# GRAMMAR-BASED COMPRESSION OF UNRANKED TREES

ADRIÀ GASCÓN, MARKUS LOHREY, SEBASTIAN MANETH, CARL PHILIPP REH, AND KURT SIEBER

Abstract. We introduce forest straight-line programs (FSLPs) 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 or commutative can be tested in polynomial time. This generalizes previous results for testing isomorphism of compressed unordered ranked trees.

## 1. Introduction

Generally speaking, 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. Henceforth, there is a great interest in problems that can be solved in polynomial time on the grammar, while requiring at least linear time on the original uncompressed object. One of the most well-known and fundamental such problems is testing equality of the strings produced by two context-free string grammars, each producing exactly one string (such grammars are also known as straight-line programs — in this paper we use the term string straight-line program, SSLP for short). Polynomial time solutions to this problem were discovered, in different contexts by different groups of people, see the survey [12] for references.

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 [6]. Such grammars are also known as tree straight-line programs, TSLPs for short, see [13] 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 [6]. 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 [15]. 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

(Adrià Gascón) Warwick University and Alan Turing Institute, UK
(Markus Lohrey, Carl Philipp Reh, Kurt Sieber) Universität Siegen, Germany
(Sebastian Maneth) Universität Bremen, Germany
*E-mail addresses*: agascon@inf.ed.ac.uk, lohrey@eti.uni-siegen.de, sebastian.maneth@gmail.com, reh@eti.uni-siegen.de, sieber@informatik.uni-siegen.de.
2010 *Mathematics Subject Classification.* E.4 Data compaction and compression.

is particularly effective for XML document trees (see [14]), and (*ii*) XML document trees can often be considered unordered (one speaks of "data-centric XML", see e.g. [1, 3, 5, 19, 20]), allowing even stronger grammar-based compressions [16].

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 $f(a, a, \ldots, a)$ with $n$ many $a$'s is not compressible with TSLPs but can be produced by an FSLP of size $O(\log n)$. FSLPs are based on the operations of horizontal and vertical forest composition from forest algebras [4]. The main contributions of this paper are the following:

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 [14] 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 5 and 6).

Top dags are another formalism for compressing unranked trees [2]. 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 3), the reverse transformation (from an FSLP to a top dag) needs time $O(\sigma \cdot n)$ and involves a multiplicative blow-up of size $O(\sigma)$ where $\sigma$ is the number of node labels of the tree (Proposition 4). A simple example (Example 6) shows that this $\sigma$-factor is unavoidable. The reason for the $\sigma$-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 $\sigma$-blow-up 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 6 and Theorem 16 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 $\Sigma$ 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 $t_1, t_2, \ldots, t_n$ (*i*) we do not distinguish between the trees $c(t_1, \ldots, t_n)$ and $c(t_{\sigma(1)}, \ldots, t_{\sigma(n)})$, where $\sigma$ is any permutation (commutativity), and (*ii*) we do not distinguish the trees $a(t_1, \ldots, t_n)$ and $a(t_1, \ldots, t_{i-1}, a(t_i, \ldots, t_{j-1}), t_j, \ldots, t_n)$ for $1 \leq i \leq j \leq n+1$ (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 16). Note that unordered tree isomorphism corresponds to the case $\mathcal{C} = \Sigma$ and $\mathcal{A} = \emptyset$ (in particular we generalize the result from [15] for ranked unordered trees). Theorem 16 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 16 also shows the utility 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.

## 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 $\mathcal{A} = (A, f_1, \dots, f_k)$ 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 $f_i$. Algebraic expressions over $\mathcal{A}$ are defined in the usual way: if $e_1, \dots, e_{n_i}$ are algebraic expressions over $\mathcal{A}$, then also $f_i(e_1, \dots, e_{n_i})$ 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 $\rho$ maps every variable $A \in V$ to an expression of the form $f_i(A_1, \dots, A_{n_i})$ (the so called *right-hand side* of $A$) such that $A_1, \dots, A_{n_i} \in V$ and the edge relation $E(P) = \{(A, B) \in V \times V \mid B \text{ occurs in } \rho(A)\}$ is acyclic. This allows to define for every variable $A \in V$ its value $[\![A]\!]_P$ inductively by $[\![A]\!]_P = f_i([\![A_1]\!]_P, \dots, [\![A_{n_i}]\!]_P)$ if $\rho(A) = f_i(A_1, \dots, A_{n_i})$. 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. In our concrete setting, validity of an SLP can be tested by a simple syntax check. The value of $P$ is $[\![P]\!] = [\![S]\!]_P$. Usually, we prove properties of SLPs by induction along the partial order $E(P)^*$.

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 $f_1, \dots, f_k$ in all right-hand sides (which is the number of variables if all right-hand sides have the standard form $f_i(A_1, \dots, A_{n_i})$).

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 $\rho(A) = f_i(A_1, \dots, A_{n_i})$. Note that the outgoing edges $(A, A_1), \dots, (A, A_{n_i})$ 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.

### 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 $\varepsilon$ 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 $\Sigma$, we speak of an SSLP over $\Sigma$. In many papers, SSLPs are just called straight-line programs; see [12] for a survey. Occasionally we consider SSLPs without a start variable $S$ and then write $(V, \rho)$.

**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 $\Sigma$ of node labels for the rest of the paper. We consider $\Sigma$-labelled rooted ordered trees, where "ordered" means that the children of a node are totally ordered. Every node has a label from $\Sigma$. 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 $\Sigma$-labelled forests is denoted by $\mathcal{F}_0(\Sigma)$ and the set of all $\Sigma$-labelled trees is denoted by $\mathcal{T}_0(\Sigma)$. As usual, we can identify trees with expressions built up from symbols in $\Sigma$ 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 $\Sigma$-labelled trees with $n \geq 0$, then the string $t_1 t_2 \cdots t_n$ is a $\Sigma$-labelled forest (in particular, the empty string $\varepsilon$ is a $\Sigma$-labelled forest).
- If $f$ is a $\Sigma$-labelled forest and $a \in \Sigma$, then $a(f)$ is a $\Sigma$-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 $\mathcal{F}_1(\Sigma)$. Let $\mathcal{T}_1(\Sigma) = \mathcal{F}_1(\Sigma) \cap \mathcal{T}_0(\Sigma \cup \{x\})$. 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 [4], we define the *forest algebra* $\mathsf{FA}(\Sigma) = (\mathcal{F}(\Sigma), \boxdot, \boxminus, (a)_{a \in \Sigma}, \varepsilon, x)$ 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).
- $\boxminus$ is the vertical concatenation operator: for forests $f_1, f_2 \in \mathcal{F}(\Sigma)$, $f_1 \boxminus f_2$ is defined if $f_1 \in \mathcal{F}_1(\Sigma)$ and in this case $f_1 \boxminus f_2$ 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 $a : \mathcal{F}(\Sigma) \to \mathcal{T}(\Sigma)$ 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 $f \boxminus g$, $fg$ instead of $f \boxdot g$, and $a$ instead of $a(\varepsilon)$. Note that a forest $f \in \mathcal{F}(\Sigma)$ can be also viewed as an algebraic expression over $\mathsf{FA}(\Sigma)$, which evaluates to $f$ itself (analogously to the free term algebra).

