Acta Cybernetica **22** (2015) 373–392.

Context-Free Tree Grammars are as Powerful as Context-Free Jungle Grammars

Frank Drewes* and Joost Engelfriet†

Dedicated to the memory of Ferenc Gécseg

Abstract

Jungles generalize trees by sharing subtrees and allowing garbage. It is shown that IO context-free tree grammars generate the same jungle languages as context-free jungle grammars. Also, they define the same subsets of any algebra.

Keywords: context-free tree grammar, jungle, delegation network

1 Introduction

One of the main motivations for studying tree language theory is that a tree over a ranked alphabet is a term, which can be interpreted as an element of any algebra, see, e.g., Sections I.2, I.3, II.1 and II.2 of the influential book of Gécseg and Steinby [13]. Thus, the interpretation of a tree language, i.e., a set of trees, becomes a subset of the algebra. Regular tree grammars [13, Section II.3] and context-free tree grammars [14, Section 15] generate tree languages. However, as shown by Mezei and Wright in [17], a regular tree grammar can also naturally be viewed as a system of equations (or, more informally, as a recursive program) that has a least fixed point semantics in any algebra, and thus defines a subset of the algebra. The main result of [17] is that, for any algebra, the semantics of a regular tree grammar G equals the interpretation of the tree language $L(G)$ generated by G. Thus, the semantics of the program G is determined by the set of syntactic objects it generates. This program-schematic result was generalized to context-free tree grammars in [11], both for call by value semantics vs. inside-out (IO) generation, and for call by name semantics vs. outside-in (OI) generation. However, in the call by value case it holds for deterministic algebras, but *not for nondeterministic algebras*. In a usual, deterministic algebra, each operator symbol of rank k

*Department of Computing Science, Umeå University, S–901 87 Umeå, Sweden, E-mail: drewes@cs.umu.se

†Leiden Institute of Advanced Computer Science, Leiden University, P.O. Box 9612, NL–2300 RA Leiden, The Netherlands, E-mail: j.engelfriet@liacs.leidenuniv.nl

is interpreted as a k-ary operation on the domain of the algebra, whereas in a nondeterministic algebra, it is interpreted as a $(k+1)$-ary relation (see, e.g., [13, Section II.2]). Thus, in a nondeterministic algebra, a tree is interpreted as a subset of the algebra; as in the deterministic case, a tree language is also interpreted as a subset of the algebra, viz. the union of the interpretations of its elements. Since grammars are essentially nondeterministic programs, it is natural to interpret them in nondeterministic algebras.

It was shown in [5] that the call by value case for nondeterministic algebras can be handled by considering *jungles* (or DOAGs, directed ordered acyclic graphs) instead of trees. A jungle is a representation of a tree, in which equal subtrees can be shared, and in which "garbage" can occur that is not used in the tree, see, e.g., [1, 15, 16, 18].[1] Jungles can be interpreted in any nondeterministic algebra, in a natural way. The sharing of subtrees allows to fix a nondeterministic choice for later multiple use, whereas the garbage allows to force the evaluation of trees that are later disregarded. As shown in [5], a context-free tree grammar G can be turned into a graph grammar that generates jungles, in a straightforward way, such that the call by value semantics of G equals the interpretation of the "jungle language" $L_{\mathrm{J}}(G)$ generated by G, for any nondeterministic algebra.

On the basis of the above "Mezei-and-Wright-like" result for $L_{\mathrm{J}}(G)$, one may ask whether context-free tree grammars have the same jungle generating power as *context-free jungle grammars*, which are context-free graph grammars in which all right-hand sides of rules are jungles (see [5, Definition 7.6]). In this paper, we answer this question affirmatively. Moreover, we define the least fixed point semantics of a context-free jungle grammar in any nondeterministic algebra, viewing the grammar as a system of equations, and we prove that context-free jungle grammars define the same subsets of the algebra as IO context-free tree grammars. As a corollary we obtain that the above Mezei-and-Wright-like result also holds for context-free jungle grammars G. Finally, a context-free jungle grammar generates the tree language obtained by unfolding the generated jungles, and we show that context-free jungle grammars generate the same tree languages as IO context-free tree grammars.

Thus we conclude that, in all respects, IO context-free tree grammars have the same power as context-free jungle grammars.

In Section 2 we define basic concepts, such as trees and IO context-free tree grammars. Jungles are defined in Section 3, and we define the substitution of one jungle for a node of another jungle. It is shown that this substitution is confluent and associative (in the sense of [3]), a folklore result. In Section 4 we define context-free jungle grammars (CFJGs), in such a way that context-free tree grammars (CFTGs) are a special case. The derivations of a CFJG use the jungle substitution defined in the previous section. We show the simple fact that the rules of a CFJG can be substituted into one another (generalizing the corresponding property of context-free string grammars), and we prove our main result: for every CFJG G there is a CFTG H that generates the same jungle language. As a corollary we obtain that CFJGs generate the same tree languages as IO CFTGs. In

[1] When trees are called terms, jungles are called term graphs.

Section 5 we turn to semantics. We recall (nondeterministic) algebras and define the interpretation of jungles in such an algebra. Then we introduce the notion of a *jungle delegation network*, which is a CFJG G together with an algebra A. It generalizes the (finitary, tree) delegation network of [4, 5], which is an IO CFTG with an algebra. Finally, we define the least fixed point semantics of a jungle delegation network (G, A), and prove that it defines the same subset of A as the tree delegation network (H, A), where H is as above. As a corollary we obtain that that subset is equal to the interpretation in A of the jungle language $L_{\mathrm{J}}(G)$ generated by G: our Mezei-and-Wright-like result for context-free jungle grammars.

The results of this paper were already suggested in the Conclusion of [5].

2 Basic Terminology

The set of all natural numbers (including zero) is denoted $\mathbb{N}$. For $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. The set of all finite strings (or sequences) over a set A is denoted by A^*, and λ denotes the empty string. The length of a string u is denoted by $|u|$.

We assume functions to be total, i.e., if $f\colon A \to B$ is a function, then $f(a)$ is defined for every $a \in A$. Functions from A to B are a special case of binary relations $r \subseteq A \times B$. As usual, we let $r(A') = \{b \in B \mid \exists a \in A'\colon (a, b) \in r\}$ for $A' \subseteq A$, and $r(a) = r(\{a\})$ for $a \in A$. Note that, in this way, r can be viewed as a "nondeterministic function" $r\colon A \to \mathcal{P}(B)$ where $\mathcal{P}(B)$ is the powerset of B.

A *signature* (or ranked alphabet) is a pair (Σ, rk), where Σ is a finite set of symbols, and rk assigns to every $\mathtt{f} \in \Sigma$ a *rank* $rk(\mathtt{f}) \in \mathbb{N}$. We will denote (Σ, rk) simply by Σ. If necessary, the rank k of a symbol $\mathtt{f}$ is indicated by writing $\mathtt{f}$ as $\mathtt{f}^{(k)}$.

The set of all *trees over* Σ is denoted by T_Σ. It is the smallest set of strings such that for all $k \in \mathbb{N}$, $\mathtt{f}^{(k)}$ in Σ, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$, the string $\mathtt{f}(t_1, \ldots, t_k)$ is in T_Σ (where the parentheses and the comma are assumed to be special symbols not in Σ). A tree of the form $\mathtt{f}(\,)$, where $\mathtt{f}$ has rank 0, is identified with the string $\mathtt{f}$ of length 1. A subset of T_Σ is a *tree language*.

As usual, a tree $t \in \mathrm{T}_\Sigma$ will be identified with a graph whose nodes are labelled with symbols in Σ. A node is a string in $(\mathbb{N} \setminus \{0\})^*$ which, intuitively, represents the Dewey path from the root to the node. Thus, λ is the root of t and vi is the i-th child of node v. Formally, we define the set $V(t)$ of *nodes* of t, the *subtree* t/v at a node v, and the *label* $\ell_t(v)$ of node v inductively, as follows. If $t = \mathtt{f}(t_1, \ldots, t_k)$, then $V(t) = \{\lambda\} \cup \{iv \mid i \in [k], v \in V(t_i)\}$; furthermore, $t/\lambda = t$, $\ell_t(\lambda) = \mathtt{f}$, and, for all $i \in [k]$ and $v \in V(t_i)$, $t/iv = t_i/v$ and $\ell_t(iv) = \ell_{t_i}(v)$. A node v of t is said to be an occurrence of the symbol $\ell_t(v)$.

As usual, to define the substitution of a tree s for a node v of a tree t, we use the set of *variables* $X = \{x_1, x_2, x_3, \ldots\}$. For $k \in \mathbb{N}$, $X_k = \{x_1, \ldots, x_k\}$ is a signature such that x_i has rank 0 for every $i \in [k]$. We assume X to be disjoint with all the usual signatures. For such a signature Σ, the set $\mathrm{T}_{\Sigma \cup X_k}$ is denoted by $\mathrm{T}_\Sigma(X_k)$; it is the set of *trees with k variables*.

For $t \in \mathrm{T}_\Sigma$, $v \in V(t)$ and $s \in \mathrm{T}_\Sigma(X_k)$, where $k = rk(\ell_t(v))$, the *substitution* of s for v in t, denoted $t[v \leftarrow s]$, is defined as follows:

- $t[\lambda \leftarrow s]$ is the result of substituting t/i for each occurrence of x_i in s;
- if $t = \mathtt{f}(t_1, \ldots, t_m)$, then $t[iv \leftarrow s] = \mathtt{f}(t_1, \ldots, t_{i-1}, t_i[v \leftarrow s], t_{i+1}, \ldots, t_m)$.

This notion of substitution leads to the definition of context-free tree grammars (see, e.g., [14, Section 15]).

