The Generative Power of Delegation Networks

Frank Drewes
Department of Computing Science, Umeå University
S–901 87 Umeå, Sweden
drewes@cs.umu.se

Joost Engelfriet
Leiden Institute of Advanced Computer Science, Leiden University
P.O. Box 9612, NL-2300 RA Leiden, The Netherlands
engelfri@liacs.nl

Abstract. A device that generates trees over a ranked alphabet Σ, together with an interpretation of the symbols in Σ as functions or relations on a domain $\mathbb{A}$, generates subsets of $\mathbb{A}$. This concept of tree-based generators is well known and essentially already present in the seminal paper by Mezei and Wright from 1967. A delegation network is a system consisting of a finite set of such generators that can "delegate" parts of the generation process to each other. It can be viewed as consisting of an (extended) IO context-free tree grammar and an interpretation. We investigate the language-theoretic properties of these systems and establish several characterizations of the generated languages. In particular, we obtain results in the style of Mezei and Wright. We also study the hierarchy of tree language classes obtained by iterating the concept of delegation, and show that this hierarchy is properly contained in the closure of the regular tree languages under nondeterministic macro tree transductions, but not in the IO-hierarchy.

1 Introduction

The theory of tree languages and tree transformations is an important and lively field of theoretical computer science [GS84, NP92, GS97, FV98, CDG$^+$02, FV09]. For the most part, this theory is concerned with formal devices that generate, recognize or transform trees over a finite ranked alphabet. In such an alphabet Σ, every symbol $\mathtt{f} \in \Sigma$ has a rank k that determines the number of (direct) subtrees that it requires: $\mathtt{f}[t_1, \ldots, t_k]$ denotes the tree with root label $\mathtt{f}$ and subtrees $t_1, \ldots, t_k$.

Let us use the term *tree generator* for any device γ that defines a tree language $L(\gamma) \subseteq \mathrm{T}_\Sigma$, where T_Σ denotes the set of all trees over Σ. A tree $t \in \mathrm{T}_\Sigma$ can be seen as a formal expression composed of abstract operation symbols, where ranks are arities. The usefulness of devices dealing with such trees is to a large extent based on the fact that trees can be interpreted by choosing a domain

$\mathbb{A}$ and associating a (total) function $f\colon \mathbb{A}^k \to \mathbb{A}$ with each symbol $\mathtt{f}$ of rank k. Thus, given such an interpretation σ, a tree t evaluates recursively to an element $\sigma(t)$ of $\mathbb{A}$: $\sigma(\mathtt{f}[t_1, \ldots, t_k]) = f(\sigma(t_1), \ldots, \sigma(t_k))$. Consequently, a device that generates trees provides the syntactic component of a *tree-based generator* – a pair consisting of the tree generator γ with $L(\gamma) \subseteq \mathrm{T}_\Sigma$ and an interpretation σ of the symbols of Σ. By definition, it generates a subset of $\mathbb{A}$, viz. the set $\sigma(L(\gamma)) = \{\sigma(t) \mid t \in L(\gamma)\} \subseteq \mathbb{A}$. The application of this idea to the area of picture generation is studied in [Dre00, Dre01, Dre06].

In this paper we study so-called delegation networks, which were introduced in [Dre07a, Dre07b]. Such a delegation network consists of a finite number of generators that can "delegate" parts of the generation process to each other. The purpose of delegation is to make it possible to combine several generators, that may even be of different types, to generate complex objects. Intuitively, an individual generator in a delegation network can be compared to a nondeterministic procedure that can make use of other such procedures in order to accomplish its task. For example, one could envision the generation of 3D models of cities. In a delegation network for this purpose, one generator could generate the overall structure of the city, placing roads, parks and buildings in appropriate places while delegating the generation of these details to other generators. The "park generator" could in turn delegate the generation of plants to a generator that uses a suitable type of grammar to generate the branching structure of a plant, while the form of leaves may be generated by a different type of grammar and their texture by yet another one (see [Dre06] for discussions of such devices). Delegation networks also make it possible to use non-grammatical generators by viewing them as so-called primitives – "black boxes" that turn input into output in a possibly nondeterministic way (see below). For example, in the hypothetical scenario above, textures may be generated by a primitive whose internal workings are based on random noise or cellular automata. Such a primitive could, for instance, take a 3D object as input and map a generated texture onto its surface. For concrete examples of delegation networks that combine different types of picture-generating mechanisms, see [Dre07b].

Let us discuss the formal notions underlying delegation networks in more detail. Just as tree-based generators, delegation networks use tree generators to generate subsets of a domain $\mathbb{A}$. However, in order to accomplish the ideas sketched above, they generalize tree-based generators in two ways.

The first generalization concerns the type of interpretations used. As in [ES77, ES78], where they are called nondeterministic Σ-algebras, we consider interpretations in which f, the interpretation of a symbol $\mathtt{f}$ of rank k, may be an arbitrary relation $f \subseteq \mathbb{A}^k \times \mathbb{A}$ rather than being required to be a function from $\mathbb{A}^k$ to $\mathbb{A}$. Clearly, ordinary interpretations are a special case if we regard a function as a relation in the usual way. A tuple $((a_1, \ldots, a_k), a) \in f$ represents the fact that an application of f to the arguments $a_1, \ldots, a_k$ may, nondeterministically, produce the value a. We write $f(a_1, \ldots, a_k)$ for the set $\{a \in \mathbb{A} \mid ((a_1, \ldots, a_k), a) \in f\}$

of possible values of the application of f to $a_1, \ldots, a_k$.

As a consequence of this generalization, the evaluation of a tree now determines a subset of $\mathbb{A}$, which is the set of its possible values:

$$\sigma(\mathtt{f}[t_1, \ldots, t_k]) = \bigcup\{f(a_1, \ldots, a_k) \mid a_1 \in \sigma(t_1), \ldots, a_k \in \sigma(t_k)\},$$

and $\sigma(L(\gamma)) = \bigcup\{\sigma(t) \mid t \in L(\gamma)\} \subseteq \mathbb{A}$.

We note here that delegation networks were defined over many-sorted signatures in [Dre07a, Dre07b], using a finite set of domains instead of a single domain $\mathbb{A}$. Hence, in that setting, the interpretation of a symbol $\mathtt{f}$ of rank k is a relation $f \subseteq (\mathbb{A}_1 \times \cdots \times \mathbb{A}_k) \times \mathbb{A}_{k+1}$, where $\mathbb{A}_1, \ldots, \mathbb{A}_{k+1}$ may differ from each other. In view of the motivation of delegation networks sketched above, it seems natural to adopt this more general setting. However, many-sortedness does not affect any of the formal reasonings in this paper – its sole effect is to complicate notation. Therefore, we have chosen to restrict the exposition to one-sorted signatures (which are ranked alphabets) in the present paper.

The second generalization that leads from tree-based generators to delegation networks is obtained by evaluating trees that contain n formal parameters. More precisely, we consider trees $t \in \mathrm{T}_{\Sigma,n} = \mathrm{T}_{\Sigma\cup\{x_1,\ldots,x_n\}}$, where the so-called parameter symbols x_i are of rank 0. Then $\sigma(t) \subseteq \mathbb{A}^n \times \mathbb{A}$, as follows. For n arguments $a_1, \ldots, a_n$ (the actual parameters), t is recursively evaluated as before, with the addition that each x_i is interpreted as a_i. To be precise, $\sigma(x_i)(a_1, \ldots, a_n) = \{a_i\}$ and $\sigma(\mathtt{f}[t_1, \ldots, t_k])(a_1, \ldots, a_n)$ is the union of all $f(b_1, \ldots, b_k)$ where $b_j \in \sigma(t_j)(a_1, \ldots, a_n)$.

Note that, if we now consider a tree generator γ that generates trees with parameter symbols, i.e., $L(\gamma) \subseteq \mathrm{T}_{\Sigma,n}$, then the straightforward definition of $\sigma(L(\gamma))$ yields a subset of $\mathbb{A}^n \times \mathbb{A}$: for $a_1, \ldots, a_n \in \mathbb{A}$, $\sigma(L(\gamma))(a_1, \ldots, a_n) = \bigcup\{\sigma(t)(a_1, \ldots, a_n) \mid t \in L(\gamma)\}$. Thus, a tree generator with k parameter symbols can be used to define the interpretation $g \subseteq \mathbb{A}^k \times \mathbb{A}$ of a symbol $\mathtt{g} \in \Sigma$ of rank k.

The last observation provides the formal basis for delegation, as it allows us to consider a finite set of tree generators that call each other recursively. More precisely, we divide Σ into two ranked alphabets Π and $\mathcal{G}$ of so-called primitives and generator symbols, respectively. The primitives come with an a priori interpretation π, whereas every generator symbol $\mathtt{g}$ of rank k is associated with a tree generator $\gamma_{\mathtt{g}}$ that generates a tree language in $\mathrm{T}_{\mathcal{G}\cup\Pi,k}$. Intuitively, the purpose of $\gamma_{\mathtt{g}}$ is to generate the interpretation of $\mathtt{g}$ in the sense of the previous paragraph. Whenever $\mathtt{g}$ appears in a tree generated by $\gamma_{\mathtt{g}'}$ for some $\mathtt{g}' \in \mathcal{G}$, this signals a delegation to $\gamma_{\mathtt{g}}$ by $\gamma_{\mathtt{g}'}$, i.e., $\gamma_{\mathtt{g}'}$ calls $\gamma_{\mathtt{g}}$. The semantics of delegation is straightforward in the case where $L(\gamma_{\mathtt{g}}) \subseteq \mathrm{T}_{\Pi,k}$. Then $\mathtt{g}$ is interpreted as $\pi(L(\gamma_{\mathtt{g}}))$. The general case (in which $L(\gamma_{\mathtt{g}}) \subseteq \mathrm{T}_{\mathcal{G}\cup\Pi,k}$) is slightly more complicated since delegation can be cyclic, which means that we will naturally adopt a least fixed point semantics.

Every delegation network $\mathcal{N}$ contains a designated main generator symbol $\mathtt{g}_0$ of rank 0. According to the semantics of delegation networks, $\mathtt{g}_0$ generates a

subset of $\mathbb{A}$. This set is considered to be the "language" $L(\mathcal{N})$ generated by $\mathcal{N}$. As we will see, an appropriate operational view of delegation is that of parallel distributed processes executing (copies of) the tree generators $\gamma_{\mathtt{g}}$ and evaluating the resulting trees, as follows. Initially, a process corresponding to $\mathtt{g}_0$ is started. This process uses $\gamma_{\mathtt{g}_0}$ to generate a tree. Then, for every individual occurrence of a symbol $\mathtt{g}$ in that tree, a new process working in a similar manner (using $\gamma_{\mathtt{g}}$ instead of $\gamma_{\mathtt{g}_0}$) is started recursively. Evaluation is taken care of locally in these processes as well, as each process evaluates the tree it has generated. To be able to do this, its parent process must eventually provide it with each of its actual parameters (call-by-value). Evaluation of a tree works as follows: as soon as the subtrees of (an occurrence of) a symbol $\mathtt{f} \in \Pi$ have been evaluated, f is applied to them nondeterministically. When a subtree of a symbol $\mathtt{g} \in \mathcal{G}$ has been evaluated, the resulting value is passed to the child process that evaluates the occurrence of $\mathtt{g}$, providing it with one of its actual parameters. When the child process has received all its actual parameters and has finished the evaluation of its tree (i.e., has arrived at the root), it reports the value it has computed to the parent process and terminates. The parent process can then continue the evaluation of its own tree. Eventually, when the main process has finished the evaluation of its tree, the resulting value is returned, being an element of the language generated by the delegation network.

A delegation network can be seen as an "extended" IO context-free tree grammar that is interpreted in a nondeterministic algebra $(\mathbb{A}, \pi)$. In this sense, the present paper contributes to the theory of context-free tree grammars, which are of interest in, e.g., recursive program scheme theory, computational linguistics and data compression theory.[1] The primitives and generator symbols of the network correspond to the terminals and nonterminals of the grammar, respectively, and the tree language $L(\gamma_{\mathtt{g}})$ is the set of right-hand sides of the rules of $\mathtt{g}$ (which have the left-hand side $\mathtt{g}[x_1, \ldots, x_k]$ when $\mathtt{g}$ has rank k). Thus, as in the extended Backus-Naur Form, there may be infinitely many right-hand sides for a given left-hand side. Non-extended ("finitary") IO context-free tree grammars and their interpretation in nondeterministic algebras were studied in [ES77, ES78]. In the present paper, we generalize some of the results of [ES77, ES78], showing that they carry over to the extended case. In particular, we prove several MW-like results. Here, 'MW' refers to the seminal paper [MW67], in which Mezei and Wright showed that the equational subsets of an algebra are exactly those that can be obtained by evaluating regular tree languages with respect to the algebra in question.

Let us discuss the main results of this paper. In Section 3 (Theorem 3.5), we prove an operational characterization of the language generated by a delegation network (defined as a least fixed point in Section 2). This characterization formalizes the operational view discussed above and shows that this view is adequate.

[1] Recent contributions in these fields are [Sch11], [KR11, ME12], and [LMSS12], respectively.

Theorem 5.6 in Section 5 is our main MW-like result. It solves a problem left open in [ES78] (for the finitary case), namely to find an MW-like characterization of the languages generated by delegation networks in the presence of nondeterministic primitives, i.e., primitives that are interpreted as non-functional relations. We accomplish this by generating and interpreting so-called jungles [HKP91] instead of trees.

Essentially, a jungle J over the ranked alphabet Σ is a DOAG (directed ordered acyclic graph) with a designated node, such that each node with k outgoing edges (which are linearly ordered) is labeled with a symbol from Σ of rank k, cf. [AG68]. Each node v of J determines a tree over Σ obtained by unfolding the subgraph induced by all nodes that are reachable from v. Thus, a jungle J can be viewed as a set of trees over Σ of which the nodes can be shared, and it represents the tree determined by its designated node, denoted by $\mathrm{tree}(J)$. The nodes that are not reachable from the designated node are said to be garbage, as they do not contribute to $\mathrm{tree}(J)$. Given an interpretation σ, the set of possible values of J consists of all $\alpha(v)$, where v is the designated node of J and α is a mapping that assigns to each node of v a value in $\mathbb{A}$, such that if v has label $\mathtt{f}$ and its outgoing edges lead to nodes $v_1, \ldots, v_k$ (in that order), then $((\alpha(v_1), \ldots, \alpha(v_k)), \alpha(v)) \in f$. For a set $\mathcal{J}$ of jungles, $\sigma(\mathcal{J}) = \bigcup_{J \in \mathcal{J}} \sigma(J)$.

In Section 5 we let each delegation network $\mathcal{N}$ generate a set of jungles, denoted $L_{\mathrm{J}}(\mathcal{N})$, and we show that $L(\mathcal{N}) = \pi(L_{\mathrm{J}}(\mathcal{N}))$, our main MW-like result. In fact, it is straightforward to turn an IO context-free tree grammar into a (context-free) graph grammar that generates jungles, by viewing each tree t in $L(\gamma_{\mathtt{g}})$ as a jungle $\mathrm{J}_k(t)$ that contains exactly one node with label x_i for every parameter symbol x_i (where $1 \leq i \leq k$ and k is the rank of $\mathtt{g}$). Thus, all nodes of t with label x_i are shared, and a garbage node with label x_i is created when x_i does not occur in t. Intuitively, this is in accordance with the call-by-value semantics of $\mathcal{N}$: whenever x_i is used, the same (nondeterministic) value of x_i must be used; moreover, every parameter x_i must have at least one possible value.

Jungles are defined in Section 4, not as DOAGs with labeled nodes, but as acyclic hypergraphs with labeled hyperedges (as in [HKP91]; see also [HP91, CR93, Plu99]). This simplifies the definition of $L_{\mathrm{J}}(\mathcal{N})$, which can then be generated by a so-called hyperedge-replacement graph grammar; see, e.g., [BC87, HK87, Hab92, Eng97, DHK97, CE12]. These (hyper)graph grammars are well known, well investigated and easy to understand.

In Section 6, we obtain further MW-like characterizations for the special case of deterministic primitives, i.e., primitives that are interpreted as functions.

- The delegation network $\mathcal{N}_{\mathrm{free}}$, which is $\mathcal{N}$ with π replaced by the free interpretation of symbols in Π, generates trees over Π. Our second MW-like result states that $L(\mathcal{N}) = \pi(L(\mathcal{N}_{\mathrm{free}}))$ for delegation networks with deterministic primitives (Theorem 6.2).
- Alternatively, trees can be generated operationally by viewing $\mathcal{N}$ as an (extended) IO context-free tree grammar. We show in Theorem 6.4 that

the language $L_{\mathrm{T}}(\mathcal{N})$ generated in this way is equal to $L(\mathcal{N}_{\mathrm{free}})$, which yields Theorem 6.6, our third MW-like characterization: $L(\mathcal{N}) = \pi(L_{\mathrm{T}}(\mathcal{N}))$, still under the assumption of deterministic primitives. This generalizes [ES78, Theorem 5.10 and Corollary 5.11].

- We also show, in Theorem 6.7, that these two MW-like characterizations hold in the presence of nondeterministic primitives, provided that $\mathcal{N}$ is linear and nondeleting, i.e., each parameter of $\mathtt{g}$ occurs exactly once in each tree of $L(\gamma_{\mathtt{g}})$.

Since the tree language $L_{\mathrm{T}}(\mathcal{N})$ generated by a delegation network $\mathcal{N}$ only depends on the tree languages $L(\gamma_{\mathtt{g}})$ ($\mathtt{g} \in \mathcal{G}$), delegation can be regarded as an operator DEL on classes of tree languages: for a class C, DEL(C) is the class of tree languages $L_{\mathrm{T}}(\mathcal{N})$ such that $L(\gamma_{\mathtt{g}}) \in C$ for every $\mathtt{g} \in \mathcal{G}$. In Sections 7 and 8, we study the hierarchy $\mathrm{DEL}^*(\mathrm{FIN}) = \bigcup_{n \geq 0} \mathrm{DEL}^n(\mathrm{FIN})$, which is obtained by starting with the class FIN of finite tree languages and iterating the delegation operator, similar to the IO-hierarchy studied in [Mai74] and [ES78, Section 7]. Clearly, $\mathrm{DEL}^*(\mathrm{FIN})$ is the smallest class of tree languages that contains FIN and is closed under DEL: if the tree generators of a delegation network $\mathcal{N}$ generate tree languages in that class, then so does $\mathcal{N}$.

In Section 7 we establish a result that can be used to prove that certain tree languages are not in $\mathrm{DEL}^*(\mathrm{FIN})$. As usual, we encode a path $(v_1, \ldots, v_n)$ in a tree t over Σ, from the root v_1 to some node v_n ($n \geq 1$), as the string $\langle \mathtt{f}_1, j_1 \rangle \cdots \langle \mathtt{f}_{n-1}, j_{n-1} \rangle \langle \mathtt{f}_n, 0 \rangle$ where $\mathtt{f}_i$ is the label of v_i in t, and v_{i+1} is the j_i-th child of v_i in t. The result states that the set of (encoded) paths in the trees of each tree language in $\mathrm{DEL}^*(\mathrm{FIN})$ is a context-free language (Theorem 7.23). The proof is long and complicated. It makes essential use of jungles, in particular of the fact that $L_{\mathrm{T}}(\mathcal{N}) = \mathrm{tree}(L_{\mathrm{J}}(\mathcal{N}))$, which is proved in Corollary 6.5.

In Section 8 we prove that $\mathrm{DEL}^*(\mathrm{FIN})$ is included in $\mathrm{MTT}^*(\mathrm{REG})$, the closure of the class of regular tree languages under nondeterministic macro tree transductions (Theorem 8.15), but not in the smaller class $\mathrm{YIELD}^*(\mathrm{REG})$ of tree languages of the IO-hierarchy (Theorem 8.17). By the result of Section 7, $\mathrm{DEL}^*(\mathrm{FIN})$ is properly included in the first class and incomparable with the second. Since the inclusion of $\mathrm{DEL}^*(\mathrm{FIN})$ in $\mathrm{MTT}^*(\mathrm{REG})$ is effective, we immediately obtain the decidability of the membership, emptiness and finiteness problems for every language in $\mathrm{DEL}^*(\mathrm{FIN})$, as shown for $\mathrm{MTT}^*(\mathrm{REG})$ in [EV85, DE98]. Finally, we prove that, in fact, $\mathrm{MTT}^*(\mathrm{REG})$ is closed under DEL (Theorem 8.19).

2 Delegation Networks

Before turning to the definition of delegation networks, which will be given later in this section, let us summarize some basic terminology and notation.

We denote the set of all natural numbers (including zero) by $\mathbb{N}$. For $n \in \mathbb{N}$, we let $[n]$ denote $\{1, \ldots, n\}$. By convention, this is the empty set if $n = 0$. The powerset of a set A is denoted by $\wp(A)$. The set of all finite strings (or sequences)

over A is denoted by A^*; λ denotes the empty string, and $A^+ = A^* \setminus \{\lambda\}$. For a string $u = a_1 \cdots a_k$ over A (where $a_1, \ldots, a_k \in A$), we let $[u] = \{a_1, \ldots, a_k\}$. In other words, $[u]$ is the smallest set $A' \subseteq A$ such that $u \in A'^*$. For sets A and strings u, the cardinality of A and the length of u are denoted by $|A|$ and $|u|$, respectively. For a string u as above and $i \in [k]$, a_i is also denoted by $u(i)$. More generally, for a function $f\colon S \to A^*$ and $s \in S$, if $f(s) = a_1 \cdots a_k$, then $f(s, i)$ denotes a_i, for all $i \in [k]$.

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 that are not necessarily total, are called partial functions. For a function $f\colon A \to B$, the canonical extensions of f to subsets of A and to strings over A are denoted by f as well, i.e., $f(A') = \{f(a) \mid a \in A'\}$ for all $A' \subseteq A$, and $f(a_1 \cdots a_k) = f(a_1) \cdots f(a_k)$ for $a_1, \ldots, a_k \in A$.

A binary relation $r \subseteq A \times B$ may alternatively be viewed as a function $r\colon A \to \wp(B)$ that maps elements of A to sets of elements of B. As usual, we let $r(a) = \{b \in B \mid (a, b) \in r\}$ for $a \in A$, and $r(A') = \bigcup_{a \in A'} r(a) = \{b \in B \mid \exists a \in A'\colon (a, b) \in r\}$ for $A' \subseteq A$. Note that this is consistent with the definition of $f(A')$ above (where we identify a function $f\colon A \to B$ with the relation $\{(a, f(a)) \mid a \in A\}$, as usual). Observe that A can be a cartesian product $A_1 \times \cdots \times A_k$, i.e., r may be of the form $r \subseteq (A_1 \times \cdots \times A_k) \times B$. In that case, we let $r(A'_1, \ldots, A'_k) = r(A'_1 \times \cdots \times A'_k)$ for $A'_1 \subseteq A_1, \ldots, A'_k \subseteq A_k$. If $k = 0$, we identify r with the set $r() \subseteq B$. As usual, the composition of binary relations $r \subseteq A \times B$ and $s \subseteq B \times C$ is given by $s \circ r = \{(a, c) \in A \times C \mid c \in s(r(a))\}$.

2.1 Signatures and Trees

A *signature* (or ranked alphabet) is a pair (Σ, rk), where Σ is a set of symbols, and rk assigns to every $\mathtt{f} \in \Sigma$ a *rank* $rk(\mathtt{f}) \in \mathbb{N}$. In the following, we shall simply denote (Σ, rk) by Σ. If necessary, the rank k of a symbol $\mathtt{f}$ is indicated by denoting $\mathtt{f}$ as $\mathtt{f}^{(k)}$. Symbols of rank 0 are called *constant symbols*.

The set of all *trees over* Σ is denoted by T_Σ. By definition, it is the smallest set of strings satisfying the following condition:

> For all $\mathtt{f}^{(k)}$ in Σ ($k \in \mathbb{N}$) and all $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$, the string $\mathtt{f}[t_1, \ldots, t_k]$ is in T_Σ. (Here, the square brackets and the comma are assumed to be special symbols not in Σ.)

A tree of the form $\mathtt{f}[\,]$ (i.e., where $\mathtt{f}$ is a constant symbol in Σ) is identified with the string $\mathtt{f}$ of length 1. By this convention, all constant symbols are trees in T_Σ.

A tree $t \in \mathrm{T}_\Sigma$ can be viewed as a graph whose nodes are labelled with symbols in Σ. A node is a string in $(\mathbb{N} \setminus \{0\})^*$ which, intuitively, represents the path from the root to the node in question. 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 that node 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 $\ell_t(iv) = \ell_{t_i}(v)$. The node λ is the *root* of t. As usual, a tree s is a *subtree* of t if $s = t/v$ for some $v \in V(t)$. A node v of t is said to be an *occurrence* of the symbol $\ell_t(v)$ and of the subtree t/v. If $t = \mathtt{f}[t_1, ..., t_k]$, then the subtrees $t_1, \ldots, t_k$ are also called the subtrees of (this particular occurrence of) the symbol $\mathtt{f}$. We will not always distinguish precisely between a symbol (or subtree) and one of its occurrences; this should be clear from the context.

For a signature Σ', we let $V_{\Sigma'}(t) = \{v \in V(t) \mid \ell_t(v) \in \Sigma'\}$. We say that t is *linear in* Σ' if each symbol in Σ' occurs at most once in t and *nondeleting in* Σ' if each symbol in Σ' occurs at least once in t (formally, $|V_{\{\sigma\}}(t)| \leq 1$, respectively $|V_{\{\sigma\}}(t)| \geq 1$, for every $\sigma \in \Sigma'$).

For finite Σ, a subset of T_Σ is a *tree language*. Any device γ that generates such a tree language is a *tree generator*; Σ is said to be its *output signature*, and the generated tree language is denoted by $L(\gamma)$.

2.2 Interpretations

Let $\mathbb{A}$ be a *domain*, i.e., a set. We wish to be able to interpret every symbol of a signature Σ, and to use this interpretation for evaluating trees 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, as mentioned in the Introduction, we generalize this in order to be able to model nondeterminism[2]. For this purpose, symbols are interpreted as relations $f \subseteq \mathbb{A}^k \times \mathbb{A}$ rather than as functions. One may view such a relation f as a "nondeterministic function" $f\colon \mathbb{A}^k \to \wp(\mathbb{A})$ that, for an argument tuple in $\mathbb{A}^k$, yields the set of possible results. Of course, functions and partial functions are special cases.

A Σ-*interpretation (into* $\mathbb{A}$*)* is a function σ that assigns to every symbol $\mathtt{f}^{(k)} \in \Sigma$ a relation $\sigma(\mathtt{f}) \subseteq \mathbb{A}^k \times \mathbb{A}$. If $\sigma(\mathtt{f})$ is a function for all $\mathtt{f} \in \Sigma$, then we call σ a *deterministic* Σ-*interpretation*; note that this implies totality of $\sigma(\mathtt{f})$. Throughout the rest of the paper, we will use the typographical convention that the interpretation $\sigma(\mathtt{f})$ of a symbol $\mathtt{f}$ is denoted by f. Thus, if $\mathtt{f}^{(k)} \in \Sigma$, then $f \subseteq \mathbb{A}^k \times \mathbb{A}$ (and $f\colon \mathbb{A}^k \to \mathbb{A}$ if σ is deterministic).

In the following, we want to represent complex operations by trees. For this purpose, let us define a *parameter signature of rank* $n \in \mathbb{N}$ to be a signature $Y = \{y_1, \ldots, y_n\}$ consisting of pairwise distinct indexed symbols $y_1, \ldots, y_n$, where y_i has rank 0 for every $i \in [n]$. Note that the indices provide Y with a linear order. For a signature Σ disjoint with Y, the set $\mathrm{T}_{\Sigma \cup Y}$ is denoted by $\mathrm{T}_{\Sigma,Y}$ to emphasize the special role of parameter symbols. For trees $t, t_1, \ldots, t_n \in \mathrm{T}_{\Sigma,Y}$, we denote by $t(t_1, \ldots, t_n)$ the substitution of t_i for each occurrence of y_i in t ($i \in [n]$). (In this notation, the parameter signature in question will always be clear from the context.)

Given a Σ-interpretation σ and a parameter signature Y as above, we can

[2]See also [ES77, ES78].

evaluate trees t in $\mathrm{T}_{\Sigma,Y}$. The result of this evaluation, denoted by $\sigma_Y(t)$, is the relation $\varphi \subseteq \mathbb{A}^n \times \mathbb{A}$ such that $\varphi(a_1, \ldots, a_n)$ is given as follows, for all $a_1, \ldots, a_n \in \mathbb{A}$:

- If $t = y_i$, then $\varphi(a_1, \ldots, a_n) = \{a_i\}$.
- Otherwise, if $t = \mathtt{f}[t_1, \ldots, t_k]$ with $\varphi_i = \sigma_Y(t_i)$ for all $i \in [k]$, then

$$\varphi(a_1, \ldots, a_n) = f(\varphi_1(a_1, \ldots, a_n), \ldots, \varphi_k(a_1, \ldots, a_n)).$$

If the parameter signature Y is clear from the context, we omit the subscript, thus denoting $\sigma_Y(t)$ by $\sigma(t)$. Note that $\sigma(t)$ is a function from $\mathbb{A}^n$ to $\mathbb{A}$ if σ is deterministic, and that it is a subset of $\mathbb{A}$ if $n = 0$ (i.e., $Y = \emptyset$ and $t \in \mathrm{T}_\Sigma$). In particular, if σ is deterministic and $n = 0$, then $\sigma(t) \in \mathbb{A}$.

Given a set of trees $T \subseteq \mathrm{T}_{\Sigma,Y}$ rather than a single tree, we let $\sigma(T) = \bigcup_{t \in T} \sigma(t)$. We note here that in [ES77, ES78], σ is called a nondeterministic Σ-algebra, and $\sigma(T)$ is called the derived relation of T over σ (see [ES78, Definition 5.8]).

A well-known possibility is to choose as σ the (deterministic) *free interpretation* free_Σ of symbols in Σ. For this, let $\mathbb{A} = \mathrm{T}_\Sigma$, and let $f = \mathit{free}_\Sigma(\mathtt{f})$ be the function given by $f(t_1, \ldots, t_k) = \mathtt{f}[t_1, \ldots, t_k]$, for every symbol $\mathtt{f}^{(k)} \in \Sigma$ and all trees $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$. Thus, free_Σ is the identity on T_Σ. More generally, for a parameter signature Y of rank n, if $t \in \mathrm{T}_{\Sigma,Y}$ and $t_1, \ldots, t_n \in \mathrm{T}_\Sigma$, then $\mathit{free}_\Sigma(t)(t_1, \ldots, t_n) = t(t_1, \ldots, t_n)$.

Lemma 2.1 Let σ be a Σ-interpretation, for a signature Σ, and let Y be a parameter signature of rank n. For all trees $t \in \mathrm{T}_{\Sigma,Y}$ and $t_1, \ldots, t_n \in \mathrm{T}_\Sigma$, if $|\sigma(t_i)| = 1$ for all $i \in [n]$, then

$$\sigma(t(t_1, \ldots, t_n)) = \sigma(t)(\sigma(t_1), \ldots, \sigma(t_n)).$$

Proof This follows easily from the definition of $\sigma(t)$, using induction on t. Obviously, the equation holds if $t = y_i$ for some $i \in [n]$. Further, if $t = \mathtt{f}[s_1, \ldots, s_k]$, and $a_i = \sigma(t_i)$ for every $i \in [n]$, we get

$$\begin{aligned}
\sigma(t)(a_1, \ldots, a_n) &= f(\sigma(s_1)(a_1, \ldots, a_n), \ldots, \sigma(s_k)(a_1, \ldots, a_n)) \\
&= f(\sigma(s_1(t_1, \ldots, t_n)), \ldots, \sigma(s_k(t_1, \ldots, t_n))) \\
&= \sigma(f[s_1(t_1, \ldots, t_n), \ldots, s_k(t_1, \ldots, t_n)]) \\
&= \sigma(t(t_1, \ldots, t_n)),
\end{aligned}$$

as claimed. ∎

Note that the first equality of the display in the proof above does not hold without the requirement that $|\sigma(t_i)| = 1$: with $A_i = \sigma(t_i) \subseteq \mathbb{A}$, we get

$$\begin{aligned}
\sigma(t)(A_1, \ldots, A_n) &= \textstyle\bigcup_{a_i \in A_i} \sigma(t)(a_1, \ldots, a_n) \\
&= \textstyle\bigcup_{a_i \in A_i} f(\sigma(s_1)(a_1, \ldots, a_n), \ldots, \sigma(s_k)(a_1, \ldots, a_n)),
\end{aligned}$$

in contrast to the fact that

$$\begin{aligned} & f(\sigma(s_1)(A_1,\ldots,A_n),\ldots,\sigma(s_k)(A_1,\ldots,A_n)) \\ & \quad = f(\textstyle\bigcup_{a_i\in A_i}\sigma(s_1)(a_1,\ldots,a_n),\ldots,\bigcup_{a_i\in A_i}\sigma(s_k)(a_1,\ldots,a_n)) \\ & \quad = \textstyle\bigcup_{a_i^j\in A_i} f(\sigma(s_1)(a_1^1,\ldots,a_n^1),\ldots,\sigma(s_k)(a_1^k,\ldots,a_n^k)). \end{aligned}$$

This difference is what makes the usual MW-like result fail in the presence of nondeterministic interpretations.

2.3 Delegation Networks and Their Semantics

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

Definition 2.2 (delegation network) A *delegation network* is a system $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$, where

- $\mathcal{G}$ and Π are disjoint finite signatures of *generator symbols* and *primitives*, resp.,
- $\Gamma = (\gamma_{\mathtt{g}}, Y_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}$ is a $\mathcal{G}$-indexed family of pairs, each consisting of a tree generator $\gamma_{\mathtt{g}}$ and a parameter signature $Y_{\mathtt{g}}$ of the same rank as $\mathtt{g}$,
- $\mathtt{g}_0$ is a constant symbol in $\mathcal{G}$,
- $\mathbb{A}$ is a domain, and
- π is a Π-interpretation into $\mathbb{A}$.

We require that each parameter signature $Y_{\mathtt{g}}$, $\mathtt{g}\in\mathcal{G}$, is disjoint with $\mathcal{G}\cup\Pi$, and that $L(\gamma_{\mathtt{g}}) \subseteq \mathrm{T}_{\mathcal{G}\cup\Pi, Y_{\mathtt{g}}}$. The signature $\mathcal{G}\cup\Pi$ is denoted by $\Sigma_{\mathcal{N}}$.

By the definition above, $\Sigma_{\mathcal{N}}$ contains two kinds of symbols: the generator symbols $\mathtt{g}\in\mathcal{G}$, each of which has an associated tree generator $\gamma_{\mathtt{g}}$, and the primitives $\mathtt{f}\in\Pi$. In the context of the particular delegation network $\mathcal{N}$, the latter are given a fixed interpretation by means of π. Following our general convention, we shall therefore generally denote the interpretation $\pi(\mathtt{f})$ of such a symbol by f.

The semantics of $\mathcal{N}$ is obtained by constructing a $\Sigma_{\mathcal{N}}$-interpretation $\sigma_{\mathcal{N}}$ that agrees with π on Π. To take into account the fact that delegation can be cyclic, we choose a least fixed point semantics, using Tarski's fixed point theorem.

Proposition 2.3 ([Tar55, Theorem 1]) Let C be a complete lattice with partial order $\leq$. Every monotonically increasing mapping τ on C has a least fixed point, and this least fixed point is equal to the greatest lower bound of all $c\in C$ such that $\tau(c)\leq c$.

To continue, recall that the set of all relations $r\subseteq\mathbb{A}^k$ becomes a complete lattice if we choose $\subseteq$ as its partial order. Its least element is the empty relation, and its largest is $\mathbb{A}^k$. We extend this ordering to $\Sigma_{\mathcal{N}}$-interpretations σ,σ' in the obvious way: $\sigma\leq\sigma'$ if and only if $\sigma(\mathtt{f})\subseteq\sigma'(\mathtt{f})$ for all $\mathtt{f}\in\Sigma_{\mathcal{N}}$.

We can now define the semantics of delegation networks. According to this semantics, a delegation network generates a subset of $\mathbb{A}$. Typically, the reader may think of generating strings, trees, graphs, pictures, etc. In contexts where

the grammatical generation of sets of such objects is studied, they are sometimes called languages, generalizing the terminology commonly used in the string case. Here, we follow this tradition.

Definition 2.4 (semantics of delegation networks) Let $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$ be a delegation network, where $\Gamma = (\gamma_\mathtt{g}, Y_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}$.

1. The mapping $\mathrm{iterate}_\mathcal{N}$ on the complete lattice of all $\Sigma_\mathcal{N}$-interpretations into $\mathbb{A}$ is defined as follows: for every $\Sigma_\mathcal{N}$-interpretation σ, and every symbol $\mathtt{f} \in \Sigma_\mathcal{N}$,
$$\mathrm{iterate}_\mathcal{N}(\sigma)(\mathtt{f}) = \begin{cases} \sigma_{Y_\mathtt{f}}(L(\gamma_\mathtt{f})) & \text{if } \mathtt{f} \in \mathcal{G} \\ f & \text{if } \mathtt{f} \in \Pi. \end{cases}$$
2. The least fixed point of $\mathrm{iterate}_\mathcal{N}$ is denoted by $\sigma_\mathcal{N}$. (Note that by Proposition 2.3, $\sigma_\mathcal{N}$ exists, as it is straightforward to verify that $\mathrm{iterate}_\mathcal{N}$ is monotonically increasing.)
3. The *language generated by* $\mathcal{N}$ is $L(\mathcal{N}) = \sigma_\mathcal{N}(\mathtt{g}_0)$.

Note that the language generated by $\mathcal{N}$ is a subset of $\mathbb{A}$, because the rank of $\mathtt{g}_0$ is 0. Observe furthermore that, for $\mathtt{f} \in \Pi$, we have $\sigma_\mathcal{N}(\mathtt{f}) = f$. As a consequence, for all $\mathtt{g} \in \mathcal{G}$ with $L(\gamma_\mathtt{g}) \subseteq \mathrm{T}_\Pi$, $\sigma_\mathcal{N}(\mathtt{g})$ is just $\pi(L(\gamma_\mathtt{g}))$. This shows that delegation networks can be seen as a generalization of the tree-based generators in [Dre06]: a tree-based generator is a delegation network $\mathcal{N}$ with $\mathcal{G} = \{\mathtt{g}\}$, where $\gamma_\mathtt{g}$ has the output signature Π (i.e., does not delegate to itself), and $\pi(\mathtt{f})$ is a function for all $\mathtt{f} \in \Pi$.

Another obvious fact that nevertheless may be worth pointing out is that the semantics of delegation networks does not depend on the particular devices $\gamma_\mathtt{g}$ that generate the tree languages $L(\gamma_\mathtt{g})$. In fact, delegation networks are closely related to the systems of equations over nondeterministic Π-algebras studied in [ES78]. If we restrict delegation networks to the *finitary* case, i.e., require that all $L(\gamma_\mathtt{g})$ ($\mathtt{g} \in \mathcal{G}$) are finite, then the syntactic part $(\mathcal{G}, \Pi, \Gamma)$ of a delegation network is what is called a system of context-free Π-equations in [ES78, Definition 5.1][3], and the least fixed point $\sigma_\mathcal{N}$ is the solution of that system in the algebra of relations over π (see [ES78, Definition 5.3]). Furthermore, for every $\mathtt{g} \in \mathcal{G}$, $\sigma_\mathcal{N}(\mathtt{g})$ is what is proposed to be called the call by value relation computed by $(\mathcal{G}, \Pi, \Gamma, \mathtt{g})$ over π in the discussion after Corollary 5.7 in [ES78]. In the MW-like result [ES78, Theorem 5.10] it is shown that this relation is uniquely determined by the tree language generated by $\mathtt{g}$ in the IO context-free tree grammar $(\mathcal{G}, \Pi, \Gamma)$. Note that, in the finitary case, there is no reason at all to distinguish between $\gamma_\mathtt{g}$ and $L(\gamma_\mathtt{g})$, because finite tree languages can be specified by simply listing their elements. In this sense, a finite tree language is its own generator.

A simplification that can be made in large parts of the paper and, in particular, throughout Sections 3–6 and 8, concerns the parameter signatures used by the

[3]It is just another, more suggestive, name for a context-free tree grammar with terminal signature Π.

tree generators γ_{g}. Unless the contrary is clear from the context, we assume that Y_{g} is the standard parameter signature $X_k = \{x_1, \ldots, x_k\}$, where k is the rank of g. In this case, Y_{g} will not be specified explicitly, i.e., we let $\Gamma = (\gamma_{\mathsf{g}})_{\mathsf{g} \in \mathcal{G}}$. Moreover, we abbreviate T_{Σ, X_k} as $\mathrm{T}_{\Sigma,k}$. Thus, $L(\gamma_{\mathsf{g}}) \subseteq \mathrm{T}_{\Sigma_{\mathcal{N}},k}$. More generally, we will assume that in an expression $t(t_1, \ldots, t_n)$ the parameter signature X_n is used, unless mentioned otherwise.