3.2. **First-child/next-sibling encoding.** The first-child/next-sibling encoding transforms a forest over some alphabet $\Sigma$ into a binary tree over $\Sigma \uplus \{\bot\}$. We define $\mathrm{fcns}\colon \mathcal{F}_0(\Sigma) \to \mathcal{T}_0(\Sigma \uplus \{\bot\})$ inductively by: (*i*) $\mathrm{fcns}(\varepsilon) = \bot$ and (*ii*) $\mathrm{fcns}(a(f)g) = a(\mathrm{fcns}(f)\mathrm{fcns}(g))$ for $f, g \in \mathcal{F}_0(\Sigma)$, $a \in \Sigma$. Thus, the left (resp., right) child of a node in $\mathrm{fcns}(f)$ is the first child (resp., right sibling) of the node in $f$ or a $\bot$-labelled leaf if it does not exist.

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

$$\begin{aligned} fcns(f) &= fcns(a(bc)d(e)) = a(fcns(bc)fcns(d(e))) \\ &= a(b(\bot fcns(c))d(fcns(e)\bot)) = a(b(\bot c(\bot\bot))d(e(\bot\bot)\bot)). \end{aligned}$$

3.3. **Forest straight-line programs.** A *forest straight-line program* over $\Sigma$, 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.** *Consider the FSLP* $F = (\{S, A_0, A_1, \dots, A_n, B_0, B_1, \dots, B_n\}, S, \rho)$ *over* $\{a, b, c\}$ *with* $\rho$ *defined by* $\rho(A_0) = a$, $\rho(A_i) = A_{i-1}A_{i-1}$ *for* $1 \le i \le n$, $\rho(B_0) = b(A_n x A_n)$, $\rho(B_i) = B_{i-1}\langle B_{i-1}\rangle$ *for* $1 \le i \le n$, *and* $\rho(S) = B_n\langle c\rangle$. *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.*

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

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

*Note that, although* $F$ *has size* $O(n)$, $[\![F]\!]$ *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* $(e(e(ab)c))^{2^n+1}(e(a\, e(bc)))^{2^n}$.

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

- $V'_0 = \{D, E_0, \dots, E_n, E, S_2\}$,
- $V'_1 = \{F_0, \dots, F_n\}$,
- $\rho(D) = e(abc)$,
- $\rho(E_0) = DD$,
- $\rho(E_i) = E_{i-1}E_{i-1}$ *for* $1 \le i \le n$,
- $\rho(E) = E_n D$,
- $\rho(F_0) = d(Ex)$,
- $\rho(F_i) = F_{i-1}\langle F_{i-1}\rangle$ *for* $1 \le i \le n$, *and*
- $\rho(S_2) = F_n\langle E\rangle$.

*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* $e(abc)^{2^{n+1}+1}$.

*Note that if we consider* $e$ *as associative (meaning that* $e(s\, e(tu)) = e(e(st)u)$ *for all trees* $s, t, u$*), then* $f$ *and* $f'$ *represent the same forest. If in addition we consider* $d$ *as commutative (meaning that* $d(st) = d(ts)$ *for all trees* $s, t$*) 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.*

FSLPs generalize *tree straight-line programs* (TSLPs for short) that have been used for the compression of ranked trees before, see e.g. [13]. We only need TSLPs for binary trees. A TSLP over $\Sigma$ 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 in order to compress the fcns-encoding of an unranked tree; see also [14]. 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 $|T| \in O(|F|)$. This is an easy corollary of our normal form for FSLPs that we introduce next (see also the proof of Proposition 5).

3.4. **Factorization of SSLPs.** Let $\Sigma$ 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 $i \in \{1, \dots, n\}$, which we call the $\Sigma_1$-*factorization of* $w$. Let $G = (V, \rho)$ and $G' = (V', \rho')$ be SSLPs over $\Sigma$. We call $G'$ a $\Sigma_1$-*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

$$\rho'(V) \subseteq \mathcal{L} \cup \mathcal{L}\Sigma_1\mathcal{L} \cup \mathcal{L}\mathcal{U}\Sigma_1\mathcal{L} \qquad \rho'(\mathcal{U}) \subseteq \Sigma_1\mathcal{L} \cup \mathcal{U}\mathcal{U} \qquad \rho'(\mathcal{L}) \subseteq \{\varepsilon\} \cup \Sigma_2 \cup \mathcal{L}\mathcal{L}.$$

Note that the partition $V' = V \uplus \mathcal{U} \uplus \mathcal{L}$ is uniquely determined by $V'$ and $\rho$. 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 $\Sigma_1$-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 $i \in \{1, \dots, n-1\}$.

**Lemma 1.** *Given an SSLP $G = (V, \rho)$ over $\Sigma$ and $\Sigma_1 \subseteq \Sigma$, one can compute in linear time a $\Sigma_1$-factorization of $G$ of size $O(|G|)$.*

*Proof.* Let $G = (V, \rho)$ be an SSLP over $\Sigma$, $\Sigma_1 \subseteq \Sigma$ and $\Sigma_2 = \Sigma \setminus \Sigma_1$. W.l.o.g. we can assume that $\rho(V) \subseteq VV \cup \Sigma$. For every string $w \in \Sigma^*$ with $\Sigma_1$-factorization $w = v_0 a_1 v_1 \cdots a_n v_n$ let $w_\ell, w_m, w_r \in \Sigma^*$ and $\sigma_w \in \Sigma_1 \cup \{\varepsilon\}$ be defined 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 the $\Sigma_1$-factorization $G' = (V \uplus \mathcal{U} \uplus \mathcal{L}, \rho')$ of $G$. Take new variables $A_\ell, A_m, A_r, U_{BC}, L_{BC} \notin V$ and let

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

For every $A \in V$ we define $\sigma_A \in \Sigma_1 \cup \{\varepsilon\}$ and the right-hand sides of the new variables as follows:

- If $\rho(A) = \varepsilon$ then $\rho'(A_\ell) = \rho'(A_m) = \sigma_A = \rho'(A_r) = \varepsilon$.
- 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 $\rho'(A_\ell) = B_\ell C_\ell$, $\rho'(A_m) = C_m$, $\sigma_A = \sigma_C$ and $\rho'(A_r) = C_r$,

– if $\sigma_B \in \Sigma_1$ and $\sigma_C = \varepsilon$ then $\rho'(A_\ell) = B_\ell$, $\rho'(A_m) = B_m$, $\sigma_A = \sigma_B$ and $\rho'(A_r) = B_r C_\ell$,
– if $\sigma_B, \sigma_C \in \Sigma_1$ then $\rho'(A_\ell) = B_\ell$, $\rho'(A_m) = B_m U_{BC} C_m$ with $\rho'(U_{BC}) = \sigma_B L_{BC}$ and $\rho'(L_{BC}) = B_r C_\ell$, $\sigma_A = \sigma_C$ and $\rho'(A_r) = C_r$.