Definition 1. *A* context-free tree grammar *(abbreviated CFTG) is a four-tuple* $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ *such that*

- Ξ *and* Σ *are disjoint signatures of* nonterminals *and* terminals*, respectively,*
- R *is a finite set of* rules *of the form* $\mathtt{g}(x_1, \ldots, x_k) \to s$*, where* $k \in \mathbb{N}$*,* $\mathtt{g}^{(k)} \in \Xi$ *and* $s \in \mathrm{T}_{\Xi \cup \Sigma}(X_k)$*, and*
- $\mathtt{g}_{\mathrm{in}} \in \Xi$ *is the* initial nonterminal*, of rank* 0.

For trees $t, t' \in \mathrm{T}_{\Xi \cup \Sigma}$*, there is an IO derivation step* $t \Rightarrow_{G,\mathrm{IO}} t'$ *if there are a node* $v \in V(t)$ *and a rule* $\mathtt{g}(x_1, \ldots, x_k) \to s$ *in* R *such that* $\ell_t(v) = \mathtt{g}$*,* $t/vi \in T_\Sigma$ *for every* $i \in [k]$*, and* $t' = t[v \leftarrow s]$*. The* tree language IO-generated by G *is* $L_{\mathrm{IO}}(G) = \{t \in \mathrm{T}_\Sigma \mid \mathtt{g}_{\mathrm{in}} \Rightarrow^*_{G,\mathrm{IO}} t\}$.

Example 1. We consider a very simple example of a CFTG $G_1 = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$. It has signatures $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$ and $\Xi = \{\mathtt{g}_{\mathrm{in}}^{(0)}, \mathtt{g}_1^{(2)}, \mathtt{g}_2^{(1)}, \mathtt{g}_3^{(2)}\}$, and R consists of the rules

$$\begin{array}{rcl} \mathtt{g}_{\mathrm{in}} & \to & \mathtt{g}_2(\mathtt{f}(\mathtt{a}, \mathtt{g}_1(\mathtt{a}, \mathtt{c}))), \\ \mathtt{g}_1(x_1, x_2) & \to & \mathtt{f}(x_1, x_1), \\ \mathtt{g}_2(x_1) & \to & \mathtt{g}_3(x_1, \mathtt{d}(x_1)), \text{ and} \\ \mathtt{g}_3(x_1, x_2) & \to & x_1. \end{array}$$

This grammar has exactly one derivation, viz., $\mathtt{g}_{\mathrm{in}} \Rightarrow_{G_1,\mathrm{IO}} \mathtt{g}_2(\mathtt{f}(\mathtt{a}, \mathtt{g}_1(\mathtt{a}, \mathtt{c}))) \Rightarrow_{G_1,\mathrm{IO}} \mathtt{g}_2(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))) \Rightarrow_{G_1,\mathrm{IO}} \mathtt{g}_3(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a})), \mathtt{d}(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a})))) \Rightarrow_{G_1,\mathrm{IO}} \mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))$, and so $L_{\mathrm{IO}}(G_1) = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$. □

3 Jungles and their Substitution

In this section, we recall some notions regarding jungles [1, 15, 16, 18], and present some elementary properties of jungles.

Jungles can either be defined as node-labeled graphs (see, e.g., [1, 2]) or, equivalently, as edge-labeled hypergraphs (see, e.g., [5, 15, 18]). Here we choose to define them as node-labeled graphs, which are technically more convenient for our purposes (and which are closer to trees).

3.1 Jungles

Intuitively, a jungle is a directed ordered acyclic graph representing a tree. In such a jungle, subtrees can be shared and unreachable subtrees, so-called garbage, may occur.

Let Σ be a signature. A *directed ordered graph* (abbreviated DOG) over Σ is a triple $G = (V, lab, arg)$ consisting of a finite set V of *nodes*, a *labelling function* $lab\colon V \to \Sigma$, and an *argument function* $arg\colon V \to V^*$ such that $|arg(v)| = rk(lab(v))$ for every $v \in V$.

We define the *rank* of a node v as the rank of its label, i.e., $rk(v) = rk(lab(v))$. The elements of the sequence $arg(v)$ are called the *arguments* of v. In particular, the i-th element of the sequence will be denoted $arg(v, i)$, and is called the i-th argument of v. The DOG G can be visualized as an ordinary directed graph (V, E) with labelled nodes and edges, where the set of edges is $E = \{(v, arg(v, i)) \mid v \in V, i \in [rk(v)]\}$ and the label of the edge $(v, arg(v, i))$ is the natural number i. Accordingly, for nodes v and w of G, a (directed) *path from v to w* is a sequence $v_1 \cdots v_n \in V^*$ such that $n \geq 1$, $v = v_1$, $w = v_n$ and v_{j+1} is an argument of v_j for every $j \in [n-1]$. The DOG G is *acyclic* (in short, a DOAG) if, for every $v \in V$, the only path from v to v is v. A *topological order* of a DOG G is a linear order $<$ on its set V of nodes such that $arg(v, i) < v$ for every $v \in V$ and $i \in [rk(v)]$. It is well known (and easy to see) that a DOG is a DOAG if and only if it has a topological order.

A *jungle* over Σ is a DOAG with a designated node, i.e., it is a four-tuple $J = (V, res, lab, arg)$ where (V, lab, arg) is a DOAG over Σ, and $res \in V$ is the *result node* of J. The set of jungles over Σ is denoted J_Σ. A subset of J_Σ is a *jungle language*. For $k \in \mathbb{N}$, we denote $\mathrm{J}_{\Sigma \cup X_k}$ by $\mathrm{J}_\Sigma(X_k)$; it is the set of *jungles with k variables*. Note that $\mathrm{J}_\Sigma(X_0) = \mathrm{J}_\Sigma$. If necessary, the components of a jungle J will be denoted by V_J, res_J, lab_J, arg_J, respectively, and similarly for derived notions such as E_J, rk_J, etc. Two jungles J and K are *disjoint* if $V_J \cap V_K = \emptyset$. As usual, we do not distinguish between isomorphic jungles, i.e., jungles that are identical up to a bijective renaming of their nodes.

Figure 1 shows three example jungles: $K, K'' \in \mathrm{J}_\Sigma$ and $K' \in \mathrm{J}_\Sigma(X_1)$ where Σ is the signature $\{\mathtt{f}^{(2)}, \mathtt{h}^{(1)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$. All edges are assumed to be directed downwards. Outgoing edges of the same node are assumed to be ordered from left to right. Result nodes are indicated by dashed circles. Thus, $K'' = (V, res, lab, arg)$ with, e.g., $V = \{d, f_1, a_1, f_2, a_2, c\}$ and $res = f_1$, $lab(d) = \mathtt{d}$, $lab(f_1) = lab(f_2) = \mathtt{f}$, $lab(a_1) = lab(a_2) = \mathtt{a}$, $lab(c) = \mathtt{c}$, $arg(d) = f_1$, $arg(f_1) = a_1 f_2$, $arg(f_2) = a_2 a_2$, and $arg(c) = \lambda$. A topological order of K'' is $c < a_2 < f_2 < a_1 < f_1 < d$.

Since trees over Σ are identified with graphs in the usual way, we will view T_Σ as a subset of J_Σ. To be precise, every tree $t \in \mathrm{T}_\Sigma$ will be identified with the jungle (V, res, lab, arg) where $V = V(t)$, $res = \lambda$, and for every $v \in V$, $lab(v) = \ell_t(v)$ and $arg(v, i) = vi$ for $i \in [rk(\ell_t(v))]$. In this way, $\mathrm{T}_\Sigma \subseteq \mathrm{J}_\Sigma$ and $\mathrm{T}_\Sigma(X_k) \subseteq \mathrm{J}_\Sigma(X_k)$ for every $k \in \mathbb{N}$.

Jungles generalize trees by allowing nodes (and hence whole subtrees) to be shared. A node w of a jungle J is shared if there are distinct pairs $(v, j), (v', j')$

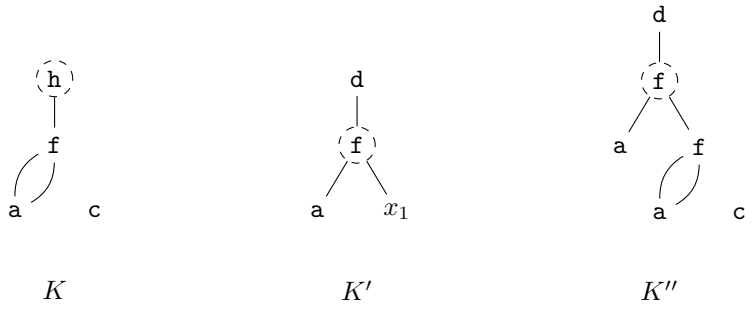

Figure 1: Jungles K, K' and K''. Using jungle substitution as defined later, $K'' = K[v \leftarrow K']$, where v is the node of K with label `h`.

such that $arg_J(v, j) = w = arg_J(v', j')$. Moreover, jungles contain garbage nodes, i.e., nodes w for which there is no path in J from res_J to w. When jungles are interpreted in a nondeterministic algebra (as we will do in Section 5), a node may have no value, one value, or several possible values. Then shared nodes and garbage nodes force a call by value semantics: a node must be evaluated, even if it will not be used (i.e., is garbage); and when it is used several times (i.e., is shared), the same value must be taken each time. In Figure 1, the node with label `a` of K is shared (and so is the node a_2 of K''), and the nodes with label `c` and `d` are garbage.