We will also make use of special parameter symbols $\Box_1, \Box_2, \ldots$, where $\Box$ abbreviates $\Box_1$, mainly for the purpose of decomposing trees. To avoid confusion when using both the parameter signatures X_n and $\{\Box_1, \ldots, \Box_n\}$, we denote an expression of the form $t(t_1, \ldots, t_n)$ by $t[\![t_1, \ldots, t_n]\!]$ if it is to be evaluated with respect to the parameter signature $\{\Box_1, \ldots, \Box_n\}$. If, moreover, t is linear and nondeleting in $\{\Box_1, \ldots, \Box_n\}$, then we denote by $t \cdot (t_1, \ldots, t_n)$ the *concatenation of t and $t_1, \ldots, t_n$* given by $t \cdot (t_1, \ldots, t_n) = t[\![t_1, \ldots, t_n]\!]$. Thus, denoting a tree s in the form $t \cdot (t_1, \ldots, t_n)$ indicates that (a) $s = t(t_1, \ldots, t_n)$ where $\{\Box_1, \ldots, \Box_n\}$ is assumed to be the relevant parameter signature and (b) t is assumed to be linear and nondeleting in $\{\Box_1, \ldots, \Box_n\}$. Hence, each t_i corresponds to a unique occurrence of t_i as a subtree of s. In particular, in the case where $n = 1$, $s = t \cdot t_1$ is a tree obtained from t by replacing the unique occurrence of $\Box$ in t with t_1.

We conclude this section with a series of small examples that illustrate some of the properties of delegation networks. Bigger examples that, moreover, discuss the many-sorted case of delegation networks, can be found in [Dre07b]. In the examples below, but also in Section 8, regular tree grammars will be used as generators.

Definition 2.5 (regular tree grammar) A *regular tree grammar* is a tuple $\gamma = (\Xi, \Sigma, R, \xi_0)$, where

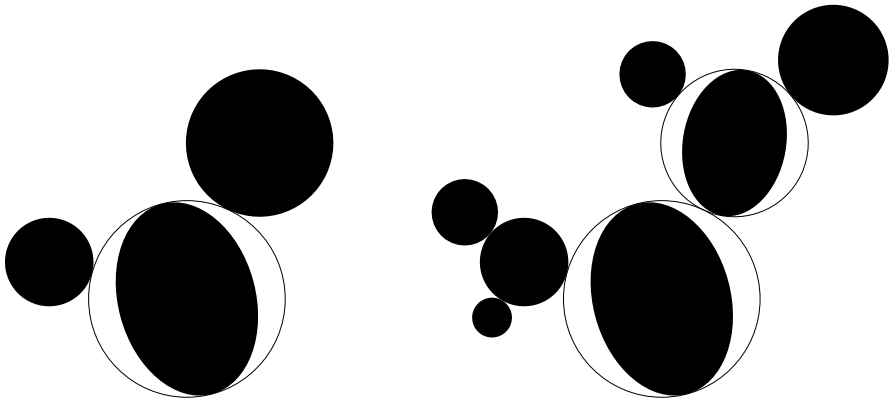
- Ξ is a finite signature of *nonterminals* of rank 0,
- Σ is a finite signature of *terminals* disjoint with Ξ,
- R is a finite set of *rules* $\xi \to r$, where $\xi \in \Xi$ and $r \in \mathrm{T}_{\Xi \cup \Sigma}$, and
- $\xi_0 \in \Xi$ is the *initial nonterminal*.

A *derivation step* $s \Rightarrow_\gamma t$ (or simply $s \Rightarrow t$, if γ is understood) consists of two trees $s, t \in \mathrm{T}_{\Xi \cup \Sigma}$ such that t is obtained from s by replacing a single occurrence of a nonterminal ξ with r, where the rule $\xi \to r$ belongs to R. The tree language generated by γ, called a *regular tree language*, is

$$L(\gamma) = \{t \in \mathrm{T}_\Sigma \mid \xi_0 \Rightarrow^* t\}.$$

The class of all regular tree languages is denoted by REG.

Example 2.6 In this example and those that follow, we interpret primitives as picture operations in the sense of [Dre00, Dre01, Dre06]. Here, a *picture* is a nonempty subset of the Euclidean plane, typically being a union of objects such as squares and circles. In other words, $\mathbb{A}$ is the set $\mathbb{P}$ of all nonempty subsets of $\mathbb{R}^2$. A picture operation of arity k is a function $f \colon \mathbb{P}^k \to \mathbb{P}$ which is given by a picture P_0 and affine transformations $\tau_1, \ldots, \tau_k$ of $\mathbb{R}^2$, where $f(P_1, \ldots, P_k) = P_0 \cup \bigcup_{i=1}^k \tau_i(P_i)$.



$f(r(\bullet), \bullet, \bullet)$ $\qquad$ $f(r(\bullet), f(\bullet, \bullet, \bullet), f(r(\bullet), \bullet, \bullet))$

Figure 1: Two expressions and the pictures they yield

Let $\Pi = \{\mathtt{f}^{(3)}, \mathtt{r}^{(1)}, \mathtt{d}^{(0)}\}$, and define a deterministic interpretation π as follows.

- $\pi(\mathtt{d}) = \bullet$ is the picture consisting of the unit disk centered at the origin.
- $\pi(\mathtt{r}) = r$ scales its argument horizontally by a factor of 0.7, rotates it by 15 degrees[4], and adds a unit circle.
- $\pi(\mathtt{f}) = f$ keeps the first argument unchanged. The second argument is uniformly scaled by the factor 0.45, translated vertically by 1.45 units, and rotated by 75 degrees. The third argument is transformed in a similar way, except that the scaling factor is 0.75, the vertical translation is by 1.75 units, and the rotation is by -25 degrees.

Two pictures that can be obtained from expressions over these operations are shown in Figure 1. In both pictures, the largest circle is the unit circle whose center is the origin of the coordinate system.

(a) Now, consider the delegation network $\mathcal{N} = (\{\mathtt{g}_0^{(0)}, \mathtt{h}^{(0)}\}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$, where $\gamma_{\mathtt{g}_0}$ and $\gamma_{\mathtt{h}}$ are the regular tree grammars with nonterminal signature $\{\xi_0\}$, terminal signature Π, and sets of rules $R_{\mathtt{g}_0}$ and $R_{\mathtt{h}}$, respectively, where

$$R_{\mathtt{g}_0} = \{\xi_0 \to \mathtt{f}[\mathtt{h}, \xi_0, \xi_0] \mid \mathtt{h}\} \text{ and } R_{\mathtt{h}} = \{\xi_0 \to \mathtt{r}[\xi_0] \mid \mathtt{d}\}.^5$$

Thus, $L(\gamma_{\mathtt{h}}) = \mathrm{T}_{\{\mathtt{r},\mathtt{d}\}}$, i.e., it consists of all trees of the form $\mathtt{r}[\cdots \mathtt{r}[\mathtt{d}] \cdots]$, which means that $\sigma_{\mathcal{N}}(\mathtt{h}) = \pi(L(\gamma_{\mathtt{h}}))$. The fact that $\mathtt{h}$ occurs in the trees of $L(\gamma_{\mathtt{g}_0})$, means that the generator $\gamma_{\mathtt{g}_0}$ delegates parts of its task to the generator $\gamma_{\mathtt{h}}$. As a consequence, the elements of $L(\mathcal{N})$ are the pictures that are obtained by interpreting the trees in $L(\gamma_{\mathtt{g}_0})$ with respect to π', the extension of π to $\Pi \cup \{\mathtt{h}\}$ given by $\pi'(\mathtt{h}) = \pi(\mathrm{T}_{\{\mathtt{r},\mathtt{d}\}})$. One of the elements of $L(\mathcal{N})$ is shown in Figure 2.

(b) Alternatively, $L(\mathcal{N}) = L(\mathcal{N}')$, where $\mathcal{N}' = (\{\mathtt{g}_0^{(0)}\}, \Pi, \Gamma', \mathtt{g}_0, \mathbb{A}, \pi)$ is the tree-based generator[6] given by the regular tree grammar $\gamma'_{\mathtt{g}_0} = (\{\xi_0, \xi\}, \Pi, R, \xi_0)$

[4]By convention, positive rotation direction means counterclockwise rotation.

[5]As usual, $\xi \to t_1 \mid \cdots \mid t_n$ abbreviates n rules $\xi \to t_1, \ldots, \xi \to t_n$.

[6]Cf. the remark following Definition 2.4.

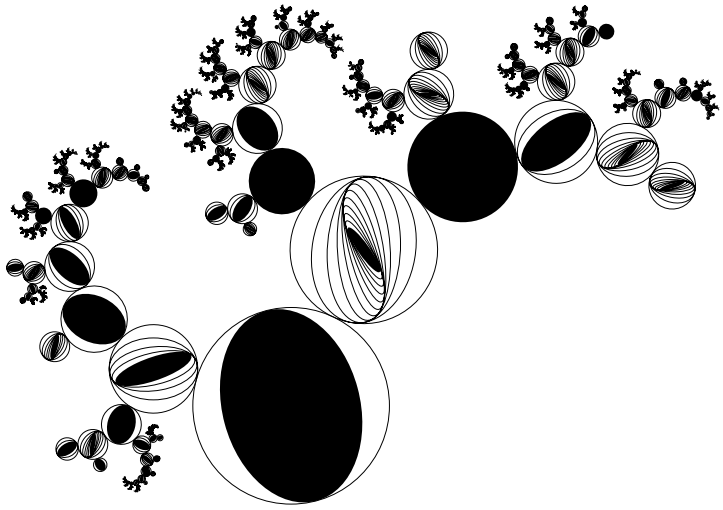


Figure 2: A picture generated by the delegation network in Example 2.6

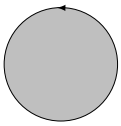
with

$$R = \{\xi_0 \to \mathtt{f}[\xi, \xi_0, \xi_0] \mid \xi,\ \xi \to \mathtt{r}[\xi] \mid \mathtt{d}\}.$$

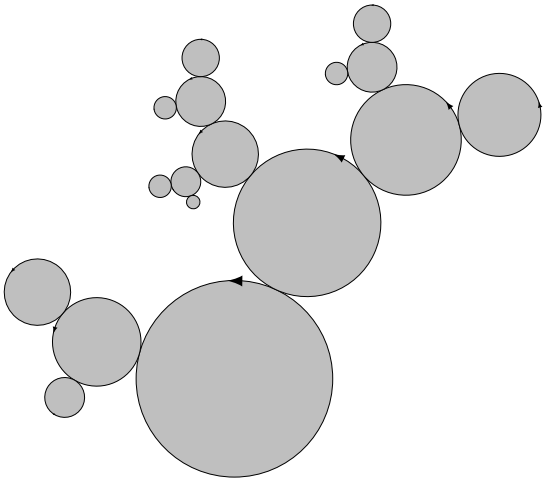
(c) Instead of removing all the delegation from $\mathcal{N}$, as in $\mathcal{N}'$, we could also use the finitary delegation network that is identical to $\mathcal{N}$, except that $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{f}[\mathtt{h}, \mathtt{g}_0, \mathtt{g}_0], \mathtt{h}\}$ and $L(\gamma_{\mathtt{h}}) = \{\mathtt{r}[\mathtt{h}], \mathtt{d}\}$. In this way, we generate the same picture language $L(\mathcal{N})$ using self-delegation, i.e., the trees in $L(\gamma_{\mathtt{g}_0})$ contain $\mathtt{g}_0$, and similarly for $L(\gamma_{\mathtt{h}})$.

Example 2.7 We now modify the delegation network of Example 2.6 such that $\gamma_{\mathtt{g}_0}$ delegates additionally to $\gamma_{\mathtt{g}}$, where $\mathtt{g}$ is of rank 1. Following our convention regarding parameter signatures, $Y_{\mathtt{g}} = X_1 = \{x_1\}$. Throughout the example, $L(\gamma_{\mathtt{h}})$ is as in Example 2.6.

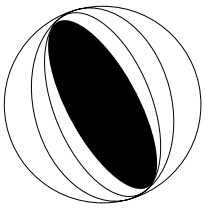
(a) Let $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{h}]\}$, and let $L(\gamma_{\mathtt{g}})$ be given by the regular tree grammar whose rules are $\xi_0 \to \mathtt{f}[x_1, \xi_0, \xi_0] \mid x_1$. Thus, $L(\gamma_{\mathtt{g}})$ is the set of all trees $t \in \mathrm{T}_{\{\mathtt{f}\},1} = \mathrm{T}_{\{\mathtt{f},x_1\}}$ such that the first child of every occurrence of $\mathtt{f}$ is x_1. The interpretation of such a tree t with respect to π yields a (deterministic) unary operation on pictures that duplicates the argument as many times as there are occurrences of x_1 in t and puts them into the corresponding places. Interpreting x_1 as the (for illustration purposes shaded and oriented) unit disk



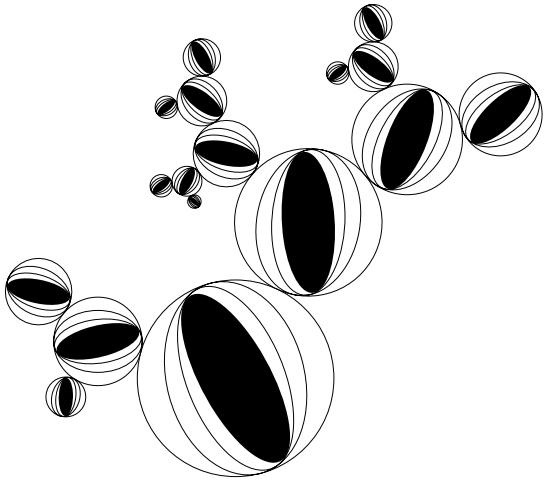
rather than as a parameter, we obtain a graphical representation of the operation $\sigma_{\mathcal{N}}(t) = \pi(t)$ and its application to an element of $\sigma_{\mathcal{N}}(\mathtt{h}) = \pi(L(\gamma_{\mathtt{h}}))$, as shown in Figure 3. The interpretation $\sigma_{\mathcal{N}}(\mathtt{g})$ of $\mathtt{g}$ in $\mathcal{N}$ is the nondeterministic unary



sample operation $\pi(t) = \sigma_{\mathcal{N}}(t)$



sample argument in $\sigma_{\mathcal{N}}(\mathtt{h})$



result in $\sigma_{\mathcal{N}}(\mathtt{g}[\mathtt{h}]) = \sigma_{\mathcal{N}}(\mathtt{g}_0) = L(\mathcal{N})$

Figure 3: Graphical representation of applying $\pi(t)$ to an element of $\sigma_{\mathcal{N}}(\mathtt{h})$ in Example 2.7(a)

operation g such that, for a picture P, $g(P) = \{\pi(t)(P) \mid t \in L(\gamma_{\mathtt{g}})\}$. Thus, $L(\mathcal{N}) = \{g(P) \mid P \in \pi(L(\gamma_{\mathtt{h}}))\}$.

It may be instructive to have a look at the relation between $L(\mathcal{N})$ and the tree language $L(\mathcal{N}_{\text{free}})$, where $\mathcal{N}_{\text{free}}$ is identical to $\mathcal{N}$, except that its interpretation π of primitives is the free interpretation $free_{\Pi}$ with domain T_{Π}. Thus, each tree in $L(\mathcal{N}_{\text{free}})$ is obtained from a tree in $L(\gamma_{\mathtt{g}})$ by substituting a tree in $L(\gamma_{\mathtt{h}}) = \mathrm{T}_{\{\mathtt{r},\mathtt{d}\}}$ for the occurrences of the parameter x_1: $L(\mathcal{N}_{\text{free}}) = \{t(t_1) \mid t \in L(\gamma_{\mathtt{g}}), t_1 \in \mathrm{T}_{\{\mathtt{r},\mathtt{d}\}}\}$. The fact that, in every picture in $L(\mathcal{N})$, the inner decorations of all circles are identical, corresponds to the fact that, in every tree in $L(\mathcal{N}_{\text{free}})$, the first subtrees of all occurrences of $\mathtt{f}$ are identical. By the MW-like Theorem 6.2 in Section 6, it holds that $L(\mathcal{N}) = \pi(L(\mathcal{N}_{\text{free}}))$. However, this relies on the fact that the interpretations of the primitives are deterministic, i.e., they are functions. Suppose that we change $\mathcal{N}$ into $\mathcal{N}'$ by removing $\mathtt{h}$ from the signature of generator symbols and turning it into a nondeterministic primitive with $\pi'(\mathtt{h}) = \sigma_{\mathcal{N}}(\mathtt{h})$. From Definition 2.4, it is immediately clear that $L(\mathcal{N}') = L(\mathcal{N})$. In contrast, $\pi'(L(\mathcal{N}'_{\text{free}}))$ is equal to the picture language generated in Example 2.6. This discrepancy is caused by the fact that the trees in $L(\mathcal{N}'_{\text{free}})$ are of the form $t(\mathtt{h})$ with $t \in L(\gamma_{\mathtt{g}})$, in which $\mathtt{h}$ has several occurrences. When the interpretation π' is applied to $t(\mathtt{h})$, these occurrences of $\mathtt{h}$ are evaluated individually. Thus, each occurrence of $\mathtt{h}$ gives nondeterministically rise to a potentially different element of $\pi'(\mathtt{h})$. We will show in Section 5 how this can be avoided by generating and evaluating jungles instead of trees. In a jungle generated by $\mathcal{N}'$, there will be only one occurrence of $\mathtt{h}$, which is shared. Hence, it is evaluated only once, which makes the evaluation of the entire jungle return the desired result.

Using self-delegation as in Example 2.6(c), $\mathcal{N}$ can be turned into a finitary delegation network that generates the same picture language $L(\mathcal{N})$, by setting $L(\gamma_{\mathtt{g}}) = \{\mathtt{f}[x_1, \mathtt{g}[x_1], \mathtt{g}[x_1]], x_1\}$. Then $\mathcal{N}_{\text{free}}$ can be viewed as an IO context-free tree grammar that generates the language $L(\mathcal{N}_{\text{free}})$ using the rules $\mathtt{g}_0 \to \mathtt{g}[\mathtt{h}]$, $\mathtt{g}[x_1] \to \mathtt{f}[x_1, \mathtt{g}[x_1], \mathtt{g}[x_1]]$, $\mathtt{g}[x_1] \to x_1$, $\mathtt{h} \to r[\mathtt{h}]$ and $\mathtt{h} \to \mathtt{d}$. Similarly, a grammar for $L(\mathcal{N}')$ is obtained by dropping the last two rules.

(b) A modification of $L(\mathcal{N})$ is obtained by taking as $\gamma_{\mathtt{g}}$ the regular tree grammar whose rules are $\xi_0 \to \mathtt{f}[x_1, \mathtt{g}_0, \xi_0] \mid x_1$. This is an example of cyclic delegation (similar to mutual recursion in programs), because $\gamma_{\mathtt{g}_0}$ delegates to $\gamma_{\mathtt{g}}$ and vice versa. The change implies that, while the inner decorations of the major spine of the picture (extending to the right) are still identical, the subpictures that correspond to the second subtrees of occurrences of $\mathtt{f}$ are recursively obtained (transformed versions of) arbitrary elements of $\sigma_{\mathcal{N}}(\mathtt{g}_0) = L(\mathcal{N})$, thus differing with respect to their decorations. One of these pictures is shown in Figure 4.

(c) A variant of (b) with a slightly different domain is the following one. We use colored pictures[7] and change the interpretation of $\mathtt{r}$. It is now interpreted

[7]We omit a formal definition, since it is not needed to understand the example. See [Dre06, Chapter 8] for a discussion of color operations.

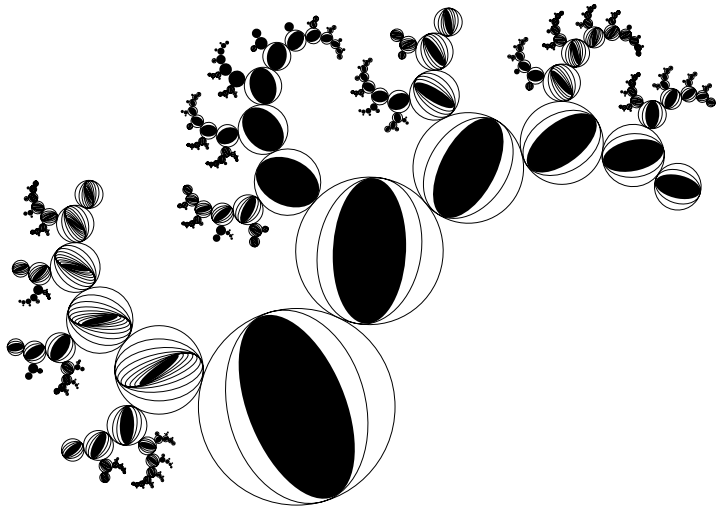


Figure 4: A picture generated by the delegation network in Example 2.7(b)

as a nondeterministic primitive that, when being applied to a picture, changes the color of each point in the picture by increasing the intensity of any of the channels *red*, *green*, and *blue* by a certain fixed amount. The color of $\pi(\mathtt{d}) = \bullet$ is black (i.e., the value of all channels is zero). Thus, $r^n(\bullet)$ yields unit disks in $(n+1)^3$ different colors.

Apart from this change of π, the delegation network is as in (b). The operation f is assumed to preserve color. Thus, similar to (b) the major spine of the picture is uniformly colored, whereas the subpictures that correspond to the second subtrees of occurrences of $\mathtt{f}$ are, recursively, colored in the same fashion, but with their individual coloring. One of these pictures can be seen in Figure 5.

Example 2.8 A phenomenon that must be taken into account when proving MW-like results is deletion. As an (artificial) example, let us modify the delegation network $\mathcal{N}$ of Example 2.6(a) in the following way, yielding $\mathcal{N}^\perp$. The generator symbol $\mathtt{h}$ is turned into a symbol of rank 1. In the rules of $\gamma_{\mathtt{g}_0}$, every occurrence of $\mathtt{h}$ is replaced by $\mathtt{h}[\perp]$, where $\perp$ is a new primitive of rank 0 that the extended interpretation $\pi^\perp$ interprets as undefined, i.e., $\pi^\perp(\perp) = \emptyset$. Keeping $\gamma_{\mathtt{h}}$ unchanged, $\sigma_{\mathcal{N}^\perp}(\mathtt{h})$ thus becomes the unary nondeterministic function h given by $h(P) = \sigma_{\mathcal{N}}(\mathtt{h})$, because $\gamma_{\mathtt{h}}$ never makes use of the parameter x_1. Nevertheless, $L(\mathcal{N}^\perp) = \emptyset$, because $\sigma_{\mathcal{N}^\perp}(t)$ is undefined for all $t \in L(\gamma_{\mathtt{g}_0})$, owing to the occurrences of $\perp$ in t.

In contrast to $\pi^\perp(\perp)$, the free interpretation of $\perp$ yields a result (namely $\perp$ itself), which implies that $\sigma_{\mathcal{N}^\perp_{\mathrm{free}}}(\mathtt{h}[\perp]) = \sigma_{\mathcal{N}}[\mathtt{h}]$ and thus $L(\mathcal{N}^\perp_{\mathrm{free}}) = L(\mathcal{N}_{\mathrm{free}})$. In other words, if we use the free interpretation, the offending occurrences of $\perp$ are deleted, which causes $\pi^\perp(L(\mathcal{N}^\perp_{\mathrm{free}}))$ to be different from $L(\mathcal{N}^\perp)$. We will show in Section 5 how this discrepancy can be removed by generating jungles instead of

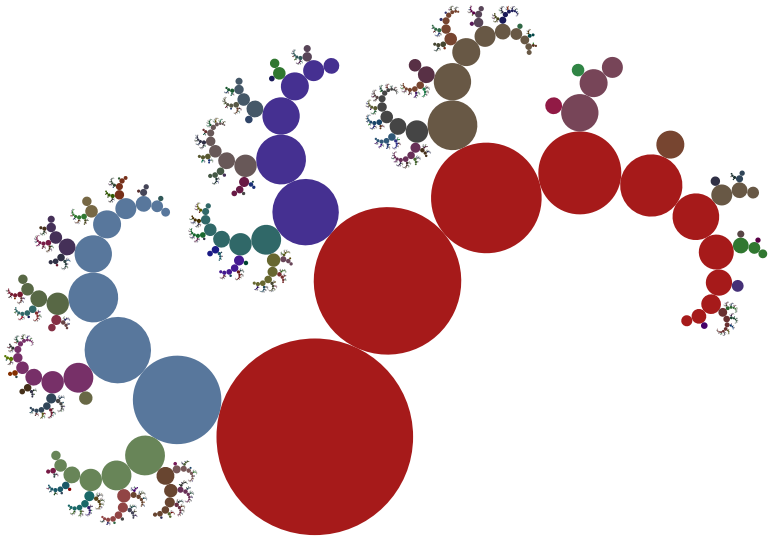


Figure 5: A picture generated by the delegation network in Example 2.7(c)

trees. In a generated jungle, the unused copies of $\bot$ will be retained (as garbage), so that the evaluation of the jungle yields the expected result.

Example 2.9 Since delegation networks with the free interpretation generate trees, they can be used as $\gamma_{\mathtt{g}}$ in other delegation networks. This will be studied in detail in Sections 7 and 8. We consider two examples.

(a) Let $\mathcal{N}_0$ be the delegation network that uses the free interpretation, with the generator symbols $\mathtt{g}_0^{(0)}$ and $\mathtt{g}^{(1)}$, primitives $\mathtt{f}^{(3)}$ and $\mathtt{h}^{(0)}$, and $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{h}]\}$ and $L(\gamma_{\mathtt{g}}) = \{\mathtt{g}[\mathtt{f}[\mathtt{h}, x_1, x_1]], x_1\}$. Then, $\mathcal{N}_0$ generates all trees in $\mathrm{T}_{\{\mathtt{f},\mathtt{h}\}}$ such that the first child of each $\mathtt{f}$ is $\mathtt{h}$ and the binary tree obtained by considering only the second and third child of each occurrence of $\mathtt{f}$ is fully balanced.

Now, let $\mathcal{N}$ use the interpretation in Example 2.6, with generators $\mathtt{g}_0$ and $\mathtt{h}$, where $\gamma_{\mathtt{g}_0} = \mathcal{N}_0$. Again, $\gamma_{\mathtt{h}}$ is as in the previous examples. Then all "arms" of each generated picture are equally refined (by the fact that $\mathcal{N}_0$ generates fully balanced trees), but the inner decorations of the circles vary, since each is generated by an individual copy of $\gamma_{\mathtt{h}}$; see Figure 6 (in which the fully balanced tree has height 10). It is worth comparing this result with the one obtained by just combining the two delegation networks rather than using $\mathcal{N}_0$ as a tree generator in $\mathcal{N}$. More precisely, let $\mathcal{N}' = (\{\mathtt{g}_0, \mathtt{g}, \mathtt{h}\}, \Pi, \Gamma, \mathtt{g}_0, \pi)$, where $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{h}]\}$, $L(\gamma_{\mathtt{g}}) = \{\mathtt{g}[\mathtt{f}[\mathtt{h}, x_1, x_1]], x_1\}$, and $\gamma_{\mathtt{h}}$ and π are again as in the preceding examples. Now, as shown in Figure 7, the two substructures attached to each circle are (transformed) copies of each other, including the inner decorations of the circles.

(b) Let us now assume, as in Example 2.7(c), that pictures are colored, and let us turn $\mathtt{d}$ into a nondeterministic primitive such that $\pi(\mathtt{d})$ is the set of all uniformly colored unit disks. The primitives $\mathtt{r}$ and $\mathtt{f}$ are interpreted as in Exam-

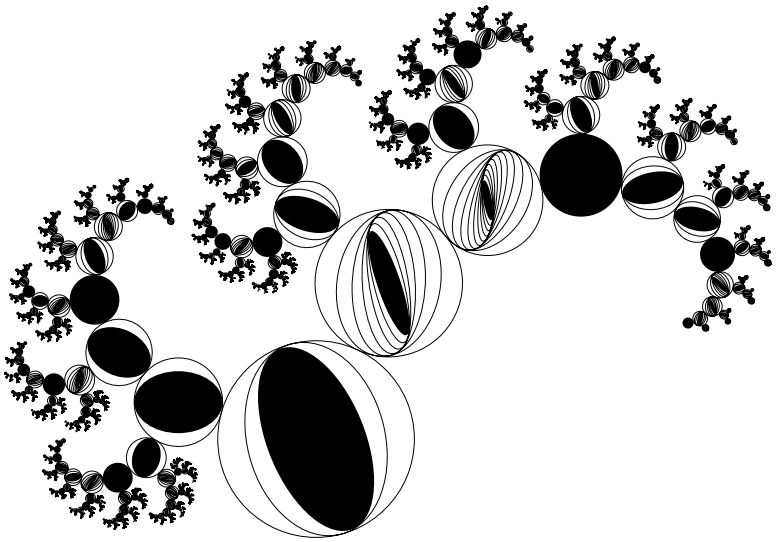


Figure 6: A picture generated by the delegation network $\mathcal{N}$ in Example 2.9

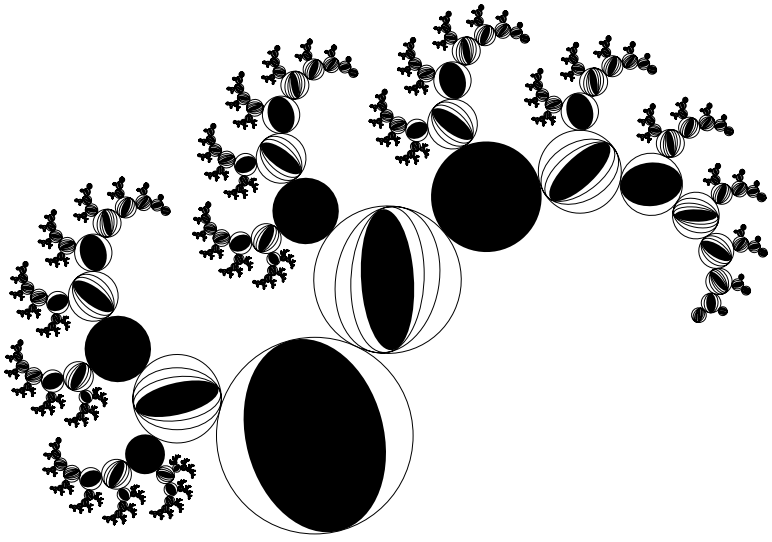


Figure 7: A picture generated by the delegation network $\mathcal{N}'$ in Example 2.9

ple 2.6 (which implies that they preserve color). We wish to define a delegation network $\mathcal{N}''$ that, except for the coloring, generates the same pictures as $\mathcal{N}$ above. In each picture, we want all (transformed) disks in the inner decorations of circles to have the same color. The network $\mathcal{N}''$ has generator symbols $\mathtt{g}_0^{(0)}$, $\mathtt{k}^{(1)}$ and $\mathtt{h}^{(1)}$, with $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{k}[\mathtt{d}]\}$, $L(\gamma_{\mathtt{h}}) = \{\mathtt{r}[\mathtt{h}[x_1]], x_1\}$ and $L(\gamma_{\mathtt{k}})$ being the tree language consisting of all trees of $L(\mathcal{N}_0)$ in which every occurrence of $\mathtt{h}$ is replaced by $\mathtt{h}[x_1]$. The tree language $L(\gamma_{\mathtt{k}})$ can be generated by an obvious variation $\mathcal{N}_0''$ of $\mathcal{N}_0$. Since x_1 is a primitive of $\mathcal{N}_0''$, $\mathcal{N}_0''$ cannot use it as a parameter symbol. Therefore, it uses the parameter signature $Y_{\mathtt{g}} = \{y_1\}$. Thus, in $\mathcal{N}_0''$, $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{h}[x_1]]\}$ and $L(\gamma_{\mathtt{g}}) = \{\mathtt{g}[\mathtt{f}[\mathtt{h}[x_1], y_1, y_1]], y_1\}$. This explains why, in general, we need other parameter signatures than X_k.

3 An Operational Characterization

Throughout this section, let $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$ be a delegation network, where $\Gamma = (\gamma_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}$. The goal of the section is to develop an operational characterization of the semantics of delegation networks. Such an operational characterization allows us to avoid the explicit use of the fixed-point construction of Definition 2.4 in proofs. To achieve this, we define a derivation relation $\rightarrowtail_{\mathcal{N}}$ that, intuitively, works on trees over a larger (possibly infinite) signature, with additional symbols $\langle a\rangle$ of rank 0 that represent values $a \in \mathbb{A}$. Intuitively, $\rightarrowtail_{\mathcal{N}}$ works bottom up and performs both evaluation and generation steps. Evaluation steps apply the interpretation of a primitive to already computed values, i.e., they nondeterministically replace $\mathtt{f}[\langle a_1\rangle, \ldots, \langle a_k\rangle]$ with $\mathtt{f}^{(k)} \in \Pi$ by any $\langle a\rangle$ with $a \in f(a_1, \ldots, a_k)$. Generation steps nondeterministically replace $\mathtt{g}[\langle a_1\rangle, \ldots, \langle a_k\rangle]$ with $\mathtt{g}^{(k)} \in \mathcal{G}$ by any tree $u(\langle a_1\rangle, \ldots, \langle a_k\rangle)$, where $u \in L(\gamma_{\mathtt{g}})$.

To start with, we have the following easy lemma.

Lemma 3.1 Let Σ and σ be a signature and a Σ-interpretation, resp. For every tree $s \cdot t \in \mathrm{T}_\Sigma$, if $t' \in \mathrm{T}_\Sigma$ is such that $\sigma(t) \subseteq \sigma(t')$, then $\sigma(s \cdot t) \subseteq \sigma(s \cdot t')$.

Proof Straigthforward structural induction on s. ∎

We now define the derivation relation $\rightarrowtail_{\mathcal{N}}$ mentioned above, that will be the basis for our operational characterization of the semantics of delegation networks.

Definition 3.2 Let $\underline{\Sigma}_{\mathcal{N}} = \Sigma_{\mathcal{N}} \cup \{\langle a\rangle^{(0)} \mid a \in \mathbb{A}\}$.[8] The derivation relation $\rightarrowtail_{\mathcal{N}}$ on $\mathrm{T}_{\underline{\Sigma}_{\mathcal{N}}}$ is given as follows. For trees $s \cdot t, s \cdot t' \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}}}$, we let $s \cdot t \rightarrowtail_{\mathcal{N}} s \cdot t'$ if one of the following holds:

- t is of the form $\mathtt{f}[\langle a_1\rangle, \ldots, \langle a_k\rangle]$, where $\mathtt{f} \in \Pi$, and $t' = \langle a\rangle$ for some $a \in f(a_1, \ldots, a_k)$, or
- t is of the form $\mathtt{g}[\langle a_1\rangle, \ldots, \langle a_k\rangle]$, where $\mathtt{g} \in \mathcal{G}$, and $t' = u(\langle a_1\rangle, \ldots, \langle a_k\rangle)$ for a tree $u \in L(\gamma_{\mathtt{g}})$.

[8] Here, $\langle a\rangle$ is assumed to be chosen in such a way that $\langle a\rangle \notin \Sigma_{\mathcal{N}}$.

Thus, the first item describes an evaluation step applying f to $a_1, \ldots, a_k$, whereas the second yields a tree generation step using $\gamma_{\mathtt{g}}$. We are going to show that $L(\mathcal{N}) = \{a \in \mathbb{A} \mid \mathtt{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle a \rangle\}$. For the first inclusion of this equality, which is provided by the following lemma, it is technically convenient to extend $\sigma_{\mathcal{N}}$ to $\underline{\Sigma}_{\mathcal{N}}$ in the obvious way: $\sigma_{\mathcal{N}}(\langle a \rangle) = \{a\}$ for all $a \in \mathbb{A}$.

Lemma 3.3 For all trees $t, t' \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}}}$, $t \rightarrowtail^*_{\mathcal{N}} t'$ implies $\sigma_{\mathcal{N}}(t') \subseteq \sigma_{\mathcal{N}}(t)$.

Proof It suffices to consider the relation $\rightarrowtail_{\mathcal{N}}$ instead of $\rightarrowtail^*_{\mathcal{N}}$. According to the definition of $\rightarrowtail_{\mathcal{N}}$, we have to consider two cases. Let $t = s \cdot t_0$ and $t' = s \cdot t'_0$.

Case 1 We have $t_0 = \mathtt{f}[\langle a_1 \rangle, \ldots, \langle a_k \rangle]$, where $\mathtt{f} \in \Pi$, and $t'_0 = \langle a \rangle$ for some $a \in f(a_1, \ldots, a_k)$.

In this case, we immediately get

$$\sigma_{\mathcal{N}}(t'_0) = \{a\} \subseteq f(a_1, \ldots, a_k) = \sigma_{\mathcal{N}}(t_0).$$

Thus, Lemma 3.1 yields $\sigma_{\mathcal{N}}(t') \subseteq \sigma_{\mathcal{N}}(t)$, as claimed.

Case 2 We have $t_0 = \mathtt{g}[\langle a_1 \rangle, \ldots, \langle a_k \rangle]$, where $\mathtt{g} \in \mathcal{G}$, and $t'_0 = u(\langle a_1 \rangle, \ldots, \langle a_k \rangle)$ for some $u \in L(\gamma_{\mathtt{g}})$.

This yields

$$\begin{aligned} \sigma_{\mathcal{N}}(t'_0) &= \sigma_{\mathcal{N}}(u)(a_1, \ldots, a_k) && \text{(Lemma 2.1)} \\ &\subseteq \sigma_{\mathcal{N}}(L(\gamma_{\mathtt{g}}))(a_1, \ldots, a_k) \\ &= \sigma_{\mathcal{N}}(\mathtt{g})(a_1, \ldots, a_k) \\ &= \sigma_{\mathcal{N}}(t_0), \end{aligned}$$

where the equality in the third row holds because $\sigma_{\mathcal{N}}$ is, by definition, a fixed point of $\mathrm{iterate}_{\mathcal{N}}$ (see Definition 2.4). As a consequence, again by Lemma 3.1, $\sigma_{\mathcal{N}}(t') \subseteq \sigma_{\mathcal{N}}(t)$. ∎

By the previous lemma, $\mathtt{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle a \rangle$ only if $a \in L(\mathcal{N})$. The next lemma proves that the other direction is valid as well.

Lemma 3.4 For all $a \in L(\mathcal{N})$, $\mathtt{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle a \rangle$.

Proof Let σ be the $\Sigma_{\mathcal{N}}$-interpretation such that $\sigma(\mathtt{f}) = f$ for all $\mathtt{f} \in \Pi$, and

$$\sigma(\mathtt{g})(a_1, \ldots, a_k) = \{a' \in \mathbb{A} \mid \mathtt{g}[\langle a_1 \rangle, \ldots, \langle a_k \rangle] \rightarrowtail^*_{\mathcal{N}} \langle a' \rangle\}$$

for all $\mathtt{g}^{(k)} \in \mathcal{G}$ and $a_1, \ldots, a_k \in \mathbb{A}$. We prove the inequality

$$\mathrm{iterate}_{\mathcal{N}}(\sigma) \leq \sigma. \tag{1}$$

This finishes the proof of the lemma, because it yields $\sigma_{\mathcal{N}} \leq \sigma$, by Proposition 2.3. In particular, $\sigma_{\mathcal{N}}(\mathtt{g}_0) \subseteq \sigma(\mathtt{g}_0)$, which is the same as saying that $a' \in L(\mathcal{N}) = \sigma_{\mathcal{N}}(\mathtt{g}_0)$ implies $\mathtt{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle a' \rangle$ (by the definition of $\sigma(\mathtt{g}_0)$).

By the definition of $\mathrm{iterate}_{\mathcal{N}}$, to prove (1), it suffices to show that $\sigma(L(\gamma_{\mathtt{g}})) \subseteq \sigma(\mathtt{g})$, for all $\mathtt{g}^{(k)} \in \mathcal{G}$. In other words, we have to show that $\mathtt{g}[\langle a_1 \rangle, \ldots, \langle a_k \rangle] \rightarrowtail^*_{\mathcal{N}}$

$\langle a' \rangle$ for all $a_1, \ldots, a_k \in \mathbb{A}$ and $a' \in \sigma(L(\gamma_{\mathtt{g}}))(a_1, \ldots, a_k)$. For this, by the second item of Definition 3.2, it suffices to prove the following claim.

Claim Let $k \in \mathbb{N}$. For all trees $t \in \mathrm{T}_{\Sigma_{\mathcal{N}},k}$ and all $a' \in \sigma(t)(a_1, \ldots, a_k)$, we have $t(\langle a_1 \rangle, \ldots, \langle a_k \rangle) \rightarrowtail^*_{\mathcal{N}} \langle a' \rangle$.