Finally we define the new right-hand side for every $A \in V$: If $\sigma_A = \varepsilon$ then $\rho'(A) = A_\ell \in \mathcal{L}$. If $\sigma_A \in \Sigma_1$ and $[\![A_m]\!]_{G'} = \varepsilon$ 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 the structure of the SSLP $G$ 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 $[\![A]\!]_{G'} = ([\![A_\ell]\!]_{G'})([\![A_m]\!]_{G'})\sigma_A([\![A_r]\!]_{G'}) = w_\ell w_m \sigma_w w_r = w$. □

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 $\rho$. 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* $F^{\boxminus} = (V_1^\top, \rho_1)$ over $V_1^\perp$ with $\rho_1(A) = BC$ whenever $\rho(A) = B\langle C\rangle$. For every $A \in V_1$ the string $[\![A]\!]_{F^{\boxminus}} \in (V_1^\perp)^*$ is called the *spine* of $A$ (in $F$), 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 $\rho_0$ is the restriction of $\rho$ to $V_0^\top$. For every $A \in V_0$ we use $\mathrm{hor}(A)$ to denote the string $[\![A]\!]_{F^{\boxdot}} \in (V_0^\perp)^*$. 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 $\mathcal{F}_1(\Sigma) \setminus \mathcal{T}_1(\Sigma)$ can be eliminated without significant size blow-up.

**Theorem 2.** *From a given FSLP $F$ one can construct in linear time an FSLP $F'$ in normal form such that $[\![F']\!] = [\![F]\!]$ and $|F'| \in O(|F|)$.*

*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 $A \in V_1$, $a \in \Sigma$,
- $\rho(A) = BxC$, 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 $a \in \Sigma$, $B, C \in V$. 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 spine SSLP as the SSLP $F^{\boxminus} = (V_1^\top, \rho_1)$ over $V_1^\perp$ with $\rho_1(A) = BC$ whenever $\rho(A) = B\langle C\rangle$.

Let $\mathcal{V} = \{A \in V_1^\perp \mid \rho(A) \text{ has the form } a(x)\}$ 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 $A_0A_1 \cdots A_n$ with $A_0 \in \mathcal{V}$ and $A_1, \ldots, A_n \in \mathcal{H}$ in $[\![A]\!]_{F^{\boxminus}}$ (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 $1 \le i \le n$. Then, $A_0\langle A_1\langle \cdots \langle A_n\rangle \cdots\rangle\rangle$ can be rewritten into $a(B_1B_2 \cdots B_n x C_n \cdots C_2C_1)$. We will introduce additional variables in order to produce the horizontal concatenations $B_1B_2 \cdots B_n$ and $C_n \cdots C_2C_1$ 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 $A_0A_1 \cdots A_n$ considered in Section 3.4 are explicitly generated by the $\mathcal{V}$-factorization of the spine SSLP $F^{\boxminus}$. 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 $F^{\boxminus}$ with $|G| \in O(|F^{\boxminus}|) \le O(|F|)$. From $F$ and $G$ we obtain the FSLP $F' = (V_0 \uplus \{A_\ell, A_r \mid A \in \mathcal{L}\} \uplus \mathcal{U}, S, \rho')$ with new variables $A_\ell, A_r$ and $\rho'$ 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 the partial order of the dag shows that

- if $B \in \mathcal{L}$ with $[\![B]\!]_{F^{\boxminus}} = H_1 \cdots H_n \in \mathcal{H}^*$ then $[\![H_1\langle \cdots \langle H_n\rangle \cdots\rangle]\!]_F = [\![B_\ell x B_r]\!]_{F'}$,

- if $A \in V_0$ then $[\![A]\!]_F = [\![A]\!]_{F'}$.

From the last point we finally obtain $[\![F]\!] = [\![S]\!]_F = [\![S]\!]_{F'} = [\![F']\!]$. $\square$

## 4. Cluster algebras and top dags

In this section we introduce top dags [2, 10] 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 $\Sigma$-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 $\mathrm{rank}(s) = i$ 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 $\mathrm{rank}(s) + \mathrm{rank}(t) \leq 1$ 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 $\Sigma$ is a 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 $D = (\{S, A_0, \ldots, A_n, B_0, \ldots, B_n\}, S, \rho)$, where $\rho(A_0) = b(a)$, $\rho(A_i) = A_{i-1} \odot A_{i-1}$ for $1 \leq i \leq n$, $\rho(B_0) = A_n \odot b(\underline{b}) \odot A_n$, $\rho(B_i) = B_{i-1} \varobar B_{i-1}$ for $1 \leq i \leq n$, 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 very efficiently. 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 3.** *For a given top dag $D$ one can compute in linear time an FSLP $F$ such that $[\![F]\!] = [\![D]\!]$ and $|F| \in O(|D|)$.*

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

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

$$\varphi(s) \boxplus \varphi(t) = \varphi(s \oplus t) \tag{2}$$

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

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

Let $D = (V, S, \rho)$ be a top dag and let $\alpha$ 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 $\rho(A) = B \oplus C$ then $\rho'(A) = B \boxplus C$.

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

**Proposition 4.** *For a given FSLP $F$ with $[\![F]\!] \in \mathcal{T}_0(\Sigma)$ and $|[\![F]\!]| \geq 2$ one can compute in time $O(|\Sigma| \cdot |F|)$ a top dag $D$ such that $[\![D]\!] = [\![F]\!]$ and $|D| \in O(|\Sigma| \cdot |F|)$.*

*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 $u$:

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

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

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 2 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 $\varepsilon$ 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

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

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

(i) $[\![A]\!]_D = [\![A]\!]_F$ for every $A \in U_0^\perp$,

(ii) $[\![A^a]\!]_D = a([\![A]\!]_F)$ for every $A \in V_0$,
(iii) $[\![A_a]\!]_D = \psi_a([\![A]\!]_F)$ for every $A \in V_1$.

In order to obtain these identities, we define $\rho'$ 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 $\rho'(A_a) = B^b \odot b(a) \odot C^b$,
- if $\rho(A) = b(Bx)$ for $A \in V_1$, $B \in V_0$ then $\rho'(A_a) = B^b \odot b(a)$,
- if $\rho(A) = b(xC)$ for $A \in V_1$, $C \in V_0$ then $\rho'(A_a) = b(a) \odot C^b$,
- if $\rho(A) = b(x)$ for $A \in V_1$ then $\rho'(A_a) = b(a)$.

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 4 is sharp:

**Example 6.** *Let $\Sigma = \{a, a_1, ..., a_\sigma\}$ 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 $O(n)$: using $n = \log m$ many variables we can produce the forest $a^m$ and then $O(n)$ 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 5.** *Let $f \in \mathcal{F}(\Sigma)$ be a forest and let $F$ be an FSLP (or TSLP) over $\Sigma \uplus \{\perp\}$ with $[\![F]\!] = fcns(f)$. Then we can transform $F$ in linear time into an FSLP $F'$ over $\Sigma$ with $[\![F']\!] = f$ and $|F'| \in O(|F|)$.*

*Proof.* Let $F = (V, S, \rho)$ be an FSLP over $\Sigma \cup \{\perp\}$. By Theorem 2, we may assume that $F$ is in normal form and every variable is reachable from $S$. This implies $|\mathrm{hor}(A)| \leq 2$ for every $A \in V_0$, because $\mathrm{fcns}(f)$ is a binary tree. Hence we can compute the strings $\mathrm{hor}(A) = [\![A]\!]_{F^{\boxdot}} \in (V_0^{\perp})^*$ with $A \in V_0^{\top}$ all together in linear time, substitute $\mathrm{hor}(A)$ 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 $\varepsilon$ 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 $|\mathrm{hor}(B)| + |\mathrm{hor}(C)| = 1$. Hence the substitution leads to $a(Dx)$ or $a(xD)$ with $D \in V_0^{\perp}$. In the second case ($a(B)$) either $a = \perp$ and $|\mathrm{hor}(B)| = 0$ or $a \in \Sigma$ and $|\mathrm{hor}(B)| = 2$, hence the substitution leads to $\perp$ 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:

- $\rho(A) = \perp$
- $\rho(A) = a(BC)$
- $\rho(A) = a(Bx)$

- $\rho(A) = a(xB)$
- $\rho(A) = B\langle C\rangle$

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 $f$:

- $\rho(A) = \perp$ 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:

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

These are shown using a simple induction and cases analysis:

- $\rho(A) = \perp$: $\mathrm{fcns}([\![A]\!]_{F'}) = \mathrm{fcns}(\varepsilon) = \perp = [\![A]\!]_F$.
- $\rho(A) = a(BC)$: 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

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

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

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

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

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

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

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

This concludes the proof of the proposition. □

**Proposition 6.** *For every FSLP $F$ over $\Sigma$, we can construct in linear time a TSLP $T$ over $\Sigma \cup \{\perp\}$ with $[\![T]\!] = \mathit{fcns}([\![F]\!])$ and $|T| \in O(|F|)$.*

*Proof.* Let $F = (V, S, \rho)$ be an FSLP over $\Sigma$. We may assume that $F$ is already in normal form. We construct a TSLP $T = (V', S, \rho')$ over $\Sigma \cup \{\perp\}$ where

- $V'_0 = \{A_\triangle \mid A \in V_0^\perp\} \uplus \{S\}$
- $V'_1 = \{A_\triangle \mid A \in V_1\} \uplus \{A^\pi \mid A \in V_0\}$

with new variables $A_\triangle, A^\pi \notin V$. For every $A \in V_1$ let $R_A \in V_0$ be defined by

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

Thus, $[\![R_A]\!]_F$ is the list of right siblings of the parameter $x$ in $[\![A]\!]_F$. For $A \in V_0^\perp$ we define the top symbol $\alpha_A \in \Sigma$ as in the Proposition 4. We then define $\rho'$ by

- $\rho'(S) = S^\pi\langle\perp\rangle$
- $\rho'(A_\triangle) = B^\pi\langle\perp\rangle$ if $\rho(A) = a(B)$ for $A \in V_0^\perp, B \in V_0$
- $\rho'(A_\triangle) = B_\triangle\langle \alpha_C(C_\triangle\, R_B^\pi\langle\perp\rangle)\rangle$ if $\rho(A) = B\langle C\rangle$
- $\rho'(A_\triangle) = B^\pi$ if $\rho(A) = a(BxC)$ for $A \in V_1^\perp$, $B, C \in V_0$
- $\rho'(A^\pi) = \alpha_A(A_\triangle\, x)$ for every $A \in V_0^\perp$
- $\rho'(A^\pi) = x$ if $\rho(A) = \varepsilon$ for $A \in V_0^\top$
- $\rho'(A^\pi) = B^\pi\langle C^\pi\rangle$ if $\rho(A) = BC$ for $A \in V_0^\top$, $B, C \in V_0$.

Note that in $\rho'(A_\triangle) = B_\triangle\langle \alpha_C(C_\triangle\, R_B^\pi\langle\perp\rangle)\rangle$ we may have $C_\triangle \in V'_0$ (if $C \in V_0^\perp$) or $C_\triangle \in V'_1$ (if $C \in V_1$). In the latter case we obtain for every $f \in \mathcal{F}_0(\Sigma)$,

$$[\![A_\triangle]\!]_{F'}\langle f\rangle = [\![B_\triangle]\!]_{F'}\langle \alpha_C([\![C_\triangle]\!]_{F'}\langle f\rangle\, [\![R_B^\pi]\!]_{F'}\langle\perp\rangle)\rangle.$$

Let $\Delta\colon \mathcal{T}_0(\Sigma) \to \mathcal{F}_0(\Sigma)$ be defined by $\Delta(a(f)) = f$. We will prove the following equations, which express the role of the new variables in $V'$.

(1) $[\![A_\triangle]\!]_{F'} = \mathrm{fcns}(\Delta([\![A]\!]_F))$ for every $A \in V_0^\perp$.
(2) $[\![A^\pi]\!]_{F'}\langle \mathrm{fcns}(f)\rangle = \mathrm{fcns}([\![A]\!]_F\, f)$ for every $A \in V_0, f \in \mathcal{F}_0(\Sigma)$.
(3) $[\![A_\triangle]\!]_{F'}\langle \mathrm{fcns}(t\,[\![R_A]\!]_F)\rangle = \mathrm{fcns}(\Delta([\![A]\!]_F\langle t\rangle))$ for every $A \in V_1, t \in \mathcal{T}_0(\Sigma)$.

From (2) we obtain $[\![A^\pi]\!]_{F'}\langle\perp\rangle = [\![A^\pi]\!]_{F'}\langle \mathrm{fcns}(\varepsilon)\rangle = \mathrm{fcns}([\![A]\!]_F)$ for every $A \in V_0$. This implies $[\![F']\!] = [\![S]\!]_{F'} = [\![S^\pi]\!]_{F'}\langle\perp\rangle = \mathrm{fcns}([\![S]\!]_F) = \mathrm{fcns}([\![F]\!])$ which concludes the proof of Proposition 6. Hence only equations (1) to (3) remain to be proved, which is done by the following induction on the partial order induced by the dag $F$. Let $A \in V$:

(1) must be proved for every $A \in V_0^\perp$:

- If $\rho(A) = a(B)$ then $[\![A_\triangle]\!]_{F'} = [\![B^\pi]\!]_{F'}\langle\perp\rangle = \mathrm{fcns}([\![B]\!]_F) = \mathrm{fcns}(\Delta([\![A]\!]_F))$.

- If $\rho(A) = B\langle C\rangle$ with $B \in V_1$, $C \in V_0^\perp$ then

$$\begin{aligned}
[\![A_\triangle]\!]_{F'} &= [\![B_\triangle]\!]_{F'}\langle \alpha_C([\![C_\triangle]\!]_{F'}\,[\![R_B^\pi]\!]_{F'}\langle\perp\rangle)\rangle \\
&= [\![B_\triangle]\!]_{F'}\langle \alpha_C(\mathrm{fcns}(\Delta([\![C]\!]_F))\,\mathrm{fcns}([\![R_B]\!]_F))\rangle \\
&\quad \text{by induction for } C \in V_0^\perp \text{ and } R_B \in V_0 \\
&= [\![B_\triangle]\!]_{F'}\langle \mathrm{fcns}(\alpha_C(\Delta([\![C]\!]_F))\,[\![R_B]\!]_F)\rangle \\
&\quad \text{by definition of fcns} \\
&= [\![B_\triangle]\!]_{F'}\langle \mathrm{fcns}([\![C]\!]_F\,[\![R_B]\!]_F)\rangle \\
&= \mathrm{fcns}(\Delta([\![B]\!]_F\langle [\![C]\!]_F\rangle)) \\
&\quad \text{by induction for } B \in V_1 \\
&= \mathrm{fcns}(\Delta([\![A]\!]_F)).
\end{aligned}$$

(2) must be proved for every $A \in V_0$:

- If $A \in V_0^\perp$ then

$$\begin{aligned}
[\![A^\pi]\!]_{F'}\langle \mathrm{fcns}(f)\rangle &= \alpha_A([\![A_\triangle]\!]_{F'}\,\mathrm{fcns}(f)) \\
&= \alpha_A(\mathrm{fcns}(\Delta([\![A]\!]_F))\,\mathrm{fcns}(f)) \\
&\quad \text{by equation (1) for } A_\triangle \\
&= \mathrm{fcns}(\alpha_A(\Delta([\![A]\!]_F))\,f) \\
&\quad \text{by definition of fcns} \\
&= \mathrm{fcns}([\![A]\!]_F\,f).
\end{aligned}$$

- If $\rho(A) = \varepsilon$ then $[\![A^\pi]\!]_{F'}\langle \mathrm{fcns}(f)\rangle = \mathrm{fcns}(f) = \mathrm{fcns}(\varepsilon\, f) = \mathrm{fcns}([\![A]\!]_F\, f)$.
- If $\rho(A) = BC$ then

$$\begin{aligned}
[\![A^\pi]\!]_{F'}\langle \mathrm{fcns}(f)\rangle &= [\![B^\pi]\!]_{F'}\langle [\![C^\pi]\!]_{F'}\langle \mathrm{fcns}(f)\rangle\rangle \\
&= \mathrm{fcns}\langle [\![B]\!]_F\,[\![C]\!]_F\,f\rangle \\
&\quad \text{by induction for } B \text{ and } C \\
&= \mathrm{fcns}([\![BC]\!]_F\,f) \\
&\quad \text{by definition of fcns} \\
&= \mathrm{fcns}([\![A]\!]_F\,f).
\end{aligned}$$

(3) must be proved for every $A \in V_1$:

- If $\rho(A) = a(BxC)$ then

$$\begin{aligned}
[\![A_\triangle]\!]_{F'}\langle \mathrm{fcns}(t\,[\![R_A]\!]_F)\rangle &= [\![B^\pi]\!]_{F'}\langle \mathrm{fcns}(t\,[\![R_A]\!]_F)\rangle \\
&= \mathrm{fcns}([\![B]\!]_F\,t\,[\![C]\!]_F) \\
&\quad \text{by induction for } B \text{ and because } R_A = C \\
&= \mathrm{fcns}(\Delta([\![a(BxC)]\!]_F\langle t\rangle)) \\
&= \mathrm{fcns}(\Delta([\![A]\!]_F\langle t\rangle)).
\end{aligned}$$

- If $\rho(A) = B\langle C\rangle$ with $A, B, C \in V_1$ then

$$\begin{aligned}
[\![A_\triangle]\!]_{F'}\langle \mathrm{fcns}(t\,[\![R_A]\!]_F)\rangle &= [\![B_\triangle]\!]_{F'}\langle \alpha_C([\![C_\triangle]\!]_{F'}\langle \mathrm{fcns}(t\,[\![R_A]\!]_F)\rangle\,[\![R_B^\pi]\!]_{F'}\langle\bot\rangle)\rangle \\
&= [\![B_\triangle]\!]_{F'}\langle \alpha_C([\![C_\triangle]\!]_{F'}\langle \mathrm{fcns}(t\,[\![R_C]\!]_F)\rangle\,\mathrm{fcns}([\![R_B]\!]_F))\rangle \\
&\quad\ \text{by induction for } R_B \text{ and because } R_A = R_C \\
&= [\![B_\triangle]\!]_{F'}\langle \alpha_C(\mathrm{fcns}(\Delta([\![C]\!]_F\langle t\rangle))\,\mathrm{fcns}([\![R_B]\!]_F))\rangle \\
&\quad\ \text{by induction for } C \\
&= [\![B_\triangle]\!]_{F'}\langle \mathrm{fcns}(\alpha_C(\Delta([\![C]\!]_F\langle t\rangle))\,[\![R_B]\!]_F)\rangle \\
&\quad\ \text{by definition of fcns} \\
&= [\![B_\triangle]\!]_{F'}\langle \mathrm{fcns}([\![C]\!]_F\langle t\rangle\,[\![R_B]\!]_F)\rangle \\
&= \mathrm{fcns}(\Delta([\![B]\!]_F\langle [\![C]\!]_F\langle t\rangle\rangle)) \\
&\quad\ \text{by induction for } B \\
&= \mathrm{fcns}(\Delta([\![A]\!]_F\langle t\rangle)).
\end{aligned}$$

This concludes the proof of the proposition. □

Proposition 6 and the construction from [7, Proposition 8.3.2] allow to reduce the evaluation of forest automata on FSLPs (for a definition of forest and tree automata, see [7]) to the evaluation of ordinary tree automata on binary trees. The latter problem can be solved in polynomial time [17], which yields:

**Corollary 7.** *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 $\mathrm{fcns}([\![F]\!])$ using Proposition 6. We also convert $A$ in polynomial time into a tree automaton $A'$ such that $A'$ accepts $\mathrm{fcns}(f)$ if and only if $A$ accepts $f$, using the construction from [7, Proposition 8.3.2]. Finally, we use the result from [17] to check in polynomial time whether $A'$ accepts $[\![T]\!]$. □

In [2], a linear time algorithm is presented that constructs from a tree of size $n$ with $\sigma$ many node labels a top dag of size $O(n/\log_\sigma^{0.19} n)$. In [10] this bound was improved to $O(n \log\log n/\log_\sigma n)$ (for the same algorithm as in [2]). In [18] we recently presented an alternative construction that achieves the information-theoretic optimum of $O(n/\log_\sigma n)$. Moreover, as in [2], the constructed top dag satisfies the additional size bound $O(d \cdot \log n)$, where $d$ is the size of the minimal dag of $t$. With Proposition 3 and 6 we get:

**Corollary 8.** *Given a tree $t$ of size $n$ with $\sigma$ many node labels, one can construct in linear time an FSLP for $t$ (or an TSLP for fcns$(t)$) of size $O(n/\log_\sigma n) \cap O(d \cdot \log n)$, 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 utility 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 $t_1, \ldots, t_n \in \mathcal{T}_0(\Sigma)$, all permutations $\sigma\colon \{1,\ldots,n\} \to \{1,\ldots,n\}$, and all $1 \le i \le j \le n+1$:

$$a(t_1 \cdots t_n) = a(t_1 \cdots t_{i-1} a(t_i \cdots t_{j-1}) t_j \cdots t_n) \tag{8}$$

$$c(t_1 \cdots t_n) = c(t_{\sigma(1)} \cdots t_{\sigma(n)}). \tag{9}$$

Note that the standard law of associativity for a binary symbol $\circ$ (i.e., $x \circ (y \circ z) = (x \circ y) \circ z$) can be captured by making $\circ$ an (unranked) associative symbol in the sense of (8).

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 (8) 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 (8) 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 $f \in \mathcal{F}_0(\Sigma)$ and $t_1, \ldots, t_n \in \mathcal{T}_0(\Sigma)$:

$$\phi_a(b(f)) = \begin{cases} \phi_a(f) & \text{if } a \in \mathcal{A} \text{ and } a = b, \\ b(\phi_b(f)) & \text{otherwise,} \end{cases} \qquad \phi_a(t_1 \cdots t_n) = \phi_a(t_1) \cdots \phi_a(t_n).$$

In particular, $\phi_a(\varepsilon) = \varepsilon$. Moreover, 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*

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

To show the following simple lemma one considers the terminating and confluent rewriting system obtained by directing the equations (8) from right to left.