3.2 Jungle Substitution

We first show how to contract certain edges of jungles. Let `id` be a symbol of rank 1 that does not occur in Σ; intuitively, it stands for the identity function. For a jungle $J \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$, $n \in \mathbb{N}$, and a set W of nodes of J with label `id` we define $ctr_W(J) \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$ to be the result of contracting all edges $(v, arg(v, 1))$ of J with $v \in W$; the node that results from the identification of v and $arg(v, 1)$ receives the label of $arg(v, 1)$. Formally, for $J = (V, res, lab, arg)$ and $W \subseteq \{v \in V \mid lab(v) = \mathtt{id}\}$, we define the function $\gamma : V \to V \setminus W$ such that $\gamma(v) = v$ if $v \in V \setminus W$, and $\gamma(v) = \gamma(v')$ if $v \in W$ and v' is the (unique) argument of v; note that γ is well defined because J is acyclic. Then $ctr_W(J) = (V \setminus W, res', lab', arg')$ where $res' = \gamma(res)$, lab' is the restriction of lab to $V \setminus W$, and for $v \in V \setminus W$, if $arg(v) = v_1 \cdots v_k$ with $v_i \in V$ then $arg'(v) = \gamma(v_1) \cdots \gamma(v_k)$. Note that $ctr_W(J)$ is indeed acyclic: the restriction of a topological order of J to $V \setminus W$ is a topological order of $ctr_W(J)$. In particular, we define the *contraction* $ctr(J) \in \mathrm{J}_\Sigma(X_n)$ of J by $ctr(J) = ctr_W(J)$ where $W = \{v \in V \mid lab(v) = \mathtt{id}\}$. Thus, $V_{ctr(J)}$ consists of all nodes of J that do not have label `id`; the above function $\gamma\colon V_J \to V_{ctr(J)}$ is called the *track* function of J.

For a jungle J and a node v of rank k, we now show how to substitute a jungle K with k variables for that node v in J. Intuitively, the node v is replaced by the

result node of K, and every node of K with label x_i is replaced by the i-th argument of v. Note that in the special case where the result node of K has label x_i, the node v is replaced by its i-th argument.

Formally, let $J \in \mathrm{J}_\Sigma(X_n)$, $v \in V_J$ with $rk_J(v) = k$, and $K \in \mathrm{J}_\Sigma(X_k)$. We assume that J and K are disjoint, otherwise we consider a disjoint isomorphic copy of K. We first define the jungle $J\langle v \leftarrow K\rangle \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$ to be the union of J and K, with result node res_J, and with the following changes: $lab(v)$ is changed into $\mathtt{id}$ and $arg(v)$ into res_K, and for every $w \in V_K$ and $i \in [k]$, if $lab(w) = x_i$, then $lab(w)$ is changed into $\mathtt{id}$ and $arg(w)$ into $arg(v,i)$.[2] It should be clear that $U = J\langle v \leftarrow K\rangle$ is acyclic: if $<_J$ and $<_K$ are topological orders for J and K, respectively, then a topological order for U is obtained by inserting $<_K$ just before v in $<_J$, i.e., $<_U$ is the union of $<_J$, $<_K$, $\{(v', w) \mid v' \in V_J,\, v' < v,\, w \in V_K\}$ and $\{(w, v') \mid w \in V_K,\, v' \in V_J,\, v \leq v'\}$.

Finally, we define $J[v \leftarrow K] \in \mathrm{J}_\Sigma(X_n)$ to be the jungle $ctr(J\langle v \leftarrow K\rangle)$. It is called the *substitution* of K for v in J. Note that $V_{J[v\leftarrow K]}$ is the union of $V_J \setminus \{v\}$ and $V_K \setminus \{w \in V_K \mid lab_K(w) \in X\}$. Note also that the track function γ of $J\langle v \leftarrow K\rangle$ is the identity on $V_{J[v\leftarrow K]}$; moreover, $\gamma(v) = res_K$ if $lab_K(res_K) \notin X$, $\gamma(v) = \arg_J(v,i)$ if $lab_K(res_K) = x_i$, and $\gamma(w) = \arg_J(v,i)$ for every $w \in V_K$ with $lab_K(w) = x_i$. A very simple example of substitution is shown in Figure 1.

In the next section we will need the fact that jungle substitution is confluent and associative, as defined in [3]. These are natural properties that are satisfied by many notions of substitution that are used in context-free grammars for several types of structures, as shown in [3].[3] We start with a simple lemma.

Lemma 1. *For a jungle $J \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$, a node $v \in V_J$ of rank k with $lab_J(v) \neq \mathtt{id}$, and a jungle $K \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_k)$,*

$$ctr(J)[v \leftarrow ctr(K)] = ctr(J\langle v \leftarrow K\rangle).$$

Proof. The straightforward proofs of the following two equalities are left to the reader. Let W, W_1, W_2 be sets of nodes with label $\mathtt{id}$, such that $W, W_1 \subseteq V_J$ and $W_2 \subseteq V_K$.

(i) $ctr(ctr_W(J)) = ctr(J)$

(ii) $ctr_{W_1}(J)\langle v \leftarrow ctr_{W_2}(K)\rangle = ctr_{W_1\cup W_2}(J\langle v \leftarrow K\rangle)$

By (ii), $ctr(J)[v \leftarrow ctr(K)] = ctr(ctr_{W_1\cup W_2}(J\langle v \leftarrow K\rangle))$ where $W_1 = \{w \in V_J \mid lab_J(w) = \mathtt{id}\}$ and $W_2 = \{w \in V_K \mid lab_K(w) = \mathtt{id}\}$. This equals $ctr(J\langle v \leftarrow K\rangle)$ by (i), applied to $J\langle v \leftarrow K\rangle$. □

[2] To be completely formal, $U = J\langle v \leftarrow K\rangle$ is defined as follows: $V_U = V_J \cup V_K$, $res_U = res_J$,

- $lab_U(u) = lab_J(u)$ and $arg_U(u) = arg_J(u)$ if $u \in V_J$ and $u \neq v$,
- $lab_U(v) = \mathtt{id}$ and $arg_U(v) = res_K$,
- $lab_U(u) = lab_K(u)$ and $arg_U(u) = arg_K(u)$ if $u \in V_K$ and $lab_K(u) \notin X_k$, and
- $lab_U(u) = \mathtt{id}$ and $arg_U(u) = arg_J(v,i)$ if $u \in V_K$ and $lab_K(u) = x_i$, for every $i \in [k]$.

[3] When jungles are defined as hypergraphs, jungle substitution is modeled by hyperedge replacement (see [5, Section 4]). It is well known that the corresponding notion of hypergraph substitution is confluent and associative (see, e.g., [6, Section 2.2.2]).

In the next two lemmas we show that jungle substitution is confluent and associative, respectively.

Lemma 2. *For jungles* $J \in \mathrm{J}_\Sigma(X_n)$, $K_1 \in \mathrm{J}_\Sigma(X_{k_1})$, $K_2 \in \mathrm{J}_\Sigma(X_{k_2})$ *and distinct nodes* $v_1, v_2 \in V_J$ *of rank* k_1, k_2, *respectively,*

$$J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = J[v_2 \leftarrow K_2][v_1 \leftarrow K_1].$$

Proof. By Lemma 1, $J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = ctr(J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle)$. It is obvious that $J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle = J\langle v_2 \leftarrow K_2\rangle\langle v_1 \leftarrow K_1\rangle$. □

Lemma 3. *For jungles* $J \in \mathrm{J}_\Sigma(X_n)$, $K_1 \in \mathrm{J}_\Sigma(X_{k_1})$, $K_2 \in \mathrm{J}_\Sigma(X_{k_2})$ *and nodes* $v_1 \in V_J$ *of rank* k_1 *and* $v_2 \in V_{K_1}$ *of rank* k_2 *with* $lab_{K_1}(v_2) \notin X_{k_1}$,

$$J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = J[v_1 \leftarrow K_1[v_2 \leftarrow K_2]].$$

Proof. The proof is similar to the previous one. Lemma 1 implies that both $J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = ctr(J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle)$ and $J[v_1 \leftarrow K_1[v_2 \leftarrow K_2]] = ctr(J\langle v_1 \leftarrow K_1\langle v_2 \leftarrow K_2\rangle\rangle)$. And it is obvious that $J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle = J\langle v_1 \leftarrow K_1\langle v_2 \leftarrow K_2\rangle\rangle$. □

4 Context-free Jungle Grammars

Having defined jungles and their substitution, we now define the notion of a context-free jungle grammar in an obvious way, see [5, Definition 7.6].

Definition 2. *A* context-free jungle grammar *(abbreviated CFJG) is a four-tuple* $G = (\Xi, \Sigma, R, \mathsf{g}_{\mathrm{in}})$ *such that*

- Ξ *and* Σ *are disjoint signatures of* nonterminals *and* terminals, *respectively,*
- R *is a finite set of* rules *of the form* $\mathsf{g}(x_1, \ldots, x_k) \to K$, *where* $k \in \mathbb{N}$, $\mathsf{g}^{(k)} \in \Xi$ *and* $K \in \mathrm{J}_{\Xi\cup\Sigma}(X_k)$, *and*
- $\mathsf{g}_{\mathrm{in}} \in \Xi$ *is the* initial nonterminal, *of rank* 0.

For jungles $J, J' \in \mathrm{J}_{\Xi\cup\Sigma}$, *there is a derivation step* $J \Rightarrow_G J'$ *if there are a node* $v \in V_J$ *and a rule* $\mathsf{g}(x_1, \ldots, x_k) \to K$ *in* R *such that* $lab_J(v) = \mathsf{g}$ *and* $J' = J[v \leftarrow K]$. *The* jungle language generated by G *is* $L_{\mathrm{J}}(G) = \{J \in \mathrm{J}_\Sigma \mid \mathsf{g}_{\mathrm{in}} \Rightarrow_G^* J\}$.[4]

Since every tree is a jungle, every context-free tree grammar G is also a context-free jungle grammar, generating not only the tree language $L_{\mathrm{IO}}(G)$ but also the jungle language $L_{\mathrm{J}}(G)$.

[4] Note that g_{in} is a (one-node) jungle, because g_{in} has rank 0 and $T_{\Xi\cup\Sigma} \subseteq \mathrm{J}_{\Xi\cup\Sigma}$.