We proceed by structural induction on t. If $t = x_i$, then $a' = a_i$, and there is nothing to show. Thus, let $t = \mathtt{f}[t_1, \ldots, t_l]$ and $a' \in \sigma(\mathtt{f})(a'_1, \ldots, a'_l)$, for appropriate $a'_i \in \sigma(t_i)(a_1, \ldots, a_k)$ $(i \in [l])$. Then the induction hypothesis yields $t_i(\langle a_1 \rangle, \ldots, \langle a_k \rangle) \rightarrowtail^*_{\mathcal{N}} \langle a'_i \rangle$ for every $i \in [l]$, which gives

$$\begin{aligned} t(\langle a_1 \rangle, \ldots, \langle a_k \rangle) = \quad & \mathtt{f}[t_1(\langle a_1 \rangle, \ldots, \langle a_k \rangle), \ldots, t_l(\langle a_1 \rangle, \ldots, \langle a_k \rangle)] \\ \rightarrowtail^*_{\mathcal{N}} \; & \mathtt{f}[\langle a'_1 \rangle, \ldots, \langle a'_l \rangle] \\ \rightarrowtail^*_{\mathcal{N}} \; & \langle a' \rangle. \end{aligned}$$

Here, the last line holds by the first item of Definition 3.2 if $\mathtt{f} \in \Pi$, and by the definition of σ if $\mathtt{f} \in \mathcal{G}$. ∎

Combining the two previous lemmas, the desired operational characterization of the semantics of $\mathcal{N}$ follows immediately.

Theorem 3.5 $L(\mathcal{N}) = \{a \in \mathbb{A} \mid \mathtt{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle a \rangle\}$, for every delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$.

Proof By Lemmas 3.3 (where $t = \mathtt{g}_0$ and $t' = \langle a \rangle$) and 3.4. ∎

The previous theorem shows that the parallel distributed implementation of delegation networks discussed in the introduction is indeed correct. We repeat that discussion here in more detail, in order to relate it to the formal definitions. The main process takes care of the generation of an element of $L(\mathcal{N})$. For this, it first uses $\gamma_{\mathtt{g}_0}$ to generate a tree u_0 (if possible, i.e., if $L(\gamma_{\mathtt{g}_0}) \neq \emptyset$). Then, for every individual occurrence of a symbol $\mathtt{g}$ in u_0, a similar process using $\gamma_{\mathtt{g}}$ instead of $\gamma_{\mathtt{g}_0}$ is started recursively. If desired, this process may run on a different machine. Each process also takes care of the nondeterministic evaluation of the tree u it has generated. For this to be possible, its parent process must eventually provide it with values for all the parameter symbols in u, the actual parameters.[9] An evaluation step takes place as soon as the subtrees of (an occurrence of) a symbol $\mathtt{f}^{(k)} \in \Pi$ in u have been evaluated. In this situation, the process applies f to the values $a_1, \ldots, a_k$ of the subtrees as in the first item of Definition 3.2, obtains a value a' (provided that $f(a_1, \ldots, a_k) \neq \emptyset$), and replaces $\mathtt{f}$ by $\langle a' \rangle$. To implement the second item of Definition 3.2, when a subtree of a symbol $\mathtt{g} \in \mathcal{G}$ has been evaluated, the resulting value is passed to the child process that evaluates the occurrence of $\mathtt{g}$, thus providing it with one of its actual parameters. Moreover, whenever a child process has received all its actual parameters and has finished the evaluation of its own tree, it reports the value a'' computed to its parent process

[9] Note that the parameter passing mechanism is call-by-value.

and terminates.[10] The parent process can then continue the evaluation of u at symbol g, changing that symbol into $\langle a'' \rangle$. In effect, all processes together carry out a derivation $\mathsf{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle a \rangle$ according to Definition 3.2, where a is the result of the evaluation performed by the main process. Of course, a real implementation must bound the number of processes created, and it must make sure that the depth of the recursion caused by cyclic delegation does not increase ad infinitum.

Whereas Theorem 3.5 shows that $L(\mathcal{N})$ can be generated by alternatingly using semantic evaluation steps and syntactic tree generation steps, in the next two sections we will prove MW-like[11] results, to show that one can even perform all the syntactic steps first, and then the semantic ones. However, it turns out that such a result can be obtained for the general case only if the syntactic generation is defined in such a way that it yields trees with sharing and garbage, i.e., acyclic graphs, rather than ordinary trees. This case is treated in Section 5. In Section 6, we show that the use of trees is sufficient if all primitives are interpreted as functions.

4 Jungles and Hyperedge Replacement

In this section, we recall some notions regarding jungles [HP91, HKP91, Plu99], which will play a major role in Sections 5 and 7. 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. Technically, jungles are defined to be special hypergraphs, which makes it possible to generate them by means of hyperedge replacement, a context-free graph rewriting mechanism that has been studied extensively in the context of hyperedge-replacement graph grammars; see, e.g., [BC87, HK87, Hab92, Eng97, DHK97, CE12].[12]

Our notion of hypergraph is made precise in the following definition.

Definition 4.1 (Λ-hypergraph) Let Λ be a set of labels. A Λ-*hypergraph* is a tuple $H = (V, E, att, lab, ext)$ consisting of

- a finite set V of *nodes*,
- a finite set E of *hyperedges*,
- an *attachment function* $att\colon E \to V^+$ assigning to every hyperedge e a nonempty sequence $att(e)$ of attached nodes,

[10] We wish to stress that, to implement the call-by-value semantics correctly, we must make sure that a child process does not terminate its computation before receiving *all* its actual parameters, even those that do not occur in the tree it has generated. It does not necessarily have to receive them at the start of its computation.

[11] Recall that 'MW-like' refers to the seminal paper [MW67] by Mezei and Wright, where it was shown that a subset of a domain of an algebra is context-free if and only if it is the set of evaluations of a regular set of trees.

[12] Somewhat similar to the use of hyperedge replacement in this paper, [EH92] shows that the "term-generating" power of hyperedge-replacement graph grammars that generate jungles (which are then turned into trees) is equal to the power of attribute grammars. However, in the present paper, even the right-hand sides of rules (and, thus, all sentential forms) are jungles.

- a *labelling function* $lab\colon E \to \Lambda$, and
- a nonempty sequence $ext \in V^+$ of *external nodes*.

If Λ is understood or of minor interest, we simply speak of hypergraphs rather than Λ-hypergraphs. The five components of a hypergraph H may be denoted by $V_H, E_H, att_H, lab_H, ext_H$, resp.

Before continuing, let us mention that we usually do not distinguish between isomorphic hypergraphs, i.e., hypergraphs which are identical up to a bijective renaming of nodes and hyperedges. In other words, operations and relations defined on hypergraphs are meant to work on abstract hypergraphs (isomorphism classes of hypergraphs), even though, for technical convenience, we use concrete representatives in definitions, proofs, and the like.

The notion of hyperedge replacement to be defined next relies on the disjoint union $S \uplus S'$ of sets S, S' of nodes or hyperedges. To keep the technicalities as simple as possible, we shall generally assume that the hypergraphs in question are silently replaced by suitable isomorphic copies to make sure that $S \cap S' = \emptyset$. Using this convention, $S \uplus S'$ will be treated like the ordinary union of disjoint sets.

Given two hypergraphs H and H', the *addition of H' to H* yields the hypergraph $H \oplus H' = (V_H \uplus V_{H'}, E_H \uplus E_{H'}, att_H \cup att_{H'}, lab_H \cup lab_{H'}, ext_H)$. Note that $H \oplus H'$ inherits its sequence of external nodes from H. Thus, $\oplus$ is not commutative.

The *removal* of a hyperedge e from a hypergraph H yields

$$H - e = (V_H, E_H \setminus \{e\}, att, lab, ext_H),$$

where att and lab are the restrictions of att_H and lab_H, resp., to $E_H \setminus \{e\}$.

Finally, given two sequences of nodes $v = v_1 \cdots v_k$ and $w = w_1 \cdots w_k$ of a hypergraph H, of the same length, $H_{v=w}$ denotes the hypergraph obtained by identifying v with w in H. More precisely, let $\sim$ be the equivalence relation on V_H generated by $\{(v_1, w_1), \ldots, (v_k, w_k)\}$, and let $img(v')$ denote the image of $v' \in V_H$ in the quotient set $V_H/\!\sim$. Then $H_{v=w} = (V_H/\!\sim, E_H, img \circ att_H, lab_H, img(ext_H))$.

We are now ready to recall the definition of hyperedge replacement.

Definition 4.2 (hyperedge replacement) Let H be a hypergraph. A hypergraph H' *fits* a hyperedge $e \in E_H$ (in H) if $|att_H(e)| = |ext_{H'}|$. In this case, $H[e \leftarrow H']$ is the hypergraph obtained from the disjoint union of $H - e$ and H' by identifying $att_H(e)$ with $ext_{H'}$, i.e.,

$$H[e \leftarrow H'] = ((H - e) \oplus H')_{att_H(e) = ext_{H'}}.$$

Note that if $H'' = H[e \leftarrow H']$, then every node $v \in V_H \cup V_{H'}$ has an image $img(v)$ in H'', where img is as in the paragraph preceding Definition 4.2. The function img makes it possible to locate the nodes of H and H' in H''. In the following, it will therefore be called the *locator function* associated with the hyperedge replacement. By construction, img maps the ith attached node of

e in H to the same node as the ith external node of H', but is injective on $(V_H \setminus [att_H(e)]) \cup (V_{H'} \setminus [ext_{H'}])$.[13]

As mentioned above, we are interested in a special kind of hypergraph, the so-called jungle. Let Σ be a signature. A jungle is essentially a tree in $\mathrm{T}_{\Sigma,n}$ (with garbage), in which subtrees may be shared and multiple occurrences of each parameter symbol x_i must be shared.[14] To make this precise, let J be a Σ-hypergraph, with $ext_J = w_1 \cdots w_n w$. Assume that, for each hyperedge $e \in E_J$, if $lab_J(e) = \mathtt{f}^{(k)}$, then $att_J(e)$ has length $k+1$, say $att_J(e) = v_1 \cdots v_k v$. Intuitively, the nodes $v_1, \ldots, v_k$ represent the parameters of $\mathtt{f}$, and the nodes $w_1, \ldots, w_n$ represent the parameters of J. Therefore, we call $par_J(e) = v_1 \cdots v_k$ the sequence of *parameter nodes of* e, and $par(J) = w_1 \cdots w_n$ the sequence of *parameter nodes of* J. Similarly, v and w intuitively represent the results of $\mathtt{f}$ and J, respectively, and so $res_J(e) = v$ and $res(J) = w$ are called the *result nodes of* e *and* J, respectively. A *parameter tentacle of* e is a pair (e, j), where $j \in [k]$. We let $tent(J)$ denote the set of all parameter tentacles of hyperedges in E_J.

Now, let $v, w \in V_J$. For the definition of jungles below, and also for later use, we define a *jungle path from* v *to* w (in J) to be either the empty sequence λ, provided that $v = w$, or a nonempty sequence $(e_0, j_0) \cdots (e_m, j_m) \in tent(J)^*$ such that $v = res_J(e_0)$, $w = par_J(e_m, j_m)$, and $res_J(e_l) = par_J(e_{l-1}, j_{l-1})$ for all $l \in [m]$. The hypergraph J is *acyclic* if, for every $v \in V_J$, λ is the only jungle path from v to v.

Definition 4.3 (jungle) Let Σ be a signature. The set $\mathrm{J}_{\Sigma,n}$ of *jungles over* Σ *with* n *parameters* ($n \in \mathbb{N}$) consists of all acyclic Σ-hypergraphs J such that

- $par(J) = w_1 \cdots w_n$ for pairwise distinct nodes $w_1, \ldots, w_n$,
- for all $v \in V_J$, $v = res_J(e)$ for some $e \in E_J$ if and only if $v \notin [par(J)]$,
- for all hyperedges $e, e' \in E_J$, if $e \neq e'$ then $res_J(e) \neq res_J(e')$, and
- for every hyperedge $e \in E_J$, $|par_J(e)|$ is equal to the rank of $lab_J(e)$ in Σ.

We let $\mathrm{J}_\Sigma = \mathrm{J}_{\Sigma,0}$. For finite Σ, a *jungle language* is a subset of $\mathrm{J}_{\Sigma,n}$, where $n \in \mathbb{N}$.

Intuitively, the n pairwise distinct parameter nodes of a jungle J correspond to the parameter symbols x_i of a tree in $\mathrm{T}_{\Sigma,n}$, and $res(J)$ corresponds to its root.

Every jungle $J \in \mathrm{J}_{\Sigma,n}$ represents a unique tree $\mathrm{tree}(J) \in \mathrm{T}_{\Sigma,n}$, namely $\mathrm{tree}(J) = \mathrm{tree}(J, res(J))$, where $\mathrm{tree}(J, v)$ is defined as follows, for all $v \in V_J$. If $par(J) = w_1 \cdots w_n$ and $v = w_i$, where $i \in [n]$, then $\mathrm{tree}(J, v) = x_i$. Otherwise, let $e \in E_J$ be the unique hyperedge such that $res_J(e) = v$. Then $\mathrm{tree}(J, v) = \mathtt{f}[\mathrm{tree}(J, v_1), \ldots, \mathrm{tree}(J, v_k)]$, where $\mathtt{f}^{(k)} = lab_J(e)$ and $par_J(e) = v_1 \cdots v_k$.

Conversely, a tree $t \in \mathrm{T}_{\Sigma,n}$ can be turned into a jungle by creating a node that represents the root of s, for each (occurrence of a) subtree $s = \mathtt{f}[s_1, \ldots, s_k]$. If $\mathtt{f} \in \Sigma$, then there is a hyperedge labelled $\mathtt{f}$ whose result node is this node, and

[13] Recall that, for a string u, $[u]$ denotes the set of all symbols occurring in u.

[14] Recall from the end of Section 2 that $\mathrm{T}_{\Sigma,n} = \mathrm{T}_{\Sigma,X_n}$, where X_n is the parameter signature $\{x_1, \ldots, x_n\}$.

whose parameter nodes are the nodes representing the roots of $s_1, \ldots, s_k$. If $\mathtt{f} = x_i$, then the node representing its root becomes $par(J, i)$. Different occurrences of one and the same subtree can either be shared (the same node represents their roots) or not. However, all occurrences of one and the same parameter must be shared: we have to create a unique parameter node for every $x_i \in X_n$. The result node of the jungle is the node representing the root of t. The jungle obtained in this way with the minimum amount of sharing, is denoted by $\mathrm{J}_n(t)$.

Formally, we let

$$\mathrm{J}_n(t) = (V, E, att, lab, ext)$$

consist of the following components:

- $V = \{nod(v) \mid v \in V_\Sigma(t) \uplus X_n\}$;[15]
- $E = \{edg(v) \mid v \in V_\Sigma(t)\}$;
- $lab(edg(v)) = \ell_t(v)$ for all $v \in V_\Sigma(t)$;
- $att(edg(v)) = nod(v_1) \cdots nod(v_k) nod(v)$ for all $v \in V_\Sigma(t)$, where k is the rank of $\ell_t(v)$, and

$$v_i = \begin{cases} vi & \text{if } \ell_t(vi) \in \Sigma \\ \ell_t(vi) & \text{if } \ell_t(vi) \in X_n, \end{cases}$$

  for all $i \in [k]$;
- $ext = nod(x_1) \cdots nod(x_n) w$, where $w = nod(\lambda)$ if $\ell_t(\lambda) \in \Sigma$, and $w = nod(x_i)$ if $t = x_i$ for some $i \in [n]$.

Note that for every tree t and $n \in \mathbb{N}$, $\mathrm{tree}(\mathrm{J}_n(t)) = t$.

If $\mathrm{J}_n(t)$ is taken with respect to a parameter signature Y other than X_n, the definition is adapted in the obvious way, and the notation is turned into $\mathrm{J}_Y(t)$ to emphasize Y. However, this will only be used in Section 7.

Note that in the special case where $t = \mathtt{f}[x_1, \ldots, x_n]$ for a symbol $\mathtt{f}^{(n)}$, the jungle $J = \mathrm{J}_n(t)$ consists of a single $\mathtt{f}$-labelled hyperedge e with n pairwise distinct parameter nodes, such that $ext_J = att_J(e)$. In the following, if the rank n of $\mathtt{f}$ is clear from the context, we simply denote this jungle by $\mathrm{J}(\mathtt{f})$.

Thanks to the fact that parameter nodes of jungles are required to be pairwise distinct, it is not difficult to verify that $\mathrm{J}_{\Sigma,n}$ is closed under hyperedge replacement (see [EV94, Lemma 5.3]).

Fact 4.4 Let $J \in \mathrm{J}_{\Sigma,n}$ for a signature Σ and $n \in \mathbb{N}$. For every jungle $J' \in \mathrm{J}_{\Sigma,k}$, if J' fits hyperedge $e \in E_J$, then $J[e \leftarrow J'] \in \mathrm{J}_{\Sigma,n}$.

An important fact that needs to be kept in mind when working with jungles is that they may contain shared nodes as well as garbage. Here, a node v in a jungle J is *shared* if there are distinct parameter tentacles $(e, j), (e', j') \in tent(J)$ such that $par_J(e, j) = v = par_J(e', j')$. We say that v is *garbage* if there is no jungle path in J from $res(J)$ to v; in particular, v is a *garbage root* if $v \neq res(J)$

[15] Recall that $V_\Sigma(t)$ denotes the set of nodes of t with labels in Σ.

and $v \notin [par_J(e)]$ for all $e \in E_J$. Shared nodes and garbage roots will play an important role in the proof of Theorem 5.6.

Garbage, except parameter nodes, can be removed from a jungle J without affecting tree(J). The following definitions related to this observation will only be needed in Section 7. Given a jungle J, we let C_J denote the set of all nodes $v \in V_J$ that are not garbage. We say that J is *clean* if $V_J = C_J \cup [par(J)]$, i.e., if all garbage nodes of J are parameter nodes of J. The clean jungle clean(J) is obtained from J by restricting J to $C_J \cup [par(J)]$. Thus, clean(J) = (V, E, att, lab, ext_J), where $V = C_J \cup [par(J)]$, $E = \{e \in E_J \mid res_J(e) \in V\}$, and att and lab are the restrictions of att_J and lab_J to E. As mentioned above, we have tree(clean(J)) = tree(J). The reader should also note a few additional easy facts that will be used (in Section 7) without mention: (1) clean(J) is clean (as the notation suggests), because $C_{\text{clean}(J)} = C_J$, (2) by construction, jungles of the form $\mathrm{J}_n(t)$ are clean, and (3) if J is clean, then clean(J) = J.

Another basic fact about jungles used in Section 7 is that, intuitively, a jungle of the form $\mathrm{J}_n(t)$ is the most general jungle representation of a tree t. This can be expressed formally by saying that $\mathrm{J}_n(\text{tree}(J))$ maps to J, for every jungle J, and that both jungles represent the same tree. Here, we say that a jungle I *maps to* a jungle J if there is a structure preserving mapping $\varphi\colon V_I \to V_J$, where structure preservation means that $\varphi(ext_I) = ext_J$ and, for every hyperedge $e \in E_I$, there exists a (unique) hyperedge $e' \in E_J$ with $att_J(e') = \varphi(att_I(e))$ and $lab_J(e') = lab_I(e)$. The formal statement below is similar to [Plu99, Lemmas 1.3.6 and 1.3.9].

Lemma 4.5 For every jungle $J \in \mathrm{J}_{\Sigma,n}$, the jungle $\mathrm{J}_n(\text{tree}(J))$ maps to J. Furthermore, if I is a jungle that maps to J, then tree(I) = tree(J).

Proof As the second statement should be obvious[16], we prove only the first. Let $v \in V_J$, and let J_v be the jungle that is equal to J, except that $res(J_v) = v$. By induction on the number of hyperedges in clean(J_v), we prove that $\mathrm{J}_n(\text{tree}(J, v))$ maps to J_v. This is obvious if $v \in [par(J)]$, using the mapping φ given by $\varphi(nod(x_j)) = par(J, j)$ for all $j \in [n]$ (where $nod(x_j)$ is as in the definition of $\mathrm{J}_n(t)$). Now, consider the case where $v \notin [par(J)]$, and let $e \in E_J$ be the unique hyperedge such that $res_J(e) = v$, where $par_J(e) = v_1 \cdots v_k$. By the induction hypothesis, for all $i \in [k]$, $\mathrm{J}_n(\text{tree}(J, v_i))$ maps to J_{v_i}. Let $\varphi_1, \ldots, \varphi_k$ be the corresponding structure-preserving mappings. Then the reader may easily check that φ is a structure-preserving mapping from $\mathrm{J}_n(\text{tree}(J, v))$ to J_v, where $\varphi(nod(\lambda)) = v$, $\varphi(nod(x_j)) = par(J, j)$ for $j \in [n]$, and $\varphi(nod(iw)) = \varphi_i(nod(w))$, for $i \in [k]$ and $w \in V_\Sigma(\text{tree}(J, v_i))$. ∎

In examples, we will also use the jungle with the maximal amount of sharing

[16] By induction on the structure of I, tree(I, v) = tree($J, \varphi(v)$) for every $v \in V_I$ (cf. the proof of [Plu99, Lemma 1.3.6]).

that represents a tree $t \in \mathrm{T}_{\Sigma,n}$, denoted $\mathrm{J}_n^{\mathrm{ms}}(t)$. It is defined as

$$\mathrm{J}_n^{\mathrm{ms}}(t) = (V, E, att, lab, ext)$$

where:

- $V = \{nod(s) \mid s \text{ is a subtree of } t \text{ or } s \in X_n\}$;
- $E = \{edg(s) \mid s \text{ is a subtree of } t \text{ and } s \notin X_n\}$;
- if $s = \mathtt{f}[s_1, \ldots, s_k]$ with $\mathtt{f}^{(k)} \in \Sigma \cup X_n$ and $s_1, \ldots, s_k \in \mathrm{T}_{\Sigma,n}$, then $lab(edg(s)) = \mathtt{f}$ and $att(edg(s)) = nod(s_1) \cdots nod(s_k)nod(s)$;
- $ext = nod(x_1) \cdots nod(x_n)nod(t)$.

As an example, if $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{c}^{(0)}\}$, then $\mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\mathtt{c}, \mathtt{c}]) = (\{v, w\}, \{e_\mathtt{f}, e_\mathtt{c}\}, att, lab, v)$ with $att(e_\mathtt{f}) = wwv$, $att(e_\mathtt{c}) = w$, $lab(e_\mathtt{f}) = \mathtt{f}$, and $lab(e_\mathtt{c}) = \mathtt{c}$ (where $v = nod(\mathtt{f}[\mathtt{c}, \mathtt{c}])$, $w = nod(\mathtt{c})$, $e_\mathtt{f} = edg(\mathtt{f}[\mathtt{c}, \mathtt{c}])$, and $e_\mathtt{c} = edg(\mathtt{c})$).

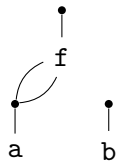
For $\mathrm{J}_n^{\mathrm{ms}}(t)$, it is easy to prove a "dual" of Lemma 4.5: for every clean jungle $J \in \mathrm{J}_{\Sigma,n}$, the jungle J maps to $\mathrm{J}_n^{\mathrm{ms}}(\mathrm{tree}(J))$ (by mapping every $v \in V_J$ to $nod(\mathrm{tree}(J, v))$). However, this property will not be used anywhere in this paper.

5 A Characterization Using Jungles

In this section, we show our first (and main) MW-like result stating that, for a delegation network $\mathcal{N}$, $L(\mathcal{N})$ can be obtained by first using $\mathcal{N}$ to generate a set of jungles and then evaluating the jungles obtained. As discussed in Examples 2.7(a) and 2.8, sharing and garbage in jungles are essential to achieve this if the interpretation of primitives is nondeterministic. Let us recall the problem by discussing another small example.

Example 5.1 Consider a delegation network $\mathcal{N}$ with primitives $\mathtt{f}^{(2)}, \mathtt{a}^{(0)}, \mathtt{b}^{(0)}$ interpreted into strings, where f is string concatenation, $a = \{\diamondsuit, \heartsuit\}$, and $b = \{\diamondsuit\}$. The generator symbols are $\mathtt{g}_0^{(0)}$ and $\mathtt{g}^{(2)}$ with $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{a}, \mathtt{b}]\}$ and $L(\gamma_\mathtt{g}) = \{\mathtt{f}[x_1, x_1]\}$. Then $L(\mathcal{N}) = \{\diamondsuit\diamondsuit, \heartsuit\heartsuit\}$. In contrast, the tree language generated by $\mathcal{N}$ (using the free interpretation of Π) consists of the single tree $\mathtt{f}[\mathtt{a}, \mathtt{a}]$, which evaluates to $\{\diamondsuit\diamondsuit, \diamondsuit\heartsuit, \heartsuit\diamondsuit, \heartsuit\heartsuit\}$, because both occurrences of $\mathtt{a}$ are evaluated independently. Moreover, if we turn b into $\emptyset$, then $L(\mathcal{N}) = \emptyset$, whereas the evaluation of the tree language generated by $\mathcal{N}$ still yields the same result, because $\mathtt{b}$ is not evaluated at all.

As might be expected, the jungle J generated by $\mathcal{N}$ will look like this:



The result node of J is the result node v_0 of the hyperedge with label $\mathtt{f}$. Note that the result node v_1 of the hyperedge with label $\mathtt{a}$ is shared (it is both the first and second parameter node of the hyperedge with label $\mathtt{f}$), and the result node

v_2 of the hyperedge with label `b` is garbage (and, in fact, a garbage root). The evaluation of J according to the definition below will evaluate the occurrences of `a` and `b` exactly once, thus yielding the correct result under both interpretations.

We start by defining how jungles are evaluated.

Definition 5.2 (jungle evaluation) Consider a Σ-interpretation σ (into $\mathbb{A}$) and a jungle $J \in \mathrm{J}_{\Sigma,n}$. Given $a_1, \ldots, a_n \in \mathbb{A}$, let $ASS_{J,\sigma}(a_1, \ldots, a_n)$ be the set of all functions $\alpha \colon V_J \to \mathbb{A}$ such that the following hold:

- $\alpha(par(J, i)) = a_i$, for all $i \in [n]$;
- if $att_J(e) = v_1 \cdots v_k v$, then $\alpha(v) \in \sigma(lab_J(e))(\alpha(v_1), \ldots, \alpha(v_k))$, for all $e \in E_J$.

Now, $\sigma(J)$ 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 $\mathcal{J} \subseteq \mathrm{J}_{\Sigma,n}$, $\sigma(\mathcal{J}) = \bigcup_{J \in \mathcal{J}} \sigma(J)$.

For instance, if J is the jungle in Example 5.1, with the nodes v_0 (the result node), v_1 (the leftmost node), and v_2 (the rightmost node), then $ASS_{J,\sigma}() = \{\alpha_1, \alpha_2\}$, where $\alpha_1(v_1) = \diamondsuit$ and $\alpha_2(v_1) = \heartsuit$ and, for $i \in [2]$, $\alpha_i(v_2) = \diamondsuit$ and $\alpha_i(v_0) = \alpha_i(v_1)\alpha_i(v_1)$. Thus, $\sigma(J) = L(\mathcal{N})$. Moreover, in the case where $b = \emptyset$, we obtain $ASS_{J,\sigma}() = \emptyset$.

The following lemma shows that the previous definition is consistent with the evaluation of trees: if we turn a tree t into the jungle $\mathrm{J}_n(t)$, then their evaluations coincide.

Lemma 5.3 Let σ be a Σ-interpretation. For all trees $t \in \mathrm{T}_{\Sigma,n}$, $\sigma(t) = \sigma(\mathrm{J}_n(t))$.

Proof Let $J = \mathrm{J}_n(t)$ and $a_1, \ldots, a_n \in \mathbb{A}$. We have to show that, for all $a \in \mathbb{A}$, $a \in \sigma(t)(a_1, \ldots, a_n)$ if and only if there is some $\alpha \in ASS_{J,\sigma}(a_1, \ldots, a_n)$ with $\alpha(res(J)) = a$.

It follows directly from the definition of $\sigma(t)$ (by using an obvious induction), that $a \in \sigma(t)(a_1, \ldots, a_n)$ if and only if there is a function $\alpha' \colon V(t) \to \mathbb{A}$ such that $\alpha'(\lambda) = a$, $\alpha'(v) = a_i$ for all $v \in V_{X_n}(t)$ with $\ell_t(v) = x_i$, and $\alpha'(v) \in f(\alpha'(v1), \ldots, \alpha'(vk))$ for all $v \in V_\Sigma(t)$ with $\ell_t(v) = \mathtt{f}^{(k)}$. By the definition of $ASS_{J,\sigma}(a_1, \ldots, a_n)$, this is the case if and only if there is some $\alpha \in ASS_{J,\sigma}(a_1, \ldots, a_n)$ with $\alpha(res(J)) = a$, which completes the proof. ∎

It should be noticed that the validity of the previous proof depends on a subtlety whose importance might not be obvious at first sight, namely the fact that $\mathrm{J}_n(t)$ uses minimal sharing and has no garbage roots. In other words, it is *not* the case that $\sigma(\mathrm{tree}(J)) = \sigma(J)$ for every jungle J. To see this, recall Example 5.1, where the sharing in J restricts nondeterminism, yielding $\sigma(J) = \{\diamondsuit\diamondsuit, \heartsuit\heartsuit\}$. In contrast, $\sigma(\mathrm{tree}(J)) = \sigma(\mathtt{f}[\mathtt{a}, \mathtt{a}]) = \{\diamondsuit\diamondsuit, \diamondsuit\heartsuit, \heartsuit\diamondsuit, \heartsuit\heartsuit\}$, because in this case the occurrences of `a` are evaluated independently. Moreover, as mentioned in Example 5.1 too, the existence of garbage in J may make $\sigma(J)$ undefined.

We now show that interpretation of jungles is compatible with hyperedge replacement, similar to Lemma 3.1.

Lemma 5.4 Let σ be a Σ-interpretation, $\mathtt{f}^{(k)} \in \Sigma$, $J \in \mathrm{J}_{\Sigma,n}$, and $e \in E_J$ with $lab_J(e) = \mathtt{f}$ (where $k, n \in \mathbb{N}$). For all jungles $J' \in \mathrm{J}_{\Sigma,k}$, if $\sigma(J') \subseteq \sigma(\mathtt{f})$, then $\sigma(J[e \leftarrow J']) \subseteq \sigma(J)$.

Proof Let $J'' = J[e \leftarrow J']$, and let $\alpha'' \in ASS_{J'',\sigma}(a_1, \ldots, a_n)$ for $a_1, \ldots, a_n \in \mathbb{A}$, with $\alpha''(res(J'')) = a \in \mathbb{A}$. Let $img\colon V_J \cup V_{J'} \to V_{J''}$ be the locator function associated with the hyperedge replacement (see the paragraph following Definition 4.2). With $\alpha''(img(ext(J'))) = b_1 \cdots b_k b$, we get $\alpha' \in ASS_{J',\sigma}(b_1, \ldots, b_k)$ by defining $\alpha'(v) = \alpha''(img(v))$, for all $v \in V_{J'}$. It follows that $b \in f(b_1, \ldots, b_k)$, which means that we get $\alpha \in ASS_{J,\sigma}(a_1, \ldots, a_n)$ by defining $\alpha(v) = \alpha''(img(v))$, for all $v \in V_J$. Note that $\alpha(att_J(e)) = b_1 \cdots b_k b$, because $img(att_J(e)) = img(ext(J'))$. This proves that $a \in \sigma(J)(a_1, \ldots, a_n)$ and completes the proof. ∎

As mentioned in the introduction, it is straightforward to turn an IO context-free tree grammar into a (context-free) graph grammar that generates jungles, by viewing each tree t in $L(\gamma_{\mathtt{g}})$ as the jungle $\mathrm{J}_k(t)$ which contains exactly one node with label x_i for every parameter symbol x_i (where $i \in [k]$ and k is the rank of $\mathtt{g}$). With this in mind, the previous lemma motivates the following definition, which leads to Theorem 5.6, our main MW-like result.

Definition 5.5 For every delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}})_{\mathtt{g} \in \mathcal{G}}, \mathtt{g}_0, \mathbb{A}, \pi)$, the binary relation $\Rightarrow_{\mathcal{N}}$ on $\mathrm{J}_{\Sigma_{\mathcal{N}},n}$ ($n \in \mathbb{N}$) is given as follows. If $J \in \mathrm{J}_{\Sigma_{\mathcal{N}},n}$ and $e \in E_J$, where $lab_J(e) = \mathtt{g}^{(k)} \in \mathcal{G}$, then $J \Rightarrow_{\mathcal{N}} J[e \leftarrow \mathrm{J}_k(t)]$ for all trees $t \in L(\gamma_{\mathtt{g}})$.

The *jungle language generated by* $\mathcal{N}$ is given by

$$L_{\mathrm{J}}(\mathcal{N}) = \{J \in \mathrm{J}_\Pi \mid \mathrm{J}(\mathtt{g}_0) \Rightarrow_{\mathcal{N}}^* J\}.$$

Note that $\Rightarrow_{\mathcal{N}}$ and, thus, $L_{\mathrm{J}}(\mathcal{N})$ do not depend on the last two components $\mathbb{A}$ and π of $\mathcal{N}$. Thus, $L_{\mathrm{J}}(\mathcal{N})$ is generated in a purely syntactical way, cf. the last paragraph of Section 3. The reader should also note that the jungles generated by $\mathcal{N}$, i.e., the elements of $L_{\mathrm{J}}(\mathcal{N})$, do not have parameter nodes, because $\mathtt{g}_0$ has rank 0.

Theorem 5.6 For every delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}})_{\mathtt{g} \in \mathcal{G}}, \mathtt{g}_0, \mathbb{A}, \pi)$,

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

Proof The inclusion $\pi(L_{\mathrm{J}}(\mathcal{N})) \subseteq L(\mathcal{N})$ follows from Lemmas 5.3 and 5.4, as follows. By Lemma 5.3, $\sigma_{\mathcal{N}}(\mathrm{J}(\mathtt{g}_0)) = \sigma_{\mathcal{N}}(\mathtt{g}_0)$. Moreover, by the definition of $\mathrm{iterate}_{\mathcal{N}}$ (and the fact that $\sigma_{\mathcal{N}}$ is a fixed point of $\mathrm{iterate}_{\mathcal{N}}$), we have $\sigma_{\mathcal{N}}(t) \subseteq \sigma_{\mathcal{N}}(\mathtt{g})$ for all $\mathtt{g} \in \mathcal{G}$ and $t \in L(\gamma_{\mathtt{g}})$. Using this, together with Lemmas 5.3 and 5.4, induction on the length of derivations yields $\sigma_{\mathcal{N}}(J) \subseteq \sigma_{\mathcal{N}}(\mathtt{g}_0)$, for all jungles $J \in \mathrm{J}_{\Sigma_{\mathcal{N}}}$ such that $\mathrm{J}(\mathtt{g}_0) \Rightarrow_{\mathcal{N}}^* J$. Since $\sigma_{\mathcal{N}}(J) = \pi(J)$ for all $J \in \mathrm{J}_\Pi$, this proves that $\pi(L_{\mathrm{J}}(\mathcal{N})) \subseteq \sigma_{\mathcal{N}}(\mathtt{g}_0) = L(\mathcal{N})$.

It remains to prove that $L(\mathcal{N}) \subseteq \pi(L_{\mathrm{J}}(\mathcal{N}))$. For this, we use the following definitions:

1. For a jungle $J \in \mathrm{J}_{\Sigma_\mathcal{N}}$, we let ASS'_J denote the set of all partial functions $\alpha\colon V_J \to \mathbb{A}$ such that the following hold:
   (a) α is defined on all shared nodes and garbage roots of J;[17]
   (b) for all $e \in E_J$ with $lab_J(e) = \mathtt{f}^{(k)}$ and $att_J(e) = v_1 \cdots v_k v$, if $\alpha(v)$ is defined, then $\mathtt{f} \in \Pi$, $\alpha(v_i)$ is defined for every $i \in [k]$, and $\alpha(v) \in f(\alpha(v_1), \ldots, \alpha(v_k))$.

   Note that if $\alpha(res(J))$ is defined, then α is total, and so $J \in \mathrm{J}_\Pi$ and $\alpha(res(J)) \in \pi(J)$.
2. For $J \in \mathrm{J}_{\Sigma_\mathcal{N}}$, $\alpha \in ASS'_J$, and $v \in V_J$, define $\mathrm{tree}_\alpha(J, v) \in \mathrm{T}_{\Sigma_\mathcal{N}}$ as follows: if $\alpha(v)$ is defined, then $\mathrm{tree}_\alpha(J, v) = \langle \alpha(v) \rangle$. Otherwise, $\mathrm{tree}_\alpha(J, v) = \mathtt{f}[\mathrm{tree}_\alpha(J, v_1), \ldots, \mathrm{tree}_\alpha(J, v_k)]$, where $e \in E_J$ is the unique hyperedge such that $res_J(e) = v$, with $lab_J(e) = \mathtt{f}^{(k)}$ and $par_J(e) = v_1 \cdots v_k$. We use the abbreviation $\mathrm{tree}_\alpha(J)$ to denote $\mathrm{tree}_\alpha(J, res(J))$.
   Note that if $\mathrm{tree}_\alpha(J) = \langle a \rangle$ for some $a \in \mathbb{A}$, then $\alpha(res(J))$ is defined and equals a, which means that $J \in \mathrm{J}_\Pi$ and $a \in \pi(J)$.

Claim For all trees $t \in \mathrm{T}_{\Sigma_\mathcal{N}}$ such that $\mathtt{g}_0 \rightarrowtail^*_\mathcal{N} t$, there are a jungle $J \in \mathrm{J}_{\Sigma_\mathcal{N}}$ and a partial function $\alpha \in ASS'_J$ such that $\mathrm{J}(\mathtt{g}_0) \Rightarrow^*_\mathcal{N} J$ and $\mathrm{tree}_\alpha(J) = t$.

By Theorem 3.5, this proves the required inclusion since $a \in L(\mathcal{N})$ implies $\mathtt{g}_0 \rightarrowtail^*_\mathcal{N} \langle a \rangle$, and hence there exists $J \in L_{\mathrm{J}}(\mathcal{N})$ such that $a \in \pi(J)$.

We proceed by induction on the length of derivations. For $t = \mathtt{g}_0$, the claim is true with $J = \mathrm{J}(\mathtt{g}_0)$ and α being the totally undefined function. Now, let $\mathtt{g}_0 \rightarrowtail^*_\mathcal{N} t' \rightarrowtail_\mathcal{N} t$, and assume that the claim holds for the initial part $\mathtt{g}_0 \rightarrowtail^*_\mathcal{N} t'$ of the derivation, where $J' \in \mathrm{J}_{\Sigma_\mathcal{N}}$ and $\alpha' \in ASS'_{J'}$ satisfy the requirements given. Let $t' = s \bullet \mathtt{f}[\langle a_1 \rangle, \ldots, \langle a_k \rangle]$, and let $v \in V_{J'}$ be the node yielding the root of $\mathtt{f}[\langle a_1 \rangle, \ldots, \langle a_k \rangle]$ under $\mathrm{tree}_{\alpha'}$, and $e \in E_{J'}$ the unique hyperedge such that $res_{J'}(e) = v$.[18] Note that by the definition of $\mathrm{tree}_{\alpha'}(J', v)$, $\alpha'(v)$ is undefined, and thus v is not shared. Furthermore, $lab_{J'}(e) = \mathtt{f}$, and if $par_{J'}(e) = v_1 \cdots v_k$, then $\alpha'(v_i) = a_i$ (also by the definition of $\mathrm{tree}_{\alpha'}(J', v)$).

To complete the proof of the claim, we have to check two cases.

Case 1 We have $\mathtt{f} \in \Pi$ and $t = s \bullet \langle a \rangle$ for some $a \in f(a_1, \ldots, a_k)$.

Let $J = J'$, and let α be the extension of α' such that $\alpha(v) = a$. Clearly, this yields $\alpha \in ASS'_J$. Moreover, the fact that v is not shared implies that $\mathrm{tree}_\alpha(J) = t$. (Note that if v would be shared, setting $\alpha(v) = a$ could turn several subtrees of $t' = \mathrm{tree}_{\alpha'}(J')$ into $\langle a \rangle$, thus yielding $\mathrm{tree}_\alpha(J) \neq t$.)

Case 2 We have $\mathtt{f} \in \mathcal{G}$ and $t = s \bullet s'(\langle a_1 \rangle, \ldots, \langle a_k \rangle)$ for a tree $t' \in L(\gamma_\mathtt{f})$.