**Lemma 9.** *For two forests $f_1, f_2 \in \mathcal{F}_0(\Sigma)$, $\mathit{nf}_{\mathcal{A}}(f_1) = \mathit{nf}_{\mathcal{A}}(f_2)$ if and only if $f_1$ and $f_2$ are equal modulo the identities in* (8) *for all $a \in \mathcal{A}$.*

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

$$a(t_1 \cdots t_{i-1} a(t_i \cdots t_{j-1}) t_j \cdots t_n) \to a(t_1 \cdots t_n) \tag{10}$$

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 $\to$ 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 (8) 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 10.** *From a given FSLP $F = (V, S, \rho)$ over $\Sigma$ one can construct in time $\mathcal{O}(|F| \cdot |\Sigma|)$ an FSLP $F'$ with $[\![F']\!] = \mathit{nf}_{\mathcal{A}}([\![F]\!])$.*

*Proof.* By Theorem 2, we assume that $F$ 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 $\omega_B$ 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 $\rho'$ 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 $\Sigma$ 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 (9) for all $c \in \mathcal{C}$.

We start with a general definition: Let $\Delta$ be a possibly infinite alphabet together with a total order $<$. Let $\leq$ be the reflexive closure of $<$. Define the function $\mathrm{sort}^< \colon \Delta^* \to \Delta^*$ 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 11.** *Let $G$ be an SSLP over $\Delta$ and let $<$ be some total order on $\Delta$. 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 $\Delta$ where $V' = V \uplus \{A_a \mid A \in V, a \in \Delta\}$ with new variables $A_a \notin V$, and $\rho'$ 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 $\rho(A) = BC$,
- $\rho'(S') = A_{a_1} \ldots A_{a_n}$ 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 $[\![A]\!]_G$. □

In order to define the commutative normal form, we need a total order on $\mathcal{F}_0(\Sigma)$. Recall that elements of $\mathcal{F}_0(\Sigma)$ are particular strings over the alphabet $\Gamma := \Sigma \cup \{(,)\}$. Fix an arbitrary total order on $\Gamma$ and let $<_{\mathrm{llex}}$ be the *length-lexicographic order* on $\Gamma^*$ induced by $<$: for $x, y \in \Gamma^*$ we have $x <_{\mathrm{llex}} y$ if $|x| < |y|$ or ($|x| = |y|$, $x = uav$, $y = ubv'$, and $a < b$ for $u, v, v' \in \Gamma^*$ and $a, b \in \Gamma$). We now consider the restriction of $<_{\mathrm{llex}}$ to $\mathcal{F}_0(\Sigma) \subseteq \Gamma^*$. For the proof of the following lemma one first constructs SSLPs for the strings $[\![F_1]\!], [\![F_2]\!] \in \Gamma^*$ (the construction is similar to the case of TSLPs, see [6]) and then uses [15, Lemma 3] according to which SSLP-encoded strings can be compared in polynomial time with respect to $<_{\mathrm{llex}}$.

**Lemma 12.** *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 $[\![F_2]\!]$, respectively, where the latter are viewed as a string over the alphabet $\Sigma \cup \{(,)\}$. The construction is similar to the case of TSLPs; see [6]: Consider $F_1 = (V, S, \rho)$. By Theorem 2 we can assume that $F_1$ is in normal form. We define the SSLP $G_1 = (V', S, \rho')$ over $\Sigma \cup \{(,)\}$, where $V' = V_0 \cup \{A_1, A_2 \mid A \in V_1\}$ and $\rho'$ is defined as follows:

- If $\rho(A) = \varepsilon$ or $\rho(A) = BC$ then $\rho'(A) = \rho(A)$,
- If $\rho(A) = a(B)$ then $\rho'(A) = a(B)$.
- If $\rho(A) = B\langle C\rangle$ with $C \in V_0$ then $\rho'(A) = B_1 C B_2$.
- If $\rho(A) = a(BxC)$ then $\rho'(A_1) = a(B$, and $\rho'(A_2) = C)$.
- If $\rho(A) = B\langle C\rangle$ with $C \in V_1$ then $\rho'(A_1) = B_1 C_1$ and $\rho'(A_2) = C_2 B_2$.

The correctness of the construction can be easily verified.

The rest of the proof follows immediately from [15, Lemma 3]: Given SSLPs $G_1$ and $G_2$ over the same terminal alphabet $\Gamma$, we can check in polynomial time whether $[\![G_1]\!] <_{\mathrm{llex}} [\![G_2]\!]$, $[\![G_2]\!] <_{\mathrm{llex}} [\![G_1]\!]$ or $[\![G_1]\!] = [\![G_2]\!]$. □

From the restriction of $<_{\mathrm{llex}}$ to $\mathcal{T}_0(\Sigma) \subseteq \Gamma^*$ we obtain the function $\mathrm{sort}^{<_{\mathrm{llex}}}$ on $\mathcal{T}_0(\Sigma)^* = \mathcal{F}_0(\Sigma)$. We define $\mathrm{nf}_{\mathcal{C}} \colon \mathcal{F}_0(\Sigma) \to \mathcal{F}_0(\Sigma)$ by

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

Obviously, $f_1, f_2 \in \mathcal{F}(\Sigma)$ are equal modulo the identities in (9) for all $c \in \mathcal{C}$ if and only if $\mathrm{nf}_{\mathcal{C}}(f_1) = \mathrm{nf}_{\mathcal{C}}(f_2)$. Using this fact and Lemma 9 it is not hard to show:

**Lemma 13.** *For $f_1, f_2 \in \mathcal{F}_0(\Sigma)$ we have $nf_{\mathcal{C}}(nf_{\mathcal{A}}(f_1)) = nf_{\mathcal{C}}(nf_{\mathcal{A}}(f_2))$ if and only if $f_1$ and $f_2$ are equal modulo the identities in* (8) *and* (9) *for all $a \in \mathcal{A}$, $c \in \mathcal{C}$.*

*Proof.* It suffices to show that $\mathrm{nf}_{\mathcal{C}}(\mathrm{nf}_{\mathcal{A}}(f_1)) = \mathrm{nf}_{\mathcal{C}}(\mathrm{nf}_{\mathcal{A}}(f_2))$ if $f_1$ and $f_2$ can be transformed into each other by a single application of (8) or (9); let us write $f_1 =_{(8)} f_2$ or $f_1 =_{(9)} f_2$, respectively, for the latter. The case $f_1 =_{(8)} f_2$ is clear, since this implies $\mathrm{nf}_{\mathcal{A}}(f_1) = \mathrm{nf}_{\mathcal{A}}(f_2)$ by Lemma 9. Now assume that $f_1 =_{(9)} f_2$. As in the proof of Lemma 9, consider the infinite rewriting system with the rules from (10) and the associated rewrite relation $\to$. The crucial observation is that $f_1 =_{(9)} f_2 \to f_2'$ implies that there exists $f_1'$ such that $f_1 \to f_1' =_{(9)} f_2'$. Since $f_2 \to^* \mathrm{nf}_{\mathcal{A}}(f_2)$, it follows that there exists $f_1'$ such that $f_1 \to^* f_1' =_{(9)} \mathrm{nf}_{\mathcal{A}}(f_2)$. But this implies that $f_1'$ is irreducible with respect to $\to$, i.e., $f_1' = \mathrm{nf}_{\mathcal{A}}(f_1)$. We obtain $\mathrm{nf}_{\mathcal{A}}(f_1) =_{(9)} \mathrm{nf}_{\mathcal{A}}(f_2)$ and hence $\mathrm{nf}_{\mathcal{C}}(\mathrm{nf}_{\mathcal{A}}(f_1)) = \mathrm{nf}_{\mathcal{C}}(\mathrm{nf}_{\mathcal{A}}(f_2))$. $\square$