Example 2. We consider a very simple example of a CFJG $G_2 = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$. It has the same terminal signature $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$ as the CFTG G_1 of Example 1. The nonterminal signature is $\Xi = \{\mathtt{g}^{(0)}, \mathtt{h}^{(1)}\}$ with $\mathtt{g}_{\mathrm{in}} = \mathtt{g}$, and the set R consists of the two rules $\mathtt{g} \to K$ and $\mathtt{h}(x_1) \to K'$, where K and K' are given in Figure 1. The unique derivation of this grammar is $\mathtt{g} \Rightarrow_G K \Rightarrow_G K[v \leftarrow K']$, where v is the node of K with label $\mathtt{h}$. Thus $L_{\mathrm{J}}(G_2) = \{K''\}$, where K'' is given in Figure 1.

As another simple example we consider the context-free jungle grammar G_1 of Example 1. A derivation of G_1 is shown in Figure 2; it generates the jungle K''. The other two derivations of G_1 also generate the jungle K'' (in accordance with Lemma 2). Thus, $L_{\mathrm{J}}(G_1) = L_{\mathrm{J}}(G_2) = \{K''\}$.

An interpretation of grammars G_2 and G_1 will be given in Examples 5 and 6. □

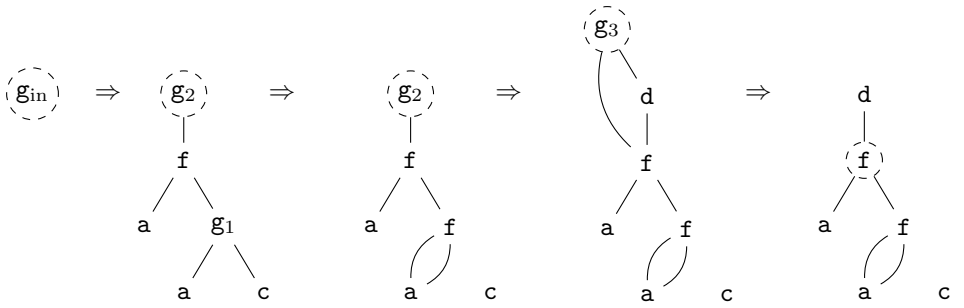

Figure 2: A derivation of G_1.

Our next aim is to show that rules of a CFJG can be substituted into each other, without changing the generated jungle language. More precisely, consider a rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ and let $v \in V_K$ be a node of rank m with nonterminal label $\mathtt{h}$. Then this rule can be replaced by all rules

$$\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$$

where K' is the right-hand side of a rule with left-hand side $\mathtt{h}(x_1, \ldots, x_m)$. This clearly holds for the CFJG G_2 of Example 2: the resulting grammar has rules $\mathtt{g} \to K''$ and $\mathtt{h}(x_1) \to K'$, and thus generates the same jungle language $\{K''\}$; note that the second rule has become superfluous.

The above property is well known for context-free string grammars, and for several types of context-free graph grammars. Its proof is based on the fact that jungle substitution is confluent and associative, as shown in the previous section.

Lemma 4. *Let $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ be a CFJG. Let $K \in \mathrm{J}_{\Xi \cup \Sigma}$ and let $v \in V_K$ be such that $lab_K(v) = \mathtt{h}^{(m)} \in \Xi$. Then, for every $J \in \mathrm{J}_\Sigma$ and $n \in \mathbb{N}$, $K \Rightarrow_G^n J$ if and only if there exists a rule $\mathtt{h}(x_1, \ldots, x_m) \to K'$ in R such that $K[v \leftarrow K'] \Rightarrow_G^{n-1} J$.*

Proof. The if direction is obvious, because $K \Rightarrow_G K[v \leftarrow K']$. The only-if direction is proved by induction on n. Let $K \Rightarrow_G K[w \leftarrow L] \Rightarrow_G^{n-1} J$ be the first step of the derivation. If $w = v$ then there is a rule $\mathtt{h}(x_1, \ldots, x_m) \to L$ in R, and we are ready. Now assume that $w \neq v$. By the induction hypothesis, there is a rule $\mathtt{h}(x_1, \ldots, x_m) \to K'$ in R such that $K[w \leftarrow L][v \leftarrow K'] \Rightarrow_G^{n-2} J$. Hence $K[v \leftarrow K'][w \leftarrow L] \Rightarrow_G^{n-2} J$ by Lemma 2. This implies that

$$K[v \leftarrow K'] \Rightarrow_G K[v \leftarrow K'][w \leftarrow L] \Rightarrow_G^{n-2} J,$$

and so $K[v \leftarrow K'] \Rightarrow_G^{n-1} J$. □

Theorem 1. *Let $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ be a CFJG. Let $\mathtt{g}(x_1, \ldots, x_k) \to K$ be a rule in R and let $v \in V_K$ be such that $lab_K(v) = \mathtt{h}^{(m)} \in \Xi$. Let G' be the CFJG $(\Xi, \Sigma, R', \mathtt{g}_{\mathrm{in}})$ where R' is obtained from R by replacing the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by all rules $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ where $\mathtt{h}(x_1, \ldots, x_m) \to K'$ is in R. Then $L_{\mathrm{J}}(G') = L_{\mathrm{J}}(G)$.*

Proof. We prove by induction on the length of the derivations that for all $I \in \mathrm{J}_{\Xi \cup \Sigma}$ and $J \in \mathrm{J}_\Sigma$, $I \Rightarrow_G^* J$ if and only if $I \Rightarrow_{G'}^* J$.

For the only-if direction, we consider the first step of the derivation $I \Rightarrow_G^* J$. It clearly suffices to consider the case that the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ is applied in this step. Thus, let $I \Rightarrow_G I[w \leftarrow K] \Rightarrow_G^* J$, where $lab_I(w) = \mathtt{g}$. By Lemma 4 there is a rule $\mathtt{h}(x_1, \ldots, x_m) \to K'$ in R such that $I[w \leftarrow K][v \leftarrow K'] \Rightarrow_G^* J$, and this derivation is shorter than the derivation $I \Rightarrow_G^* J$. Hence, by the induction hypothesis, $I[w \leftarrow K][v \leftarrow K'] \Rightarrow_{G'}^* J$. Now, by Lemma 3, we have $I[w \leftarrow K][v \leftarrow K'] = I[w \leftarrow K[v \leftarrow K']]$, and so $I \Rightarrow_{G'} I[w \leftarrow K[v \leftarrow K']] \Rightarrow_{G'}^* J$, where the rule $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ of G' is applied in the first step.

The if direction is similar, but slightly easier. For the first step of the derivation $I \Rightarrow_{G'}^* J$ it suffices to consider a rule $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ in R'. If $I \Rightarrow_{G'} I[w \leftarrow K[v \leftarrow K']] \Rightarrow_{G'}^* J$, then $I[w \leftarrow K][v \leftarrow K'] \Rightarrow_G^* J$ by Lemma 3 and the induction hypothesis, and so $I \Rightarrow_G I[w \leftarrow K] \Rightarrow_G^* J$ by Lemma 4. □

Before proving our main result, we discuss an easy normal form of context-free jungle grammars. A CFJG $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ is in *variable normal form* if, for every rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ in R and every $i \in [k]$, exactly one node of K has label x_i. It is easy to see that for every CFJG G an equivalent CFJG G' can be constructed that is in variable normal form, as follows. If $\mathtt{g}(x_1, \ldots, x_k) \to K$ is a rule of G, then $\mathtt{g}(x_1, \ldots, x_k) \to K'$ is a rule of G', where K' is obtained from K by identifying all nodes with label x_i, for each $i \in [k]$, and adding an isolated node with label x_i if K does not have such a node. To be precise, $K' = \mathtt{g}(x_1, \ldots, x_k)[\lambda \leftarrow K]$, i.e., the substitution of K for the node with label $\mathtt{g}$ in the jungle $\mathtt{g}(x_1, \ldots, x_k)$. It follows from Lemma 3 (and is also easy to see) that $J[v \leftarrow K'] = J[v \leftarrow K]$ for every jungle J and every node $v \in V_J$ with label $\mathtt{g}$, and hence $L_{\mathrm{J}}(G') = L_{\mathrm{J}}(G)$.

The equivalence of G' and G can also be proved by Theorem 1, as follows. Let G_1 be the CFJG obtained from G by replacing every rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by the rules $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$ and $\mathtt{g}_0(x_1, \ldots, x_k) \to K$, where $\mathtt{g}_0$ is a

new nonterminal. Obviously, $L_{\mathrm{J}}(G_1) = L_{\mathrm{J}}(G)$. Application of Theorem 1 to each rule $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$, changes G_1 into the equivalent grammar G_1' in which that rule is changed into all rules $\mathtt{g}(x_1, \ldots, x_k) \to K'$. Since the rules $\mathtt{g}_0(x_1, \ldots, x_k) \to K$ have become useless in G_1', we obtain that $L_{\mathrm{J}}(G_1') = L_{\mathrm{J}}(G')$ and hence $L_{\mathrm{J}}(G) = L_{\mathrm{J}}(G')$.

We now show the main result of this paper: context-free tree grammars have the same jungle generating power as context-free jungle grammars.

Theorem 2. *For every context-free jungle grammar G there is a context-free tree grammar H such that $L_{\mathrm{J}}(H) = L_{\mathrm{J}}(G)$.*

Proof. Let $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ be a CFJG in variable normal form, which can be assumed by the discussion above. Consider a rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ in R. We will construct an equivalent CFJG G' in which this rule is replaced by a finite set of CFTG rules. By repeating this construction we obtain the required CFTG H. In the CFJG $G' = (\Xi', \Sigma, R', \mathtt{g}_{\mathrm{in}})$, the rule will be simulated by a sequence of rules that build the jungle K node by node, in a bottom-up fashion. This is similar to, but slightly more complicated than, the construction of Chomsky normal form for context-free string grammars.