Let $J = J'[e \leftarrow \mathrm{J}_k(s')]$, and define α as follows. If $s' \notin X_k$, then $\alpha = \alpha' \circ img^{-1}$, where img is the locator function associated with the hyperedge replacement,

[17] Recall the definition of shared nodes and garbage roots following Fact 4.4.

[18] More precisely, v and e are uniquely determined by the requirements that $v = res_{J'}(e)$ and, if $J' - e = (V, E, att, lab, w)$ and $J'' = (V, E, att, lab, vw)$, then $\mathrm{tree}(J'') = s$.

restricted to $V_{J'}$. (Note that *img* is injective if $s' \notin X_k$, which makes α well defined.) It is straightforward to check that $\alpha \in ASS'_J$ and $\mathrm{tree}_\alpha(J) = t$, again using the fact that v is not shared.[19] Finally, if $s' = x_i$ for some $i \in [k]$, this means that J is obtained from $J' - e$ by identifying v with v_i, where $par_{J'}(e) = v_1 \cdots v_k$. Let v' be the resulting node and define

$$\alpha(w) = \begin{cases} \alpha'(v_i) & \text{if } w = v' \\ \alpha'(w) & \text{otherwise.} \end{cases}$$

Again, it is straightforward to check that this fulfils the requirements. This completes the proof of the theorem. ∎

6 Deterministic Primitives

In this section, we consider *delegation networks with deterministic primitives*. These are delegation networks whose interpretation π of primitives is deterministic.[20] Looking at Example 5.1, the use of jungles in Theorem 5.6 seems to be essential only in the presence of nondeterministic primitives. In a delegation network with deterministic primitives, multiple evaluations of a tree always yield the same result, and never fail. Therefore, neither sharing nor garbage should be needed in the case of deterministic primitives. We confirm this intuition by proving another MW-like result, where the syntactic generation phase yields trees rather than jungles, but only delegation networks with deterministic primitives are considered.

We start by observing that, in the deterministic case, Lemma 5.3 holds even without the assumption of minimal sharing and absence of garbage roots; cf. the discussion following the proof of Lemma 5.3. (The next lemma can already be found in [AG68, Lemma 1.16].)

Lemma 6.1 Let σ be a deterministic Σ-interpretation. For all jungles $J \in \mathrm{J}_{\Sigma,n}$, we have $\sigma(\mathrm{tree}(J)) = \sigma(J)$.

Proof Consider $a_1, \ldots, a_n \in \mathbb{A}$. Since σ is deterministic, an obvious induction can be used to verify that $ASS_{J,\sigma}(a_1, \ldots, a_n) = \{\alpha\}$, where α is given by $\alpha(v) = \sigma(\mathrm{tree}(J, v))(a_1, \ldots, a_n)$ for all $v \in V_J$. In particular, $\sigma(J)(a_1, \ldots, a_n) = \alpha(res(J)) = \sigma(\mathrm{tree}(J))(a_1, \ldots, a_n)$, as claimed. ∎

We can now prove our second MW-like result. For this, given a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$, let $\mathcal{N}_{\mathrm{free}} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathrm{T}_\Pi, free_\Pi)$.

[19] Note that α is defined on all shared nodes and garbage roots of J. This should be obvious for shared nodes. For garbage roots, it follows from the fact that all garbage roots of $\mathrm{J}_k(s')$ are in $[par(\mathrm{J}_k(s'))]$, which means that every garbage root of J must be of the form $img(v)$, where v is a garbage root of J' or is in $[src_{J'}(e)]$. In both cases, $\alpha'(v)$ is defined.

[20] Recall that π is said to be deterministic if $\pi(\mathtt{f})$ is a function for every $\mathtt{f} \in \Pi$.

Theorem 6.2 For every delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathsf{g}_0, \mathbb{A}, \pi)$ with deterministic primitives, $L(\mathcal{N}) = \pi(L(\mathcal{N}_{\text{free}}))$.

Proof By Theorem 5.6 and Lemma 6.1, the equation

$$L(\mathcal{N}) = \pi(L_{\mathrm{J}}(\mathcal{N})) = \pi(\mathrm{tree}(L_{\mathrm{J}}(\mathcal{N}))) \tag{2}$$

holds. Since $\mathcal{N}_{\text{free}}$ has deterministic primitives as well, free_Π is the identity on T_Π, and $L_{\mathrm{J}}(\mathcal{N}) = L_{\mathrm{J}}(\mathcal{N}_{\text{free}})$, another instance of this equation is

$$L(\mathcal{N}_{\text{free}}) = \mathrm{tree}(L_{\mathrm{J}}(\mathcal{N})). \tag{3}$$

Thus, substituting the left-hand side of (3) for $\mathrm{tree}(L_{\mathrm{J}}(\mathcal{N}))$ in (2) yields the claimed equality $L(\mathcal{N}) = \pi(L(\mathcal{N}_{\text{free}}))$. ∎

Motivated by the previous result, we now ask ourselves whether $L(\mathcal{N}_{\text{free}})$ can be characterized in terms of derivations on trees. We show that such a characterization is obtained by using the well-known mechanism of IO context-free tree rewriting, using all IO context-free rules of the form $\mathsf{g}[x_1, \ldots, x_k] \to u$, where $\mathsf{g}^{(k)} \in \mathcal{G}$ and $u \in L(\gamma_{\mathsf{g}})$.

Definition 6.3 Let $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathsf{g}})_{\mathsf{g}\in\mathcal{G}}, \mathsf{g}_0, \mathbb{A}, \pi)$ be a delegation network, and $t, t' \in \mathrm{T}_{\Sigma_\mathcal{N}}$. Then $t \Rightarrow_{\mathcal{N},\mathrm{IO}} t'$ if t can be decomposed as $t = t_0 \cdot \mathsf{g}[s_1, \ldots, s_k]$ such that

- $k \in \mathbb{N}$, $\mathsf{g}^{(k)} \in \mathcal{G}$ and $s_1, \ldots, s_k \in \mathrm{T}_\Pi$, and
- $t' = t_0 \cdot u(s_1, \ldots, s_k)$ for a tree $u \in L(\gamma_{\mathsf{g}})$.

The *tree language generated by* $\mathcal{N}$ is given by $L_{\mathrm{T}}(\mathcal{N}) = \{s \in \mathrm{T}_\Pi \mid \mathsf{g}_0 \Rightarrow^*_{\mathcal{N},\mathrm{IO}} s\}$.

Note that as far as $L_{\mathrm{T}}(\mathcal{N})$ is concerned, the last two components of a delegation network, which define the interpretation of primitives, are irrelevant. Thus, as mentioned after Definition 2.4, finitary delegation networks[21] with the semantics $L_{\mathrm{T}}(\mathcal{N})$ may be identified with IO context-free tree grammars. In fact, looking at the general case, we may consider the class $F_\mathcal{N}$ of all finitary delegation networks $\mathcal{N}' = (\mathcal{G}, \Pi, (\gamma'_{\mathsf{g}})_{\mathsf{g}\in\mathcal{G}}, \mathsf{g}_0, \mathbb{A}, \pi)$ such that, for all $\mathsf{g} \in \mathcal{G}$, $L(\gamma'_{\mathsf{g}}) \subseteq L(\gamma_{\mathsf{g}})$. Then, obviously, $L_{\mathrm{T}}(\mathcal{N}) = \bigcup_{\mathcal{N}'\in F_\mathcal{N}} L_{\mathrm{T}}(\mathcal{N}')$.

Thus, still viewing $\mathcal{N}$ as a tree-generating device in the sense of Definition 6.3, it may be seen as an IO context-free tree grammar which is *extended* in the sense that the set of right-hand sides for a given left-hand side $\mathsf{g}[x_1, \ldots, x_k]$ may be an infinite language taken from some class C of tree languages (i.e., $L(\gamma_{\mathsf{g}}) \in C$). This is similar to the notion of C-extended context-free string grammars and ET0L systems, which have been studied in the context of AFL theory; see, e.g., [Lee74, Asv78]. Recall also the *extended Backus-Naur Form*, where the adjective extended has a similar meaning.

These relationships explain why the results proved in this section (except Corollary 6.5) can be seen as "extended" versions of results in [ES77, ES78]. The

[21] Recall that $\mathcal{N}$ is said to be finitary if each of the tree languages $L(\gamma_{\mathsf{g}})$, $\mathsf{g} \in \mathcal{G}$, is finite.

next theorem shows that $L_{\mathrm{T}}(\mathcal{N})$ provides the desired operational characterization of $L(\mathcal{N}_{\mathrm{free}})$. In the other direction, given the definition of $L(\mathcal{N}_{\mathrm{free}})$, this can be seen as a least fixed point characterization of $L_{\mathrm{T}}(\mathcal{N})$, and thus as the extended version of the least fixed point characterization of the IO context-free tree languages discussed after Definition 5.12 in [ES78].

Theorem 6.4 For every delegation network $\mathcal{N}$, $L_{\mathrm{T}}(\mathcal{N}) = L(\mathcal{N}_{\mathrm{free}})$.

Proof Let $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathsf{g}})_{\mathsf{g}\in\mathcal{G}}, \mathsf{g}_0, \mathrm{T}_\Pi, \mathit{free}_\Pi)$, assuming without loss of generality that $\mathcal{N} = \mathcal{N}_{\mathrm{free}}$. By Theorem 3.5, it suffices to show that

$$L_{\mathrm{T}}(\mathcal{N}) = \{s \in \mathrm{T}_\Pi \mid \mathsf{g}_0 \rightarrowtail^*_{\mathcal{N}} \langle s\rangle\}.$$

Recall that $\underline{\Sigma}_{\mathcal{N}}$ consists of all symbols in $\Sigma_{\mathcal{N}}$, and all constant symbols $\langle s\rangle$ such that $s \in \mathrm{T}_\Pi$. Given a tree $t \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}},\{\Box\}}$, we make use of the following notation: $\mathit{rem}(t)$ denotes the tree in $\mathrm{T}_{\Sigma_{\mathcal{N}},\{\Box\}}$ obtained from t by replacing every occurrence of all symbols of the form $\langle s\rangle$ ($s \in \mathrm{T}_\Pi$) by the tree s.[22] Observe that, for all trees $t' \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}}}$ with $\mathit{rem}(t') \in \mathrm{T}_\Pi$, we have $t' \rightarrowtail^*_{\mathcal{N}} \langle \mathit{rem}(t')\rangle$, by performing derivation steps according to the first case of Definition 3.2.

We show by induction on the length of derivations that, for every tree $t \in \mathrm{T}_{\Sigma_{\mathcal{N}}}$, there is a derivation $\mathsf{g}_0 \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t$ if and only if there is a tree $z \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}}}$ such that $\mathsf{g}_0 \rightarrowtail^*_{\mathcal{N}} z$ and $\mathit{rem}(z) = t$. This completes the proof since, by the observation above, $z \rightarrowtail^*_{\mathcal{N}} \langle s\rangle$ in the special case where $\mathit{rem}(z) = s \in \mathrm{T}_\Pi$.

Clearly, the statement is trivially true in both directions for derivations of length 0, because $\mathit{rem}(\mathsf{g}_0) = \mathsf{g}_0$. It remains to be shown that single steps preserve the equivalence.

('⇒') Assume that $t \Rightarrow_{\mathcal{N},\mathrm{IO}} t'$, where the trees $t = t_0 \cdot \mathsf{g}[s_1, \ldots, s_k]$ and $t' = t_0 \cdot u(s_1, \ldots, s_k)$ are as in Definition 6.3. In particular, $s_1, \ldots, s_k \in \mathrm{T}_\Pi$. Let $z \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}}}$ be such that $\mathit{rem}(z) = t$. Thus, z has the form $z = z_0 \cdot \mathsf{g}[z_1, \ldots, z_k]$, where $\mathit{rem}(z_0) = t_0$ and $\mathit{rem}(z_i) = s_i$ for all $i \in [k]$. By the observation above and the second case of Definition 3.2, this yields

$$z \rightarrowtail^*_{\mathcal{N}} z_0 \cdot \mathsf{g}[\langle s_1\rangle, \ldots, \langle s_k\rangle] \rightarrowtail_{\mathcal{N}} z_0 \cdot u(\langle s_1\rangle, \ldots, \langle s_k\rangle),$$

which ends this part of the proof, as $\mathit{rem}(z_0 \cdot u(\langle s_1\rangle, \ldots, \langle s_k\rangle)) = t_0 \cdot u(s_1, \ldots, s_k)$.

('⇐') Let $z \rightarrowtail_{\mathcal{N}} z'$ with $\mathit{rem}(z) = t$. If this step is performed according to the first case of Definition 3.2, then $\mathit{rem}(z') = t$, so there is nothing to show, because $t \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t$. Therefore, assume that $z = z_0 \cdot \mathsf{g}[\langle s_1\rangle, \ldots, \langle s_k\rangle]$ and $z' = z_0 \cdot u(\langle s_1\rangle, \ldots, \langle s_k\rangle)$. Setting $t_0 = \mathit{rem}(z_0)$, we get $t = t_0 \cdot \mathsf{g}[s_1, \ldots, s_k] \Rightarrow_{\mathcal{N},\mathrm{IO}} t_0 \cdot u(s_1, \ldots, s_k) = \mathit{rem}(z')$, as required. ∎

Using (3), we get the following as an immediate consequence.

[22] Intuitively, all that *rem* does is remove all angular brackets. Thus, as a recursive definition, $\mathit{rem}(\mathsf{g}[t_1, \ldots, t_k]) = \mathsf{g}[\mathit{rem}(t_1), \ldots, \mathit{rem}(t_k)]$ for all $\mathsf{g}^{(k)} \in \Sigma_{\mathcal{N}}$ and $t_1, \ldots, t_k \in \mathrm{T}_{\underline{\Sigma}_{\mathcal{N}},\{\Box\}}$, $\mathit{rem}(\langle s\rangle) = s$ for all trees $s \in \mathrm{T}_\Pi$, and $\mathit{rem}(\Box) = \Box$.

Corollary 6.5 For every delegation network $\mathcal{N}$, $L_{\mathrm{T}}(\mathcal{N}) = \mathrm{tree}(L_{\mathrm{J}}(\mathcal{N}))$.

As another immediate consequence, we obtain an alternative formulation of our MW-like result in Theorem 6.2, similar to Theorem 5.6. For the finitary case, this was proved in [ES78, Theorem 5.10, Corollary 5.11]; see also the discussion after Definition 2.4.

Theorem 6.6 For every delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathsf{g}_0, \mathbb{A}, \pi)$ with deterministic primitives, $L(\mathcal{N}) = \pi(L_{\mathrm{T}}(\mathcal{N}))$.

The failure of (the finitary case of) Theorem 6.6 for delegation networks with nondeterministic primitives was pointed out in [ES78, p. 72]. Theorem 5.6, whose finitary counterpart is missing in [ES77, ES78], can be seen as a solution to this problem. Intuitively, it shows that one has to avoid copying (by sharing) and deletion (by garbage) to make Theorem 6.6 work in the presence of nondeterministic primitives. This intuition gives rise to an obvious conjecture, namely that the assumption of deterministic primitives in Theorem 6.6 can be dropped if neither copying nor deletion takes place. We conclude this section with the observation that this is indeed true. More precisely, let a *linear nondeleting delegation network* $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathsf{g}_0, \mathbb{A}, \pi)$ be a delegation network such that, for all $\mathsf{g}^{(k)} \in \mathcal{G}$, all trees $t \in L(\gamma_{\mathsf{g}})$ are linear and nondeleting in X_k. (Recall that this means that the parameter symbol x_i occurs exactly once in t, for every $i \in [k]$.)

Theorem 6.7 For every linear nondeleting delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathsf{g}_0, \mathbb{A}, \pi)$, we have $L(\mathcal{N}) = \pi(L_{\mathrm{T}}(\mathcal{N}))$.

Proof For the proof, we make use of the "unrestricted" derivation relation $\Rightarrow_{\mathcal{N},\mathrm{unr}}$, which is defined exactly as $\Rightarrow_{\mathcal{N},\mathrm{IO}}$ in Definition 6.3, except that the trees $s_1, \ldots, s_k$ (the subtrees of the generator symbol that is being replaced) are allowed to be arbitrary trees in $\mathrm{T}_{\Sigma_{\mathcal{N}},\{\Box\}}$. Thus, in contrast to $\Rightarrow_{\mathcal{N},\mathrm{IO}}$, arbitrary generator symbols rather than only the innermost ones can be replaced. To be able to use tree concatenation, we also consider rewriting of trees containing the parameter $\Box$. Thus, it makes sense to note that, for every tree $t = t_0 \cdot t_1$, if $t_0 \Rightarrow_{\mathcal{N},\mathrm{unr}} t_0'$ for a tree t_0', then $t \Rightarrow_{\mathcal{N},\mathrm{unr}} t_0' \cdot t_1$. In particular, $t_0' \cdot t_1$ is defined, because t_0' contains exactly one occurrence of $\Box$: thanks to the assumption that $\mathcal{N}$ is linear and nondeleting, the unique occurrence of $\Box$ in t_0 is neither deleted nor copied. We define the tree language generated by $\mathcal{N}$ in unrestricted derivation mode to be $L_{\mathrm{T},\mathrm{unr}}(\mathcal{N}) = \{s \in \mathrm{T}_{\Pi} \mid \mathsf{g}_0 \Rightarrow^*_{\mathcal{N},\mathrm{unr}} s\}$.

For a derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^n_{\mathcal{N}} J$, using the assumption that $\mathcal{N}$ is linear and nondeleting, it follows by a straightforward induction on n that J is a jungle containing neither garbage roots nor shared nodes, and so $J = \mathrm{J}_0(\mathrm{tree}(J))$. In particular, every hyperedge $e \in E_J$ corresponds to a unique occurrence of $lab_J(e)$ in $\mathrm{tree}(J)$. Therefore, every derivation $\mathsf{g}_0 \Rightarrow^n_{\mathcal{N},\mathrm{unr}} s$ can, in the obvious way, be turned into a derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^n_{\mathcal{N}} \mathrm{J}_0(s)$; vice versa, every derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^n_{\mathcal{N}}$

$\mathrm{J}_0(s)$ can be turned into a derivation $\mathsf{g}_0 \Rightarrow^n_{\mathcal{N},\mathrm{unr}} s$. This shows that

$$L_{\mathrm{J}}(\mathcal{N}) = \mathrm{J}_0(L_{\mathrm{T,unr}}(\mathcal{N})). \tag{4}$$

Next we show that $L_{\mathrm{T,unr}}(\mathcal{N})$, in fact, coincides with $L_{\mathrm{T}}(\mathcal{N})$:

$$L_{\mathrm{T}}(\mathcal{N}) = L_{\mathrm{T,unr}}(\mathcal{N}). \tag{5}$$

To prove (5), we show by induction on $n \in \mathbb{N}$ that the following is true for all trees $t = t_0 \cdot \mathsf{g}[s_1, \ldots, s_k] \in \mathrm{T}_{\Sigma_\mathcal{N}}$ with $\mathsf{g}^{(k)} \in \mathcal{G}$ and $s_1, \ldots, s_k \in \mathrm{T}_\Pi$: if $t \Rightarrow^n_{\mathcal{N},\mathrm{unr}} s$ for a tree $s \in \mathrm{T}_\Pi$, then there is a tree $u \in L(\gamma_\mathsf{g})$ such that $t \Rightarrow_{\mathcal{N},\mathrm{unr}} t_0 \cdot u(s_1, \ldots, s_k) \Rightarrow^{n-1}_{\mathcal{N},\mathrm{unr}} s$. From this, it follows that $t \Rightarrow^n_{\mathcal{N},\mathrm{IO}} s$ (by induction, since $t \Rightarrow_{\mathcal{N},\mathrm{IO}} t_0 \cdot u(s_1, \ldots, s_k)$), which proves (5). To prove the property claimed, if the derivation $t \Rightarrow^n_{\mathcal{N},\mathrm{unr}} s$ is not already of the form required, it has the form $t \Rightarrow_{\mathcal{N},\mathrm{unr}} t'_0 \cdot \mathsf{g}[s_1, \ldots, s_k] \Rightarrow^{n-1}_{\mathcal{N},\mathrm{unr}} s$, i.e., we have $t_0 \Rightarrow_{\mathcal{N},\mathrm{unr}} t'_0$. By the induction hypothesis, $t'_0 \cdot \mathsf{g}[s_1, \ldots, s_k] \Rightarrow_{\mathcal{N},\mathrm{unr}} t'_0 \cdot u(s_1, \ldots, s_k) \Rightarrow^{n-2}_{\mathcal{N},\mathrm{unr}} s$ for an appropriate tree $u \in L(\gamma_\mathsf{g})$. It follows that

$$\begin{aligned} t &\Rightarrow_{\mathcal{N},\mathrm{unr}} t_0 \cdot u(s_1, \ldots, s_k) && \text{(by the definition of } \Rightarrow_{\mathcal{N},\mathrm{unr}}) \\ &\Rightarrow_{\mathcal{N},\mathrm{unr}} t'_0 \cdot u(s_1, \ldots, s_k) && \text{(since } t_0 \Rightarrow_{\mathcal{N},\mathrm{unr}} t'_0) \\ &\Rightarrow^{n-2}_{\mathcal{N},\mathrm{unr}} s, \end{aligned}$$

as claimed.

Finally, using (4) and (5), we get

$$\begin{aligned} L(\mathcal{N}) &= \pi(L_{\mathrm{J}}(\mathcal{N})) && \text{(by Theorem 5.6)} \\ &= \pi(\mathrm{J}_0(L_{\mathrm{T,unr}}(\mathcal{N}))) && \text{(by (4))} \\ &= \pi(L_{\mathrm{T,unr}}(\mathcal{N})) && \text{(by Lemma 5.3)} \\ &= \pi(L_{\mathrm{T}}(\mathcal{N})) && \text{(by (5)),} \end{aligned}$$

which completes the proof of the theorem. ∎

We note here that the nondeleting property of $\mathcal{N}$ is essential for the correctness of (5). In [KM06, Corollary 5] it is wrongly stated that the corresponding equation holds for linear context-free tree languages, which is not the case unless the context-free tree grammar is assumed to be nondeleting. The erroneous corollary is, moreover, used in [SO07, Section 3].

7 Delegation as an Operator on Language Classes

In this section and the next, we view delegation networks as tree generators in the sense of Definition 6.3. Just as $L(\mathcal{N})$, the tree language $L_{\mathrm{T}}(\mathcal{N})$ generated by a delegation network $\mathcal{N}$ depends only on the tree languages $L(\gamma_\mathsf{g})$; the specific devices γ_g generating them are unimportant. Delegation, in the form realized by delegation networks, can therefore be considered as an operator on classes of tree languages (see also Example 2.9). The next definition formalizes this and introduces the resulting hierarchy of tree language classes, which we call

the delegation hierarchy. The tree languages in this hierarchy are generated by delegation networks that themselves use delegation networks as tree generators.

Definition 7.1 For every class C of tree languages, $\mathrm{DEL}(C)$ denotes the class of all tree languages of the form $L_{\mathrm{T}}(\mathcal{N})$, such that $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0, \mathbb{A}, \pi)$ is a delegation network with $L(\gamma_{\mathtt{g}}) \in C$, for all $\mathtt{g} \in \mathcal{G}$ (where $\Gamma = (\gamma_{\mathtt{g}}, Y_{\mathtt{g}})_{\mathtt{g} \in \mathcal{G}}$). Furthermore, $\mathrm{DEL}^0(C) = C$, $\mathrm{DEL}^{n+1}(C) = \mathrm{DEL}(\mathrm{DEL}^n(C))$ for all $n \in \mathbb{N}$, and $\mathrm{DEL}^*(C) = \bigcup_{n \in \mathbb{N}} \mathrm{DEL}^n(C)$.

The *delegation hierarchy* is the hierarchy $(\mathrm{DEL}^n(\mathrm{FIN}))_{n \in \mathbb{N}}$, where FIN denotes the class of all finite tree languages.

A delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}}, Y_{\mathtt{g}})_{\mathtt{g} \in \mathcal{G}}, \mathtt{g}_0, \mathbb{A}, \pi)$ is *hierarchical* if it is n-hierarchical for some $n \geq 1$; it is 1-hierarchical if it is finitary (i.e., if $L(\gamma_{\mathtt{g}})$ is finite for every $\mathtt{g} \in \mathcal{G}$), and, for $n \geq 2$, it is n-hierarchical if $\gamma_{\mathtt{g}}$ is an $(n-1)$-hierarchical delegation network for every $\mathtt{g} \in \mathcal{G}$ (with $L(\gamma_{\mathtt{g}}) = L_{\mathrm{T}}(\gamma_{\mathtt{g}})$). Thus, $\mathrm{DEL}^*(\mathrm{FIN})$ is the class of tree languages generated by hierarchical delegation networks, and, for $n \geq 1$, $\mathrm{DEL}^n(\mathrm{FIN})$ is the class generated by n-hierarchical ones.

Since $L_{\mathrm{T}}(\mathcal{N})$ does not depend on the semantic components $\mathbb{A}$ and π of $\mathcal{N}$, as discussed after Definition 6.3, we will from now on denote a delegation network as a 4-tuple $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$.

We start our discussion of the delegation hierarchy with an example of a tree language in $\mathrm{DEL}^2(\mathrm{FIN})$. Note that $\mathrm{DEL}(\mathrm{FIN})$ is the class of tree languages generated by finitary delegation networks, i.e., by IO context-free tree grammars (cf. the last paragraph of Example 2.7(a)). Hence, $\mathrm{DEL}^2(\mathrm{FIN})$ is the set of tree languages generated by delegation networks of which each tree generator is an IO context-free tree grammar.

Example 7.2 This example, of a tree language in $\mathrm{DEL}^2(\mathrm{FIN})$, is similar to Example 2.9(a). We first observe that the tree language consisting of all fully balanced binary trees of height ≥ 1 over the signature $\{\mathtt{f}^{(2)}, \mathtt{h}^{(0)}\}$ is in $\mathrm{DEL}(\mathrm{FIN})$: it is $L_{\mathrm{T}}(\mathcal{N}_{\mathrm{bin}})$ for the delegation network $\mathcal{N}_{\mathrm{bin}}$ with generator symbols $\mathtt{g}_0^{(0)}$ and $\mathtt{g}^{(1)}$, and with $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{h}]\}$ and $L(\gamma_{\mathtt{g}}) = \{\mathtt{g}[\mathtt{f}[x_1, x_1]], \mathtt{f}[x_1, x_1]\}$ where $Y_{\mathtt{g}} = \{x_1\}$. Now consider the language L_{expseq} consisting of all trees of the form $s \boldsymbol{\cdot} (t_1, \ldots, t_{2^n})$, $n \geq 1$, where s is a fully balanced binary tree of height n over the signature $\{\mathtt{f}, \square_1, \ldots, \square_{2^n}\}$ and $t_1, \ldots, t_{2^n}$ are arbitrary trees over the signature $\{\mathtt{k}^{(2)}, \mathtt{a}^{(0)}\}$. This language is equal to $L_{\mathrm{T}}(\mathcal{N}_{\mathrm{expseq}})$ for the 2-hierarchical delegation network $\mathcal{N}_{\mathrm{expseq}}$ with generator symbols $\mathtt{h}_0^{(0)}$ and $\mathtt{h}^{(0)}$, and with $L(\gamma_{\mathtt{h}_0}) = L_{\mathrm{T}}(\mathcal{N}_{\mathrm{bin}})$ and $L(\gamma_{\mathtt{h}}) = \mathrm{T}_{\{\mathtt{k},\mathtt{a}\}}$. Obviously, $\mathrm{T}_{\{\mathtt{k},\mathtt{a}\}}$ is in $\mathrm{DEL}(\mathrm{FIN})$ (with $L(\gamma_{\mathtt{g}_0'}) = \{\mathtt{k}[\mathtt{g}_0', \mathtt{g}_0'], \mathtt{a}\}$). Hence L_{expseq} is in $\mathrm{DEL}^2(\mathrm{FIN})$. ∎

In the preceding example, the tree generators $\gamma_{\mathtt{h}_0} = \mathcal{N}_{\mathrm{bin}}$ and $\gamma_{\mathtt{h}}$ of the 2-hierarchical delegation network $\mathcal{N}_{\mathrm{expseq}}$ do not have parameters, whereas the tree generator $\gamma_{\mathtt{g}}$ of the 1-hierarchical delegation network $\mathcal{N}_{\mathrm{bin}}$ does have parameters. In general however, for an n-hierarchical delegation network $\mathcal{N} =$

$(\mathcal{G}, \Pi, (\gamma_{\mathtt{g}}, Y_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0, \mathbb{A}, \pi)$, the trees generated by the $(n-1)$-hierarchical delegation network $\gamma_{\mathtt{g}}$ contain the parameters in $Y_{\mathtt{g}}$. As a consequence, the tree generators of the network $\gamma_{\mathtt{g}}$ should use parameters that are not in $Y_{\mathtt{g}}$ (see Example 2.9(b)). A careful treatment of this formal detail will be important in Lemma 7.22 and Theorem 7.23 (and their proofs); otherwise, we will assume as usual that $Y_{\mathtt{g}} = X_k$ where k is the rank of $\mathtt{g}$.

In the remainder of the paper, we want to study the class DEL*(FIN) of tree languages generated by hierarchical delegation networks, and compare it with known tree language classes. We start by looking at the so-called path languages of tree languages in DEL*(FIN).

For a signature Σ, let $\langle\Sigma\rangle$ be the (finite unranked) alphabet

$$\langle\Sigma\rangle = \{\langle \mathtt{f}, j\rangle \mid k \in \mathbb{N}, \mathtt{f}^{(k)} \in \Sigma, 0 \leq j \leq k\}.$$

Given a tree $t \in \mathrm{T}_{\Sigma,n}$ and $i \in [n]$, we let $\mathrm{paths}_i(t) \subseteq \langle\Sigma\rangle^*$ be the string language of all downwards directed paths in t from the root to an occurrence of x_i (where x_i is excluded). Formally,

$$\mathrm{paths}_i(t) = \begin{cases} \{\lambda\} & \text{if } t = x_i \\ \emptyset & \text{if } t = x_j,\ j \neq i \\ \{\langle \mathtt{f}, j\rangle p \mid j \in [k], p \in \mathrm{paths}_i(t_j)\} & \text{if } t = \mathtt{f}[t_1, \ldots, t_k], \text{ with } \mathtt{f} \in \Sigma. \end{cases}$$

Thus, a symbol $\langle \mathtt{f}, j\rangle$ indicates that the path continues at the jth subtree of the symbol $\mathtt{f}$. The string language of paths from the root ending at a non-parameter symbol, a subset of $\langle\Sigma\rangle^+$, is denoted by $\mathrm{paths}(t)$. Thus, $\mathrm{paths}(x_i) = \emptyset$ for all $i \in [n]$, and, for $t = \mathtt{f}[t_1, \ldots, t_k]$ with $\mathtt{f}^{(k)} \in \Sigma$,

$$\mathrm{paths}(t) = \{\langle \mathtt{f}, 0\rangle\} \cup \{\langle \mathtt{f}, j\rangle p \mid j \in [k], p \in \mathrm{paths}(t_j)\}.$$

For technical convenience, we use 'paths_0' as a synonym for 'paths'. For a tree language $L \subseteq \mathrm{T}_{\Sigma,n}$ and $i \in \{0, \ldots, n\}$, we let $\mathrm{paths}_i(L) = \bigcup_{t\in L} \mathrm{paths}_i(t)$. For a jungle $J \in \mathrm{J}_{\Sigma,n}$, let $\mathrm{paths}_i(J) = \mathrm{paths}_i(\mathrm{tree}(J))$, and similarly for jungle languages. If L is a tree language or a jungle language, then $\mathrm{paths}(L)$ is called the *path language of* L. Note that, by Corollary 6.5, for every delegation network $\mathcal{N}$, $L_{\mathrm{J}}(\mathcal{N})$ and $L_{\mathrm{T}}(\mathcal{N})$ have the same path language. This will be the key observation that is used to prove the main result of this section.

Of course, $\mathrm{paths}_i(J)$ can easily be obtained from the jungle paths in J without taking the detour via $\mathrm{tree}(J)$. For $i \in [n]$, it should be clear that $\mathrm{paths}_i(J)$ is equal to the set of all $\langle lab_J(e_1), j_1\rangle \cdots \langle lab_J(e_m), j_m\rangle$, such that $(e_1, j_1) \cdots (e_m, j_m)$ is a jungle path from $res(J)$ to $par(J, i)$. Similarly, $\mathrm{paths}(J)$ is the set of all sequences of the form $\langle lab_J(e_1), j_1\rangle \cdots \langle lab_J(e_m), j_m\rangle\langle lab_J(e), 0\rangle$, where $e \in E_J$, and $(e_1, j_1) \cdots (e_m, j_m)$ is a jungle path from $res(J)$ to $res_J(e)$.

Our intention is to prove in this section that the path language of every tree language in DEL*(FIN) is a context-free language. The proof would be easy if we would only consider proper delegation networks in Definition 7.1, where a

delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$ is *proper* if for every $t \in \mathrm{T}_{\Sigma_\mathcal{N}}$ there exists $t' \in \mathrm{T}_\Pi$ such that $t \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t'$.

To sketch the proof of this result (for the proper case), recall that a context-free grammar is a tuple $G = (\Xi, \Sigma, R, S)$ where Ξ and Σ are disjoint finite sets of nonterminals and terminals, $S \in \Xi$, and R is a finite set of rules of the form $\mathtt{f} \to \alpha$ with $\mathtt{f} \in \Xi$ and $\alpha \in (\Xi \cup \Sigma)^*$. A derivation step is denoted $\beta_1 \mathtt{f} \beta_2 \Rightarrow_G \beta_1 \alpha \beta_2$, and $L(G) = \{w \in \Sigma^* \mid S \Rightarrow^*_G w\} = \{w \in \Sigma^* \mid S \Rightarrow^*_{G,\mathrm{rm}} w\}$ where $\Rightarrow^*_{G,\mathrm{rm}}$ indicates rightmost derivations (meaning that $\beta_2 \in \Sigma^*$). A *cf-extended* context-free grammar is the same as a context-free grammar except that R can be infinite, provided the language $\{\alpha \mid (\mathtt{f} \to \alpha) \in R\}$ is context-free, for every $\mathtt{f} \in \Xi$. It is well known (and easy to see) that every cf-extended context-free grammar has an equivalent context-free grammar.

Proposition 7.3 Let $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$ be a proper delegation network such that $\mathrm{paths}_i(L(\gamma_\mathtt{g}))$ is context-free for every $\mathtt{g}^{(k)} \in \mathcal{G}$ and every $i \in \{0, \ldots, k\}$. Then $\mathrm{paths}(L_\mathrm{T}(\mathcal{N}))$ is context-free.

Proof sketch The proof is similar to the one of [ES79, Corollary 4.2], which treats the case where $\mathcal{N}$ is finitary. We construct the cf-extended context-free grammar $G = (\langle\mathcal{G}\rangle, \langle\Pi\rangle, R, \langle \mathtt{g}_0, 0\rangle)$ where R is the set of all rules $\langle \mathtt{g}, i\rangle \to p$ with $p \in \mathrm{paths}_i(L(\gamma_\mathtt{g}))$. It can be shown (cf. Lemmas 4.2 and 4.3 of [ES79]) that for every $w \in \langle\Sigma_\mathcal{N}\rangle^*$, $\langle \mathtt{g}_0, 0\rangle \Rightarrow^*_{G,\mathrm{rm}} w$ if and only if there exists $t \in \mathrm{T}_{\Sigma_\mathcal{N}}$ with $w \in \mathrm{paths}(t)$, such that $\mathtt{g}_0 \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t$. Properness of $\mathcal{N}$ is used in the proof of the only-if direction, and it is used to conclude from this statement that $L(G) = \mathrm{paths}(L_\mathrm{T}(\mathcal{N}))$.

To illustrate the need for properness, consider the delegation network $\mathcal{N}$ with $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{g}[\mathtt{a}, \mathtt{g}']\}$, $L(\gamma_\mathtt{g}) = \{x_1\}$ and $L(\gamma_{\mathtt{g}'}) = \emptyset$. Then $\mathrm{paths}(L_\mathrm{T}(\mathcal{N})) = \emptyset$, but G has a derivation $\langle \mathtt{g}_0, 0\rangle \Rightarrow \langle \mathtt{g}, 1\rangle\langle \mathtt{a}, 0\rangle \Rightarrow \langle \mathtt{a}, 0\rangle$. ∎

It would not be difficult to generalize this lemma in such a way that it can be used iteratively, showing that the path language of every tree language in $\mathrm{DEL}^*(\mathrm{FIN})$ is context-free, provided only proper delegation networks are used. Now, it is easy to transform a delegation network $\mathcal{N}$ into an equivalent proper one (assuming that $L_\mathrm{T}(\mathcal{N}) \neq \emptyset$). In fact, as for context-free grammars, it suffices to determine the set $\mathrm{use}(\mathcal{N}) \subseteq \mathcal{G}$ of *useful* generator symbols: it is the smallest subset of $\mathcal{G}$ such that for every $\mathtt{g}^{(k)} \in \mathcal{G}$ and $t \in L(\gamma_\mathtt{g})$, if $t \in \mathrm{T}_{\mathrm{use}(\mathcal{N})\cup\Pi,k}$ then $\mathtt{g} \in \mathrm{use}(\mathcal{N})$. An equivalent proper network is then obtained from $\mathcal{N}$ by removing all generator symbols that are not useful and by replacing $L(\gamma_\mathtt{g})$ with $L(\gamma_\mathtt{g}) \cap \mathrm{T}_{\mathrm{use}(\mathcal{N})\cup\Pi,k}$ for every $\mathtt{g}^{(k)} \in \mathrm{use}(\mathcal{N})$. The problem with this construction is that we do not know whether $\mathrm{DEL}^n(\mathrm{FIN})$ is closed under intersection with regular tree languages, not even when they are of the form T_Σ for some signature Σ. Thus, we also do not know whether the restriction to proper delegation networks leads to the same class $\mathrm{DEL}^*(\mathrm{FIN})$ or to a smaller one, and for that reason we do not wish to impose that restriction (though, as we will show after Corollary 8.16, it

is decidable whether a hierarchical delegation network is proper).

Thus, the straightforward approach to proving that the path language of every tree language in DEL*(FIN) is context-free, is flawed. We now start our, rather technical, alternative proof. We first show that it suffices to consider a restriction of the operator DEL, because this restriction will be needed later. Let us call a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}}, Y_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ *propagating* if $L(\gamma_{\mathtt{g}}) \cap Y_{\mathtt{g}} = \emptyset$, for all $\mathtt{g} \in \mathcal{G}$. We let pDEL denote the operator which is defined in the same way as DEL, except that only propagating delegation networks are considered. The definitions of pDEL$^n(C)$ and pDEL$^*(C)$ carry over from the general case in the obvious way.

Now, let $*^{(1)} \notin \Pi$ be a special symbol reserved for its use in the following context. Define erase$_*$: $\mathrm{T}_{\Pi\cup\{*\}} \to \mathrm{T}_\Pi$ to be the mapping given by erase$_*(*[t]) =$ erase$_*(t)$ and erase$_*(\mathtt{f}[t_1, \ldots, t_k]) = \mathtt{f}[\mathrm{erase}_*(t_1), \ldots, \mathrm{erase}_*(t_k)]$ for all $\mathtt{f}^{(k)} \in \Pi$ and $t_1, \ldots, t_k \in \mathrm{T}_{\Pi\cup\{*\}}$. Hence, erase$_*$ is a linear and nondeleting tree homomorphism.[23] We have the following lemma.

Lemma 7.4 For every tree language $L \in$ DEL*(FIN), there is a tree language $L' \in$ pDEL*(FIN), such that $L =$ erase$_*(L')$.

Proof By induction on n, where $L \in$ DEL^n(FIN), we prove the more general statement that there is a tree language $L' \in$ pDEL*(FIN), such that $L =$ erase$_*(L')$ and the root label of every tree in L' is $*$.

For $n = 0$, this is obvious, by defining $L' = \{*[t] \mid t \in L\}$. For $n > 0$, let $L = L_{\mathrm{T}}(\mathcal{N})$, where $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ with $L(\gamma_{\mathtt{g}}) \in$ DEL^{n-1}(FIN). Using the induction hypothesis, define $\mathcal{N}' = (\mathcal{G}, \Pi \cup \{*\}, (\gamma'_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$, where $L(\gamma_{\mathtt{g}}) =$ erase$_*(L(\gamma'_{\mathtt{g}}))$ and each tree in $L(\gamma'_{\mathtt{g}})$ has $*$ as its root label, for all $\mathtt{g} \in \mathcal{G}$. Clearly, by looking at the very first step of a derivation in $\mathcal{N}'$, this means that each tree in $L_{\mathrm{T}}(\mathcal{N}')$ has $*$ as its root label.

It remains to argue that $L =$ erase$_*(L_{\mathrm{T}}(\mathcal{N}'))$. This follows directly from the following claim, using induction on the length of derivations.