For our main technical result (Theorem 15) we need a strengthening of our FSLP normal form. Recall the notion of the *spine* from Section 3. We say that an FSLP $F = (V, S, \rho)$ is in *strong normal form* if it is in normal form and for every $A \in V_0^{\perp}$ with $\rho(A) = B\langle C \rangle$ either $B \in V_1^{\perp}$ or $|[\![C]\!]_F| \geq |[\![D]\!]_F| - 1$ for every $D \in V_1^{\perp}$ which occurs in $\mathrm{spine}(B)$ (note that $|[\![D]\!]_F| - 1$ is the number of nodes in $[\![D]\!]_F$ except for the parameter $x$).

**Lemma 14.** *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 modify the right-hand sides of variables $A \in V_0^{\perp}$ with $\rho(A) = B\langle C \rangle$ and $|\mathrm{spine}(B)| \geq 2$. Basically, we replace the vertical concatenations $B\langle C \rangle$ by polynomially many vertical concatenations $B_i\langle C_i \rangle$ which satisfy the condition of the strong normal form.

$F'$ is obtained from $F$ by modifying (only) the right-hand sides of variables $A \in V_0^{\perp}$ with $\rho(A) = B\langle C \rangle$ and $|\mathrm{spine}_F(B)| > 1$. The modification for such a variable $A$ works as follows.

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

For every $1 \leq i \leq m$ we can construct in polynomial time an SSLP $G_i = (N_i, E_i, \rho_i)$ over $V_1^{\perp}$ such that $[\![G_i]\!] = B_{p_{i+1}+1} \cdots B_{p_i - 1}$ (see e.g. [17, Lemma 1]), hence $\mathrm{spine}_F(B) = [\![G_m]\!] B_{p_m} \cdots [\![G_1]\!] B_{p_1} = [\![E_m]\!]_{G_m} D_m \cdots [\![E_1]\!]_{G_1} D_1$. We may assume that the variable sets $N_i$ are pairwise disjoint and also disjoint from $V$, and that $\rho_i(N_i) \subseteq V_1^{\perp} \cup N_i N_i$ whenever $[\![G_i]\!] \neq \varepsilon$. Hence we can add each $X \in N_i$ (with $[\![G_i]\!] \neq \varepsilon$) 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 $\rho_i(X) = D \in V_1^{\perp}$.

Thus we obtain $[\![B]\!]_{F'} = [\![E_m\langle D_m\langle \cdots E_1\langle D_1\rangle \cdots\rangle\rangle]\!]_{F'}$.

Now we add new variables $A_i$ for $1 \le i \le m-1$ and $C_i$ for $1 \le i \le m$ to the variable set $V_0'$ of $F'$ and define

- $\rho'(C_1) = D_1\langle C\rangle$,
- $\rho'(C_i) = D_i\langle A_{i-1}\rangle$ for $2 \le i \le m$,
- $\rho'(A_i) = E_i\langle C_i\rangle$, if $[\![E_i]\!]_G \neq \varepsilon$, otherwise $\rho'(A_i) = \rho'(C_i)$ for $1 \le i \le m-1$,
- $\rho'(A) = E_m\langle C_m\rangle$, if $[\![E_m]\!]_G \neq \varepsilon$, otherwise $\rho'(A) = \rho'(C_m)$.

Obviously, $[\![C_i]\!]_{F'} = [\![D_i\langle \ldots D_1\langle C\rangle \ldots\rangle]\!]_{F'}$ for $1 \le i \le m$, which implies $|[\![C_i]\!]_{F'}| \ge |[\![D_j]\!]_{F'}| - 1$ for all $1 \le j \le i \le m$ (equality holds if $i = m$ and $[\![C]\!]_{F'} = \varepsilon$, since the parameter $x$ of $D_m$ disappears in this case). Hence, the right-hand sides $\rho'(A_i)$ and $\rho'(A)$ meet the definition of strong normal form. Moreover, $[\![A]\!]_{F'} = [\![E_m\langle D_m\langle \ldots D_1\langle C\rangle \ldots\rangle]\!]_{F'} = [\![B\langle C\rangle]\!]_{F'}$. By induction on the partial order of the dag, this implies $[\![A]\!]_{F'} = [\![A]\!]_F$ for *all* $A \in V$, because the right-hand sides of other variables in $V$ are not modified. In particular, $[\![F']\!] = [\![S]\!]_{F'} = [\![S]\!]_F = [\![F]\!]$, which concludes the proof. □

**Theorem 15.** *From a given FSLP $F$ we can construct in polynomial time an FSLP $F'$ with $[\![F']\!] = nf_{\mathcal{C}}([\![F]\!])$.*

*Proof.* Let $F = (V, S, \rho)$. By Theorem 2 and Lemma 14 we may assume that $F$ is in strong normal form. For every $A \in V_1$ let

$$\mathrm{args}(A) = \{t \in \mathcal{T}_0(\Sigma) \mid |t| \ge |[\![D]\!]_F| - 1 \text{ for each symbol } D \text{ in spine}(A)\}$$

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 $A \in V_1$, $t \in \mathrm{args}(A)$.

From (1) we obtain $[\![F']\!] = [\![S]\!]_{F'} = \mathrm{nf}_{\mathcal{C}}([\![S]\!]_F) = \mathrm{nf}_{\mathcal{C}}([\![F]\!])$ which concludes the proof.

To define $\rho'$, let $V^c = V_0^c \cup V_1^c$ with $V_1^c = \{A \in V_1 \mid \rho(A) = a(BxC) \text{ with } a \in \mathcal{C}\}$ and $V_0^c = \{A \in V_0 \mid \rho(A) = a(B) \text{ with } a \in \mathcal{C} \text{ or } \rho(A) = D\langle C\rangle \text{ with } D \in V_1^c\}$ be the set of *commutative variables*. We set $\rho'(A) = \rho(A)$ for $A \in V \setminus V^c$. For $A \in V^c$ we define $\rho'(A)$ by induction along the partial order of the dag:

(1) $\rho(A) = a(B)$: Let $M_A$ be the set of all $C \in V_0^{\perp}$ which are below $A$ in the dag, and let $w = \mathrm{hor}(B) = [\![B]\!]_{F^{\boxdot}} \in M_A^*$. By induction, $\rho'$ is already defined on $M_A$, and thus $[\![C]\!]_{F'}$ is defined for every $C \in M_A$. By Lemma 12, we can compute in polynomial time a total order $<$ on $M_A$ such that $C < D$ implies $[\![C]\!]_{F'} \le_{\mathrm{llex}} [\![D]\!]_{F'}$ for all $C, D \in M_A$. By Lemma 11, 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 we finally set $\rho'(A) = a(S_w)$.
(2) $\rho(A) = B\langle C\rangle$: Let $\rho(B) = a(DxE)$. We define $G_w = (V_w, S_w, \rho_w)$ as before, but with $w = [\![DCE]\!]_{F^{\boxdot}}$ instead of $w = [\![B]\!]_{F^{\boxdot}}$, and we set $\rho'(A) = a(S_w)$.
(3) $\rho(A) = a(BxC)$: We define $G_w = (V_w, S_w, \rho_w)$ as before, this time with $w = [\![BC]\!]_{F^{\boxdot}}$, and we set $\rho'(B) = a(S_w x)$.