Let $v_1 < \cdots < v_k < v_{k+1} < \cdots < v_{k+\ell}$, $\ell \geq 0$, be a topological order of K such that $v_1, \ldots, v_k$ are the (unique) nodes of K with labels $x_1, \ldots, x_k$, respectively. Obviously such a topological order exists, because the nodes $v_1, \ldots, v_k$ have no arguments. We define $\Xi' = \Xi \cup \{\mathtt{g}_0, \mathtt{g}_1, \ldots, \mathtt{g}_\ell\}$ where $\mathtt{g}_i$ is a new nonterminal of rank $k + i$, for $0 \leq i \leq \ell$. Moreover, we define R' to be the set of rules obtained from R by replacing the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by the CFTG rules

- $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$,
- $\mathtt{g}_{i-1}(x_1, \ldots, x_{k+i-1}) \to \mathtt{g}_i(x_1, \ldots, x_{k+i-1}, \mathtt{f}_i(x_{j_1}, \ldots, x_{j_p}))$ for $i \in [\ell]$, where $\mathtt{f}_i^{(p)} = \mathit{lab}_K(v_{k+i})$ and $\mathit{arg}_K(v_{k+i}, q) = v_{j_q}$ for every $q \in [p]$, and
- $\mathtt{g}_\ell(x_1, \ldots, x_{k+\ell}) \to x_j$, where $\mathit{res}_K = v_j$.

For $i \in [\ell + 1]$, let t_i be the right-hand side of the rule with left-hand side $\mathtt{g}_{i-1}(x_1, \ldots, x_{k+i-1})$. Note that in the second item $j_q \in [k + i - 1]$, because $<$ is a topological order. Thus, $t_i \in \mathrm{T}_{\Xi \cup \Sigma}(X_{k+i-1})$ as required.

To prove the correctness of this construction, we take new nodes $r_0, \ldots, r_\ell$, we assume that the nodes in the right-hand side $\mathtt{g}_0(x_1, \ldots, x_k)$ of the first rule are $r_0, v_1, \ldots, v_k$ with respective labels $\mathtt{g}_0, x_1, \ldots, x_k$, and we assume for $i \in [\ell]$, that the nodes in t_i with labels $\mathtt{g}_i$ and $\mathtt{f}_i$ are r_i and v_{k+i}, respectively. For $0 \leq i \leq \ell$, we define the jungle $K_i = (V, \mathit{res}, \mathit{lab}, \mathit{arg}) \in \mathrm{J}_{\Xi' \cup \Sigma}(X_k)$ as follows: $V = \{r_i, v_1, \ldots, v_{k+i}\}$, $\mathit{res} = r_i$, $\mathit{lab}(r_i) = \mathtt{g}_i$, $\mathit{arg}(r_i) = v_1 \cdots v_{k+i}$, and for $j \in [k + i]$, $\mathit{lab}(v_j) = \mathit{lab}_K(v_j)$ and $\mathit{arg}(v_j) = \mathit{arg}_K(v_j)$. Moreover, we define $K_{\ell+1} = K$.

It can easily be checked that $K_0 = \mathtt{g}_0(x_1, \ldots, x_k)$ and $K_i = K_{i-1}[r_{i-1} \leftarrow t_i]$ for every $i \in [\ell + 1]$. Thus, by iterated application of Theorem 1 (i.e., formally by induction on i), the grammar G' is equivalent to the grammar G_i' that is obtained from G' by changing the rule $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$ into the rule

$\mathtt{g}(x_1, \ldots, x_k) \to K_i$. Hence, since $K_{\ell+1} = K$, the grammar $G'_{\ell+1}$ has all rules of G plus all rules $\mathtt{g}_i(x_1, \ldots, x_{k+i}) \to t_{i+1}$ for $0 \leq i \leq \ell$. However, since $\mathtt{g}_0$ does not appear in any right-hand side of a rule of $G'_{\ell+1}$, the rules $\mathtt{g}_i(x_1, \ldots, x_{k+i}) \to t_{i+1}$ have become useless in $G'_{\ell+1}$, and hence $G'_{\ell+1}$ is equivalent to G.

This shows that $L_{\mathrm{J}}(G') = L_{\mathrm{J}}(G)$. □

Example 3. We illustrate the construction in the proof of Theorem 2 with the CFJG G_2 of Example 2. It has initial nonterminal $\mathtt{g}$ and it has the two rules $\mathtt{g} \to K$ and $\mathtt{h}(x_1) \to K'$, where K and K' are given in Figure 1. The resulting CFTG H has the same initial nonterminal $\mathtt{g}$. Its rules are constructed based on the following topological orders of K and K' (where we indicate nodes by their labels): $\mathtt{a} < \mathtt{c} < \mathtt{f} < \mathtt{h}$ for K, and $x_1 < \mathtt{a} < \mathtt{f} < \mathtt{d}$ for K'. This gives the following rules:

$$\begin{aligned}
\mathtt{g} &\to \mathtt{g}_0 \\
\mathtt{g}_0 &\to \mathtt{g}_1(\mathtt{a}) \\
\mathtt{g}_1(x_1) &\to \mathtt{g}_2(x_1, \mathtt{c}) \\
\mathtt{g}_2(x_1, x_2) &\to \mathtt{g}_3(x_1, x_2, \mathtt{f}(x_1, x_1)) \\
\mathtt{g}_3(x_1, x_2, x_3) &\to \mathtt{g}_4(x_1, x_2, x_3, \mathtt{h}(x_3)) \\
\mathtt{g}_4(x_1, x_2, x_3, x_4) &\to x_4 \\
\mathtt{h}(x_1) &\to \mathtt{h}_0(x_1) \\
\mathtt{h}_0(x_1) &\to \mathtt{h}_1(x_1, \mathtt{a}) \\
\mathtt{h}_1(x_1, x_2) &\to \mathtt{h}_2(x_1, x_2, \mathtt{f}(x_2, x_1)) \\
\mathtt{h}_2(x_1, x_2, x_3) &\to \mathtt{h}_3(x_1, x_2, x_3, \mathtt{d}(x_3)) \\
\mathtt{h}_3(x_1, x_2, x_3, x_4) &\to x_3
\end{aligned}$$

Of course, this is not the simplest CFTG that generates the same jungle language as G_2. A simpler one is the grammar G_1 of Example 1, as we saw in Example 2. □

Thus, context-free tree grammars have the same jungle generating power as context-free jungle grammars. Vice versa, every jungle represents a tree (by unfolding) and we will show that context-free jungle grammars have the same tree generating power as context-free tree grammars.

Every jungle $J \in \mathrm{J}_\Sigma$ represents a unique tree $\mathrm{tree}(J) \in \mathrm{T}_\Sigma$, namely $\mathrm{tree}(J) = \mathrm{tree}(J, \mathit{res}_J)$, where $\mathrm{tree}(J, v)$ is defined as follows, for all $v \in V_J$: if $\mathit{lab}_J(v) = \mathtt{f}^{(k)}$ and $\mathit{arg}_J(v) = v_1 \cdots v_k$, then $\mathrm{tree}(J, v) = \mathtt{f}(\mathrm{tree}(J, v_1), \ldots, \mathrm{tree}(J, v_k))$. For example, $\mathrm{tree}(K'') = \mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))$ where K'' is given in Figure 1.

For a context-free jungle grammar G we define the *tree language generated by* G as $L_{\mathrm{T}}(G) = \{\mathrm{tree}(J) \mid J \in L_{\mathrm{J}}(G)\}$.

Theorem 3. *A tree language can be generated by a context-free jungle grammar if and only if it can be IO-generated by a context-free tree grammar.*

Proof. (If) It is shown in [5, Corollary 6.5] that $L_{\mathrm{T}}(G) = L_{\mathrm{IO}}(G)$ for every context-free tree grammar G.[5]

[5] As an example, $L_{\mathrm{IO}}(G_1) = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$ for the context-free tree grammar G_1 of Example 1, and, by Example 2, $L_{\mathrm{J}}(G_1) = \{K''\}$ and so $L_{\mathrm{T}}(G_1) = \{\mathrm{tree}(K'')\} = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$.

(Only if) For a context-free jungle grammar G, let H be a context-free tree grammar such that $L_{\mathrm{J}}(H) = L_{\mathrm{J}}(G)$, which exists by Theorem 2. By the previous paragraph, $L_{\mathrm{IO}}(H) = L_{\mathrm{T}}(H)$ and so $L_{\mathrm{IO}}(H) = \{\mathrm{tree}(J) \mid J \in L_{\mathrm{J}}(H)\} = \{\mathrm{tree}(J) \mid J \in L_{\mathrm{J}}(G)\} = L_{\mathrm{T}}(G)$. □

Related results are proved in [7, 9, 12, 10]. It is shown in [7] (see also [8]) that IO context-free tree grammars generate the same tree languages as attribute grammars with one synthesized attribute. As shown in [9], arbitrary attribute grammars generate the same tree languages as jungle generating context-free graph grammars (which generalize context-free jungle grammars). In [12] it is proved that total deterministic macro tree transducers compute the same tree translations as top-down tree-to-jungle transducers. Finally, in [10] context-free tree grammars are considered such that for every rule $\mathtt{g}(x_1, \ldots, x_k) \to s$, each parameter x_i occurs exactly once in s; it should be clear that $L_{\mathrm{IO}}(G) = L_{\mathrm{J}}(G) = L_{\mathrm{T}}(G)$ for such a grammar G (cf. [5, Theorem 6.7]).

5 Jungle Delegation Networks

In this section we interpret context-free jungle grammars in algebras and call them jungle delegation networks.

5.1 Interpretations and Algebras

We wish to interpret every symbol of a signature Σ as an operation on the elements of a set $\mathbb{A}$, and to use this interpretation for evaluating jungles over Σ. Usually, the interpretation of a symbol $\mathtt{f}^{(k)}$ would be a k-ary function $f\colon \mathbb{A}^k \to \mathbb{A}$. However, we wish to consider the more general case of nondeterministic operations (as in [13, Section II.2]; see also [5, 11]). For this purpose, symbols are interpreted as relations $f \subseteq \mathbb{A}^k \times \mathbb{A}$ rather than as functions. Of course, functions and partial functions are special cases.