Claim Let $s \in \mathrm{T}_{\Sigma_\mathcal{N}}$ and $t \in \mathrm{T}_{\Sigma_\mathcal{N}\cup\{*\}}$ be such that erase$_*(t) = s$. Then, for every derivation step $s \Rightarrow_{\mathcal{N},\mathrm{IO}} s'$, there is a derivation step $t \Rightarrow_{\mathcal{N}',\mathrm{IO}} t'$ such that erase$_*(t') = s'$. Vice versa, $s \Rightarrow_{\mathcal{N},\mathrm{IO}}$ erase$_*(t')$ is a derivation step, for every derivation step $t \Rightarrow_{\mathcal{N}',\mathrm{IO}} t'$.

For the first direction, following Definition 6.3, let $s = s_0 \cdot \mathtt{g}[z_1, \ldots, z_k]$ and $s' = s_0 \cdot u(z_1, \ldots, z_k)$, where $s_0 \in \mathrm{T}_{\Sigma_\mathcal{N},\{\Box\}}$, $\mathtt{g}^{(k)} \in \mathcal{G}$, $z_1, \ldots, z_k \in \mathrm{T}_\Pi$, and $u \in L(\gamma_{\mathtt{g}})$. Since erase$_*(t) = s$, it should be clear that we can write t as $t_0 \cdot \mathtt{g}(z'_1, \ldots, z'_k)$, where erase$_*(t_0) = s_0$ and erase$_*(z'_i) = z_i$ for all $i \in [k]$ (extending erase$_*$ to trees in $\mathrm{T}_{\Sigma_\mathcal{N},\{\Box\}}$ in the obvious way). In particular, since erase$_*$ is linear and nondeleting, t_0 contains exactly one occurrence of $\Box$, because s_0 does. Hence, defining $t' = t_0 \cdot u'(z'_1, \ldots, z'_k)$ for a tree $u' \in L(\gamma'_{\mathtt{g}})$ such

[23] Cf. the next section, after Definition 8.1.

that $\mathrm{erase}_*(u') = u$, we get $t \Rightarrow_{\mathcal{N}',\mathrm{IO}} t'$ and, by construction, $\mathrm{erase}_*(t') = s'$, as claimed.

We omit the proof of the other direction, because it is very similar to the preceding one. ∎

Together with Lemma 7.4, the next lemma shows that it suffices to consider tree languages in pDEL*(FIN) to establish that the path languages of tree languages in DEL*(FIN) are context-free.

Lemma 7.5 For every tree language L, if $\mathrm{paths}(L)$ is a context-free language, then $\mathrm{paths}(\mathrm{erase}_*(L))$ is a context-free language.

Proof Let $L \subseteq \mathrm{T}_{\Sigma\cup\{*\}}$. Obviously, $\mathrm{paths}(\mathrm{erase}_*(L)) = h(\mathrm{paths}(L) \cap R)$, where R is the regular language of all strings in $\langle \Sigma \cup \{*\}\rangle^*$ that do not end on $\langle *, 0\rangle$, and h is the string homomorphism that erases the symbol $\langle *, 1\rangle$. The result follows from the fact that the class of context-free languages is closed under homomorphisms and intersection with regular languages. ∎

Our proof of the context-freeness of $\mathrm{paths}(L)$ for $L \in \mathrm{pDEL}^*(\mathrm{FIN})$ makes use of the notion of context-free jungle languages. These are sets of jungles generated by the special case of hyperedge-replacement graph grammars in which all right-hand sides are jungles. Throughout the rest of this section, we assume that the reader is familiar with some of the most basic properties of hyperedge-replacement graph grammars.

Definition 7.6 (context-free jungle grammar) A *context-free jungle grammar* (abbreviated CFJG) is a tuple $G = (\Xi, \Sigma, R, \mathtt{f}_0)$ such that

- Ξ and Σ are disjoint finite signatures of *nonterminals* and *terminals*, resp.,
- R is a finite set of rules of the form $\mathtt{f} \to K$, where $\mathtt{f}^{(k)} \in \Xi$ and $K \in \mathrm{J}_{\Xi\cup\Sigma,k}$ ($k \in \mathbb{N}$), and
- $\mathtt{f}_0^{(n)} \in \Xi$, where $n \in \mathbb{N}$, is the *initial nonterminal*.

For jungles $J, J' \in \mathrm{J}_{\Xi\cup\Sigma,n}$, there is a derivation step $J \Rightarrow_G J'$ if there are a hyperedge $e \in E_J$ and a rule $lab_J(e) \to K$ in R, such that $J' = J[e \leftarrow K]$. The *jungle language generated by* G is given by

$$L(G) = \{J \in \mathrm{J}_{\Sigma,n} \mid \mathrm{J}(\mathtt{f}_0) \Rightarrow_G^* J\}.$$

A jungle language that can be generated by a CFJG is said to be *context-free*. An *extended CFJG*, its derivations, and the generated jungle language are defined in precisely the same way, with the exception that R may be infinite. Clearly, for a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$, the jungle language $L_{\mathrm{J}}(\mathcal{N})$, as defined in Definition 5.5, is generated by the extended CFJG $(\mathcal{G}, \Pi, R, \mathtt{g}_0)$ where R consists of all rules $\mathtt{g} \to \mathrm{J}_k(t)$ with $\mathtt{g}^{(k)} \in \mathcal{G}$ and $t \in L(\gamma_{\mathtt{g}})$. A *cf-extended CFJG* is an extended CFJG such that the jungle language $L_{\mathtt{f}} = \{J \mid (\mathtt{f} \to J) \in R\}$ is context-free, for every $\mathtt{f} \in \Xi$. In this case, we denote by $\mathtt{f} \to L_{\mathtt{f}}$ the set of all rules in R with left-hand side $\mathtt{f}$.

For a CFJG (and similarly for an extended CFJG) $G = (\Xi, \Sigma, R, \mathtt{f}_0)$, we say that a nonterminal $\mathtt{f}^{(k)} \in \Xi$ is *useful* if there exists $J \in \mathrm{J}_{\Sigma,k}$ such that $\mathrm{J}(\mathtt{f}) \Rightarrow_G^* J$. Obviously, all nonterminals (except $\mathtt{f}_0$) that are not useful can be discarded, together with the rules in which they occur, without changing $L(G)$. Therefore, except in the trivial case where $L(G) = \emptyset$, we may always assume without loss of generality that all nonterminals of a CFJG considered are useful.

To prove that every tree language in DEL*(FIN) has a context-free path language, we will show that for every tree language $L \in$ pDEL*(FIN) there is a context-free jungle language having the same path language as L. More precisely, for every hierarchical delegation network $\mathcal{N}$ with $L_{\mathrm{T}}(\mathcal{N}) = L$ we will define a CFJG G such that $L(G)$ has the same path language as $L_{\mathrm{J}}(\mathcal{N})$. Then the following lemma yields the main result of this section (together with Lemmas 7.4 and 7.5).

Lemma 7.7 For every context-free jungle language L, paths(L) is context-free.

Proof Let $L = L(G)$, where $G = (\Xi, \Sigma, R, \mathtt{f}_0^{(n)})$ contains only useful nonterminals. Let G' be the context-free grammar given by $G' = (\langle \Xi \rangle, \langle \Sigma \rangle, R', \langle \mathtt{f}_0, 0 \rangle)$, where R' is the set of all rules $\langle \mathtt{f}, i \rangle \to p$ such that $\mathtt{f}^{(k)} \in \Xi$, $(\mathtt{f} \to J) \in R$, $i \in \{0, \ldots, k\}$, and $p \in \mathrm{paths}_i(J)$. Then the following can be shown in a straightforward manner, by induction on the length of derivations: for all $w \in \langle \Xi \cup \Sigma \rangle^+$, $\langle \mathtt{f}_0, 0 \rangle \Rightarrow_{G'}^* w$ if and only if there exists a jungle $J \in \mathrm{J}_{\Xi \cup \Sigma}$ with $w \in \mathrm{paths}(J)$, such that $\mathrm{J}(\mathtt{f}_0) \Rightarrow_G^* J$.[24] From this, together with the assumption that all nonterminals of G are useful, it follows that $L(G') = \mathrm{paths}(L(G))$. ∎

The following two lemmas show that it suffices to construct a cf-extended CFJG if we want to prove that a certain jungle language is context-free, and that we can always remove garbage from the jungles generated.

Lemma 7.8 For every cf-extended CFJG G, the jungle language $L(G)$ is context-free.

Proof This is similar to the proof of the fact that cf-extended context-free string grammars generate context-free languages. We only give the construction, whose correctness can be verified using the well-known fact that hyperedge replacement is confluent and associative [DHK97, Eng97].

Let $G = (\Xi, \Sigma, R, \mathtt{ini})$ be a cf-extended CFJG, where $R = \bigcup_{\mathtt{f} \in \Xi} \mathtt{f} \to L_{\mathtt{f}}$. For every $\mathtt{f} \in \Xi$, let $G_{\mathtt{f}} = (\Xi_{\mathtt{f}}, \Xi \cup \Sigma, R_{\mathtt{f}}, \mathtt{ini}_{\mathtt{f}})$ be a CFJG with $L(G_{\mathtt{f}}) = L_{\mathtt{f}}$, and assume without loss of generality that the signatures $\Xi_{\mathtt{f}}$, $\mathtt{f} \in \Xi$, are pairwise disjoint. Then $L(G) = L(G')$, where $G' = (\Xi', \Sigma, R', \mathtt{ini})$ is the CFJG given by

[24] Recall that we use $\Rightarrow_{G'}$ to denote the derivation relation of G', and $L(G')$ to denote its generated language.

$\Xi' = \Xi \cup \bigcup_{\mathtt{f}\in\Xi} \Xi_{\mathtt{f}}$ and $R = \{\mathtt{f} \to \mathrm{J}(\mathtt{ini}_{\mathtt{f}}) \mid \mathtt{f} \in \Xi\} \cup \bigcup_{\mathtt{f}\in\Xi} R_{\mathtt{f}}$. ∎

For the remainder of this section, recall the definitions regarding clean jungles after Fact 4.4, and, in particular, that C_J denotes the set of all non-garbage nodes of a jungle J, and that clean(J) denotes the clean jungle obtained from J by restricting J to $C_J \cup [\mathit{par}(J)]$.

Lemma 7.9 For every CFJG G, there is a CFJG G' with $L(G') = \mathrm{clean}(L(G))$.

Proof Let $G = (\Xi, \Sigma, R, \mathtt{f}_0^{(n)})$. Assuming that all nonterminals in G are useful, we construct $G' = (\Xi', \Sigma, R', \mathtt{f}_0^{(n)})$ based on an easy guess-and-verify strategy, in the following way. The set of nonterminals is given by

$$\Xi' = \{\mathtt{f}_0\} \cup \{\mathtt{f}_S^{(|S|)} \mid k \in \mathbb{N}, \mathtt{f}^{(k)} \in \Xi, S \subseteq [k]\}.$$

The intuition is that $\mathtt{f}_S$ labels a hyperedge e such that (a) $\mathit{res}(e)$ is not (and will not become) garbage, and (b) the parameter tentacles (e, j) with $j \notin S$ have been removed as they have been guessed to be useless in the further course of the derivation, in the sense that, in the jungle J derived from e, there will not be a jungle path from $\mathit{res}(J)$ to $\mathit{par}(J, j)$. Using finite sets INI and K_S, for $K \in \mathrm{J}_{\Xi\cup\Sigma,k}$ and $S \subseteq [k]$, which implement the guess-and-verify strategy, the set of rules is given by

$$R' = \{\mathtt{f}_0 \to I \mid I \in \mathrm{INI}\} \cup \bigcup_{(\mathtt{f}^{(k)}\to K)\in R} \{\mathtt{f}_S \to K' \mid S \subseteq [k], K' \in K_S\}.$$

The definition of INI and K_S makes use of the following notation. Given a sequence $a = a_1 \cdots a_k$ and a set $S = \{i_1, \ldots, i_l\} \subseteq [k]$, where $i_1 < \cdots < i_l$, we let $a|_S = a_{i_1} \cdots a_{i_l}$.

Now, the set INI consists of all (clean) jungles I with $V_I = \{w_1, \ldots, w_n, w\}$, $E_I = \{e\}$, $\mathit{lab}_I(e) = (\mathtt{f}_0)_S$ and $\mathit{att}_I(e) = w_1 \cdots w_n w|_S$ for a set $S \subseteq [n]$, and $\mathit{ext}_I = w_1 \cdots w_n w$. Intuitively, in the first step of a derivation, we guess which parameter nodes of the generated jungle $J \in \mathrm{J}_{\Sigma,n}$ are *used*, meaning that they belong to C_J.

To define K_S for a jungle $K \in \mathrm{J}_{\Xi\cup\Sigma,k}$ and a set $S \subseteq [k]$, let us say that a jungle K' is *garbage-free* if $V_{K'} = C_{K'}$, i.e., if it is clean and all parameter nodes are used. Now, K_S is defined to be the set of all garbage-free jungles $K' \in \mathrm{J}_{\Xi'\cup\Sigma,|S|}$ such that

- $V_{K'} \subseteq V_K$, $E_{K'} \subseteq E_K$, $\mathit{res}(K') = \mathit{res}(K)$, and $\mathit{res}_{K'}(e) = \mathit{res}_K(e)$ for all $e \in E_{K'}$,
- for all $e \in E_{K'}$ with $\mathit{lab}_K(e) \in \Sigma$, $\mathit{lab}_{K'}(e) = \mathit{lab}_K(e)$ and $\mathit{par}_{K'}(e) = \mathit{par}_K(e)$,
- for all $e \in E_{K'}$ with $\mathit{lab}_K(e) = \mathtt{f}^{(m)} \in \Xi$, there is $S' \subseteq [m]$ such that $\mathit{lab}_{K'}(e) = \mathtt{f}_{S'}$ and $\mathit{par}_{K'}(e) = \mathit{par}_K(e)|_{S'}$, and
- $\mathit{par}(K') = \mathit{par}(K)|_S$.

Intuitively, the first three items above implement the guessing, whereas the last implements the verification in the guess-and-verify strategy mentioned above. By induction on the length of derivations, we can show for all $\mathtt{f}^{(k)} \in \Xi$, $S \subseteq [k]$, and $J' \in \mathrm{J}_{\Xi' \cup \Sigma, |S|}$, that $\mathrm{J}(\mathtt{f}_S) \Rightarrow_{G'}^* J'$ if and only if there is a jungle $J \in \mathrm{J}_{\Xi \cup \Sigma, k}$ such that $\mathrm{J}(\mathtt{f}) \Rightarrow_G^* J$ and $J' \in J_S$.

To complete the proof, note that, for $J \in \mathrm{J}_{\Sigma,n}$, there is a unique $S \subseteq [n]$ such that $J_S \neq \emptyset$; moreover, for that S, J_S is a singleton, say $J_S = \{\overline{J}\}$, and $\mathrm{clean}(J) = I[e \leftarrow \overline{J}]$ with $I \in \mathrm{INI}$ and $lab_I(e) = (\mathtt{f}_0)_S$. Thus, taking into account the rules in R' of the form $\mathtt{f}_0 \to I$ with $I \in \mathrm{INI}$, we get $L(G') = \mathrm{clean}(L(G))$. (Note that the usefulness of the nonterminals of G is required to obtain the inclusion $L(G') \subseteq \mathrm{clean}(L(G))$.) This completes the proof. ∎

To turn a (propagating) hierarchical delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathtt{g})_{\mathtt{g} \in \mathcal{G}}, \mathtt{g}_0)$ into a cf-extended CFJG G, and to be able to relate their generated jungle languages, we need a few rather technical preparations. Intuitively, we cannot define G such that it generates exactly the jungles in $L_\mathrm{J}(\mathcal{N})$, because of the following problem. Assume inductively that every $\gamma_\mathtt{g}$ ($\mathtt{g}^{(k)} \in \mathcal{G}$) is a delegation network for which we are given a CFJG $G_\mathtt{g}$ such that $L(G_\mathtt{g}) = L_\mathrm{J}(\gamma_\mathtt{g}) \subseteq \mathrm{J}_{\Sigma_\mathcal{N} \cup X_k}$. By a straightforward construction (see the proof of Lemma 7.22), $G_\mathtt{g}$ can be transformed into a CFJG $G'_\mathtt{g}$ such that $L(G'_\mathtt{g}) = \{J' \mid J \in L(G_\mathtt{g})\} \subseteq \mathrm{J}_{\Sigma_\mathcal{N}, k}$ where the jungle J' (with k parameters) is obtained from the jungle J (without parameters) as follows: for each $i \in [k]$, the ith parameter node v_i of J' is newly created, and v_i is identified with all result nodes of hyperedges that are labeled with x_i; moreover, all hyperedges with labels in X_k are removed. Now, to construct a CFJG G such that $L(G) = L_\mathrm{J}(\mathcal{N})$, it would be natural to use the (cf-extended) rules $\mathtt{g} \to L(G'_\mathtt{g})$. However, by Definition 5.5, we should actually use the rules $\mathtt{g} \to \mathrm{J}_k(\mathrm{tree}(L(G_\mathtt{g})))$ instead. In other words, the inductively obtained jungles in $L(G_\mathtt{g})$ should be replaced by jungles with minimal sharing. Clearly this is not possible because there is, e.g., no CFJG that generates the set of all fully balanced binary trees (without sharing), cf. Example 7.2. Thus, intuitively, the problem is that a CFJG generates trees with too much sharing. To circumvent this problem, we will show (in Lemma 7.20) that we can define the grammar G such that it generates a jungle language that differs from $L_\mathrm{J}(\mathcal{N})$ but, nevertheless, has the same path language; more precisely, we will say that $L(G)$ is a "path variant" of $L_\mathrm{J}(\mathcal{N})$, see Definition 7.17. For this, we invent a notion of redirecting parameter tentacles between two nodes v and w of a jungle J, an operation that can potentially increase the amount of sharing (but also can create garbage). Then G can be constructed with the rules $\mathtt{g} \to L(G'_\mathtt{g})$, where it is assumed that $L(G_\mathtt{g})$ is a path variant of $L_\mathrm{J}(\gamma_\mathtt{g})$.

We now start the technical preparations. Consider a jungle J and nodes $v, w \in V_J$. The set $\mathrm{cand}_J(v, w)$ of all parameter tentacles which are *candidates for redirection from v to w* consists of all $(e, j) \in \mathit{tent}(J)$ with $\mathit{par}_J(e, j) = v$, such that there does not exist a jungle path in J from w to $\mathit{res}_J(e)$. Now,

for every set $\Theta \subseteq \mathrm{cand}_J(v,w)$, we let $J\langle\Theta, w\rangle$ denote the jungle obtained from J by redirecting $par_J(e,j)$ to w for all $(e,j) \in \Theta$, and cleaning the result. More precisely, $J\langle\Theta, w\rangle = \mathrm{clean}(J_{\Theta ::= w})$, where $J_{\Theta ::= w}$ is equal to J, except that $att_{J_{\Theta ::= w}}(e,j) = w$, for all $(e,j) \in \Theta$. By $J\langle v, w\rangle$, we abbreviate the set $\{J\langle\Theta, w\rangle \mid \Theta \subseteq \mathrm{cand}_J(v,w)\}$. By the assumption that $\Theta \subseteq \mathrm{cand}_J(v,w)$, redirection does not introduce cycles (i.e., nonempty jungle paths from a node to itself). Using this, one can easily show that $J_{\Theta ::= w}$ is a jungle, and hence $J\langle\Theta, w\rangle$ is a clean jungle. Note that we always have $\mathrm{clean}(J) \in J\langle v, w\rangle$, by taking $\Theta = \emptyset$. For technical convenience, J itself is not required to be clean. However, if $v, w \in C_J$, then we have that $C_J \supseteq C_{J_{\Theta ::= w}}$, which means that $J\langle\Theta, w\rangle = \mathrm{clean}(J)\langle\Theta', w\rangle$, where $\Theta' = \Theta \cap tent(\mathrm{clean}(J)) = \{(e,j) \in \Theta \mid res_J(e) \in C_J\}$. (In particular, $\mathrm{clean}(J)\langle\Theta', w\rangle$ is defined, because $\Theta' \subseteq \mathrm{cand}_{\mathrm{clean}(J)}(v,w)$.) As a consequence, we then have $J\langle v, w\rangle = \mathrm{clean}(J)\langle v, w\rangle$. In the case that $v \notin C_J$, we obviously have $J\langle v, w\rangle = \{\mathrm{clean}(J)\}$.

Below, the interplay of hyperedge replacement and redirection (in the sense just defined) will be studied. For this, we now introduce a rewrite relation $\Rightarrow\!\!\!\!\!\!\Rightarrow$ on sets of clean jungles, which plays a central role in the reasoning on the following pages. The relation $\Rightarrow\!\!\!\!\!\!\Rightarrow$ is defined on sets of clean jungles in such a way that redirection from v_0 to v_1 is always paired with redirection from v_1 to v_0, thus replacing one jungle J in the set by two jungles J_0 and J_1. Intuitively, this makes sure that, in total, no paths of the original jungle are lost (by turning them into garbage).

Definition 7.10 For sets $\mathcal{J}, \mathcal{J}'$ of clean jungles, $\mathcal{J} \Rightarrow\!\!\!\!\!\!\Rightarrow \mathcal{J}'$ if there exist a jungle $J \in \mathcal{J}$, nodes $v_0, v_1 \in C_J$, and jungles $J_i \in J\langle v_i, v_{1-i}\rangle$ for $i \in \{0,1\}$, such that $\mathcal{J}' = \mathcal{J} \setminus \{J\} \cup \{J_0, J_1\}$. If $\mathcal{J}$ is a singleton $\{J\}$, we may write $J \Rightarrow\!\!\!\!\!\!\Rightarrow \mathcal{J}'$ instead of $\{J\} \Rightarrow\!\!\!\!\!\!\Rightarrow \mathcal{J}'$.

Note that, in the preceding definition, any of J, J_0, and J_1 may be equal. Before giving an example, we introduce a notation that will also be used in the example, generalizing the operation $t \cdot t'$ of concatenation of trees to jungles. If J is a jungle in $\mathrm{J}_{\Sigma\cup\{\Box\}}$ such that there is exactly one hyperedge $e \in E_J$ with label $\Box$, and J' is a jungle in J_Σ, then $J \cdot J'$ denotes the jungle $J[e \leftarrow J']$.

Example 7.11 Let $t = \mathtt{f}[t_0, t_1] \in \mathrm{T}_\Sigma$ and let $J = \mathrm{J}_0(t)$. Consider the nodes $v_0 = nod(1)$ and $v_1 = nod(2)$ of J, i.e., $v_0 = par_J(e,1)$ and $v_1 = par_J(e,2)$ where e is the unique hyperedge with $res_J(e) = res(J)$ and $lab_J(e) = \mathtt{f}$. Let $\Theta_0 = \mathrm{cand}_J(v_0, v_1) = \{(e,1)\}$ and $\Theta_1 = \mathrm{cand}_J(v_1, v_0) = \{(e,2)\}$. Then $J_{\Theta_0 ::= v_1} = J\langle\Theta_0, v_1\rangle \oplus \mathrm{J}_0(t_0)$, and cleaning this jungle (by removing $\mathrm{J}_0(t_0)$) yields $J\langle\Theta_0, v_1\rangle = \mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0(t_1)$.[25] Similarly, $J_{\Theta_1 ::= v_0} = J\langle\Theta_1, v_0\rangle \oplus \mathrm{J}_0(t_1)$ and $J\langle\Theta_1, v_0\rangle = \mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0(t_0)$; for an illustration see Figure 8. This shows that $J \Rightarrow\!\!\!\!\!\!\Rightarrow \{J_0, J_1\}$ where $J = \mathrm{J}_0(\mathtt{f}[t_0, t_1])$ and $J_i = \mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0(t_{1-i})$ for $i \in \{0,1\}$. ■

[25] For $\oplus$ see the beginning, and for $\mathrm{J}_0^{\mathrm{ms}}$ see the end of Section 4.

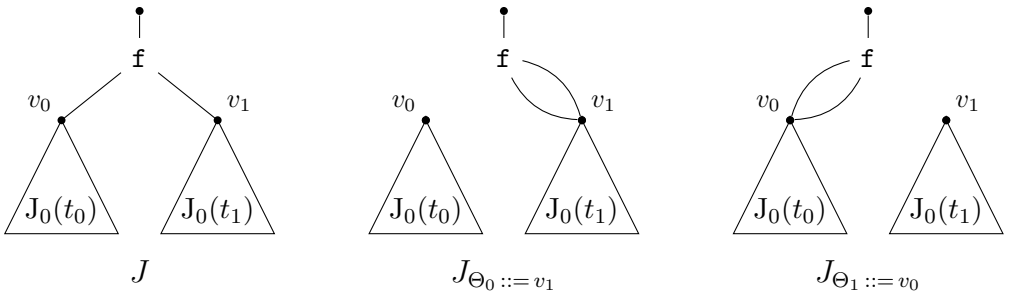


Figure 8: Redirections in Example 7.11

Note that, in the preceding example, $\mathrm{paths}(J) \subseteq \mathrm{paths}(\{J_0, J_1\})$. As we show next, this is not a coincidence.

Lemma 7.12 For clean jungles J, J_0, and J_1, if $J \Rightarrowtriangle \{J_0, J_1\}$, then $\mathrm{paths}(J) \subseteq \mathrm{paths}(\{J_0, J_1\})$.

Proof Let $v_0, v_1 \in C_J$, and $\Theta_i \subseteq \mathrm{cand}_J(v_i, v_{1-i})$, for $i \in \{0, 1\}$. We have to show that every jungle path $p = (e_1, j_1) \cdots (e_m, j_m)$ from v to w in J is a jungle path from v to w in $J\langle\Theta_0, v_1\rangle$ or in $J\langle\Theta_1, v_0\rangle$. For this, note that p is a jungle path from v to w in $J\langle\Theta_i, v_{1-i}\rangle$ unless one of its tentacles is in Θ_i. Hence, if p is neither a jungle path from v to w in $J\langle\Theta_0, v_1\rangle$ nor in $J\langle\Theta_1, v_0\rangle$, we can find $k, l \in [m]$ such that $(e_k, j_k) \in \Theta_0$ and $(e_l, j_l) \in \Theta_1$. Assuming without loss of generality that $k > l$, it follows that there is a jungle path in J from $par_J(e_l, j_l) = v_1$ to $res_J(e_k)$, which contradicts the assumption that $(e_k, j_k) \in \mathrm{cand}_J(v_0, v_1)$. ■

Let us prove two basic lemmas regarding clean jungles and redirection. The first lemma makes proofs by induction on the length of derivations $\mathcal{J} \Rightarrowtriangle^* \mathcal{J}'$ possible. It allows us to (a) split a derivation $J \Rightarrowtriangle^{n+1} \mathcal{J}$ into a first step and two subderivations and (b) combine two derivations into one by taking unions.

Lemma 7.13

(a) Let J be a clean jungle. If $\mathcal{J}$ is a set of clean jungles such that $J \Rightarrowtriangle^{n+1} \mathcal{J}$, then there are J_i and $\mathcal{J}_i$ ($i \in \{0, 1\}$), such that $J \Rightarrowtriangle \{J_0, J_1\}$, $J_i \Rightarrowtriangle^n \mathcal{J}_i$ for $i \in \{0, 1\}$, and $\mathcal{J} = \mathcal{J}_0 \cup \mathcal{J}_1$.

(b) Let $\mathcal{J}_0, \mathcal{J}_1, \mathcal{J}_0', \mathcal{J}_1'$ be sets of clean jungles. If $\mathcal{J}_i \Rightarrowtriangle^* \mathcal{J}_i'$ for $i \in \{0, 1\}$, then $\mathcal{J}_0 \cup \mathcal{J}_1 \Rightarrowtriangle^* \mathcal{J}_0' \cup \mathcal{J}_1'$.

Proof By the definition of $\Rightarrowtriangle$, the derivation considered in (a) has the form $J \Rightarrowtriangle \{J_0, J_1\} \Rightarrowtriangle^n \mathcal{J}$. Thus, to prove (a), it suffices to show the following.

> Let $\mathcal{J}_0 = \{J_1, J_2\}$ and $\mathcal{J}$ be sets of clean jungles, J_1 and J_2 not necessarily being distinct. For $n \in \mathbb{N}$, if $\mathcal{J}_0 \Rightarrowtriangle^n \mathcal{J}$, then there are $\mathcal{J}_1, \mathcal{J}_2$ such that (i) $\mathcal{J} = \mathcal{J}_1 \cup \mathcal{J}_2$ and (ii) $J_i \Rightarrowtriangle^n \mathcal{J}_i$ for $i \in \{1, 2\}$.

To prove this statement, we first observe that $\mathcal{I} \Mapsto \mathcal{I}$ for all nonempty sets $\mathcal{I}$ of clean jungles, because $I \in I\langle v, w\rangle$ for every clean jungle I and $v, w \in C_I$ (taking $\Theta = \emptyset$). We proceed by induction on n. For $n = 0$, simply choose $\mathcal{J}_i = \{J_i\}$, $i \in \{1, 2\}$. For $n > 1$, the derivation considered takes the form $\mathcal{J}_0 \Mapsto^{n-1} \mathcal{J}' \Mapsto \mathcal{J}$, where $\mathcal{J} = \mathcal{J}' \setminus \{J\} \cup \{J', J''\}$, for jungles J, J', and J''. Let $\mathcal{J}'_1, \mathcal{J}'_2$ be the subsets of $\mathcal{J}'$ obtained by applying the induction hypothesis to $\mathcal{J}_0 \Mapsto^{n-1} \mathcal{J}'$. For $i \in \{1, 2\}$, let $\mathcal{J}_i = \mathcal{J}'_i \setminus \{J\} \cup \{J', J''\}$ if $J \in \mathcal{J}'_i$, and $\mathcal{J}_i = \mathcal{J}'_i$ if $J \notin \mathcal{J}'_i$. Then assertion (i) is obviously satisfied. Furthermore, $J_i \Mapsto^{n-1} \mathcal{J}'_i \Mapsto \mathcal{J}_i$. The first part of this derivation exists due to the induction hypothesis. To see that also the last step exists, consider the two possible cases. If $J \in \mathcal{J}'_i$, then $\mathcal{J}'_i \Mapsto \mathcal{J}'_i \setminus \{J\} \cup \{J', J''\} = \mathcal{J}_i$. If $J \notin \mathcal{J}'_i$, then $\mathcal{J}_i \Mapsto \mathcal{J}'_i$ by the observation mentioned above, since both sets are equal. This completes the proof of (a).

The proof of (b) consists mainly of the verification of two claims.

Claim 1 For sets $\mathcal{J}$ and $\mathcal{J}'$ of clean jungles, if $\mathcal{J} \Mapsto \mathcal{J}'$, then $\mathcal{J} \Mapsto^2 \mathcal{J} \cup \mathcal{J}'$.

To see this, let $\mathcal{J}' = \mathcal{J} \setminus \{J\} \cup \{J_0, J_1\}$, where $J \in \mathcal{J}$, $v_0, v_1 \in C_J$, and $J_i \in J\langle v_i, v_{1-i}\rangle$ for $i \in \{0, 1\}$. Exploiting again the fact that $J \in J\langle v_i, v_{1-i}\rangle$, we get

$$\begin{aligned}
\mathcal{J} &\Mapsto \mathcal{J} \setminus \{J\} \cup \{J, J_1\} \\
&= \mathcal{J} \cup \{J_1\} \\
&\Mapsto \mathcal{J} \cup \{J_1\} \setminus \{J\} \cup \{J_0, J\} \\
&= \mathcal{J} \cup \{J_0, J_1\},
\end{aligned}$$

as claimed.

Claim 2 If $\mathcal{J}$, $\mathcal{J}'$, and $\mathcal{I}$ are sets of clean jungles such that $\mathcal{J} \Mapsto^* \mathcal{J}'$, then $\mathcal{J} \cup \mathcal{I} \Mapsto^* \mathcal{J}' \cup \mathcal{I}$.

For derivations of length 0, Claim 2 is obviously valid. Thus, consider a derivation $\mathcal{J} \Mapsto^n \mathcal{J}'' \Mapsto \mathcal{J}'$. Let $J \in \mathcal{J}''$ and $v_0, v_1 \in C_J$ be such that $\mathcal{J}' = \mathcal{J}'' \setminus \{J\} \cup \{J_0, J_1\}$ for jungles $J_i \in J\langle v_i, v_{1-i}\rangle$ ($i \in \{0, 1\}$). By the induction hypothesis, $\mathcal{J} \cup \mathcal{I} \Mapsto^* \mathcal{J}'' \cup \mathcal{I}$. Furthermore, $\mathcal{J}'' \cup \mathcal{I} \Mapsto \mathcal{J}'' \cup \mathcal{I} \setminus \{J\} \cup \{J_0, J_1\} = \mathcal{J}' \cup (\mathcal{I} \setminus \{J\})$. For $J \notin \mathcal{I}$, the latter is obviously equal to $\mathcal{J}' \cup \mathcal{I}$, whereas for $J \in \mathcal{I}$, Claim 1 yields $\mathcal{J}'' \cup \mathcal{I} \Mapsto^2 \mathcal{J}'' \cup \mathcal{I} \cup \mathcal{J}' \cup (\mathcal{I} \setminus \{J\}) = \mathcal{J}' \cup \mathcal{I}$, as required.

Now, a twofold application of Claim 2 proves (b): $\mathcal{J}_0 \cup \mathcal{J}_1 \Mapsto^* \mathcal{J}'_0 \cup \mathcal{J}_1 \Mapsto^* \mathcal{J}'_0 \cup \mathcal{J}'_1$. ■

It must be emphasized that Lemma 7.13 holds regardless of whether or not we consider $\Mapsto$ to be a relation on abstract or concrete jungles. In fact, let us denote the abstract jungle obtained from a concrete jungle J by $[J]$, and similarly for sets of jungles. Using (the two versions of) Lemma 7.13, one can easily verify the following fact, which allows us to switch freely between abstract and concrete jungles even in connection with $\Mapsto$:

> Let J be a concrete jungle, and let $\mathcal{J}$ be a set of abstract jungles. Then $[J] \Mapsto^* \mathcal{J}$ if and only if there is a set $\mathcal{J}'$ of concrete jungles such

that $\mathcal{J} = [\mathcal{J}']$ and $J \Rightarrow^* \mathcal{J}'$.

Note that, even though this fact is easy to prove, it is not entirely trivial. In particular, the *if* direction needs Claim 1 (from the proof of Lemma 7.13) to "duplicate" an abstract jungle that is represented by two distinct but isomorphic concrete jungles.

The next lemma states a few facts regarding the interplay between redirection and hyperedge replacement. Owing to the application of 'clean', redirection may remove a hyperedge that would otherwise have been replaced. For example, we may consider a jungle $I = J[e \leftarrow K]$, and study the effect of turning from J to $J' = J\langle\Theta, w\rangle$. Since e may or may not be in J', we use the notation $J'[e \overset{?}{\leftarrow} K]$ to capture the two possible cases:

$$J'[e \overset{?}{\leftarrow} K] = \begin{cases} J'[e \leftarrow K] & \text{if } e \in E_{J'} \\ J' & \text{if } e \notin E_{J'}. \end{cases}$$

The lemma below considers a jungle $J[e \leftarrow K]$. Parts (b) and (c) of the lemma state that, if we perform redirections in K or J *before* replacing e, we only get results that are also obtained by applying redirection (to the same nodes) *after* having replaced e.

Lemma 7.14 Let $I = J[e \leftarrow K]$ be a jungle, and let img be the associated locator function.[26]
(a) $\mathrm{clean}(I) = \mathrm{clean}(\mathrm{clean}(J)[e \overset{?}{\leftarrow} \mathrm{clean}(K)])$.
(b) For all $v, w \in V_K$,

$$I\langle img(v), img(w)\rangle \supseteq \{\mathrm{clean}(J[e \leftarrow K']) \mid K' \in K\langle v, w\rangle\}.$$

(c) For all $v, w \in V_J$, if $res(K) \notin [par(K)]$, then

$$I\langle img(v), img(w)\rangle \supseteq \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K]) \mid J' \in J\langle v, w\rangle\}.$$

Remark Before giving the proof of the lemma, let us note that the condition $res(K) \notin [par(K)]$ in (c) is indeed necessary. To see this, let $J = \mathrm{J}_0(\mathtt{g}[\mathtt{h}[\mathtt{a}]])$, where e and e' are the hyperedges labelled with $\mathtt{g}$ and $\mathtt{h}$, resp., $par_J(e) = v$, and $par_J(e') = w$. Choosing $K = \mathrm{J}_1(x_1)$ and $\Theta = \{(e, 1)\}$, we obtain $\mathrm{clean}(I) = I = \mathrm{J}_0(\mathtt{h}[\mathtt{a}])$, whereas $J' = J\langle\Theta, w\rangle = \mathrm{J}_0(\mathtt{g}[\mathtt{a}])$ and thus $J'[e \leftarrow K] = \mathrm{J}(\mathtt{a})$. Obviously, $\mathrm{J}(\mathtt{a}) \notin I\langle img(v), img(w)\rangle = \{I\}$.

Proof Property (a) is obvious from the fact that $img(C_J) \supseteq C_I \cap img(V_J)$ and $img(C_K \cup [par(K)]) \supseteq C_I \cap img(V_K)$.

For proving (b), we show that $\mathrm{cand}_K(v, w) \subseteq \mathrm{cand}_I(img(v), img(w))$ and

$$I\langle\Theta, img(w)\rangle = \mathrm{clean}(J[e \leftarrow K\langle\Theta, w\rangle]),$$

for all $\Theta \subseteq \mathrm{cand}_K(v, w)$.

[26] See the paragraph after Definition 4.2.

Consider $v, w \in V_K$ and $(e', j) \in \mathit{tent}(K)$ with $\mathit{par}_K(e', j) = v$, such that there is a jungle path p in I from $\mathit{img}(w)$ to $\mathit{res}_I(e') = \mathit{img}(\mathit{res}_K(e'))$. If p is not a jungle path in K (from w to $\mathit{res}_K(e')$), then, since $\mathit{res}_K(e') \notin [\mathit{par}(K)]$, p must contain a subsequence which is a jungle path in I from some $\mathit{img}(\mathit{par}(K, i))$ to $\mathit{img}(\mathit{res}(K))$ and hence is a jungle path in J from $\mathit{par}_J(e, i)$ to $\mathit{res}_J(e)$, which is impossible as J is acyclic. We can thus conclude that $\mathrm{cand}_K(v, w) \subseteq \mathrm{cand}_I(\mathit{img}(v), \mathit{img}(w))$, which verifies the well-definedness of $I\langle\Theta, \mathit{img}(w)\rangle$.

Furthermore, it should be obvious that $I_{\Theta ::= \mathit{img}(w)} = J[e \leftarrow K_{\Theta ::= w}]$. Using (a), this shows that $I\langle\Theta, \mathit{img}(w)\rangle = \mathrm{clean}(J[e \leftarrow K\langle\Theta, w\rangle])$, as claimed.

Now, let us prove (c). We show that, for all $\Theta \subseteq \mathrm{cand}_J(v, w)$, there is $\Theta' \subseteq \mathrm{cand}_I(\mathit{img}(v), \mathit{img}(w))$ such that

$$I\langle\Theta', \mathit{img}(w)\rangle = \mathrm{clean}(J\langle\Theta, w\rangle[e \overset{?}{\leftarrow} K]).$$

Let $\Theta(e) = \Theta \cap (\{e\} \times \mathbb{N})$ be the set of parameter tentacles of e in Θ. (Thus, $\Theta(e)$ is empty, unless $v \in [\mathit{par}_J(e)]$). Now, define

$$\Theta' = (\Theta \setminus \Theta(e)) \cup \bigcup_{(e,j) \in \Theta(e)} \{(e', j') \in \mathit{tent}(K) \mid \mathit{par}_K(e', j') = \mathit{par}(K, j)\}.$$

Note that the condition $\mathit{par}_K(e', j') = \mathit{par}(K, j)$ is not equivalent to saying that $\mathit{img}(\mathit{par}_K(e', j')) = \mathit{img}(v)$.

To see that $\Theta' \subseteq \mathrm{cand}_I(\mathit{img}(v), \mathit{img}(w))$ consider the possible cases. For $(e', j') \in \Theta \setminus \Theta(e)$, if there is a jungle path p in I from $\mathit{img}(w)$ to $\mathit{res}_I(e')$, then p is either a jungle path in J from w to $\mathit{res}_J(e')$, or it contains a subsequence which is a jungle path in K from $\mathit{res}(K)$ to $\mathit{par}(K, j)$, for some $j \in [|\mathit{par}(K)|]$. In the second case, p can be turned into a jungle path in J from w to $\mathit{res}_J(e')$ by replacing this subsequence with (e, j). Thus, both cases contradict the assumption that $\Theta \subseteq \mathrm{cand}_J(v, w)$.