The main idea is that the strong normal form ensures that in right-hand sides of the form $a(DxE)$ with $a \in \mathcal{C}$ one can move the parameter $x$ to the last position (see point 3 above), since only trees that are larger than all trees produced from $D$ and $E$ are substituted for $x$.

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

(i) $\rho(A) = a(B)$ with $a \in \mathcal{C}$:

Let $w = [\![B]\!]_{F^{\boxdot}} = A_1 \cdots A_m$ with $m \geq 0$. Then

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

(ii) $\rho(A) = B\langle C\rangle$ with $A, C \in V_0^{\perp}$ and $B \in V_1^{c}$, i.e., $\rho(B) = a(DxE)$ with $a \in \mathcal{C}$:

Let $w = [\![DCE]\!]_{F^{\boxdot}} = A_1 \cdots A_m$ with $m \geq 0$. Then

$$
\begin{aligned}
\mathrm{nf}_{\mathcal{C}}([\![A]\!]_F) &= \mathrm{nf}_{\mathcal{C}}(a([\![DCE]\!]_F)) \\
&= a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![DCE]\!]_F))) \text{ by definition of } \mathrm{nf}_{\mathcal{C}} \text{ since } a \in \mathcal{C} \\
&= a(\mathrm{sort}^{<_{\mathrm{llex}}}(\mathrm{nf}_{\mathcal{C}}([\![A_1]\!]_F) \cdots \mathrm{nf}_{\mathcal{C}}([\![A_m]\!]_F))) \\
&= a([\![S_w]\!]_{F'}) \text{ as in (i)} \\
&= [\![A]\!]_{F'}
\end{aligned}
$$

(iii) $\rho(A) = a(BxC)$ with $a \in \mathcal{C}$:

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 $t \in \mathrm{args}(A)$ and $1 \leq i \leq m$ we have $|\mathrm{nf}_{\mathcal{C}}(t)| = |t| \geq |[\![A]\!]_F| - 1 > |[\![BC]\!]_F| \geq |[\![A_i]\!]_F| = |\mathrm{nf}_{\mathcal{C}}([\![A_i]\!]_F)|$, hence $\mathrm{nf}_{\mathcal{C}}([\![A_i]\!]_{F'}) \leq_{\mathrm{llex}} \mathrm{nf}_{\mathcal{C}}(t)$. Thus we obtain

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

(iv) $\rho(A) = B\langle C\rangle$ with $A, C \in V_0^{\perp}$ and $B \in V_1^{\top}$:

Then $\rho'(A) = B\langle C\rangle$ and $|[\![C]\!]_F| \geq |[\![D]\!]_F| - 1$ for every $D$ which occurs in $\mathrm{spine}(B)$, i.e., $[\![C]\!]_F \in \mathrm{args}(B)$. Hence

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

(v) $\rho(A) = B\langle C\rangle$ with $A, B, C \in V_1$:

Let $t \in \mathrm{args}(A) \subseteq \mathrm{args}(B) \cap \mathrm{args}(C)$. Then $[\![C]\!]_F\langle t\rangle \in \mathrm{args}(B)$, and hence

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

This concludes the proof of the theorem. □

**Theorem 16.** *For trees $s, t$ we can test in polynomial time whether $s$ and $t$ are equal modulo the identities in* (8) *and* (9)*, if $s$ and $t$ are given succinctly by one of the following three formalisms: (i) FSLPs, (ii) top dags, (iii) TSLPs for the fcns-encodings of $s, t$.*

*Proof.* By Proposition 3 and 5 it suffices to show Theorem 16 for the case that $t_1$ and $t_2$ are given by FSLPs $F_1$ and $F_2$, respectively. By Lemma 13 and Lemma 12 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 10 and Theorem 15. □

## 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 that was finally solved by Jeż [11], 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 [9], context matching [9], and one-context unification [8] 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.

## References


[1] S. Abiteboul, P. Bourhis, and V. Vianu. Highly expressive query languages for unordered data trees. *Theor. Comput. Syst.*, 57(4):927–966, 2015.

[2] P. Bille, I. L. Gørtz, G. M. Landau, and O. Weimann. Tree compression with top trees. *Inf. Comput.*, 243:166–177, 2015.

[3] A. Boiret, V. Hugot, J. Niehren, and R. Treinen. Logics for unordered trees with data constraints on siblings. In *Proc. LATA 2015*, LNCS 8977, 175–187. Springer, 2015.

[4] M. Bojańczyk and I. Walukiewicz. Forest algebras. In *Proc. 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.

[5] I. Boneva, R. Ciucanu, and S. Staworko. Schemas for unordered XML on a DIME. *Theor. Comput. Syst.*, 57(2):337–376, 2015.

[6] G. Busatto, M. Lohrey, and S. Maneth. Efficient memory representation of XML document trees. *Information Systems*, 33(4-5):456–474, 2008.

[7] 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.

[8] C. Creus, A. Gascón, and G. Godoy. One-context unification with STG-compressed terms is in NP. In *Proc. RTA 2012*, LIPIcs 15, 149–164. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2012.

[9] 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.

[10] L. Hübschle-Schneider and R. Raman. Tree compression with top trees revisited. In *Proc. SEA 2015*, LNCS 9125, 15–27. Springer, 2015.

[11] A. Jeż. Context unification is in PSPACE. In *Proc. ICALP 2014, Part II*, LNCS 8573, 244–255. Springer, 2014.

[12] M. Lohrey. Algorithmics on SLP-compressed strings: a survey. *Groups Complexity Cryptology*, 4(2):241–299, 2012.

[13] M. Lohrey. Grammar-based tree compression. In *Proc. DLT 2015*, LNCS 9168, 46–57. Springer, 2015.

[14] M. Lohrey, S. Maneth, and R. Mennicke. XML tree structure compression using RePair. *Information Systems*, 38(8):1150–1167, 2013.

[15] M. Lohrey, S. Maneth, and F. Peternek. Compressed tree canonization. In *Proc. ICALP 2015, Part II*, 337–349. Springer, 2015.

[16] M. Lohrey, S. Maneth, and C. P. Reh. Compression of unordered XML trees. In *Proc. ICDT 2017*, LIPIcs 68, 18:1–18:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2017.

[17] M. Lohrey, S. Maneth, and M. Schmidt-Schauß. Parameter reduction and automata evaluation for grammar-compressed trees. *J. Comput. Syst. Sci.*, 78(5):1651–1669, 2012.

[18] M. Lohrey, P. Reh, and K. Sieber. Optimal top dag construction. `https://arxiv.org/abs/1712.05822`, arXiv.org, 2017.

[19] 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.

[20] S. Zhang, Z. Du, and J. T. Wang. New techniques for mining frequent patterns in unordered trees. *IEEE Trans. Cybern.*, 45(6):1113–1125, 2015.