A Σ-*interpretation into* $\mathbb{A}$ is a function σ that maps every symbol $\mathtt{f}^{(k)} \in \Sigma$ to a relation $\sigma(\mathtt{f}) \subseteq \mathbb{A}^k \times \mathbb{A}$; in particular, if $k = 0$ then $\sigma(\mathtt{f}) \subseteq \mathbb{A}$. The pair $(\mathbb{A}, \sigma)$ is called a (*nondeterministic*) Σ-*algebra*. If $\sigma(\mathtt{f})$ is a function for all $\mathtt{f} \in \Sigma$, then $(\mathbb{A}, \sigma)$ is a *deterministic* Σ-*algebra*.

Jungles with n variables can now be interpreted as derived operations of a given Σ-algebra $(\mathbb{A}, \sigma)$, in an obvious way (see [5, Definition 5.2]).

Definition 3 (jungle evaluation)**.** *Consider a* Σ*-algebra* $(\mathbb{A}, \sigma)$ *and a jungle* $J \in \mathrm{J}_\Sigma(X_n)$*. Given* $a_1, \ldots, a_n \in \mathbb{A}$*, let* $ASS_{J,\sigma}(a_1, \ldots, a_n)$ *be the set of all assignments (i.e., functions)* $\alpha\colon V_J \to \mathbb{A}$ *such that every* $v \in V_J$*:*

- *if* $lab_J(v) = x_i$*, then* $\alpha(v) = a_i$*; and*
- *if* $lab_J(v) = \mathtt{f} \in \Sigma$ *and* $arg_J(v) = v_1 \cdots v_k$*, then* $\alpha(v) \in f(\alpha(v_1), \ldots, \alpha(v_k))$ *where* $f = \sigma(\mathtt{f})$*.*

Now, $\sigma(J) \subseteq \mathbb{A}^n \times \mathbb{A}$ is the relation given by

$$\sigma(J)(a_1, \ldots, a_n) = \{\alpha(res_J) \mid \alpha \in ASS_{J,\sigma}(a_1, \ldots, a_n)\},$$

for all $a_1 \ldots, a_n \in \mathbb{A}$. For a set of jungles $\mathcal{J} \subseteq \mathrm{J}_\Sigma(X_n)$, $\sigma(\mathcal{J}) = \bigcup_{J \in \mathcal{J}} \sigma(J)$.

Since every tree is a jungle, this also defines $\sigma(t)$ for every tree $t \in \mathrm{T}_\Sigma(X_n)$. It should be clear that $\sigma(t)$ is the usual evaluation of t in a (nondeterministic) Σ-algebra, see, e.g., [5, Lemma 5.3]. For a set of trees $\mathcal{T} \subseteq \mathrm{T}_\Sigma(X_n)$, $\sigma(\mathcal{T})$ is called the derived relation of $\mathcal{T}$ over $(\mathbb{A}, \sigma)$ in [11, Definition 5.8].

Example 4. We extend [5, Example 5.1]. Let $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$ be the signature of Examples 1 and 2, and consider the Σ-algebra $(\mathbb{A}, \sigma)$ where $\mathbb{A} = \{\diamondsuit, \heartsuit\}^*$, $\sigma(\mathtt{f})$ is string concatenation, $\sigma(\mathtt{a}) = \{\diamondsuit, \heartsuit\}$, $\sigma(\mathtt{c}) = \{\diamondsuit\}$, and $\sigma(\mathtt{d})$ is the partial function $d : \mathbb{A} \to \mathbb{A}$ such that, for every string $w \in \mathbb{A}$, $d(\diamondsuit w) = w$ and $d(\heartsuit w)$ and $d(\lambda)$ are undefined (thus, d checks that the first symbol of a string is diamonds, and deletes that symbol). Now consider the interpretation $\sigma(K'')$ of the jungle K'' of Figure 1. If one constructs $\alpha \in ASS_{K'',\sigma}$ in a bottom-up fashion, then $\alpha(res_{K''})$ can have the values $\diamondsuit\diamondsuit\diamondsuit$, $\diamondsuit\heartsuit\heartsuit$, $\heartsuit\diamondsuit\diamondsuit$, and $\heartsuit\heartsuit\heartsuit$ (note that the last two symbols are equal because of the shared node with label $\mathtt{a}$). Thus, due to the presence of the node with label $\mathtt{d}$, we have $\sigma(K'') = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$. If we redefine $\sigma(\mathtt{c}) = \emptyset$, then $\sigma(K'') = \emptyset$ because no value can be assigned to the node with label $\mathtt{c}$.

The jungle K' of Figure 1 is interpreted as the function $\sigma(K') = k' \colon \mathbb{A} \to \mathbb{A}$ such that $k'(w) = \diamondsuit w$ for every $w \in \mathbb{A}$. □

5.2 Delegation Networks

We are now ready to give the formal definition of delegation networks.

Definition 4. *A* jungle delegation network *is a system $\mathcal{N} = (G, \mathbb{A}, \sigma)$, where $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ is a context-free jungle grammar and $(\mathbb{A}, \sigma)$ is a Σ-algebra. If G is a context-free tree grammar, then $\mathcal{N}$ is a* tree delegation network.[6]

The signature $\Xi \cup \Sigma$ is denoted by $\Sigma_\mathcal{N}$. For $\mathtt{g}^{(k)} \in \Xi$ we denote by $\mathrm{rhs}_G(\mathtt{g})$ the set of right-hand sides of rules in R with left-hand side $\mathtt{g}(x_1, \ldots, x_k)$.

The semantics of $\mathcal{N}$ is obtained by defining a $\Sigma_\mathcal{N}$-interpretation $\sigma_\mathcal{N}$ into $\mathbb{A}$ that agrees with σ on Σ. Since the rules of G are recursive, it is natural to choose a least fixed point semantics, using Kleene's fixed point theorem which we state next (see, e.g., [13, Theorem I.4.8]).

Proposition 1. *Let C be a complete lattice, and let $\varphi \colon C \to C$ be an ω-continuous function. Then φ has a least fixed point, and this least fixed point is equal to the least upper bound of all $\varphi^m(0)$, $m \in \mathbb{N}$, where 0 is the zero element of C.*

[6] A tree delegation network $\mathcal{N}$ is called a finitary delegation network in [5]. In [11, Definition 5.1] the syntactic part G of $\mathcal{N}$ is called a system of context-free Σ-equations.

We recall that a complete lattice is a set C with a partial order $\leq$ such that every subset of C has a least upper bound. Moreover, ω-continuity of φ means that if $c_0 \leq c_1 \leq c_2 \leq \cdots$ (with $c_i \in C$) and $c \in C$ is the least upper bound of $\{c_i \mid i \in \mathbb{N}\}$, then $\varphi(c)$ is the least upper bound of $\{\varphi(c_i) \mid i \in \mathbb{N}\}$. The zero element 0 of C is its smallest element (i.e., the least upper bound of $\emptyset$).

As is well known, the set of all relations $r \subseteq \mathbb{A}^k \times \mathbb{A}$ is a complete lattice with $\subseteq$ as partial order. We extend this ordering to $\Sigma_{\mathcal{N}}$-interpretations τ, τ' into $\mathbb{A}$ in the usual way: $\tau \leq \tau'$ if and only if $\tau(\mathtt{f}) \subseteq \tau'(\mathtt{f})$ for all $\mathtt{f} \in \Sigma_{\mathcal{N}}$. With the partial order $\leq$, the set of all $\Sigma_{\mathcal{N}}$-interpretations into $\mathbb{A}$ is a complete lattice, as is also well known. If T is a set of such interpretations, with least upper bound υ, then, for every $\mathtt{f} \in \Sigma_{\mathcal{N}}$, $\upsilon(\mathtt{f})$ is the union of all $\tau(\mathtt{f})$, $\tau \in T$. Note that the zero element 0 of the lattice is the $\Sigma_{\mathcal{N}}$-interpretation such that $0(\mathtt{f}) = \emptyset$ for all $\mathtt{f} \in \Sigma_{\mathcal{N}}$.

The semantics of $\mathcal{N}$ is a subset of $\mathbb{A}$. It will be called the language defined by $\mathcal{N}$, generalizing the notions of string language, tree language, graph language, picture language, etc. The semantics of $\mathcal{N}$ is an obvious generalization of the one for tree delegation networks (see [5, Definition 2.4]). Intuitively, G is viewed as a system of equations $(\mathtt{g} = \mathrm{rhs}_G)_{\mathtt{g} \in \Xi}$, where rhs_G is viewed as the union of its elements, and these equations are solved in the algebra $(\mathbb{A}, \sigma)$.

Definition 5. *Let $\mathcal{N} = (G, \mathbb{A}, \sigma)$ be a jungle delegation network such that $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$. Let $C_{\mathcal{N}}$ be the complete lattice of all $\Sigma_{\mathcal{N}}$-interpretations into $\mathbb{A}$.*

1. *The function $\varphi_{\mathcal{N}} \colon C_{\mathcal{N}} \to C_{\mathcal{N}}$ is defined as follows for every $\tau \in C_{\mathcal{N}}$:*
   - $\varphi_{\mathcal{N}}(\tau)(\mathtt{f}) = \sigma(\mathtt{f})$ *for every* $\mathtt{f} \in \Sigma$, *and*
   - $\varphi_{\mathcal{N}}(\tau)(\mathtt{g}) = \tau(\mathrm{rhs}_G(\mathtt{g}))$ *for every* $\mathtt{g} \in \Xi$.
2. *The least fixed point of $\varphi_{\mathcal{N}}$ is denoted by $\sigma_{\mathcal{N}}$.*[7]
3. *The* language defined by $\mathcal{N}$ *is* $L(\mathcal{N}) = \sigma_{\mathcal{N}}(\mathtt{g}_{\mathrm{in}})$.

Note that the language defined by $\mathcal{N}$ is a subset of $\mathbb{A}$, because the rank of $\mathtt{g}_{\mathrm{in}}$ is 0. Note furthermore that, for $\mathtt{f} \in \Sigma$, we have $\sigma_{\mathcal{N}}(\mathtt{f}) = \sigma(\mathtt{f})$.