Now, consider a parameter tentacle $(e', j') \in \mathit{tent}(K)$ such that $\mathit{par}_K(e', j') = \mathit{par}(K, j)$ for a parameter tentacle $(e, j) \in \Theta(e)$. By a similar reasoning as above, if there is a path in I from $\mathit{img}(w)$ to $\mathit{res}_I(e')$, then there is a path in J from w to $\mathit{res}_J(e)$, contradicting the assumption that $(e, j) \in \mathrm{cand}_J(v, w)$. Thus, we can conclude that $\Theta' \subseteq \mathrm{cand}_I(\mathit{img}(v), \mathit{img}(w))$.

As in the proof of (b), the correctness of the equation $I\langle\Theta', \mathit{img}(w)\rangle = \mathrm{clean}(J\langle\Theta, w\rangle[e \overset{?}{\leftarrow} K])$ follows from (a), together with the obvious fact that $I_{\Theta' ::= \mathit{img}(w)} = J_{\Theta ::= w}[e \leftarrow K]$. Note, however, that this makes use of the assumption that $\mathit{res}(K) \notin [\mathit{par}(K)]$. ∎

We use Lemma 7.14 to show, in the next two lemmas, that $\Rightarrow$ is compatible with hyperedge replacement.

Lemma 7.15 Let $I = J[e \leftarrow K]$ for jungles J and K, where J is clean. If $\mathrm{clean}(K) \Rightarrow^* \mathcal{K}$, then $\mathrm{clean}(I) \Rightarrow^* \{\mathrm{clean}(J[e \leftarrow K']) \mid K' \in \mathcal{K}\}$.

Proof We proceed by induction on the length of derivations $\mathrm{clean}(K) \Mapsto^m \mathcal{K}$. By Lemma 7.14(a), the statement is true for $m = 0$, i.e., $\mathcal{K} = \{\mathrm{clean}(K)\}$. Therefore, let $\mathrm{clean}(K) \Mapsto^{n+1} \mathcal{K}$ with $\mathrm{clean}(K) \Mapsto \{K_0, K_1\}$ and $K_i \Mapsto^n \mathcal{K}_i$ for $i \in \{0,1\}$, where $\mathcal{K} = \mathcal{K}_0 \cup \mathcal{K}_1$ (see Lemma 7.13(a)). Let $v_0, v_1 \in C_K$ be such that $K_i \in K\langle v_i, v_{1-i}\rangle$.

Since J is clean, $img(v_0)$ and $img(v_1)$ are in C_I. Hence, by Lemma 7.14(b), $\mathrm{clean}(J[e \leftarrow K_i]) \in I\langle img(v_i), img(v_{1-i})\rangle = \mathrm{clean}(I)\langle img(v_i), img(v_{1-i})\rangle$. In other words, we have $\mathrm{clean}(I) \Mapsto \{\mathrm{clean}(I_0), \mathrm{clean}(I_1)\}$, where $I_i = J[e \leftarrow K_i]$ for $i \in \{0,1\}$. Applying the induction hypothesis, we get $\mathrm{clean}(I_i) \Mapsto^* \{\mathrm{clean}(J[e \leftarrow K_i']) \mid K_i' \in \mathcal{K}_i\}$. Combining these derivations into one by means of Lemma 7.13(b) yields

$$\mathrm{clean}(I) \Mapsto^* \bigcup_{i\in\{0,1\}} \{\mathrm{clean}(J[e \leftarrow K_i']) \mid K_i' \in \mathcal{K}_i\} = \{\mathrm{clean}(J[e \leftarrow K']) \mid K' \in \mathcal{K}\},$$

as claimed. ∎

Lemma 7.16 Let $I = J[e \leftarrow K]$ for jungles J and K, where $res(K) \notin [par(K)]$. If $\mathrm{clean}(J) \Mapsto^* \mathcal{J}$, then there is a set $\mathcal{J}' \subseteq \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K]) \mid J' \in \mathcal{J}\}$ such that $\mathrm{clean}(I) \Mapsto^* \mathcal{J}'$.

Proof Let $\mathrm{clean}(J) \Mapsto^m \mathcal{J}$. As in the previous proof, we proceed by induction on m, starting with the observation that the induction basis follows from Lemma 7.14(a). Again using Lemma 7.13(a), let $\mathrm{clean}(J) \Mapsto^{n+1} \mathcal{J}$ with $\mathrm{clean}(J) \Mapsto \{J_0, J_1\}$ and $J_i \Mapsto^n \mathcal{J}_i$ for $i \in \{0,1\}$, where $\mathcal{J} = \mathcal{J}_0 \cup \mathcal{J}_1$. Let $v_0, v_1 \in C_J$ be such that $J_i \in J\langle v_i, v_{1-i}\rangle$. If $e \in E_{J_i}$, then the induction hypothesis provides us with a set $\mathcal{J}_i'$ such that $\mathcal{J}_i' \subseteq \{\mathrm{clean}(J_i'[e \overset{?}{\leftarrow} K]) \mid J_i' \in \mathcal{J}_i\}$ and $\mathrm{clean}(J_i[e \overset{?}{\leftarrow} K]) \Mapsto^* \mathcal{J}_i'$. If $e \notin E_{J_i}$, then $\mathcal{J}_i' = \mathcal{J}_i$ has these properties.

We distinguish two cases.

Case 1 For an $i \in \{0,1\}$, $img(v_i) \notin C_I$ (where img is the locator function).

In this case, $I\langle img(v_i), img(v_{1-i})\rangle = \{\mathrm{clean}(I)\}$ and so Lemma 7.14(c) yields $\mathrm{clean}(I) = \mathrm{clean}(J_i[e \overset{?}{\leftarrow} K])$. Hence we can take $\mathcal{J}' = \mathcal{J}_i'$.

Case 2 $img(v_0), img(v_1) \in C_I$.

By Lemma 7.14(c), $\mathrm{clean}(I) \Mapsto \{\mathrm{clean}(J_0[e \overset{?}{\leftarrow} K]), \mathrm{clean}(J_1[e \overset{?}{\leftarrow} K])\}$. So, for $\mathcal{J}' = \mathcal{J}_0' \cup \mathcal{J}_1'$, Lemma 7.13(b) yields $\mathrm{clean}(I) \Mapsto \{\mathrm{clean}(J_0[e \overset{?}{\leftarrow} K]), \mathrm{clean}(J_1[e \overset{?}{\leftarrow} K])\} \Mapsto^* \mathcal{J}'$. Furthermore, we have $\mathcal{J}' \subseteq \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K]) \mid i \in \{0,1\}, J' \in \mathcal{J}_i\} = \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K]) \mid J' \in \mathcal{J}\}$, as claimed. ∎

We now define the previously mentioned notion of path variant that relates two jungle languages. As we shall prove directly after the definition (and an example), a path variant of a jungle language L has the same path language as L. The main result of this section will be obtained by essentially proving that,

for every tree language $L \in \mathrm{pDEL}^*(\mathrm{FIN})$, there exists a context-free path variant of $\mathrm{J}_0(L)$.

Definition 7.17 (path variant) A jungle language $L' \subseteq \mathrm{J}_{\Sigma,n}$ is a *path variant* of a jungle language $L \subseteq \mathrm{J}_{\Sigma,n}$, if the following conditions are fulfilled.
(a) For every $J \in L$, there is a set $\mathcal{J} \subseteq \mathrm{clean}(L')$, such that $\mathrm{clean}(J) \Rightarrow^* \mathcal{J}$.
(b) $\mathrm{tree}(L') \subseteq \mathrm{tree}(L)$.

Note that cleaning does not affect conditions (a) and (b). More precisely, let L' be a path variant of L. Then $\mathrm{clean}(L')$ is a path variant of L as well, and L' is a path variant of $\mathrm{clean}(L)$.

Example 7.18 Let $L = \mathrm{J}_0(L_{\mathrm{expseq}})$, where L_{expseq} is defined in Example 7.2, and let L' be the set of all jungles $\mathrm{J}_0^{\mathrm{ms}}(s \cdot (\Box, \ldots, \Box)) \cdot \mathrm{J}_0(t)$ where s is a fully balanced binary tree of height $n \geq 1$ over the signature $\{\mathtt{f}^{(2)}, \Box_1, \ldots, \Box_{2^n}\}$ and t is a tree over the signature $\{\mathtt{k}^{(2)}, \mathtt{a}^{(0)}\}$. Note that $\mathrm{J}_0^{\mathrm{ms}}(s \cdot (\Box, \ldots, \Box))$ is a jungle with $n+1$ nodes and with n $\mathtt{f}$-labelled hyperedges and one $\Box$-labelled hyperedge. We claim that L' is a path variant of L. Clearly, $\mathrm{tree}(L) = L_{\mathrm{expseq}}$ and $\mathrm{tree}(L')$ consists of all trees $s \cdot (t_1, \ldots, t_{2^n})$ such that s is as above and $t_1 = \cdots = t_{2^n} \in \mathrm{T}_{\{\mathtt{k},\mathtt{a}\}}$. Thus, $\mathrm{tree}(L') \subseteq L_{\mathrm{expseq}} = \mathrm{tree}(L)$. We now prove, for every s as above and all trees $t_1, \ldots, t_{2^n} \in \mathrm{T}_{\{\mathtt{k},\mathtt{a}\}}$, that

$$\mathrm{J}_0(s \cdot (t_1, \ldots, t_{2^n})) \Rightarrow^* \{\mathrm{J}_0^{\mathrm{ms}}(s \cdot (\Box, \ldots, \Box)) \cdot \mathrm{J}_0(t_i) \mid i \in [2^n]\}.$$

The proof is by induction on n. For $n = 1$, i.e., for $s = \mathtt{f}[\Box_1, \Box_2]$, it is immediate from Example 7.11. Now consider $s \cdot (t_1, \ldots, t_{2^{n+1}})$ and note that it equals $\mathtt{f}[s' \cdot (t_1, \ldots, t_{2^n}), s' \cdot (t_{2^n+1}, \ldots, t_{2^{n+1}})]$. Again from Example 7.11 we obtain that $\mathrm{J}_0(s \cdot (t_1, \ldots, t_{2^{n+1}})) \Rightarrow$

$$\{\mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0(s' \cdot (t_1, \ldots, t_{2^n})), \mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0(s' \cdot (t_{2^n+1}, \ldots, t_{2^{n+1}}))\}.$$

By induction, $\mathrm{J}_0(s' \cdot (t_1, \ldots, t_{2^n})) \Rightarrow^* \{\mathrm{J}_0^{\mathrm{ms}}(s' \cdot (\Box, \ldots, \Box)) \cdot \mathrm{J}_0(t_j) \mid 1 \leq j \leq 2^n\}$ and $\mathrm{J}_0(s' \cdot (t_{2^n+1}, \ldots, t_{2^{n+1}})) \Rightarrow^* \{\mathrm{J}_0^{\mathrm{ms}}(s' \cdot (\Box, \ldots, \Box)) \cdot \mathrm{J}_0(t_k) \mid 2^n+1 \leq k \leq 2^{n+1}\}$. Hence, by Lemmas 7.15 and 7.13(b), $\mathrm{J}_0(s \cdot (t_1, \ldots, t_{2^{n+1}})) \Rightarrow^*$

$$\{\mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0^{\mathrm{ms}}(s' \cdot (\Box, \ldots, \Box)) \cdot \mathrm{J}_0(t_i) \mid i \in [2^{n+1}]\}.$$

Since $\mathrm{J}_0^{\mathrm{ms}}(\mathtt{f}[\Box, \Box]) \cdot \mathrm{J}_0^{\mathrm{ms}}(s' \cdot (\Box, \ldots, \Box)) = \mathrm{J}_0^{\mathrm{ms}}(s \cdot (\Box, \ldots, \Box))$, this shows that condition (a) of Definition 7.17 is fulfilled, and thus that L' is a path variant of L. ■

Lemma 7.19 For all jungle languages L and L', if L' is a path variant of L, then $\mathrm{paths}(L) = \mathrm{paths}(L')$.

Proof Condition (b) in the definition of path variant yields immediately that $\mathrm{paths}(L') \subseteq \mathrm{paths}(L)$. For the converse direction, for sets $\mathcal{J}, \mathcal{J}'$ of clean jungles

with $\mathcal{J} \Mapsto \mathcal{J}'$, we have $\mathrm{paths}(\mathcal{J}) \subseteq \mathrm{paths}(\mathcal{J}')$ by Lemma 7.12. Using condition (a) and the fact that $\mathrm{paths}(\mathrm{clean}(J)) = \mathrm{paths}(J)$, this yields $\mathrm{paths}(L) \subseteq \mathrm{paths}(L')$, as required. ∎

The next lemma will be the major ingredient needed for the proof of the context-freeness of $\mathrm{paths}(L)$ for all $L \in \mathrm{DEL}^*(\mathrm{FIN})$.

Lemma 7.20 Let $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathsf{g})_{\mathsf{g}\in\mathcal{G}}, \mathsf{g}_0)$ be a propagating delegation network. If, for every $\mathsf{g}^{(k)} \in \mathcal{G}$, there exists a context-free path variant L_g of $\mathrm{J}_k(L(\gamma_\mathsf{g}))$, then there exists a context-free path variant of $L_\mathrm{J}(\mathcal{N})$.

Proof By Lemma 7.9, and the fact that cleaning does not affect the property of being a path variant (see the remark following Definition 7.17), we may assume that each L_g consists of clean jungles. To prove the lemma, instead of constructing a CFJG generating a context-free path variant of $L_\mathrm{J}(\mathcal{N})$, we construct a cf-extended CFJG G with this property. By Lemma 7.8, this suffices to prove the lemma. We let

$$G = (\mathcal{G}, \Pi, \bigcup\{\mathsf{g} \to L_\mathsf{g} \mid \mathsf{g} \in \mathcal{G}\}, \mathsf{g}_0).$$

We have to check that $L(G)$ is a path variant of $L_\mathrm{J}(\mathcal{N})$, i.e., that conditions (a) and (b) of Definition 7.17 are satisfied.

For (a), similar to the notion of usefulness of nonterminals in G, let us say that $\mathsf{g} \in \mathcal{G}$ is *useful in* $\mathcal{N}$ if $\mathrm{J}(\mathsf{g}) \Rightarrow^*_\mathcal{N} J$ for a jungle J over Π. For $Z \in \{G, \mathcal{N}\}$, a derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_Z J'$ is useful (in Z) if $lab_{J'}(E_{J'}) \cap \mathcal{G}$ contains only symbols that are useful in Z.

Claim 1 For every $\mathsf{g} \in \mathcal{G}$, if g is useful in $\mathcal{N}$, then g is useful in G.

By the definition of $\Mapsto$, if $K \Mapsto^* \mathcal{K}$ and $K' \in \mathcal{K}$, then all hyperedge labels in K' occur also in K, i.e., $lab_{K'}(E_{K'}) \subseteq lab_K(E_K)$. Thus, by Definition 7.17(a), for all $\mathsf{g}^{(k)} \in \mathcal{G}$ and $K \in \mathrm{J}_k(L(\gamma_\mathsf{g}))$, there is a rule $\mathsf{g} \to K'$ in G, such that $lab_{K'}(E_{K'}) \subseteq lab_K(E_K)$. By a straightforward induction on the length of derivations, this means that, for every derivation $\mathrm{J}(\mathsf{g}) \Rightarrow^*_\mathcal{N} J$, there is a derivation $\mathrm{J}(\mathsf{g}) \Rightarrow^*_G J'$ such that $lab_{J'}(E_{J'}) \subseteq lab_J(E_J)$. In particular, g is useful in G if it is useful in $\mathcal{N}$.

Now, we prove the following claim by induction on the length of derivations.

Claim 2 For every useful derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_\mathcal{N} J$, there exists a set $\mathcal{J}$ of clean jungles such that
(1) $\mathrm{clean}(J) \Mapsto^* \mathcal{J}$ and
(2) for every $J' \in \mathcal{J}$ there is a useful derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_G J''$ with $\mathrm{clean}(J'') = J'$.

Let us first argue that Claim 2 suffices to prove condition (a) of Definition 7.17. For this, let $J \in L_\mathrm{J}(\mathcal{N})$, and let $\mathcal{J}$ be such that J and $\mathcal{J}$ satisfy (1) and (2). For $J' \in \mathcal{J}$, by (2), there is a useful derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_G J''$ such that $\mathrm{clean}(J'') = J'$. As this derivation is useful, it can be extended to yield a jungle over Π, say $J'' \Rightarrow^*_G J''' \in \mathrm{J}_\Pi$. Thus, $J''' \in L(G)$. Since $J \in \mathrm{J}_\Pi$, we have $\mathcal{J} \subseteq \mathrm{J}_\Pi$, and thus $\mathrm{clean}(J'') \in \mathrm{J}_\Pi$, which means that $\mathrm{clean}(J''') = \mathrm{clean}(J'') = J' \in \mathcal{J}$. (Note that

if $I' = I[e \leftarrow K]$ for a hyperedge $e \in E_I \setminus E_{\mathrm{clean}(I)}$, then $\mathrm{clean}(I') = \mathrm{clean}(I)$.) In other words, $\mathcal{J} \subseteq \mathrm{clean}(L(G))$.

Now, to prove Claim 2, first notice, using Claim 1, that it is trivially true for derivations of length 0 (with $\mathcal{J} = \{\mathrm{J}(\mathsf{g}_0)\}$). Thus, consider a useful derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_{\mathcal{N}} J \Rightarrow_{\mathcal{N}} I = J[e \leftarrow K]$, where $lab_J(e) = \mathsf{g}^{(k)}$ and $K = \mathrm{J}_k(t)$ for a tree $t \in L(\gamma_{\mathsf{g}})$. Note that the derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_{\mathcal{N}} J$ is then also useful. Assume that $\mathcal{J}$ is a set of clean jungles satisfying (1) and (2) with respect to J.

Since $\mathcal{N}$ is propagating, we have $res(K) \notin [par(K)]$. By Lemma 7.16, this means that there is a set $\mathcal{J}' \subseteq \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K]) \mid J' \in \mathcal{J}\}$ such that $\mathrm{clean}(I) \Mapsto^* \mathcal{J}'$. Further, by the assumption that L_{g} is a path variant of $\mathrm{J}_k(L(\gamma_{\mathsf{g}}))$, there is a set $\mathcal{K} \subseteq L_{\mathsf{g}}$, such that $K \Mapsto^* \mathcal{K}$. Let $\mathcal{I} = \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K']) \mid J' \in \mathcal{J}, K' \in \mathcal{K}\}$. We claim that $\mathcal{I}$ satisfies (1) and (2) with respect to I. For checking (1), consider $I' = \mathrm{clean}(J'[e \overset{?}{\leftarrow} K]) \in \mathcal{J}'$, where $J' \in \mathcal{J}$. Using Lemma 7.15, we get $I' \Mapsto^* \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K']) \mid K' \in \mathcal{K}\}$ (which is trivial if $e \notin E_{J'}$). Hence, repeated application of Lemma 7.13(b) gives

$$\mathrm{clean}(I) \Mapsto^* \mathcal{J}' \Mapsto^* \bigcup_{J' \in \mathcal{J}'} \{\mathrm{clean}(J'[e \overset{?}{\leftarrow} K']) \mid K' \in \mathcal{K}\} = \mathcal{I},$$

as required. For checking (2), consider an element $\mathrm{clean}(J'[e \overset{?}{\leftarrow} K'])$ of $\mathcal{I}$ (i.e., $J' \in \mathcal{J}$ and $K' \in \mathcal{K}$). Let $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_G J''$ be a useful derivation such that $\mathrm{clean}(J'') = J'$. By the construction of G, and since $K' \in L_{\mathsf{g}}$, we have $J'' \Rightarrow^*_G J''[e \overset{?}{\leftarrow} K']$ by either 0 or 1 derivation steps. By Lemma 7.14(a), $\mathrm{clean}(J''[e \overset{?}{\leftarrow} K']) = \mathrm{clean}(J'[e \overset{?}{\leftarrow} K'])$, as required in (2). Furthermore, since all generator symbols in J'' and K are useful in G and $\mathcal{N}$, respectively, and $lab_{K'}(E_{K'}) \subseteq lab_K(E_K)$, Claim 1 yields that the derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_G J''[e \overset{?}{\leftarrow} K']$ is useful.

It remains to prove that condition (b) of Definition 7.17 is satisfied. Instead of proving that $\mathrm{tree}(L(G)) \subseteq \mathrm{tree}(L_{\mathrm{J}}(\mathcal{N}))$, we prove the more general statement that, for every jungle $J' \in L(G)$, there is a jungle $J \in L_{\mathrm{J}}(\mathcal{N})$ that maps to J'. By Lemma 4.5, this proves that $\mathrm{tree}(J) = \mathrm{tree}(J')$.

We proceed by induction on the length of derivations to show that, for every derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_G J'$, there is a derivation $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_{\mathcal{N}} J$ such that J maps to J'. The induction basis is trivial. Thus, consider a derivation

$$\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_G J' \Rightarrow_G I',$$

where $I' = J'[e' \leftarrow K']$, $lab_{J'}(e') = \mathsf{g}^{(k)}$, and $K' \in L_{\mathsf{g}}$.

Let $K = \mathrm{J}_k(\mathrm{tree}(K'))$. By assumption, $K \in \mathrm{J}_k(L(\gamma_{\mathsf{g}}))$, and by Lemma 4.5, K maps to K'. Furthermore, by the induction hypothesis, $\mathrm{J}(\mathsf{g}_0) \Rightarrow^*_{\mathcal{N}} J$, where J maps to J'. Let $\varphi\colon V_J \to V_{J'}$ be the corresponding structure preserving mapping, and let $e_1, \ldots, e_m \in E_J$ be the pairwise distinct hyperedges e such that $\varphi(res_J(e)) = res_{J'}(e')$. Then, using the fact that J maps to J' and K maps to

K', it is straightforward to show that $I = J[e_1 \leftarrow K] \cdots [e_m \leftarrow K]$ maps to I'. Since $J \Rightarrow^*_{\mathcal{N}} I$, this completes the inductive proof.

To finish the proof that $\mathrm{tree}(L(G)) \subseteq \mathrm{tree}(L_{\mathrm{J}}(\mathcal{N}))$, consider $J' \in L(G)$. By the reasoning above, $\mathrm{J}(\mathtt{g}_0) \Rightarrow^*_{\mathcal{N}} J$ for a jungle J that maps to J'. Furthermore, $J \in \mathrm{J}_\Pi$ since $J' \in \mathrm{J}_\Pi$, which means that $J \in L_{\mathrm{J}}(\mathcal{N})$. ∎

Example 7.21 Let $\mathcal{N}$ be the propagating delegation network $\mathcal{N}_{\mathrm{expseq}}$ of Example 7.2. Clearly, $L_{\mathrm{J}}(\mathcal{N}) = \mathrm{J}_0(L_{\mathrm{T}}(\mathcal{N})) = \mathrm{J}_0(L_{\mathrm{expseq}})$. We first consider context-free path variants $L_{\mathtt{h}_0}$ and $L_{\mathtt{h}}$ of $\mathrm{J}_0(L(\gamma_{\mathtt{h}_0}))$ and $\mathrm{J}_0(L(\gamma_{\mathtt{h}}))$, respectively. Recall that $L(\gamma_{\mathtt{h}_0}) = L_{\mathrm{T}}(\mathcal{N}_{\mathrm{bin}})$ and $L(\gamma_{\mathtt{h}}) = \mathrm{T}_{\{\mathtt{k},\mathtt{a}\}}$. Let $L_{\mathtt{h}_0} = L_{\mathrm{J}}(\mathcal{N}_{\mathrm{bin}})$. By Definitions 5.5 and 7.6, it is generated by the CFJG with rules $\mathtt{g}_0 \to \mathrm{J}_0(\mathtt{g}[\mathtt{h}])$, $\mathtt{g} \to \mathrm{J}^{\mathrm{ms}}_1(\mathtt{g}[\mathtt{f}[x_1, x_1]])$, and $\mathtt{g} \to \mathrm{J}^{\mathrm{ms}}_1(\mathtt{f}[x_1, x_1])$. It is equal to $\mathrm{J}^{\mathrm{ms}}_0(L_{\mathrm{T}}(\mathcal{N}_{\mathrm{bin}}))$, which is a path variant of $\mathrm{J}_0(L_{\mathrm{T}}(\mathcal{N}_{\mathrm{bin}}))$ as can be shown by an argument similar to the one in Example 7.18. Let $L_{\mathtt{h}} = \mathrm{J}_0(L(\gamma_{\mathtt{h}}))$. It is of course a path variant of itself, and it is generated by the CFJG with rules $\mathtt{g}'_0 \to \mathrm{J}_0(\mathtt{k}[\mathtt{g}'_0, \mathtt{g}'_0])$ and $\mathtt{g}'_0 \to \mathrm{J}_0(\mathtt{a})$. By (the proof of) Lemma 7.20, the jungle language generated by the cf-extended CFJG G with rules $\mathtt{h}_0 \to L_{\mathtt{h}_0}$ and $\mathtt{h} \to L_{\mathtt{h}}$ is a path variant of $\mathrm{J}_0(L_{\mathrm{expseq}})$. By the construction in the proof of Lemma 7.8 (with some obvious simplifications), it is generated by the CFJG G' with rules $\mathtt{g}_0 \to \mathrm{J}_0(\mathtt{g}[\mathtt{h}])$, $\mathtt{g} \to \mathrm{J}^{\mathrm{ms}}_1(\mathtt{g}[\mathtt{f}[x_1, x_1]])$, $\mathtt{g} \to \mathrm{J}^{\mathrm{ms}}_1(\mathtt{f}[x_1, x_1])$, $\mathtt{h} \to \mathrm{J}_0(\mathtt{k}[\mathtt{h}, \mathtt{h}])$ and $\mathtt{h} \to \mathrm{J}_0(\mathtt{a})$. It should be clear that $L(G') = L'$, as defined in Example 7.18, where we already showed that L' is a path variant of $\mathrm{J}_0(L_{\mathrm{expseq}})$. ∎

Now, recall Corollary 6.5, which shows that $\mathrm{paths}(L_{\mathrm{T}}(\mathcal{N})) = \mathrm{paths}(L_{\mathrm{J}}(\mathcal{N}))$. We would like to be able to apply Lemma 7.20 in an induction on the number of applications of pDEL to show that, for every propagating delegation network $\mathcal{N}$ generating a tree language $L \in \mathrm{pDEL}^*(\mathrm{FIN})$, there exists a context-free path variant of $L_{\mathrm{J}}(\mathcal{N})$. However, this yields an induction hypothesis which is too weak. What is needed is a context-free path variant of $\mathrm{J}_Y(L_{\mathrm{T}}(\mathcal{N}))$, where Y is a parameter signature. Next, we prove a lemma that yields this link and completes the series of technical preparations required to prove the main result of this section.

To formulate the lemma, the following notation is used. For a jungle $J \in \mathrm{J}_{\Sigma,k}$ and a parameter signature $Y = \{y_1, \ldots, y_n\}$, we let $J_{[Y]} \in \mathrm{J}_{\Sigma\setminus Y, k+n}$ denote the jungle obtained from J by identifying, for each $i \in [n]$, all result nodes of hyperedges $e \in E_J$ with $lab_J(e) = y_i$, which yields a new node v_i, making v_i the $(k+i)$th parameter node, and deleting all hyperedges with labels in Y. If y_i does not occur in J, then v_i becomes an isolated parameter node. For a jungle language L, we let $L_{[Y]} = \{J_{[Y]} \mid J \in L\}$. Note that, for every tree $t \in \mathrm{T}_\Sigma$, $\mathrm{J}_Y(t) = \mathrm{J}_0(t)_{[Y]}$.

Lemma 7.22 Let $L, L' \subseteq \mathrm{J}_\Sigma$, and let Y be a parameter signature. If L' is a context-free path variant of L, then $L'_{[Y]}$ is a context-free path variant of

$\mathrm{J}_Y(\mathrm{tree}(L))$.

Proof Let $Y = \{y_1, \ldots, y_n\}$. We first prove that $L'_{[Y]}$ is a path variant of $\mathrm{J}_Y(\mathrm{tree}(L))$ if L' is a path variant of L.

Clearly, $\mathrm{J}_Y(\mathrm{tree}(L))$ and $L'_{[Y]}$ satisfy condition (b) of Definition 7.17, because $\mathrm{tree}(\mathrm{J}_Y(\mathrm{tree}(L)))$ is obtained from $\mathrm{tree}(L)$ by turning each y_i into x_i for $i \in [n]$, and $\mathrm{tree}(L'_{[Y]})$ is obtained from $\mathrm{tree}(L')$ in the same way. Thus, it remains to be shown that condition (a) is satisfied. For this, we prove two claims.

Claim 1 If $J = \mathrm{J}_0(\mathrm{tree}(J'))$ for a clean jungle $J' \in \mathrm{J}_\Sigma$, then $J \Rightarrow^* \{J'\}$.

We show that, for all clean jungles J and J', if J maps to J', then $J \Rightarrow^* \{J'\}$. Using Lemma 4.5, this proves Claim 1 by choosing $J = \mathrm{J}_0(\mathrm{tree}(J'))$.

Assume that J maps to J', and let $\varphi\colon V_J \to V_{J'}$ be the corresponding structure preserving mapping. Note that φ is surjective, because J' is clean. We proceed by induction on $m = |V_J| - |V_{J'}|$. If $m = 0$, then φ is injective, and thus an isomorphism, which yields $J \Rightarrow^0 \{J'\}$. If $m > 0$, then there are distinct nodes $v_0, v_1 \in V_J$ such that $\varphi(v_0) = \varphi(v_1)$. Since φ is structure preserving, we can choose these nodes in such a way that $\mathit{par}_J(e_0) = \mathit{par}_J(e_1)$ for the hyperedges e_i with $\mathit{res}_J(e_i) = v_i$ ($i \in \{0,1\}$). Now, let J_f be the jungle obtained from J by identifying v_0 with v_1 and e_0 with e_1 (which is commonly called *folding* in the literature). Obviously, J_f is isomorphic to $J_i = J\langle\Theta_i, v_{1-i}\rangle$, for $i \in \{0,1\}$, where $\Theta_i = \{(e,j) \in \mathit{tent}(J) \mid \mathit{par}_J(e,j) = v_i\} = \mathrm{cand}_J(v_i, v_{1-i})$. In other words, $J \Rightarrow \{J_0, J_1\} = \{J_\mathrm{f}\}$. Moreover, if v is the node in J_f resulting from the identification of v_0 and v_1, then J_f maps to J' by the mapping φ' such that $\varphi'(v) = \varphi(v_0)$ and $\varphi'(v') = \varphi(v')$ for all $v \in V_J \setminus \{v_0, v_1\}$. By the induction hypothesis, this yields $J_\mathrm{f} \Rightarrow^* \{J'\}$, as claimed.

Claim 2 Let $J \in \mathrm{J}_\Sigma$ be a clean jungle, and let $\mathcal{J} \subseteq \mathrm{J}_\Sigma$ be a set of clean jungles. If $J \Rightarrow^* \mathcal{J}$, then $J_{[Y]} \Rightarrow^* \mathcal{J}_{[Y]}$.

The correctness of the claim follows by an obvious inductive argument, using the fact that, for $v, w \in C_J$ and $\Theta \subseteq \mathrm{cand}_J(v,w)$, we have $\Theta \subseteq \mathrm{cand}_{J_{[Y]}}(\mathit{img}(v), \mathit{img}(w))$ and $J\langle\Theta, w\rangle_{[Y]} = J_{[Y]}\langle\Theta, \mathit{img}(w)\rangle$, where img is the function mapping every node in J to its image in $J_{[Y]}$. This fact is a direct consequence of the relevant definitions.

To conclude that $L'_{[Y]}$ is a path variant of $\mathrm{J}_Y(\mathrm{tree}(L))$, notice first that, for every jungle $J \in \mathrm{J}_\Sigma$, we have $\mathrm{clean}(J_{[Y]}) = \mathrm{clean}(J)_{[Y]}$. Now, consider a jungle $J \in L$. We have to show that there is a set $\mathcal{J} \subseteq \mathrm{clean}(L'_{[Y]})$ such that $\mathrm{clean}(\mathrm{J}_Y(\mathrm{tree}(J))) \Rightarrow^* \mathcal{J}$. By Claim 1 and the fact that L' is a path variant of L, we have $\mathrm{J}_0(\mathrm{tree}(J)) \Rightarrow^* \{\mathrm{clean}(J)\} \Rightarrow^* \mathcal{J}'$ for a set $\mathcal{J}' \subseteq \mathrm{clean}(L')$. By Claim 2, this means that $\mathrm{J}_Y(\mathrm{tree}(J)) \Rightarrow^* \mathcal{J}$, where $\mathcal{J} = \mathcal{J}'_{[Y]}$. This set $\mathcal{J}$ is a subset of $\mathrm{clean}(L')_{[Y]}$, which, by the remark above, is equal to $\mathrm{clean}(L'_{[Y]})$. Thus, $L'_{[Y]}$ is indeed a path variant of $\mathrm{J}_Y(\mathrm{tree}(L))$.

To complete the proof of the lemma, it must be shown that $L'_{[Y]}$ is context-free if L' is context-free. For this purpose, assume that $L' = L(G)$, for a CFJG

$G = (\Xi, \Sigma, R, \mathtt{f}_0)$. Let Ξ' be obtained from Ξ by turning the rank of every symbol $\mathtt{f}^{(k)} \in \Xi$ into $k+n$; in particular, $\mathtt{f}_0$ becomes a symbol of rank n. Let $\Sigma' = \Sigma \setminus Y$, and define, for every jungle $J \in \mathrm{J}_{\Xi\cup\Sigma,k}$, $\widehat{J}_{[Y]}$ to be the jungle in $\mathrm{J}_{\Xi'\cup\Sigma',k+n}$ obtained from $J_{[Y]}$ by appending $par_{J_{[Y]}}(k+1)\cdots par_{J_{[Y]}}(k+n)$ to $par_{J_{[Y]}}(e)$, for every hyperedge $e \in E_{J_{[Y]}}$ with $lab_{J_{[Y]}}(e) \in \Xi$.

As a direct consequence of its definition, $\widehat{J}_{[Y]} = J_{[Y]}$ for all $J \in \mathrm{J}_\Sigma$. Now, for the CFJG $G' = (\Xi', \Sigma', \{\mathtt{f} \to \widehat{K}_{[Y]} \mid (\mathtt{f} \to K) \in R\}, \mathtt{f}_0)$, it follows by a straightforward induction on the length of derivations, that $\mathrm{J}(\mathtt{f}_0) \Rightarrow^*_G J$ if and only if $\mathrm{J}(\mathtt{f}_0) \Rightarrow^*_{G'} \widehat{J}_{[Y]}$. In particular, this means that $L(G') = \{\widehat{J}_{[Y]} \mid J \in L(G)\} = L'_{[Y]}$. This completes the proof. ∎

We are, finally, ready to prove the main result of this section.

Theorem 7.23 For every tree language $L \in \mathrm{DEL}^*(\mathrm{FIN})$, $\mathrm{paths}(L)$ is context-free.

Proof By Lemmas 7.4 and 7.5, it suffices to prove the statement for $L \in \mathrm{pDEL}^*(\mathrm{FIN})$.

We prove first that, for every tree language $L \in \mathrm{pDEL}^n(\mathrm{FIN})$ and every parameter signature Y, there is a context-free path variant of $\mathrm{J}_Y(L)$. The proof is by induction on n. For $n = 0$, it suffices to note that finite jungle languages are context-free, and every jungle language is a path variant of itself. Now, let $n > 0$, and assume that the statement is true for $n-1$. Let $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$ be a propagating delegation network such that $L = L_{\mathrm{T}}(\mathcal{N})$, where $\Gamma = (\gamma_\mathtt{g}, Y_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}$ with $L(\gamma_\mathtt{g}) \in \mathrm{pDEL}^{n-1}(\mathrm{FIN})$ for all $\mathtt{g} \in \mathcal{G}$. By the induction hypothesis, there is a context-free path variant $L_\mathtt{g}$ of $\mathrm{J}_{Y_\mathtt{g}}(L(\gamma_\mathtt{g}))$, for every $\mathtt{g} \in \mathcal{G}$. Using Lemma 7.20, this shows that there is a context-free path variant L' of $L_{\mathrm{J}}(\mathcal{N})$. Hence, by Lemma 7.22 (for $\Sigma = \Pi$), $L'_{[Y]}$ is a context-free path variant of $\mathrm{J}_Y(\mathrm{tree}(L_{\mathrm{J}}(\mathcal{N})))$. By Corollary 6.5, the latter is equal to $\mathrm{J}_Y(L)$, as claimed.

In particular, we have shown that there is a context-free path variant L_0 of $\mathrm{J}_0(L)$. By Lemma 7.7, $\mathrm{paths}(L_0)$ is context-free, and by Lemma 7.19, it is equal to $\mathrm{paths}(\mathrm{J}_0(L)) = \mathrm{paths}(\mathrm{tree}(\mathrm{J}_0(L))) = \mathrm{paths}(L)$. ∎

Since the proof of Theorem 7.23 is effective, and since a tree language L is empty or finite if and only if the language $\mathrm{paths}(L)$ is, it follows that the emptiness and finiteness problems are decidable for $L \in \mathrm{DEL}^*(\mathrm{FIN})$. This result is, however, not very useful as such, because it is an open problem whether $\mathrm{DEL}^*(\mathrm{FIN})$ is closed under intersection with regular tree languages. (Usually, one is not interested in deciding whether L itself is empty (or finite), but rather whether $L \cap R$ is, where R is a regular tree language that captures some type of correctness property, cf., e.g., the introduction of [DE98].) The result will be strengthened in the next section, where we obtain a more useful variant (Corollary 8.16).

8 Delegation versus Macro Tree Transducers

One of the main results of this section is that for every hierarchical delegation network $\mathcal{N}$, the tree language $L_{\mathrm{T}}(\mathcal{N})$ is the image of a regular tree language under a composition of several macro tree transductions. Macro tree transducers are a powerful type of tree transducer studied, e.g., in [EV85, EV94, FV98, DE98, EM02, CE12].

Several of the auxiliary results in this section concern classes of tree languages obtained by applying an operator such as DEL to a more or less arbitrary class C of tree languages. However, to avoid some very unnatural effects and technical difficulties, we will restrict ourselves to genuine classes of tree languages. By definition, a class C of tree languages is *genuine* if it contains all finite tree languages and, for every tree language $L \subseteq \mathrm{T}_\Sigma$ and every bijective and rank-preserving renaming $r\colon \Sigma \to \Sigma'$ of symbols, if $L \in C$ then $r(L) \in C$ (where r is extended to a function on trees in the canonical way). It is obvious that the operator DEL preserves genuineness, i.e., if C is genuine, then so is $\mathrm{DEL}(C)$. It is left to the reader to check that this also holds for the operators TD, YIELD and rDEL introduced in the remainder of this section. Note that since FIN is genuine, all the classes $\mathrm{DEL}^n(\mathrm{FIN})$ of the delegation hierarchy are genuine. Note also that the class REG of regular tree languages (see Definition 2.5) is genuine.

A *tree transduction* is a binary relation $\tau \subseteq \mathrm{T}_\Sigma \times \mathrm{T}_\Delta$, where Σ and Δ are finite signatures. Thus, viewed as a function $\tau\colon \mathrm{T}_\Sigma \to \wp(\mathrm{T}_\Delta)$, τ maps trees in T_Σ to sets of trees in T_Δ. For a class C of tree languages and a class TR of tree transductions, we let $TR(C) = \{\tau(L) \mid \tau \in TR \text{ and } L \in C\}$, $TR^0(C) = C$ and $TR^{n+1}(C) = TR(TR^n(C))$, for $n \in \mathbb{N}$. Moreover, $TR^*(C) = \bigcup_{n\in\mathbb{N}} TR^n(C)$. As usual, C is said to be *closed under* TR if $TR(C) \subseteq C$. Similarly, C is *closed under DEL* if $\mathrm{DEL}(C) \subseteq C$.

Next, we recall the definitions of two well-known classes of tree transductions, namely top-down tree transductions and YIELD mappings. These are specific macro tree transductions and, vice versa, every macro tree transduction is a composition of these transductions. Thus, for the results in this section, we need not define macro tree transducers.

Definition 8.1 (top-down tree transducer) A *top-down tree transducer* (*td transducer*, for short) is a tuple $td = (\Sigma, \Delta, Q, R, q_0)$, where

- Σ and Δ are finite signatures of *input symbols* and *output symbols*, resp.,
- Q is a finite signature of *states* of rank 1, where $Q \cap (\Sigma \cup \Delta) = \emptyset$,
- R is a finite set of *rules*, and
- $q_0 \in Q$ is the *initial state*.