As already observed in [5], if $\mathcal{N}$ is a tree delegation network, then, for every $\mathtt{g} \in \Xi$, the relation $\sigma_{\mathcal{N}}(\mathtt{g})$ is what is called the call by value relation computed by G over $(\mathbb{A}, \sigma)$ in the discussion after Corollary 5.7 in [11].

Example 5. Consider the jungle delegation network $\mathcal{N}_2 = (G_2, \mathbb{A}, \sigma)$ where G_2 is the CFJG of Example 2 and $(\mathbb{A}, \sigma)$ is the string algebra defined in Example 4. Since K' does not contain nonterminal symbols, we obtain that $\sigma_{\mathcal{N}_2}(\mathtt{h}) = \sigma(K')$, which is the function k' that puts $\diamond$ in front of a string (as observed in the latter example). This implies that $\sigma_{\mathcal{N}_2}(\mathtt{g}) = \sigma'(K)$, where σ' extends σ with $\sigma'(\mathtt{h}) = k'$. From this it is easy to see that $\mathcal{N}_2$ defines the language $L(\mathcal{N}_2) = \sigma_{\mathcal{N}_2}(\mathtt{g}) = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$.

[7] By Proposition 1, $\sigma_{\mathcal{N}}$ exists, as it is easy to verify that $\varphi_{\mathcal{N}}$ is ω-continuous. Note that by Definition 3, if $\tau_0 \leq \tau_1 \leq \tau_2 \leq \cdots$ (with $\tau_i \in C_{\mathcal{N}}$), $\tau \in C_{\mathcal{N}}$ is the least upper bound of $\{\tau_i \mid i \in \mathbb{N}\}$, and $\mathcal{J} \subseteq \mathrm{J}_\Sigma(X_n)$ is finite, then $\tau(\mathcal{J})$ is the union of all $\tau_i(\mathcal{J})$, $i \in \mathbb{N}$.

We have seen in Example 2 that $L_{\mathrm{J}}(G_2) = \{K''\}$, where K'' is given in Figure 1. Combining this with the fact that $\sigma(K'') = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$ (as seen in Example 4), we find that $L(\mathcal{N}_2) = \sigma(L_{\mathrm{J}}(G_2))$. In other words, $L(\mathcal{N}_2)$ equals the interpretation of the jungle language generated by its CFJG G_2. We will prove in Theorem 6 that this is a result that holds in general. □

To express the semantics of a substitution $J[v \leftarrow K]$ in terms of that of J and K, we need some terminology. Let v have rank k, let @ be a new symbol of rank k, and let $J[v \leftarrow @]$ be the jungle obtained from J by changing the label of v into @, i.e., $J[v \leftarrow @] = J[v \leftarrow @(x_1, \ldots, x_k)]$. The next lemma shows that the semantics of $J[v \leftarrow K]$ is equal to the semantics of $J[v \leftarrow @]$, when @ is interpreted as the semantics of K. For a Σ-algebra $(\mathbb{A}, \sigma)$ and a relation $r \subseteq \mathbb{A}^k \times \mathbb{A}$, we denote by $\sigma\langle @ := r\rangle$ the $(\Sigma \cup \{@\})$-interpretation σ' into $\mathbb{A}$ such that $\sigma'(@) = r$ and $\sigma'(\mathtt{f}) = \sigma(\mathtt{f})$ for every $\mathtt{f} \in \Sigma$.

Lemma 5. *Let $(\mathbb{A}, \sigma)$ be a Σ-algebra. Let $J \in \mathrm{J}_\Sigma(X_n)$, $v \in V_J$ with $rk_J(v) = k$ and $lab_J(v) \in \Sigma$, and $K \in \mathrm{J}_\Sigma(X_k)$. Then $\sigma(J[v \leftarrow K]) = \sigma\langle @ := \sigma(K)\rangle(J[v \leftarrow @])$.*

Proof. Without loss of generality we assume that $\mathtt{id} \in \Sigma$, that $\sigma(\mathtt{id})$ is the identity on $\mathbb{A}$, and that $\mathtt{id}$ does not occur in J and K. Obviously, $\sigma(I) = \sigma(ctr(I))$ for every $I \in \mathrm{J}_\Sigma(X_n)$. Hence $\sigma(J[v \leftarrow K]) = \sigma(J\langle v \leftarrow K\rangle)$.

If $\alpha \in ASS_{J\langle v \leftarrow K\rangle, \sigma}(a_1, \ldots, a_n)$, then the restriction α_K of α to V_K is in $ASS_{K,\sigma}(b_1, \ldots, b_k)$ where $b_i = \alpha(arg_J(v, i))$. Moreover, $\alpha_K(res_K) = \alpha(v)$. This shows that the restriction α_J of α to V_J is in $ASS_{J[v \leftarrow @],\, \sigma\langle @ := \sigma(K)\rangle}(a_1, \ldots, a_n)$, and so $\sigma(J[v \leftarrow K]) \subseteq \sigma\langle @ := \sigma(K)\rangle(J[v \leftarrow @])$.

If $\alpha_J \in ASS_{J[v \leftarrow @],\, \sigma\langle @ := \sigma(K)\rangle}(a_1, \ldots, a_n)$, then there exists an assignment $\alpha_K \in ASS_{K,\sigma}(b_1, \ldots, b_k)$ such that $b_i = \alpha_J(arg_J(v, i))$ and $\alpha_K(res_K) = \alpha_J(v)$. It is now clear that $\alpha_J \cup \alpha_K$ is in $ASS_{J\langle v \leftarrow K\rangle, \sigma}(a_1, \ldots, a_n)$, from which we can conclude that $\sigma\langle @ := \sigma(K)\rangle(J[v \leftarrow @]) \subseteq \sigma(J[v \leftarrow K])$. □

The next theorem is similar to Theorem 1 in Section 4. It shows that rules of the grammar of a jungle delegation network can be substituted into each other, without changing the language defined by the network.

Theorem 4. *Let $\mathcal{N} = (G, \mathbb{A}, \sigma)$ and $\mathcal{N}' = (G', \mathbb{A}, \sigma)$ be jungle delegation networks, where G and G' are as in Theorem 1. Then $L(\mathcal{N}') = L(\mathcal{N})$.*

Proof. By assumption, $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ and $G' = (\Xi, \Sigma, R', \mathtt{g}_{\mathrm{in}})$ where R' is obtained from R by replacing the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by all rules $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ such that $\mathtt{h}(x_1, \ldots, x_m) \to K'$ is in R; here v is a node of K such that $lab_K(v) = \mathtt{h}^{(m)} \in \Xi$.

We will prove that $\sigma_{\mathcal{N}} = \sigma_{\mathcal{N}'}$, where $\sigma_{\mathcal{N}}$ is the least fixed point of $\varphi_{\mathcal{N}}$ and similarly for $\mathcal{N}'$ (see Definition 5). For $m \in \mathbb{N}$, we will denote $\varphi^m_{\mathcal{N}}(0)$ by $\sigma_{\mathcal{N},m}$, and similarly for $\mathcal{N}'$. Thus, by Proposition 1, $\sigma_{\mathcal{N}}$ is the least upper bound of all $\sigma_{\mathcal{N},m}$, $m \in \mathbb{N}$. Note that by Definition 5, $\sigma_{\mathcal{N},m+1}(\mathtt{k}) = \sigma_{\mathcal{N},m}(\mathrm{rhs}_G(\mathtt{k}))$ for every $m \in \mathbb{N}$ and $\mathtt{k} \in \Xi$; and, of course, $\sigma_{\mathcal{N},0}(\mathtt{k}) = \emptyset$.

We first show that $\sigma_{\mathcal{N}',m} \leq \sigma_{\mathcal{N}}$ for all $m \in \mathbb{N}$, which implies that $\sigma_{\mathcal{N}'} \leq \sigma_{\mathcal{N}}$. It suffices to prove that $\sigma_{\mathcal{N}',m}(\mathtt{k}) \subseteq \sigma_{\mathcal{N}}(\mathtt{k})$ for all $\mathtt{k} \in \Xi$ and $m \in \mathbb{N}$. The proof is by

induction on m. It is trivial for $m = 0$. For the induction step, consider $\sigma_{\mathcal{N}',m+1}(\mathtt{k})$. Since this equals $\sigma_{\mathcal{N}',m}(\mathrm{rhs}_{G'}(\mathtt{k}))$, it remains to prove that $\sigma_{\mathcal{N}',m}(J) \subseteq \sigma_{\mathcal{N}}(\mathtt{k})$ for every $J \in \mathrm{rhs}_{G'}(\mathtt{k})$. We consider two cases.

Case 1: $J \in \mathrm{rhs}_G(\mathtt{k})$. By induction we have $\sigma_{\mathcal{N}',m}(J) \subseteq \sigma_{\mathcal{N}}(J)$; because, in general, if $\tau_1 \leq \tau_2$ then $\tau_1(J) \subseteq \tau_2(J)$. Moreover $\sigma_{\mathcal{N}}(J) \subseteq \sigma_{\mathcal{N}}(\mathtt{k})$, because $\sigma_{\mathcal{N}}$ is a fixed point of $\varphi_{\mathcal{N}}$.

Case 2: $\mathtt{k} = \mathtt{g}$ and $J = K[v \leftarrow K']$. By Lemma 5, $\sigma_{\mathcal{N}',m}(K[v \leftarrow K']) = \sigma_{\mathcal{N}',m}\langle @ := \sigma_{\mathcal{N}',m}(K')\rangle(K[v \leftarrow @])$. By induction, and since $\sigma_{\mathcal{N}}$ is a fixed point of $\varphi_{\mathcal{N}}$, we obtain that $\sigma_{\mathcal{N}',m}(K') \subseteq \sigma_{\mathcal{N}}(\mathtt{h})$. Thus, again by induction,

$$\sigma_{\mathcal{N}',m}(K[v \leftarrow K']) \subseteq \sigma_{\mathcal{N}}\langle @ := \sigma_{\mathcal{N}}(\mathtt{h})\rangle(K[v \leftarrow @]) = \sigma_{\mathcal{N}}(K) \subseteq \sigma_{\mathcal{N}}(\mathtt{g}).$$

In the other direction, we prove by induction on m that $\sigma_{\mathcal{N},m} \leq \sigma_{\mathcal{N}'}$. As above, it suffices to prove in the induction step that $\sigma_{\mathcal{N},m}(J) \subseteq \sigma_{\mathcal{N}'}(\mathtt{k})$ for every $J \in \mathrm{rhs}_G(\mathtt{k})$. As above there are two cases. The first case, where $J \in \mathrm{rhs}_{G'}(\mathtt{k})$, is handled as above. It remains to consider the second case, where $\mathtt{k} = \mathtt{g}$ and $J = K$. If $m = 0$ then $\sigma_{\mathcal{N},m}(K) = \emptyset$ because a non-variable (namely $\mathtt{h}$) occurs in K, and we are ready. Now let $m \geq 1$. Then

$$\begin{array}{rcl}
\sigma_{\mathcal{N},m}(K) & = & \sigma_{\mathcal{N},m}\langle @ := \sigma_{\mathcal{N},m}(\mathtt{h})\rangle(K[v \leftarrow @]) \\
& = & \sigma_{\mathcal{N},m}\langle @ := \sigma_{\mathcal{N},m-1}(\mathrm{rhs}_G(\mathtt{h}))\rangle(K[v \leftarrow @]) \\
& = & \bigcup_{K' \in \mathrm{rhs}_G(\mathtt{h})} \sigma_{\mathcal{N},m}\langle @ := \sigma_{\mathcal{N},m-1}(K')\rangle(K[v \leftarrow @]) \\
& \subseteq & \bigcup_{K' \in \mathrm{rhs}_G(\mathtt{h})} \sigma_{\mathcal{N}'}\langle @ := \sigma_{\mathcal{N}'}(K')\rangle(K[v \leftarrow @]) \\
& = & \bigcup_{K' \in \mathrm{rhs}_G(\mathtt{h})} \sigma_{\mathcal{N}'}(K[v \leftarrow K']) \\
& \subseteq & \sigma_{\mathcal{N}'}(\mathtt{g})
\end{array}$$

where the last three steps are by induction, by Lemma 5, and by the fact that $\sigma_{\mathcal{N}'}$ is a fixed point of $\varphi_{\mathcal{N}'}$, respectively. □

We can now prove that tree delegation networks are as powerful as jungle delegation networks.

Theorem 5. *For every jungle delegation network $\mathcal{N}$ there is a tree delegation network $\mathcal{N}'$ over the same algebra such that $L(\mathcal{N}') = L(\mathcal{N})$.*

Proof. Let $\mathcal{N} = (G, \mathbb{A}, \sigma)$. Then we define $\mathcal{N}' = (H, \mathbb{A}, \sigma)$ where H is the CFTG constructed in the proof of Theorem 2. Since the proof of $L_{\mathrm{J}}(H) = L_{\mathrm{J}}(G)$ was entirely based on Theorem 1, the proof of $L(\mathcal{N}') = L(\mathcal{N})$ is exactly the same, now based on Theorem 4. □

Note that the construction of $\mathcal{N}'$ in the above proof does not depend on the given algebra. Thus, Theorem 5 is a program-schematic result.

Finally, we prove a Mezei-Wright-like result for jungle delegation networks $\mathcal{N} = (G, \mathbb{A}, \sigma)$: the language defined by $\mathcal{N}$ is equal to the semantics of the jungle language generated by G.

Theorem 6. *For every jungle delegation network* $\mathcal{N} = (G, \mathbb{A}, \sigma)$,

$$L(\mathcal{N}) = \sigma(L_{\mathrm{J}}(G)).$$

If, moreover, $(\mathbb{A}, \sigma)$ *is deterministic, then* $L(\mathcal{N}) = \sigma(L_{\mathrm{T}}(G))$.

Proof. Let $\mathcal{N}' = (H, \mathbb{A}, \sigma)$ be the tree delegation network constructed in the proofs of Theorems 2 and 5. Then $L(\mathcal{N}) = L(\mathcal{N}')$ by Theorem 5, and $L_{\mathrm{J}}(G) = L_{\mathrm{J}}(H)$ by Theorem 2. It is proved in [5, Theorem 5.6] for tree delegation networks that $L(\mathcal{N}') = \sigma(L_{\mathrm{J}}(H))$.

In [5, Theorem 6.6] it is proved that if $(\mathbb{A}, \sigma)$ is deterministic, then $L(\mathcal{N}') = \sigma(L_{\mathrm{IO}}(H))$. In the proof of Theorem 3 we already saw that $L_{\mathrm{IO}}(H) = L_{\mathrm{T}}(G)$. □

Example 6. Let G_1 and G_2 be the grammars of Examples 1 and 2. We have seen in Example 2 that $L_{\mathrm{J}}(G_1) = L_{\mathrm{J}}(G_2)$. Hence, by Theorem 6, $L(\mathcal{N}_1) = L(\mathcal{N}_2)$ for all delegation networks $\mathcal{N}_1 = (G_1, \mathbb{A}, \sigma)$ and $\mathcal{N}_2 = (G_2, \mathbb{A}, \sigma)$ over the same algebra $(\mathbb{A}, \sigma)$. In other words, G_1 and G_2 are equivalent program schemes. In particular, by Example 5, we have that $L(\mathcal{N}_1) = \sigma(K'') = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$ for the string algebra $(\mathbb{A}, \sigma)$ defined in Example 4.

Note that $L(\mathcal{N}_1)$ is not equal to $\sigma(L_{\mathrm{T}}(G_1))$. In fact, $L_{\mathrm{T}}(G_1) = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$ (see the footnote in the proof of Theorem 3), and $\sigma(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a})))$ is the set of all strings of length 3 in $\mathbb{A} = \{\diamondsuit, \heartsuit\}^*$. As discussed in Example 4, this illustrates two effects: in the IO-generated tree the second and third `a` are not shared, so that the second and third symbol of the resulting strings may differ. Morover, the node with label `d` above the root of K'' (which is garbage) is not present, thus allowing the first symbol of the string to differ from $\diamondsuit$. In fact, the node with label `c` is garbage as well and thus not present in the IO-generated tree. If we redefine $\sigma(\mathtt{c}) = \emptyset$, see Example 4, then $L(\mathcal{N}_1) = L(\mathcal{N}_2) = \emptyset$ whereas $\sigma(L_{\mathrm{T}}(G_1))$ is the same as above. □

References

[1] Arbib, M. A. and Give'on, Y. Algebra automata I: Parallel programming as a prolegomena to the categorical approach. *Information and Control*, 12:331–345, 1968.

[2] Ariola, Z. M. and Klop, J. W. Equational term graph rewriting. *Fundamenta Informaticae*, 26:207–240, 1996.

[3] Courcelle, B. An axiomatic definition of context–free rewriting and its application to NLC graph grammars. *Theoretical Computer Science*, 55:141–181, 1987.

[4] Drewes, F. From tree-based generation to delegation networks. In Bozapalidis, S. and Rahonis, G., editors, *Proc. 2nd International Conference on Algebraic Informatics (CAI'07)*, volume 4728 of Lecture Notes in Computer Science, pages 48–72. Springer, 2007.

[5] Drewes, F. and Engelfriet, J. The generative power of delegation networks. Report 12.15, Umeå University, 2012. To appear in *Information and Computation*. Available online doi:10.1016/j.ic.2015.04.005

[6] Drewes, F., Habel, A. and Kreowski, H.-J. Hyperedge replacement graph grammars. In Rozenberg, G., editor, *Handbook of Graph Grammars and Computing by Graph Transformation*. Vol. 1: *Foundations*, chapter 2, pages 95–162. World Scientific, Singapore, 1997.

[7] Duske, J., Parchmann, R., Sedello, M., and Specht, J. IO-macro languages and attributed translations. *Information and Control*, 35:87–105, 1977.

[8] Engelfriet, J. and Filè, G. The formal power of one-visit attribute grammars. *Acta Informatica*, 16:275–302, 1981.

[9] Engelfriet, J. and Heyker, L. Context–free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29:161–210, 1992.

[10] Engelfriet, J. and Maneth, S. Tree languages generated by context-free graph grammars. In Ehrig, H., Engels, G., and Kreowski, H.-J., editors, *Proc. Theory and Application of Graph Transformations (TAGT'98)*, volume 1764 of *Lecture Notes in Computer Science*, pages 15–29, 2000.

[11] Engelfriet, J. and Schmidt, E. M. IO and OI. *Journal of Computer and System Sciences*, 15:328–353, 1977, and 16:67–99, 1978.

[12] Engelfriet, J. and Vogler, H. The translation power of top-down tree-to-graph transducers. *Journal of Computer and System Sciences*, 49:258–305, 1994.

[13] Gécseg, F. and Steinby, M. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[14] Gécseg, F. and Steinby, M. Tree languages. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages.* Vol. 3: *Beyond Words*, chapter 1, pages 1–68. Springer, 1997.

[15] Habel, A., Kreowski, H.-J., and Plump, D. Jungle evaluation. *Fundamenta Informaticae*, 15:37–60, 1991.

[16] Hoffmann, B. and Plump, D. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 1991.

[17] Mezei, J. and Wright, J. B. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.

[18] Plump, D. Term graph rewriting. In Ehrig, H., Engels, G., Kreowski, H.-J., and Rozenberg, G., editors, *Handbook of Graph Grammars and Computing by Graph Transformation.* Vol. 2: *Applications, Languages, and Tools*, chapter 1, pages 3–61. World Scientific, Singapore, 1999.

Received 29th April 2015