Every rule in R has the form $q[\mathtt{f}[x_1, \ldots, x_k]] \to t \cdot (q_1[x_{i_1}], \ldots, q_l[x_{i_l}])$, where $k, l \in \mathbb{N}$, $q, q_1, \ldots, q_l \in Q$, $\mathtt{f}^{(k)} \in \Sigma$, $i_1, \ldots, i_l \in [k]$, and $t \in \mathrm{T}_{\Delta,\{\Box_1,\ldots,\Box_l\}}$.

Given a tree $s = s_0 \cdot q[\mathtt{f}[s_1, \ldots, s_k]]$, if R contains a rule as above, then there is a *computation step* $s \Rightarrow_{td} s_0 \cdot (t \cdot (q_1[s_{i_1}], \ldots, q_l[s_{i_l}]))$. The subscript td can be

omitted if it is clear from the context. The *top-down tree transduction* $td \subseteq \mathrm{T}_\Sigma \times \mathrm{T}_\Delta$ *computed by* td is given by

$$td(s) = \{t \in \mathrm{T}_\Delta \mid q_0(s) \Rightarrow^* t\},$$

for all $s \in \mathrm{T}_\Sigma$. The set of all top-down tree transductions (*td transductions*, for short) is denoted by TD.

An example of a td transducer that turns out to be helpful in our reasoning is the following. Consider a finite signature Σ, and let $+^{(2)}$ and $\theta^{(0)}$ be special symbols not in Σ. Now, let $set_\Sigma = (\Sigma \cup \{+, \theta\}, \Sigma, \{q\}, R, q)$, where R consists of the rules $q[+[x_1, x_2]] \to q[x_i]$ $(i = 1, 2)$ and, for all $\mathtt{f}^{(k)} \in \Sigma$, $q[\mathtt{f}[x_1, \ldots, x_k]] \to \mathtt{f}[q[x_1], \ldots, q[x_k]]$. Obviously, for $t \in \mathrm{T}_{\Sigma\cup\{+,\theta\}}$, $set_\Sigma(t)$ is the set of trees obtained from t by interpreting $+$ as set union (of sets of trees) and θ as the empty set. In the following, we let SET denote the set of all td transductions set_Σ, i.e., for all finite signatures Σ. We usually drop the index Σ, thus writing *set* instead of set_Σ.

A td transducer is *linear* (*nondeleting*) if, for every rule $q[\mathtt{f}[x_1, \ldots, x_k]] \to t$, t is linear in X_k (nondeleting in X_k, resp.). It is *total* (*deterministic*) if, for all $q \in Q$ and $\mathtt{f}^{(k)} \in \Sigma$, there is at least one (at most one) rule with the left-hand side $q[\mathtt{f}[x_1, \ldots, x_k]]$. A td transduction is said to have one of these properties, or a combination thereof, if it can be computed by a td transducer that fulfills the requirements in question. A *tree homomorphism* is a tree transduction that can be computed by a total deterministic td transducer whose set of states is a singleton. For example, the td transducers set_Σ are linear, but neither nondeleting nor deterministic (because of the rules $q[+[x_1, x_2]] \to q[x_i]$) nor total (since there are no rules with the left-hand side $q[\theta]$).

YIELD mappings, which seem to have appeared for the first time in [Mai74], formalize the construction of trees using parameter substitution. Given a finite signature Σ and a number $m \in \mathbb{N}$ such that $\Sigma \cap X_m = \emptyset$ and $k \leq m$ for all $\mathtt{f}^{(k)} \in \Sigma$, let the *derived signature* $\Delta(\Sigma, m)$ consist of all symbols $\mathtt{subst}_n^{(n+1)}$, where $1 \leq n \leq m$, and all constant symbols $\mathtt{f}^{\langle k\rangle}$, where $\mathtt{f}^{(k)} \in \Sigma \cup X_m$. In other words, in a derived signature, each of the original symbols $\mathtt{f}^{(k)} \in \Sigma \cup X_m$ is turned into a symbol $\mathtt{f}^{\langle k\rangle}$ of rank 0. The only symbols of rank greater than 0 are the symbols $\mathtt{subst}_1, \ldots, \mathtt{subst}_m$. For a signature Ω, we denote by $\Omega^{\langle\rangle}$ the set of nullary symbols $\{\mathtt{f}^{\langle k\rangle} \mid \mathtt{f}^{(k)} \in \Omega\}$; thus, if Ω is a subset of Σ, then $\Delta(\Sigma, m) = \Delta(\Sigma\backslash\Omega, m)\cup\Omega^{\langle\rangle}$. For an arbitrary parameter signature $Z = \{y_1, \ldots, y_m\}$, instead of X_m, we similarly define the derived alphabet $\Delta(\Sigma, Z)$.

Definition 8.2 (YIELD mapping)
For a derived signature $\Delta(\Sigma, m)$, the *YIELD mapping on* $\mathrm{T}_{\Delta(\Sigma,m)}$ is the mapping $Y_{\Sigma,m} \colon \mathrm{T}_{\Delta(\Sigma,m)} \to \mathrm{T}_{\Sigma,m}$ which is defined recursively, as follows. For every tree

$s \in \mathrm{T}_{\Delta(\Sigma,m)}$,

$$Y_{\Sigma,m}(s) = \begin{cases} Y_{\Sigma,m}(s_0)(Y_{\Sigma,m}(s_1), \ldots, Y_{\Sigma,m}(s_n)) & \text{if } s = \mathtt{subst}_n[s_0, s_1, \ldots, s_n] \\ \mathtt{f}[x_1, \ldots, x_k] & \text{if } s = \mathtt{f}^{\langle k \rangle} \end{cases}$$

where $1 \leq n \leq m$ and $\mathtt{f}^{(k)} \in \Sigma \cup X_m$.[27]

For an arbitrary parameter signature $Z = \{y_1, \ldots, y_m\}$ (disjoint with Σ), we similarly define the YIELD mapping $Y_{\Sigma,Z} \colon \mathrm{T}_{\Delta(\Sigma,Z)} \to \mathrm{T}_{\Sigma,Z}$. The set of all YIELD mappings is denoted by YIELD.

Although several slightly different definitions of YIELD are found in the literature [ES77, ES78, EV85, DE98, FV98], it is easy to see that the definition given above is equivalent to each of them, in all respects relevant for this paper.

In the following, we simply denote $Y_{\Sigma,m}$ by Y, assuming that Σ and m are appropriately chosen, so that all symbols in the trees $Y_{\Sigma,m}$ is applied to are covered. Note that, in particular, $Y(x_i^{\langle 0 \rangle}) = x_i$ for all $i \in [m]$. For technical convenience, we identify the symbol $\Box_i^{\langle 0 \rangle}$ with $\Box_i$, for all $i \geq 1$. As a consequence, $Y(\Box_i) = \Box_i$.

We will need a technical lemma concerning a notion of $\mathcal{G}$-preservation, which is defined as follows. For a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$ and a tree $s \in \mathrm{T}_{\Delta(\Sigma_\mathcal{N},m)}$, we say that s is *$\mathcal{G}$-preserving* if, for every subtree of s of the form $\mathtt{subst}_n[s_0, s_1, \ldots, s_n]$ (where $n \leq m$), either $s_1, \ldots, s_n \in \mathrm{T}_{\Delta(\Pi,m)}$ or $s_0 = \mathtt{f}^{\langle n \rangle}$ for some $\mathtt{f} \in \Sigma_\mathcal{N} \cup X_m$ (or both). Note that $\Delta(\Sigma_\mathcal{N}, m) = \Delta(\Pi, m) \cup \mathcal{G}^{\langle\rangle}$.

As the term *$\mathcal{G}$-preserving* indicates, occurrences of symbols from $\mathcal{G}^{\langle\rangle}$ in s are preserved under Y. More precisely, every occurrence of a symbol $\mathtt{g}^{\langle k \rangle} \in \mathcal{G}^{\langle\rangle}$ in s corresponds to a unique occurrence of $\mathtt{g}^{(k)}$ in $Y(s)$. In fact, as the following technical lemma shows, even the property of $\mathtt{g}^{\langle k \rangle}$ occurring rightmost in s is preserved (in both directions). To formulate this, for a tree $s_0 \in \mathrm{T}_{\Delta(\Sigma_\mathcal{N},m),\{\Box\}}$ containing exactly one occurrence of $\Box$, let us say that s_0 is *rightmost* if no symbol of $\mathcal{G}^{\langle\rangle}$ occurs to the right of $\Box$ in s_0. Similarly, for a tree $t_0 \in \mathrm{T}_{\Sigma_\mathcal{N} \cup X_m,\{\Box\}}$ containing exactly one occurrence of $\Box$, we call t_0 *rightmost* if no symbol of $\mathcal{G}$ occurs to the right of $\Box$ in it.

Lemma 8.3 Let $s \in \mathrm{T}_{\Delta(\Sigma_\mathcal{N},m)}$ be $\mathcal{G}$-preserving.

1. If s has the form $s_0 \cdot \mathtt{g}^{\langle k \rangle}$ for a symbol $\mathtt{g}^{(k)} \in \mathcal{G}$ and a rightmost tree s_0, then $Y(s)$ has the form $Y(s_0) \cdot \mathtt{g}[t_1, \ldots, t_k]$, where $Y(s_0)$ is rightmost and $t_1, \ldots, t_k \in \mathrm{T}_{\Pi,m}$. Moreover, for $u \in \mathrm{T}_{\Delta(\Sigma_\mathcal{N},m)}$ with $Y(u) \in \mathrm{T}_{\Sigma_\mathcal{N},k}$, we have $Y(s_0 \cdot u) = Y(s_0) \cdot (Y(u)(t_1, \ldots, t_k))$.
2. If $Y(s) \in \mathrm{T}_{\Pi,m}$, then $s \in \mathrm{T}_{\Delta(\Pi,m)}$.

[27] For $t_0, t_1, \ldots, t_n \in \mathrm{T}_{\Sigma,m}$, we use here, and in what follows, the notation $t_0(t_1, \ldots, t_n)$ for $t_0(t_1, \ldots, t_n, x_{n+1}, \ldots, x_m)$ when $n < m$. This is consistent with the fact that $\mathrm{T}_{\Sigma,m} = \mathrm{T}_{\Sigma',n}$ where $\Sigma' = \Sigma \cup \{x_{n+1}, \ldots, x_m\}$.

3. If $Y(s)$ has the form $t_0 \cdot \mathtt{g}[t_1, \ldots, t_k]$ for a symbol $\mathtt{g} \in \mathcal{G}$, a rightmost tree t_0, and trees $t_1, \ldots, t_k \in \mathrm{T}_{\Pi,m}$, then s can be written as $s_0 \cdot \mathtt{g}^{\langle k \rangle}$, where s_0 is rightmost and $Y(s_0) = t_0$.

Proof (1) The proof is by induction on the structure of s_0. The case that $s_0 = \Box$ is obvious, with $t_i = x_i$ for $i \in [k]$. Now let $s_0 = \mathtt{subst}_n[s'_0, s_1, \ldots, s_n]$.

Case 1: $\Box$ occurs in s_i for $i \in [n]$. Then $s = \mathtt{subst}_n[s'_0, s_1, \ldots, s_i \cdot \mathtt{g}^{\langle k \rangle}, \ldots, s_n]$, and so $s'_0 = \mathtt{f}^{\langle n \rangle}$ for some $\mathtt{f} \in \Sigma_{\mathcal{N}}$ because s is $\mathcal{G}$-preserving. Hence $Y(s) = \mathtt{f}[Y(s_1), \ldots, Y(s_i \cdot \mathtt{g}^{\langle k \rangle}), \ldots, Y(s_n)]$. By the induction hypothesis,

$$Y(s_i \cdot \mathtt{g}^{\langle k \rangle}) = Y(s_i) \cdot \mathtt{g}[t_1, \ldots, t_k].$$

Consequently,

$$\begin{aligned} Y(s) &= \mathtt{f}[Y(s_1), \ldots, Y(s_i) \cdot \mathtt{g}[t_1, \ldots, t_k], \ldots, Y(s_n)] \\ &= \mathtt{f}[Y(s_1), \ldots, Y(s_i), \ldots, Y(s_n)] \cdot \mathtt{g}[t_1, \ldots, t_k] \\ &= Y(s_0) \cdot \mathtt{g}[t_1, \ldots, t_k]. \end{aligned}$$

The remaining requirements are easy to check.

Case 2: $\Box$ occurs in s'_0. Then $s = \mathtt{subst}_n[s'_0 \cdot \mathtt{g}^{\langle k \rangle}, s_1, \ldots, s_n]$. Note that $s_1, \ldots, s_n$ do not contain symbols of $\mathcal{G}^{\langle\rangle}$. By the induction hypothesis,

$$Y(s'_0 \cdot \mathtt{g}^{\langle k \rangle}) = Y(s'_0) \cdot \mathtt{g}[t_1, \ldots, t_k].$$

Consequently,

$$\begin{aligned} Y(s) &= (Y(s'_0) \cdot \mathtt{g}[t_1, \ldots, t_k])(Y(s_1), \ldots, Y(s_n)) \\ &= (Y(s'_0)(Y(s_1), \ldots, Y(s_n))) \cdot \mathtt{g}[t'_1, \ldots, t'_k] \\ &= Y(s_0) \cdot \mathtt{g}[t'_1, \ldots, t'_k] \end{aligned}$$

where $t'_i = t_i(Y(s_1), \ldots, Y(s_n)) \in \mathrm{T}_{\Pi,m}$ for $i \in [k]$. Again, the remaining requirements are easy to check.

(2) This is immediate from Statement 1: if s contains a symbol of $\mathcal{G}^{\langle\rangle}$, then one can consider the rightmost occurrence of such a symbol and write s as $s_0 \cdot \mathtt{g}^{\langle k \rangle}$ where s_0 is rightmost.

(3) The proof is by induction on the structure of s. The case that s is a constant symbol is easy: s must be equal to $\mathtt{g}^{\langle k \rangle}$, $Y(s) = \mathtt{g}[x_1, \ldots, x_k]$, $t_0 = \Box$, $t_i = x_i$ for $i \in [k]$, and $s_0 = \Box$. Now let $s = \mathtt{subst}_n[u, s_1, \ldots, s_n]$. Then $Y(s) = Y(u)(Y(s_1), \ldots, Y(s_n)) = t_0 \cdot \mathtt{g}[t_1, \ldots, t_k]$. Let $v \in V(t_0)$ be the node with label $\Box$. We distinguish two cases.

Case 1: v is a node of $Y(u)$ and its label (in $Y(u)$) is not in X_m. Then $Y(u)$ has the form $t'_0 \cdot \mathtt{g}[t'_1, \ldots, t'_k]$ and $t_i = t'_i(Y(s_1), \ldots, Y(s_n))$ for $i = 0$ and for all $i \in [k]$. By the induction hypothesis, u can be written as $u_0 \cdot \mathtt{g}^{\langle k \rangle}$ where $Y(u_0) = t'_0$. Let $s_0 = \mathtt{subst}_n[u_0, s_1, \ldots, s_n]$. Then $s = \mathtt{subst}_n[u_0 \cdot \mathtt{g}^{\langle k \rangle}, s_1, \ldots, s_n] = s_0 \cdot \mathtt{g}^{\langle k \rangle}$ and $Y(s_0) = t'_0(Y(s_1), \ldots, Y(s_n)) = t_0$.

It remains to be shown that no symbol of $\mathcal{G}^{\langle\rangle}$ occurs in $s_1, \ldots, s_n$. Since s is $\mathcal{G}$-preserving, it suffices to consider the case that $u = \mathtt{g}^{\langle k \rangle}$ and $k = n$. Then

$t_0 = t'_0 = \Box$ and $Y(s) = \mathtt{g}[Y(s_1), \ldots, Y(s_n)] = \mathtt{g}[t_1, \ldots, t_n]$. Hence no symbol of $\mathcal{G}$ occurs in $Y(s_1), \ldots, Y(s_n)$. Now Statement 2 gives the result.

Case 2: v is a node of one of the occurrences of $Y(s_1), \ldots, Y(s_n)$ in $Y(s)$. Then there exists $i \in [n]$ such that $Y(s_i)$ has the form $t'_0 \cdot \mathtt{g}[t_1, \ldots, t_k]$. Thus, s_i contains a symbol of $\mathcal{G}^{\langle\rangle}$. Since s is $\mathcal{G}$-preserving, $u = \mathtt{f}^{\langle n\rangle}$ for some $\mathtt{f} \in \Sigma_{\mathcal{N}}$. Hence $Y(s) = \mathtt{f}[Y(s_1), \ldots, t'_0 \cdot \mathtt{g}[t_1, \ldots, t_k], \ldots, Y(s_n)]$ and $t_0 = \mathtt{f}[Y(s_1), \ldots, t'_0, \ldots, Y(s_n)]$. By the induction hypothesis, s_i can be written as $s_{i,0} \cdot \mathtt{g}^{\langle k\rangle}$ where $Y(s_{i,0}) = t'_0$. Take $s_0 = \mathtt{subst}_n[u, s_1, \ldots, s_{i,0}, \ldots, s_n]$. Then $s = \mathtt{subst}_n[u, s_1, \ldots, s_{i,0} \cdot \mathtt{g}^{\langle k\rangle}, \ldots, s_n] = s_0 \cdot \mathtt{g}^{\langle k\rangle}$ and $Y(s_0) = \mathtt{f}[Y(s_1), \ldots, t'_0, \ldots, Y(s_n)] = t_0$. It follows from the form of t_0, and from the induction hypothesis and Statement 2, that s_0 is rightmost. ∎

Given a derived signature $\Delta(\Sigma, m)$, there is a useful linear and nondeleting tree homomorphism $\mathrm{comb}_{\Sigma,m} \colon \mathrm{T}_{\Sigma,m} \to \mathrm{T}_{\Delta(\Sigma,m)}$ (cf. [ES77, Definition 4.4]). It is given by

$$\mathrm{comb}_{\Sigma,m}(\mathtt{f}[t_1, \ldots, t_k]) = \mathtt{subst}_k[\mathtt{f}^{\langle k\rangle}, \mathrm{comb}_{\Sigma,m}(t_1), \ldots, \mathrm{comb}_{\Sigma,m}(t_k)],$$

for all trees $\mathtt{f}[t_1, \ldots, t_k] \in \mathrm{T}_{\Sigma,m}$ with $k \geq 1$, and $\mathrm{comb}_{\Sigma,m}(\mathtt{f}) = \mathtt{f}^{\langle 0\rangle}$ for $\mathtt{f}^{(0)} \in \Sigma \cup X_m$. In the following, we will often simply write 'comb' instead of '$\mathrm{comb}_{\Sigma,m}$'. The comb mapping has the following obvious properties.

Lemma 8.4 For every derived signature $\Delta(\Sigma, m)$ and every tree $t \in \mathrm{T}_{\Sigma,m}$,

1. $Y_{\Sigma,m}(\mathrm{comb}_{\Sigma,m}(t)) = t$, and
2. $\mathrm{comb}_{\Sigma,m}(t)$ is $\mathcal{G}$-preserving if $\Sigma = \Sigma_{\mathcal{N}}$ for some $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$.

The so-called TBY hierarchy [DE98] is obtained by starting with the regular tree languages, and then applying arbitrary td transducers and YIELD mappings to them.[28] Formally, let TBY be the set of all tree transductions of the form $\tau_m \circ \cdots \circ \tau_1$, where $m \geq 1$, $\tau_1, \ldots, \tau_m \in \mathrm{TD} \cup \mathrm{YIELD}$ and $\tau_i \in \mathrm{YIELD}$ for at most one $i \in [m]$.[29] It is not known whether the TBY hierarchy, which is given by $(\mathrm{TBY}^n(\mathrm{REG}))_{n\in\mathbb{N}}$, is proper at each level. It is closely related to the hierarchy obtained by applying n (nondeterministic) macro tree transducers to regular tree languages. In particular, $\mathrm{TBY}^*(\mathrm{REG})$ is equal to $\mathrm{MTT}^*(\mathrm{REG})$, the closure of REG under macro tree transductions. In this section, we show that $\mathrm{DEL}^*(\mathrm{FIN})$ is properly contained in $\mathrm{TBY}^*(\mathrm{REG})$, and that the latter is closed under DEL.

Example 8.5 Consider the monadic tree language $L_{\mathrm{monexp}} = \{\mathtt{f}^{2^n}[\mathtt{a}] \mid n \in \mathbb{N}\}$ consisting of all trees over $\{\mathtt{f}^{(1)}, \mathtt{a}^{(0)}\}$ that contain 2^n occurrences of $\mathtt{f}$, for $n \in \mathbb{N}$. It is well known that L_{monexp} is in $\mathrm{YIELD}(\mathrm{TD}(\mathrm{REG})) \subseteq \mathrm{TBY}(\mathrm{REG})$: use a

[28] The three letters T, B, and Y refer to *top-down tree transducers*, *bottom-up tree transducers*, and *YIELD mappings*. In the definition used here, we omit bottom-up tree transducers, because it is known that each bottom-up tree transducer can be simulated by two top-down tree transducers.

[29] Note that, by Lemma 8.4(1) and the fact that $\mathrm{comb}_{\Sigma,m} \in \mathrm{TD}$, the same class TBY is obtained if we require that $\tau_i \in \mathrm{YIELD}$ for exactly one $i \in [m]$.

regular tree grammar γ with the rules $\xi_0 \to \mathtt{f}[\xi_0]$ and $\xi_0 \to \mathtt{a}$ and a td transducer td with the rules

$$\begin{aligned} q_0[\mathtt{f}[x_1]] &\to \mathtt{subst}_1[q[x_1], \mathtt{a}^{\langle 0\rangle}] \\ q[\mathtt{f}[x_1]] &\to \mathtt{subst}_1[q[x_1], q[x_1]] \\ q[\mathtt{a}] &\to \mathtt{f}^{\langle 1\rangle}. \end{aligned}$$

Thus, $td(L(\gamma))$ consists of all trees $\mathtt{subst}_1[t, \mathtt{a}^{\langle 0\rangle}]$ where t is a fully balanced binary tree over the signature $\{\mathtt{subst}_1, \mathtt{f}^{\langle 1\rangle}\}$. Obviously, $Y(td(\mathtt{f}^{n+1}[\mathtt{a}])) = \mathtt{f}^{2^n}[\mathtt{a}]$ and, thus, $Y(td(L(\gamma))) = L_{\mathrm{monexp}}$. The tree language L_{monexp} is not in DEL*(FIN), by Theorem 7.23, because paths(L_{monexp}) is not context-free (which is clear from the fact that paths(L_{monexp}) $\cap\, \langle \mathtt{f}, 1\rangle^* \langle \mathtt{a}, 0\rangle = \{\langle \mathtt{f}, 1\rangle^{2^n} \langle \mathtt{a}, 0\rangle \mid n \geq 0\}$). ∎

Note that, of course, DEL*(FIN) = DEL*(REG), because REG $\subseteq$ DEL(FIN). In fact, we can turn the subset sign in the latter formula into an equality sign if we restrict ourselves to delegation networks that work in a "regular" manner. More precisely, we say that a delegation network is *regular* if the rank of every generator symbol is 0. As a consequence, IO-derivations generate the same trees as unrestricted derivations. Intuitively, this means that a regular delegation network (viewed as a tree-generating device) is nothing else than a regular tree grammar with a possibly infinite set of rules, where the set of right-hand sides of rules for a given left-hand side is generated by the tree generator associated with it. Thus, regular delegation networks have the following property, similar to regular tree grammars.

Lemma 8.6 Let $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ be a regular delegation network, and consider a generator symbol $\mathtt{g} \in \mathcal{G}$ and a tree $t \in \mathrm{T}_{\Sigma_{\mathcal{N}}}$. For $n \in \mathbb{N}$, $\mathtt{g} \Rightarrow^{n+1}_{\mathcal{N},\mathrm{IO}} t$ if and only if there is a tree $t_0 \cdot (\mathtt{g}_1, \ldots, \mathtt{g}_k) \in L(\gamma_{\mathtt{g}})$, where $t_0 \in \mathrm{T}_{\Pi,\{\square_1,\ldots,\square_k\}}$ and $\mathtt{g}_1, \ldots, \mathtt{g}_k \in \mathcal{G}$, and there are derivations $\mathtt{g}_1 \Rightarrow^{n_1}_{\mathcal{N},\mathrm{IO}} t_1, \ldots, \mathtt{g}_k \Rightarrow^{n_k}_{\mathcal{N},\mathrm{IO}} t_k$, such that $t = t_0 \cdot (t_1, \ldots, t_k)$ and $n = \sum_{i=1}^{k} n_i$.

For a class C of tree languages, let rDEL(C) be defined in the same way as DEL(C), except that the delegation networks considered are required to be regular. The operator rDEL will play an important role in the proofs of this section. Clearly, as illustrated by the three delegation networks of Example 2.6, REG = rDEL(FIN) = rDEL(REG). The second equality is a well-known result in disguise: extended regular tree grammars, with regular tree languages as right-hand sides, generate regular tree languages. (In turn, since trees are strings and regular tree grammars are a special case of context-free grammars, the latter is a special case of the fact that cf-extended context-free string grammars generate context-free languages.) More generally, using similar well-known arguments, we obtain the following result.

Lemma 8.7 The operator rDEL is idempotent on genuine classes C of tree languages, i.e., rDEL(rDEL(C)) = rDEL(C).

Proof Since $C' \subseteq \mathrm{rDEL}(C')$ for every class C', one inclusion is obvious. For the other inclusion the construction is similar to the one of Lemma 7.8. Consider a regular delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ and, for every $\mathtt{g} \in \mathcal{G}$, a regular delegation network $\mathcal{N}_\mathtt{g} = (\mathcal{G}_\mathtt{g}, \Sigma_\mathcal{N}, (\gamma_\mathtt{h})_{\mathtt{h}\in\mathcal{G}_\mathtt{g}}, \mathtt{h}_{\mathtt{g},0})$ such that $L_\mathrm{T}(\mathcal{N}_\mathtt{g}) = L(\gamma_\mathtt{g})$ and $L(\gamma_\mathtt{h}) \in C$ for every $\mathtt{g} \in \mathcal{G}$ and $\mathtt{h} \in \mathcal{G}_\mathtt{g}$. Since C is genuine, we may assume that the signatures $\mathcal{G}_\mathtt{g}$, $\mathtt{g} \in \mathcal{G}$, are pairwise disjoint. Then $L(\mathcal{N}) = L(\mathcal{N}')$, where $\mathcal{N}' = (\mathcal{G}', \Pi, (\gamma'_{\mathtt{g}'})_{\mathtt{g}'\in\mathcal{G}}, \mathtt{g}_0)$ is the regular delegation network such that $\mathcal{G}' = \mathcal{G} \cup \bigcup_{\mathtt{g}\in\mathcal{G}} \mathcal{G}_\mathtt{g}$ and for every $\mathtt{g} \in \mathcal{G}$ and $\mathtt{h} \in \mathcal{G}_\mathtt{g}$, $L(\gamma'_\mathtt{g}) = \{\mathtt{h}_{\mathtt{g},0}\}$ and $L(\gamma'_\mathtt{h}) = L(\gamma_\mathtt{h})$. ∎

We now prove a useful decomposition lemma. In the proof, we need the notion of rightmost derivation of a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$. A derivation step of $\mathcal{N}$ as in Definition 6.3 is *rightmost*, denoted $t_0 \cdot \mathtt{g}[s_1, \ldots, s_k] \Rightarrow_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t_0 \cdot u(s_1, \ldots, s_k)$, if t_0 is rightmost. It is easy to see that $\mathtt{g}_0 \Rightarrow^*_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t$ for every $t \in L_\mathrm{T}(\mathcal{N})$. In fact, if $t_1 \Rightarrow_{\mathcal{N},\mathrm{IO}} t_2 \Rightarrow_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t_3$ and the first derivation step is *not* rightmost, then there exists t'_2 such that $t_1 \Rightarrow_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t'_2 \Rightarrow_{\mathcal{N},\mathrm{IO}} t_3$.[30] Thus, the rightmost derivation steps can be moved to the left until all have become rightmost (because the last derivation step is necessarily rightmost).

Lemma 8.8 If C is a class of tree languages that is closed under linear nondeleting tree homomorphisms[31], then $\mathrm{DEL}(C) \subseteq \mathrm{YIELD}(\mathrm{rDEL}(C))$.

Proof Consider a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$, where $L(\gamma_\mathtt{g}) \in C$ for all $\mathtt{g} \in \mathcal{G}$. The construction of $\mathcal{N}'$ is similar to the definition of G^D in [ES77, Definition 4.5]. Let $\mathcal{N}' = (\mathcal{G}', \Delta(\Pi, m), (\gamma_{\mathtt{g}'})_{\mathtt{g}'\in\mathcal{G}'}, \mathtt{g}'_0)$, where

- m is the maximum rank of symbols in $\Sigma_\mathcal{N}$,
- $\mathcal{G}' = \mathcal{G}^{\langle\rangle}$, hence $\Sigma_{\mathcal{N}'} = \Delta(\Sigma_\mathcal{N}, m)$,
- $L(\gamma_{\mathtt{g}^{\langle k\rangle}}) = \mathrm{comb}_{\Sigma_\mathcal{N},m}(L(\gamma_\mathtt{g}))$, for all $k \in [m]$ and $\mathtt{g}^{(k)} \in \mathcal{G}$, and
- $\mathtt{g}'_0 = \mathtt{g}_0^{\langle 0\rangle}$.

Note that $L(\gamma_{\mathtt{g}^{\langle k\rangle}}) \in C$ because $\mathrm{comb}_{\Sigma_\mathcal{N},m}$ is a linear nondeleting tree homomorphism. Note also that in the case where Π contains symbols x_i, we use a derived alphabet $\Delta(\Pi, Z)$ with $Z = \{y_1, \ldots, y_m\}$ disjoint with Π, instead of $\Delta(\Pi, m)$. We have to show that $L_\mathrm{T}(\mathcal{N}) = Y(L_\mathrm{T}(\mathcal{N}'))$. To prove this, we proceed by induction on the length of rightmost derivations. We first observe that if $\mathtt{g}'_0 \Rightarrow^*_{\mathcal{N}',\mathrm{IO},\mathrm{rm}} s$ then s is $\mathcal{G}$-preserving. In fact, if $s_0 \cdot \mathtt{g}^{\langle k\rangle} \Rightarrow_{\mathcal{N}',\mathrm{IO},\mathrm{rm}} s_0 \cdot \mathrm{comb}(u)$ and $s_0 \cdot \mathtt{g}^{\langle k\rangle}$ is $\mathcal{G}$-preserving, then so is $s_0 \cdot \mathrm{comb}(u)$, as one easily proves by induction on the structure of s_0 using Lemma 8.4(2).

To show the inclusion $L_\mathrm{T}(\mathcal{N}) \subseteq Y(L_\mathrm{T}(\mathcal{N}'))$, we prove that for every derivation $\mathtt{g}_0 \Rightarrow^n_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t$, there is a derivation $\mathtt{g}'_0 \Rightarrow^*_{\mathcal{N}',\mathrm{IO},\mathrm{rm}} s$ such that $Y(s) = t$.[32] For $n = 0$,

[30] To be precise, t_1 must be of the form $t_0 \cdot (\mathtt{g}[s_1, \ldots, s_k], \mathtt{g}'[s'_1, \ldots, s'_l])$ where $\Box_1$ occurs to the left of $\Box_2$ in t_0 and no symbol of $\mathcal{G}$ occurs to the right of $\Box_2$. Also, $t_2 = t_0 \cdot (u(s_1, \ldots, s_k), \mathtt{g}'[s'_1, \ldots, s'_l])$ and $t_3 = t_0 \cdot (u(s_1, \ldots, s_k), u'(s'_1, \ldots, s'_l))$. Then $t'_2 = t_0 \cdot (\mathtt{g}[s_1, \ldots, s_k], u'(s'_1, \ldots, s'_l))$.

[31] Such a class C is genuine if and only if $\mathrm{FIN} \subseteq C$.

[32] Note that this implies the inclusion: if $t \in \mathrm{T}_\Pi$, then $s \in \mathrm{T}_{\Delta_{\Pi,m}}$ by Lemma 8.3(2).

this is trivial. Therefore, let

$$\mathtt{g}_0 \Rightarrow^n_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t_0 \cdot \mathtt{g}[t_1,\ldots,t_k] \Rightarrow_{\mathcal{N},\mathrm{IO},\mathrm{rm}} t_0 \cdot u(t_1,\ldots,t_k),$$

where $\mathtt{g} \in \mathcal{G}$, $t_1,\ldots,t_k \in \mathrm{T}_\Pi$, and $u \in L(\gamma_\mathtt{g}) \subseteq \mathrm{T}_{\Sigma_\mathcal{N},k}$. By the induction hypothesis, $\mathtt{g}'_0 \Rightarrow^n_{\mathcal{N}',\mathrm{IO},\mathrm{rm}} s$, where $Y(s) = t_0 \cdot \mathtt{g}[t_1,\ldots,t_k]$. By Lemma 8.3(3), this means that $s = s_0 \cdot \mathtt{g}^{\langle k\rangle}$, for a tree s_0 such that $Y(s_0) = t_0$ and s_0 is rightmost. By the definition of $\mathcal{N}'$, we have $s_0 \cdot \mathtt{g}^{\langle k\rangle} \Rightarrow_{\mathcal{N}',\mathrm{IO},\mathrm{rm}} s'$ with $s' = s_0 \cdot \mathrm{comb}(u)$. By Lemmas 8.3(1) and 8.4(1), $Y(s') = t_0 \cdot Y(\mathrm{comb}(u))(t_1,\ldots,t_k) = t_0 \cdot u(t_1,\ldots,t_k)$.

The proof of the inclusion $Y(L_\mathrm{T}(\mathcal{N}')) \subseteq L_\mathrm{T}(\mathcal{N})$ is similar. It only uses Lemmas 8.3(1) and 8.4. ∎

Let us have a look at two examples that illustrate the preceding decomposition lemma.

Example 8.9 The tree language $L_\mathrm{T}(\mathcal{N}_\mathrm{bin})$ of fully balanced binary trees in Example 7.2 is in DEL(FIN) and hence in YIELD(rDEL(FIN)). It is equal to $Y(L_\mathrm{T}(\mathcal{N}'_\mathrm{bin}))$ where $L(\gamma_{\mathtt{g}_0^{\langle 0\rangle}}) = \{\mathtt{subst}_1[\mathtt{g}^{\langle 1\rangle}, \mathtt{h}^{\langle 0\rangle}]\}$ and $L(\gamma_{\mathtt{g}^{\langle 1\rangle}})$ consists of the trees $\mathtt{subst}_1[\mathtt{g}^{\langle 1\rangle}, \mathtt{subst}_2[\mathtt{f}^{\langle 2\rangle}, x_1^{\langle 0\rangle}, x_1^{\langle 0\rangle}]]$ and $\mathtt{subst}_2[\mathtt{f}^{\langle 2\rangle}, x_1^{\langle 0\rangle}, x_1^{\langle 0\rangle}]$. ∎

Example 8.10 The tree language $td(L(\gamma))$ from Example 8.5 is also in the class DEL(FIN), cf. Example 7.2: it is generated by the delegation network with generator symbols $\mathtt{g}_0^{(0)}$ and $\mathtt{g}^{(1)}$, and with $L(\gamma_{\mathtt{g}_0}) = \{\mathtt{subst}_1[\mathtt{g}[\mathtt{f}^{\langle 1\rangle}], \mathtt{a}^{\langle 0\rangle}]\}$ and $L(\gamma_\mathtt{g}) = \{\mathtt{g}[\mathtt{subst}_1[x_1, x_1]], x_1\}$. Thus, by Example 8.5, the tree language L_monexp is in YIELD(DEL(FIN)) and hence, trivially, in the class YIELD(rDEL(C)) where C = DEL(FIN). However, as observed in that example, L_monexp is not in DEL(C). ∎

The last example shows that the inclusion in Lemma 8.8 is not an equality for C = DEL(FIN). This is in contrast to the well-known fact that for C = FIN equality holds: DEL(FIN) = YIELD(REG), see [ES77, Corollary 4.12]. The latter equality implies that DEL(REG) = DEL(FIN) (cf. the last paragraph of Example 2.7(a)): by Lemma 8.8, DEL(REG) ⊆ YIELD(rDEL(REG)) which is equal to YIELD(REG) by Lemma 8.7.

Next, we want to prove a lemma that allows us to reorder the operators rDEL and YIELD. To be able to do that, we need an elementary property of Y.

Lemma 8.11 For every derived signature $\Delta(\Sigma, m)$ and tree $s = s_0 \cdot (s_1,\ldots,s_l)$ in $\mathrm{T}_{\Delta(\Sigma,m)}$, if $Y(s_1),\ldots,Y(s_l) \in \mathrm{T}_\Sigma$, then $Y(s) = Y(s_0)[\![Y(s_1),\ldots,Y(s_l)]\!]$.[33]

Proof We prove the statement by structural induction on s_0, but for arbitrary $s_0 \in \mathrm{T}_{\Delta(\Sigma,m),\{\Box_1,\ldots,\Box_l\}}$, i.e., where $s = s_0[\![s_1,\ldots,s_l]\!]$. For $i \in [l]$, let $t_i = Y(s_i)$.

[33]Note that $Y(s_0)$ is in $\mathrm{T}_{\Sigma\cup X_m\cup\{\Box_1,\ldots,\Box_l\}}$ and recall furthermore from Section 2.3 that $Y(s_0)[\![Y(s_1),\ldots,Y(s_l)]\!]$ is obtained from $Y(s_0)$ by replacing each occurrence of $\Box_i$ with $Y(s_i)$.

If $s_0 = \square_i$ for an $i \in [l]$, then $Y(s) = t_i = Y(s_0)[\![t_1, \ldots, t_l]\!]$. If $s_0 = \mathtt{f}^{\langle k \rangle}$ for a symbol $\mathtt{f}^{(k)} \in \Sigma \cup X_m$, then $s = s_0$ and, thus, $Y(s) = Y(s_0) = Y(s_0)[\![t_1, \ldots, t_l]\!]$.

Finally, if $s_0 = \mathtt{subst}_k[u_0, \ldots, u_k]$, using first the definition of Y, then the induction hypothesis, and then the assumption that $t_1, \ldots, t_l \in \mathrm{T}_\Sigma$, we get

$$\begin{aligned} Y(s) &= Y(u_0[\![s_1, \ldots, s_l]\!])(Y(u_1[\![s_1, \ldots, s_l]\!]), \ldots, Y(u_k[\![s_1, \ldots, s_l]\!])) \\ &= Y(u_0)[\![t_1, \ldots, t_l]\!](Y(u_1)[\![t_1, \ldots, t_l]\!], \ldots, Y(u_k)[\![t_1, \ldots, t_l]\!]) \\ &= Y(u_0)(Y(u_1), \ldots, Y(u_k))[\![t_1, \ldots, t_l]\!] \\ &= Y(s_0)[\![t_1, \ldots, t_l]\!] \end{aligned}$$

as claimed. ∎

We can now prove the mentioned lemma, that makes it possible to reorder rDEL and YIELD.

Lemma 8.12 For every genuine class C of tree languages,

$$\mathrm{rDEL}(\mathrm{YIELD}(C)) \subseteq \mathrm{SET}(\mathrm{YIELD}(\mathrm{rDEL}(C))).$$

Proof Consider a regular delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathtt{g})_{\mathtt{g} \in \mathcal{G}}, \mathtt{g}_0)$, where $L(\gamma_\mathtt{g}) = Y(L_\mathtt{g})$ with $L_\mathtt{g} \in C$. We first treat the special case where $\Sigma_\mathcal{N}$ does not contain parameters that are used by the YIELD mappings on the tree languages $L_\mathtt{g}$. In this case there exists a derived signature $\Delta(\Sigma_\mathcal{N}, m)$ such that $L_\mathtt{g} \subseteq \mathrm{T}_{\Delta(\Sigma_\mathcal{N}, m)}$ for all $\mathtt{g} \in \mathcal{G}$. We may assume that $m \geq 2$. Note that $\Delta(\Sigma_\mathcal{N}, m) = \Delta(\Pi, m) \cup \{\mathtt{g}^{\langle 0 \rangle} \mid \mathtt{g} \in \mathcal{G}\}$.

Now, let $\mathcal{G}' = \mathcal{G} \cup \{\mathtt{g}^{\langle 0 \rangle} \mid \mathtt{g} \in \mathcal{G}\}$ and define $\mathcal{N}' = (\mathcal{G}', \Pi', (\gamma'_\mathtt{h})_{\mathtt{h} \in \mathcal{G}'}, \mathtt{g}_0)$, where $\Pi' = \Delta(\Pi \cup \{+, \theta\}, m)$ and, for every $\mathtt{g} \in \mathcal{G}$, $L(\gamma'_\mathtt{g}) = L_\mathtt{g}$ and $L(\gamma'_{\mathtt{g}^{\langle 0 \rangle}}) = \{\mathtt{subst}_2[+^{\langle 2 \rangle}, \mathtt{g}, \mathtt{g}^{\langle 0 \rangle}], \theta^{\langle 0 \rangle}\}$.

For $\mathtt{g} \in \mathcal{G}$ and $n \in \mathbb{N}$, let $\Lambda(\mathtt{g}) = \bigcup_{n \in \mathbb{N}} \Lambda(\mathtt{g}, n)$ and $\Lambda'(\mathtt{g}) = \bigcup_{n \in \mathbb{N}} \Lambda'(\mathtt{g}, n)$, where

$$\begin{aligned} \Lambda(\mathtt{g}, n) &= \{t \in \mathrm{T}_\Pi \mid \mathtt{g} \Rightarrow^m_{\mathcal{N},\mathrm{IO}} t \text{ for some } m \leq n\} \text{ and} \\ \Lambda'(\mathtt{g}, n) &= \{s \in \mathrm{T}_{\Pi'} \mid \mathtt{g} \Rightarrow^m_{\mathcal{N}',\mathrm{IO}} s \text{ for some } m \leq n\}, \end{aligned}$$

for all $n \in \mathbb{N}$. In particular, $\Lambda(\mathtt{g}_0) = L_\mathrm{T}(\mathcal{N})$ and $\Lambda'(\mathtt{g}_0) = L_\mathrm{T}(\mathcal{N}')$. We show, simultaneously for all $\mathtt{g} \in \mathcal{G}$, that $set(Y(\Lambda'(\mathtt{g}))) = \Lambda(\mathtt{g})$.

('⊆') We divide the proof in two steps, in both cases proceeding by induction on the length of derivations, using Lemma 8.6.

First, consider a tree $s \in \mathrm{T}_{\Pi'}$ such that $\mathtt{g}^{\langle 0 \rangle} \Rightarrow^n_{\mathcal{N}',\mathrm{IO}} s$ for some $n \in \mathbb{N}$. By induction on n, we show that $set(Y(s)) \subseteq set(Y(\Lambda'(\mathtt{g}, n)))$. If $s = \theta^{\langle 0 \rangle}$, then $set(Y(s)) = \emptyset$. Otherwise, $s = \mathtt{subst}_2[+^{\langle 2 \rangle}, s_1, s_2]$, where $\mathtt{g} \Rightarrow^{n_1}_{\mathcal{N}',\mathrm{IO}} s_1$ and $\mathtt{g}^{\langle 0 \rangle} \Rightarrow^{n_2}_{\mathcal{N}',\mathrm{IO}} s_2$ with $n_1 + n_2 = n - 1$. Thus,

$$set(Y(s)) = set(Y(s_1)) \cup set(Y(s_2)) \subseteq set(Y(\Lambda'(\mathtt{g}, n))),$$

since the inclusion $set(Y(s_1)) \subseteq set(Y(\Lambda'(\mathtt{g}, n_1)))$ holds by the definition of $\Lambda'(\mathtt{g}, n_1)$ and $set(Y(s_2)) \subseteq set(Y(\Lambda'(\mathtt{g}, n_2)))$ is obtained from the induction hypothesis.

Now, to show by induction on n that $set(Y(\Lambda'(\mathsf{g},n))) \subseteq \Lambda(\mathsf{g})$, let $s \in \Lambda'(\mathsf{g},n)$. By Lemma 8.6 and the construction of $\mathcal{N}'$, there is a tree $s_0 \cdot (\mathsf{g}_1^{\langle 0\rangle}, \ldots, \mathsf{g}_l^{\langle 0\rangle}) \in L_{\mathsf{g}}$, where $s_0 \in \mathrm{T}_{\Delta(\Pi,m),\{\Box_1,\ldots,\Box_l\}}$ for some $l \in \mathbb{N}$, such that $s = s_0 \cdot (s_1, \ldots, s_l)$ for trees $s_1, \ldots, s_l \in \mathrm{T}_{\Pi'}$ with $\mathsf{g}_i^{\langle 0\rangle} \Rightarrow^{n_i}_{\mathcal{N}',\mathrm{IO}} s_i$ and $\sum_{i=1}^{l} n_i = n-1$. By the previous paragraph and the induction hypothesis, the latter yields $set(Y(s_i)) \subseteq set(Y(\Lambda'(\mathsf{g}_i,n_i))) \subseteq \Lambda(\mathsf{g}_i)$ for all $i \in [l]$. In particular, $set(Y(s_i)) \subseteq \mathrm{T}_{\Pi}$, i.e., $Y(s_i) \in \mathrm{T}_{\Pi\cup\{+,\theta\}}$ for all $i \in [l]$. Consequently $Y(s) = Y(s_0)[\![Y(s_1), \ldots, Y(s_l)]\!]$ by Lemma 8.11 (for $\Sigma = \Pi \cup \{+,\theta\}$).

Since $s_0 \in \mathrm{T}_{\Delta(\Pi,m),\{\Box_1,\ldots,\Box_l\}}$, the tree $Y(s_0)$ can be written in the form $Y(s_0) = t_0 \cdot (\Box_{i_1}, \ldots, \Box_{i_k})$ with $k \in \mathbb{N}$, $t_0 \in \mathrm{T}_{\Pi\cup X_m\cup\{\Box_1,\ldots,\Box_k\}}$ and $i_1, \ldots, i_k \in [l]$. This means that

$$\begin{aligned} Y(s_0 \cdot (\mathsf{g}_1^{\langle 0\rangle}, \ldots, \mathsf{g}_l^{\langle 0\rangle})) &= Y(s_0)[\![\mathsf{g}_1, \ldots, \mathsf{g}_l]\!] && \text{(by Lemma 8.11)} \\ &= ((t_0 \cdot (\Box_{i_1}, \ldots, \Box_{i_k}))[\![\mathsf{g}_1, \ldots, \mathsf{g}_l]\!] \\ &= t_0 \cdot (\mathsf{g}_{i_1}, \ldots, \mathsf{g}_{i_k}). \end{aligned}$$

Since $Y(s_0 \cdot (\mathsf{g}_1^{\langle 0\rangle}, \ldots, \mathsf{g}_l^{\langle 0\rangle})) \in Y(L_{\mathsf{g}}) = L(\gamma_{\mathsf{g}})$, we obtain $t_0 \cdot (\mathsf{g}_{i_1}, \ldots, \mathsf{g}_{i_k}) \in L(\gamma_{\mathsf{g}})$ and $t_0 \in \mathrm{T}_{\Pi,\{\Box_1,\ldots,\Box_k\}}$.

Thus, we get

$$\begin{aligned} set(Y(s)) &= set(Y(s_0)[\![Y(s_1), \ldots, Y(s_l)]\!]) \\ &= set(t_0 \cdot (Y(s_{i_1}), \ldots, Y(s_{i_k}))) \\ &= \{t_0 \cdot (t_1, \ldots, t_k) \mid t_j \in set(Y(s_{i_j})) \text{ for } j \in [k]\} \\ &\subseteq \Lambda(\mathsf{g}) \end{aligned}$$

where the last inclusion holds by Lemma 8.6 because $t_0 \cdot (\mathsf{g}_{i_1}, \ldots, \mathsf{g}_{i_k}) \in L(\gamma_{\mathsf{g}})$ and $set(Y(s_{i_j})) \subseteq \Lambda(\mathsf{g}_{i_j})$.

('$\supseteq$') Note first that, for all trees $s_1, \ldots, s_n \in \Lambda'(\mathsf{g})$, the tree

$$s = \mathtt{subst}_2[+^{\langle 2\rangle}, s_1, \mathtt{subst}_2[+^{\langle 2\rangle}, s_2, \ldots \mathtt{subst}_2[+^{\langle 2\rangle}, s_n, \theta^{\langle 0\rangle}] \ldots]]$$

satisfies $\mathsf{g}^{\langle 0\rangle} \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s$ and $set(Y(s)) = \bigcup_{i\in[n]} set(Y(s_i))$. In particular, $t = \theta^{\langle 0\rangle}$ if $n = 0$.

We show that, for all trees $t \in \Lambda(\mathsf{g})$, there is a tree $s \in \Lambda'(\mathsf{g})$ such that $Y(s) \in \mathrm{T}_{\Pi\cup\{+,\theta\}}$ and $t \in set(Y(s))$. Again, we proceed by induction on the length of derivations. Thus, suppose there is a derivation $\mathsf{g} \Rightarrow_{\mathcal{N},\mathrm{IO}} t' \Rightarrow^n_{\mathcal{N},\mathrm{IO}} t$, and let $t' = Y(s_0 \cdot (\mathsf{g}_1^{\langle 0\rangle}, \ldots, \mathsf{g}_l^{\langle 0\rangle}))$ with $s_0 \in \mathrm{T}_{\Delta(\Pi,m),\{\Box_1,\ldots,\Box_l\}}$. As in the previous step of the proof, $Y(s_0)$ can be written as $Y(s_0) = t_0 \cdot (\Box_{i_1}, \ldots, \Box_{i_k})$ with $t_0 \in \mathrm{T}_{\Pi,\{\Box_1,\ldots,\Box_k\}}$ and $i_1, \ldots, i_k \in [l]$, such that $t' = t_0 \cdot (\mathsf{g}_{i_1}, \ldots, \mathsf{g}_{i_k})$. It follows that $t = t_0 \cdot (t_1, \ldots, t_k)$ for trees $t_1, \ldots, t_k \in \mathrm{T}_{\Pi}$ with $\mathsf{g}_{i_j} \Rightarrow^{n_j}_{\mathcal{N},\mathrm{IO}} t_j$ and $\sum_{j\in[k]} n_j = n$. Thus, the induction hypothesis yields derivations $\mathsf{g}_{i_j} \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s_j$ such that $Y(s_j) \in \mathrm{T}_{\Pi\cup\{+,\theta\}}$ and $t_j \in set(Y(s_j))$ for all $j \in [k]$.

By the observation above, the latter gives rise to derivations $\mathsf{g}_i^{\langle 0\rangle} \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} \widehat{s}_i$ such that $Y(\widehat{s}_i) \in \mathrm{T}_{\Pi\cup\{+,\theta\}}$ for all $i \in [l]$ and $t_j \in set(Y(\widehat{s}_{i_j}))$ for all $j \in [k]$. Using

this and the construction of $\mathcal{N}'$, we get

$$\mathtt{g} \Rightarrow_{\mathcal{N}',\mathrm{IO}} s_0 \cdot (\mathtt{g}_1^{\langle 0\rangle}, \ldots, \mathtt{g}_l^{\langle 0\rangle}) \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s_0 \cdot (\widehat{s}_1, \ldots, \widehat{s}_l)$$

and

$$\begin{aligned}
t &= t_0 \cdot (t_1, \ldots, t_k) \\
&\in \{t_0 \cdot (t'_1, \ldots, t'_k) \mid t'_j \in set(Y(\widehat{s}_{i_j})) \text{ for } j \in [k]\} \\
&= set(t_0 \cdot (Y(\widehat{s}_{i_1}), \ldots, Y(\widehat{s}_{i_k}))) \\
&= set(Y(s_0)[\![Y(\widehat{s}_1), \ldots, Y(\widehat{s}_l)]\!]) \\
&= set(Y(s_0 \cdot (\widehat{s}_1, \ldots, \widehat{s}_l))) \qquad \text{(by Lemma 8.11)}
\end{aligned}$$

as required. Moreover, $Y(s_0 \cdot (\widehat{s}_1, \ldots, \widehat{s}_l)) = t_0 \cdot (Y(\widehat{s}_{i_1}), \ldots, Y(\widehat{s}_{i_k}))$ is in $\mathrm{T}_{\Pi \cup \{+,\theta\}}$, because $t_0 \in \mathrm{T}_{\Pi \cup \{\square_1,\ldots,\square_k\}}$.

Finally, we consider the case where $\Sigma_{\mathcal{N}}$ *does* contain parameters that are used by the YIELD mappings on the tree languages $L_{\mathtt{g}}$. Suppose they use parameters $y_1, \ldots, y_m$ (some of which are in $\Sigma_{\mathcal{N}}$) and that the parameters $x_1, \ldots, x_m$ are not in $\Sigma_{\mathcal{N}}$. For every $\mathtt{g} \in \mathcal{G}$, let $L'_{\mathtt{g}}$ be the tree language obtained from $L_{\mathtt{g}}$ by changing every $y_i^{\langle 0\rangle}$ into $x_i^{\langle 0\rangle}$. Since C is genuine, $L'_{\mathtt{g}}$ is in C. Now define $L''_{\mathtt{g}} = \{\mathtt{subst}_m[t, y_1^{\langle 0\rangle}, \ldots, y_m^{\langle 0\rangle}] \mid t \in L'_{\mathtt{g}}\}$. It should be clear that $Y(L''_{\mathtt{g}}) = Y(L_{\mathtt{g}}) = L(\gamma_{\mathtt{g}})$ and that $\Sigma_{\mathcal{N}}$ does not contain parameters that are used by the YIELD mappings on the tree languages $L''_{\mathtt{g}}$. Hence we are in the previous case and we can define $\mathcal{N}' = (\mathcal{G}', \Pi', (\gamma'_{\mathtt{h}})_{\mathtt{h}\in\mathcal{G}'}, \mathtt{g}_0)$ as above, with $L''_{\mathtt{g}}$ instead of $L_{\mathtt{g}}$. Note, however, that $L''_{\mathtt{g}}$ need not be in C. This can be corrected by defining the equivalent regular delegation network $\mathcal{N}'' = (\mathcal{G}'', \Pi', (\gamma''_{\mathtt{h}})_{\mathtt{h}\in\mathcal{G}''}, \mathtt{g}_0)$ with $\mathcal{G}'' = \mathcal{G}' \cup \{\overline{\mathtt{g}} \mid \mathtt{g} \in \mathcal{G}\}$ and for every $\mathtt{g} \in \mathcal{G}$, $L(\gamma''_{\mathtt{g}}) = \{\mathtt{subst}_m[\overline{\mathtt{g}}, y_1^{\langle 0\rangle}, \ldots, y_m^{\langle 0\rangle}]\}$, $L(\gamma''_{\overline{\mathtt{g}}}) = L'_{\mathtt{g}}$ and $L(\gamma''_{\mathtt{g}^{\langle 0\rangle}}) = L(\gamma'_{\mathtt{g}^{\langle 0\rangle}}) = \{\mathtt{subst}_2[+^{\langle 2\rangle}, \mathtt{g}, \mathtt{g}^{\langle 0\rangle}], \theta^{\langle 0\rangle}\}$. ∎

Let us have a look at an example that illustrates the construction in the preceding proof.

Example 8.13 The language L_{expseq} of Example 7.2 is in rDEL(DEL(FIN)) and hence in rDEL(YIELD(C)) with C = rDEL(FIN), cf. Example 8.9. Hence, by Lemma 8.12, it is in SET(YIELD(rDEL(C))) = SET(YIELD(REG)). Using the construction in the proof of Lemma 8.12, with some obvious simplifications, we obtain that $L_{\mathrm{expseq}} = set(Y(L(\gamma)))$ for the regular tree grammar γ with nonter-

minals $\mathtt{h}_0, \mathtt{g}_0^{\langle 0\rangle}, \mathtt{g}^{\langle 1\rangle}, \mathtt{h}^{\langle 0\rangle}, \mathtt{h}$, initial nonterminal $\mathtt{h}_0$, and the following rules:

$$\begin{aligned}
\mathtt{h}_0 &\to \mathtt{g}_0^{\langle 0\rangle}\\
\mathtt{g}_0^{\langle 0\rangle} &\to \mathtt{subst}_1[\mathtt{g}^{\langle 1\rangle}, \mathtt{h}^{\langle 0\rangle}]\\
\mathtt{g}^{\langle 1\rangle} &\to \mathtt{subst}_1[\mathtt{g}^{\langle 1\rangle}, \mathtt{subst}_2[\mathtt{f}^{\langle 2\rangle}, x_1^{\langle 0\rangle}, x_1^{\langle 0\rangle}]]\\
\mathtt{g}^{\langle 1\rangle} &\to \mathtt{subst}_2[\mathtt{f}^{\langle 2\rangle}, x_1^{\langle 0\rangle}, x_1^{\langle 0\rangle}]\\
\mathtt{h}^{\langle 0\rangle} &\to \mathtt{subst}_2[+^{\langle 2\rangle}, \mathtt{h}, \mathtt{h}^{\langle 0\rangle}]\\
\mathtt{h}^{\langle 0\rangle} &\to \theta^{\langle 0\rangle}\\
\mathtt{h} &\to \mathtt{subst}_2[\mathtt{k}^{\langle 2\rangle}, \mathtt{h}, \mathtt{h}]\\
\mathtt{h} &\to \mathtt{a}^{\langle 0\rangle}
\end{aligned}$$

Note that if the rules for $\mathtt{h}^{\langle 0\rangle}$ are replaced by the one rule $\mathtt{h}^{\langle 0\rangle} \to \mathtt{h}$, then $set(Y(L(\gamma))) = Y(L(\gamma))$ consists of all trees of the form $s \cdot (t_1, \ldots, t_{2^n})$ (as in Example 7.2) such that $t_1 = \cdots = t_{2^n}$. ■

We can soon prove the main theorem of this section, which shows how the classes DEL*(FIN) and TBY*(REG) are related. To be able to apply Lemma 8.8 in the proof of this result, we must convince ourselves that the operator DEL preserves closedness under linear nondeleting tree homomorphisms, i.e., we need the following small result.

Lemma 8.14 For every class C of tree languages, if C is closed under linear nondeleting tree homomorphisms, then so is DEL(C).[34]

Proof Let $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ be a delegation network, and let $td\colon \mathrm{T}_\Pi \to \mathrm{T}_{\Pi'}$ be a linear nondeleting tree homomorphism. Let q be the unique state of the td transducer td. For $m \in \mathbb{N}$, we extend td to a linear nondeleting tree homomorphism from $\mathrm{T}_{\mathcal{G}\cup\Pi,m}$ to $\mathrm{T}_{\mathcal{G}\cup\Pi',m}$ by adding the rules $q[\mathtt{g}[x_1, \ldots, x_k]] \to \mathtt{g}[q[x_1], \ldots, q[x_k]]$ and $q[x_i] \to x_i$ for every $\mathtt{g}^{(k)} \in \mathcal{G}$ and $i \in [m]$. Note that in the last rule, x_i is an input symbol. Now define $\mathcal{N}' = (\mathcal{G}, \Pi', (\gamma'_\mathtt{g})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ such that $L(\gamma'(\mathtt{g})) = td(L(\gamma(\mathtt{g}))$ for $\mathtt{g} \in \mathcal{G}$. It is straightforward to prove, by induction on the length of the derivations, that $\mathtt{g}_0 \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} t'$ if and only if there exists t such that $\mathtt{g}_0 \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t$ and $td(t) = t'$. The proof uses that $td(s \cdot s') = td(s) \cdot td(s')$ (with the additional rule $q[\Box] \to \Box$) and that $td(u(t_1, \ldots, t_n)) = td(u)(td(t_1), \ldots, td(t_n))$, which can be shown by induction on the structure of s and u. Hence $td(L_\mathrm{T}(\mathcal{N})) = L_\mathrm{T}(\mathcal{N}')$. ■

As mentioned, we now have all ingredients needed for the proof of the main theorem of this section.

Theorem 8.15 The class DEL*(FIN) is a proper subset of TBY*(REG).

[34] We note here, without proof, that a much stronger result holds: DEL(C) is closed under arbitrary tree homomorphisms, for every genuine class C.

Proof To show that $\mathrm{DEL}^*(\mathrm{FIN})$ is a subset of $\mathrm{TBY}^*(\mathrm{REG})$, we prove that, in fact, $\mathrm{DEL}^{n+1}(\mathrm{FIN}) \subseteq (\mathrm{YIELD} \circ \mathrm{SET})^n(\mathrm{YIELD}(\mathrm{REG}))$ for all $n \in \mathbb{N}$. Notice first that, for every genuine class C of tree languages closed under linear nondeleting tree homomorphisms, we have

$$\begin{aligned}
\mathrm{rDEL}(\mathrm{DEL}(C)) &\subseteq \mathrm{rDEL}(\mathrm{YIELD}(\mathrm{rDEL}(C))) \quad \text{(by Lemma 8.8)} \\
&\subseteq \mathrm{SET}(\mathrm{YIELD}(\mathrm{rDEL}^2(C))) \quad \text{(by Lemma 8.12)} \\
&= \mathrm{SET}(\mathrm{YIELD}(\mathrm{rDEL}(C))) \quad \text{(by Lemma 8.7).}
\end{aligned}$$

For $n \in \mathbb{N}$, $\mathrm{DEL}^n(\mathrm{FIN})$ is closed under linear nondeleting tree homomorphisms by Lemma 8.14. Hence, n-fold application of the inclusion above shows that

$$\begin{aligned}
\mathrm{rDEL}(\mathrm{DEL}^n(\mathrm{FIN})) &\subseteq (\mathrm{SET} \circ \mathrm{YIELD})^n(\mathrm{rDEL}(\mathrm{FIN})) \\
&= (\mathrm{SET} \circ \mathrm{YIELD})^n(\mathrm{REG}),
\end{aligned}$$

and, therefore,

$$\begin{aligned}
\mathrm{DEL}^{n+1}(\mathrm{FIN}) &\subseteq \mathrm{YIELD}(\mathrm{rDEL}(\mathrm{DEL}^n(\mathrm{FIN}))) \qquad \text{(by Lemma 8.8)} \\
&\subseteq \mathrm{YIELD}((\mathrm{SET} \circ \mathrm{YIELD})^n(\mathrm{REG})) \\
&= (\mathrm{YIELD} \circ \mathrm{SET})^n(\mathrm{YIELD}(\mathrm{REG})).
\end{aligned}$$

The properness of the inclusion $\mathrm{DEL}^*(\mathrm{FIN}) \subseteq \mathrm{TBY}^*(\mathrm{REG})$ is immediate from Example 8.5. ∎

Since the tree languages in the class $\mathrm{TBY}^*(\mathrm{REG})$ are decidable, so are those in $\mathrm{DEL}^*(\mathrm{FIN})$. In fact, the proof of Theorem 8.15 is effective, in the sense that for every hierarchical delegation network $\mathcal{N}$ one can construct a regular tree grammar γ and a finite sequence $T_1, T_2, \ldots, T_n$ where each T_i is either a top-down tree transducer td or a derived signature $\Delta(\Sigma, m)$, such that $L_{\mathrm{T}}(\mathcal{N}) = T_n(\cdots T_2(T_1(L(\gamma)))\cdots)$ where we also use T_i to denote either the top-down tree transduction td or the YIELD mapping $Y_{\Sigma,m}$. From this we obtain that the uniform membership problem is decidable for hierarchical delegation networks (see [EV85, Theorem 7.5]). We also obtain from [EV85, Theorem 7.4] and [DE98, Theorem 4.5] the following decidability result, where it should be noted (again) that we do not know whether $\mathrm{DEL}^*(\mathrm{FIN})$ is (effectively) closed under intersection with regular tree languages, whereas $\mathrm{TBY}^*(\mathrm{REG})$ is.

Corollary 8.16 It is decidable, for $L \in \mathrm{DEL}^*(\mathrm{FIN})$ and $R \in \mathrm{REG}$, whether $L \cap R = \emptyset$. It is also decidable whether $L \cap R$ is finite, and if the answer is yes, the elements of $L \cap R$ can be computed.[35]

In fact, the much stronger corollary with $\tau(L)$ instead of $L \cap R$ holds as well, where τ is a composition of macro tree transductions (i.e., a composition of top-down tree transductions and YIELD mappings).

[35]Here, L is given by a hierarchical delegation network and R by a regular tree grammar.

One consequence of Corollary 8.16 is that it is decidable whether a delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathtt{g}_0)$ is proper (see Proposition 7.3), if $\gamma_{\mathtt{g}}$ is a hierarchical delegation network for every $\mathtt{g} \in \mathcal{G}$ (and hence so is $\mathcal{N}$). We have to check that all generator symbols of $\mathcal{N}$ are useful, i.e., that $\mathrm{use}(\mathcal{N}) = \mathcal{G}$. Clearly, $\mathrm{use}(\mathcal{N})$ can be computed (as for context-free grammars), because, for $\mathtt{g}^{(k)} \in \mathcal{G}$ and $\mathcal{U} \subseteq \mathcal{G}$, it is decidable whether $L(\gamma_{\mathtt{g}}) \cap \mathrm{T}_{\mathcal{U} \cup \Pi, k} \neq \emptyset$. Note also that every delegation network $\mathcal{N}$ such that $L(\gamma_{\mathtt{g}}) \in \mathrm{TBY}^*(\mathrm{REG})$ for every $\mathtt{g} \in \mathcal{G}$, can effectively be transformed into an equivalent *proper* delegation network $\mathcal{N}'$ with the same property, where 'equivalent' means that $L_{\mathrm{T}}(\mathcal{N}) = L_{\mathrm{T}}(\mathcal{N}')$.

Let dTD denote the class of td transductions computed by deterministic td transducers. The class $(\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG})$ equals the class $\mathrm{dMTT}^*(\mathrm{REG})$, the closure of REG under deterministic macro tree transductions. Recall that $\mathrm{TBY}^*(\mathrm{REG}) = (\mathrm{TD} \cup \mathrm{YIELD})^*(\mathrm{REG}) = \mathrm{MTT}^*(\mathrm{REG})$. The class $\mathrm{YIELD}^*(\mathrm{REG})$ is the so-called IO-hierarchy, see [Mai74] and [ES78, Section 7].

The next theorem strengthens the properness of the inclusion of $\mathrm{DEL}^*(\mathrm{FIN})$ in $\mathrm{TBY}^*(\mathrm{REG})$.

Theorem 8.17 The class $\mathrm{DEL}^*(\mathrm{FIN})$ is incomparable with $\mathrm{YIELD}^*(\mathrm{REG})$ and with $(\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG})$.

Proof By Example 8.10, the tree language L_{monexp} of Example 8.5 is in the class $\mathrm{YIELD}(\mathrm{DEL}(\mathrm{FIN}))$. Thus, by Lemma 8.8, it is in $\mathrm{YIELD}(\mathrm{YIELD}(\mathrm{REG}))$, as is well known. Consequently, L_{monexp} is in $\mathrm{YIELD}^*(\mathrm{REG})$, but not in $\mathrm{DEL}^*(\mathrm{FIN})$.

It remains to present a tree language in $\mathrm{DEL}^*(\mathrm{FIN})$ that is not in $(\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG})$. For a tree t, let $\mathit{yield}(t)$ denote the string of constant symbols in t, read from left to right. Now, let $L'_{\mathrm{expseq}} \in \mathrm{DEL}^2(\mathrm{FIN})$ be defined in the same way as L_{expseq} in Example 7.2, except that $L(\gamma_{\mathtt{h}})$ is the regular tree language consisting of all trees t over the signature $\{\mathtt{f}^{(2)}, \mathtt{a}^{(0)}, \mathtt{b}^{(0)}, \mathtt{0}^{(0)}, \mathtt{1}^{(0)}\}$ such that $\mathit{yield}(t) = wbw'$ for some $w, w' \in \{\mathtt{a}, \mathtt{0}, \mathtt{1}\}^*$. Using arguments from [EM02], we shall show that L'_{expseq} is not an element of $(\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG})$. The proof of this fact is similar to (the second half of) the proof of [EM02, Theorem 25]. It reads as follows.

For a set A of symbols and a string language L, let $\mathit{rub}_A(L)$ denote the set of all strings of the form $w_0 a_1 w_1 \cdots a_n w_n$ with $w_0, \ldots, w_n \in A^*$ and $a_1 \cdots a_n \in L$. Intuitively, rub_A inserts "rubbish", arbitrary strings in A^*, between the symbols of strings in L. Clearly, $\mathit{yield}(L'_{\mathrm{expseq}}) = \mathit{rub}_{\{0,1,a\}}(L_{2^n}) = \mathit{rub}_{\{0,1\}}(\mathit{rub}_{\{a\}}(L_{2^n}))$, where $L_{2^n} = \{b^{2^n} \mid n \in \mathbb{N}\}$. Now, the "bridge theorem" [EM02, Theorem 20] tells us that $\mathit{yield}(L'_{\mathrm{expseq}}) \in \mathit{yield}((\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG}))$ implies that $\mathit{rub}_{\{a\}}(L_{2^n}) \in \mathit{yield}(\mathrm{dTD}(\mathrm{REG}))$. Furthermore, by [ERS80, Theorem 3.2.14], the latter implies $L_{2^n} \in \mathit{yield}(\mathrm{TD}_{\mathrm{fc}}(\mathrm{REG}))$, where $\mathrm{TD}_{\mathrm{fc}}$ denotes the set of all finite-copying top-down tree transductions. However, by [ERS80, Corollary 3.2.7], this could only be the case if the Parikh image of L_{2^n} was semilinear, which it is not. Hence, $\mathit{yield}(L'_{\mathrm{expseq}}) \notin \mathit{yield}((\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG}))$ and thus, as claimed, $L'_{\mathrm{expseq}} \notin$

$(\mathrm{dTD} \cup \mathrm{YIELD})^*(\mathrm{REG})$. ∎

In the proof above we have defined a language L'_{expseq} that is in $\mathrm{DEL}^2(\mathrm{FIN})$, but not in YIELD(REG) and hence not in DEL(FIN) (see the paragraph after Example 8.10). It is an open problem whether the delegation hierarchy $(\mathrm{DEL}^n(\mathrm{FIN}))_{n\in\mathbb{N}}$ is proper at each level, i.e., whether $\mathrm{DEL}^n(\mathrm{FIN})$ is properly included in $\mathrm{DEL}^{n+1}(\mathrm{FIN})$ for every $n \in \mathbb{N}$. Properness of the IO-hierarchy of (classes of) tree languages was shown in [Dam82]. The properness of the corresponding string IO-hierarchy (obtained by taking yields) was established in [EM02, Section 7] by means of a so-called bridge theorem.

As a last result, we will prove that the class $\mathrm{TBY}^*(\mathrm{REG})$ is closed under DEL. For that we need, in addition to Lemmas 8.8 and 8.12, a lemma that allows us to reorder the operators rDEL and TD.

Lemma 8.18 For all genuine classes C of tree languages, $\mathrm{rDEL}(\mathrm{TD}(C)) \subseteq \mathrm{TD}(\mathrm{rDEL}(C))$.

Proof Consider a regular delegation network $\mathcal{N} = (\mathcal{G}, \Pi, (\gamma_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}}, \mathtt{g}_0)$ such that $L(\gamma_{\mathtt{g}}) = td_{\mathtt{g}}(L_{\mathtt{g}})$ for all $\mathtt{g} \in \mathcal{G}$, where $td_{\mathtt{g}}$ is a td transducer and $L_{\mathtt{g}} \in C$. Since C is genuine, we may assume that $L_{\mathtt{g}} \subseteq \mathrm{T}_{\Sigma_{\mathtt{g}}}$ for pairwise disjoint signatures $\Sigma_{\mathtt{g}}$. Consequently, we may also assume that there is a single td transducer $td \subseteq \mathrm{T}_\Sigma \times \mathrm{T}_{\Sigma_\mathcal{N}}$, where $\Sigma = \bigcup_{\mathtt{g}\in\mathcal{G}} \Sigma_{\mathtt{g}}$, such that $L(\gamma_{\mathtt{g}}) = td(L_{\mathtt{g}})$ for all $\mathtt{g} \in \mathcal{G}$. Finally, it is easy to modify td in such a way that the right-hand sides of rules $q[\mathtt{f}[x_1, \ldots, x_k]] \to t$ with $k \geq 1$ do not contain generator symbols (i.e., symbols in $\mathcal{G}$). In other words, td outputs generator symbols only when it reaches a leaf of its input tree.

We have to construct a regular delegation network $\mathcal{N}' = (\mathcal{G}', \Pi', (\gamma'_{\mathtt{g}})_{\mathtt{g}\in\mathcal{G}'}, \mathtt{g}'_0)$ and a td transducer td', such that $td'(L_{\mathrm{T}}(\mathcal{N}')) = L_{\mathrm{T}}(\mathcal{N})$ and $L(\gamma'_{\mathtt{g}}) \in C$ for all $\mathtt{g} \in \mathcal{G}'$. To see how td' and $\mathcal{N}'$ can be obtained, consider a derivation $\mathtt{g}_0 \Rightarrow_{\mathcal{N},\mathrm{IO}} t_0 \cdot (\mathtt{g}_1, \ldots, \mathtt{g}_l) \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t_0 \cdot (t_1, \ldots, t_l)$, where $t_0 \cdot (\mathtt{g}_1, \ldots, \mathtt{g}_l) \in td(s)$ for a tree $s \in L_{\mathtt{g}_0}$. The td transducer td' will, intuitively, start by working in the same manner as td, on an input tree (generated by $\mathcal{N}'$) whose topmost part looks like s. However, when td would generate one of the generator symbols $\mathtt{g}_i$ from a leaf $\mathtt{a}$ of s, td' must be able to continue the computation on a subtree of which (recursively) the topmost part looks like a tree from $L_{\mathtt{g}_i}$. Clearly, when $\mathcal{N}'$ generates this input to td', it "knows" only $\mathtt{a}$, but not $\mathtt{g}_i$. Therefore, we simply let $\mathcal{N}'$ generate one subtree for each generator symbol in $\mathcal{G}$, i.e., $\mathtt{a}$ is turned into a symbol $\#_{\mathtt{a}}$ of rank $|\mathcal{G}|$, and we let td' select the subtree at the position corresponding to $\mathtt{g}_i$ when it processes $\#_{\mathtt{a}}$. Another detail to be taken into account is that td may be nonlinear, which means that a single occurrence of $\mathtt{a}$ in the input tree s may give rise to any number of generator symbols $\mathtt{g}_{i_1}, \ldots, \mathtt{g}_{i_m}$ in the output tree $td(s)$, with $i_j = i$ for all $j \in [m]$. To achieve the right effect in td', we have to make sure that it may work on different input trees from $L_{\mathtt{g}_i}$ in different places, even though the input subtrees it works on are necessarily copies of each other. We solve this problem by generating not only one tree from

$L_{\mathtt{g}_i}$ (modified as described above), but an arbitrary sequence of such trees, from which td' nondeterministically chooses one. This is similar to the implementation of nondeterminism in the proof of Lemma 8.12.

To give the formal construction, let $\mathcal{G} = \{\mathtt{h}_1, \ldots, \mathtt{h}_p\}$. We define the relevant components of $\mathcal{N}'$ as follows.

- $\mathcal{G}'$ contains all symbols $\mathtt{a}^{(0)} \in \Sigma$ and all $\mathtt{h}_i, \mathtt{h}_i^*$ for $i \in [p]$ (which are assumed to be pairwise distinct).
- Π' consists of the symbols $\mathtt{f}^{(k)} \in \Sigma$ with $k \geq 1$, the symbols $+^{(2)}$ and $\theta^{(0)}$, and all symbols $\#_{\mathtt{a}}^{(p)}$ such that $\mathtt{a}^{(0)} \in \Sigma$. Of course, even these symbols are assumed to be pairwise distinct.
- For every $\mathtt{g} \in \mathcal{G}$, $L(\gamma'_{\mathtt{g}}) = L_{\mathtt{g}}$ and $L(\gamma'_{\mathtt{g}^*}) = \{+[\mathtt{g}, \mathtt{g}^*], \theta\}$, and for every $\mathtt{a}^{(0)} \in \Sigma$, $L(\gamma'_{\mathtt{a}}) = \{\#_{\mathtt{a}}[\mathtt{h}_1^*, \ldots, \mathtt{h}_p^*]\}$.
- Finally, $\mathtt{g}'_0 = \mathtt{g}_0$.

Next, let us construct td', assuming that $td = (\Sigma, \Sigma_{\mathcal{N}}, Q, R, q_0)$. Clearly, the input and output signatures of td' are Π' and Π, respectively. Its set of states is $Q' = Q \cup \{q_{\mathrm{sel}}\}$, where $q_{\mathrm{sel}} \notin Q$, and q_0 is the initial state also in td'. The set R' contains all rules $q[\mathtt{f}[x_1, \ldots, x_k]] \to t$ in R such that $k \geq 1$. For every rule $q[\mathtt{a}] \to t_0 \cdot (\mathtt{h}_{i_1}, \ldots, \mathtt{h}_{i_l})$ in R (where t_0 does not contain symbols belonging to $\mathcal{G}$), R' contains the rule $q[\#_{\mathtt{a}}[x_1, \ldots, x_p]] \to t_0 \cdot (q_{\mathrm{sel}}[x_{i_1}], \ldots, q_{\mathrm{sel}}[x_{i_l}])$. Thus, these rules account for the choice of the "right" input subtrees to continue the computation with, i.e., using $L_{\mathtt{h}_{i_j}}$ for $\mathtt{h}_{i_j}$. Finally, R' also contains the rules $q_{\mathrm{sel}}[+[x_1, x_2]] \to q_0[x_1]$ and $q_{\mathrm{sel}}[+[x_1, x_2]] \to q_{\mathrm{sel}}[x_2]$. These account for the nondeterministic choice of a tree in $L_{\mathtt{h}_{i_j}}$.

For the correctness of the construction, we prove that for all $\mathtt{g} \in \mathcal{G}$ and $t \in \mathrm{T}_\Pi$, $\mathtt{g} \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t$ if and only if there exists $s \in \mathrm{T}_{\Pi'}$ such that $\mathtt{g} \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s$ and $t \in td'(s)$. As in the proof of Lemma 8.12, we proceed by induction on the length of the derivations, using Lemma 8.6.

('$\Leftarrow$') Consider a derivation $\mathtt{g} \Rightarrow^{n+1}_{\mathcal{N}',\mathrm{IO}} s$. Then $s = s_0 \cdot (s_1, \ldots, s_k)$ for some $s_0 \cdot (\mathtt{a}_1, \ldots, \mathtt{a}_k) \in L_{\mathtt{g}}$ where $\mathtt{a}_1^{(0)}, \ldots, \mathtt{a}_k^{(0)} \in \Sigma$ and s_0 does not contain any symbol $\mathtt{a}^{(0)} \in \Sigma$, and $\mathtt{a}_i \Rightarrow^{n_i}_{\mathcal{N}',\mathrm{IO}} s_i$ with $n_i \leq n$ for all $i \in [k]$. Moreover, $s_i = \#_{\mathtt{a}_i}[s_{i,1}, \ldots, s_{i,p}]$ with $\mathtt{h}_j^* \Rightarrow^{m_{i,j}}_{\mathcal{N}',\mathrm{IO}} s_{i,j}$ and $m_{i,j} \leq n_i$ for all $j \in [p]$. Now consider $t \in td'(s)$. It follows from the construction of td' that there is a tree of the form $t_0 \cdot (\mathtt{h}_{j_1}, \ldots, \mathtt{h}_{j_l})$ in $td(s_0 \cdot (\mathtt{a}_1, \ldots, \mathtt{a}_k))$ such that t_0 does not contain symbols from $\mathcal{G}$ and $t = t_0 \cdot (t_1, \ldots, t_l)$ for some $t_1, \ldots, t_l$; moreover, there exist $i_1, \ldots, i_l \in [k]$ such that $q_0[s] \Rightarrow^*_{td'} t_0 \cdot (q_{\mathrm{sel}}[s_{i_1,j_1}], \ldots, q_{\mathrm{sel}}[s_{i_l,j_l}])$ and $q_{\mathrm{sel}}[s_{i_r,j_r}] \Rightarrow^*_{td'} t_r$ for all $r \in [l]$. Hence, by the rules for q_{sel} and the definition of $L(\gamma'_{\mathtt{h}^*_{j_r}})$, there exists $\overline{s}_r$ such that $t_r \in td'(\overline{s}_r)$ and $\mathtt{h}_{j_r} \Rightarrow^{\overline{m}_r}_{\mathcal{N}',\mathrm{IO}} \overline{s}_r$ with $\overline{m}_r \leq m_{i_r,j_r}$. Thus, by induction, $\mathtt{h}_{j_r} \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t_r$, and so $\mathtt{g} \Rightarrow^*_{\mathcal{N},\mathrm{IO}} t$ because $t_0 \cdot (\mathtt{h}_{j_1}, \ldots, \mathtt{h}_{j_l})$ is in $td(L_{\mathtt{g}}) = L(\gamma_{\mathtt{g}})$.

('$\Rightarrow$') Now consider a derivation $\mathtt{g} \Rightarrow^{n+1}_{\mathcal{N},\mathrm{IO}} t$. Then $t = t_0 \cdot (t_1, \ldots, t_l)$ for some $t_0 \cdot (\mathtt{h}_{j_1}, \ldots, \mathtt{h}_{j_l}) \in td(L_{\mathtt{g}})$ where t_0 does not contain symbols from $\mathcal{G}$ and $\mathtt{h}_{j_r} \Rightarrow^{n_r}_{\mathcal{N},\mathrm{IO}} t_r$ with $n_r \leq n$, for all $r \in [l]$. By induction, there exist $s_1, \ldots, s_l$

such that $\mathtt{h}_{j_r} \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s_r$ and $t_r \in td'(s_r)$. This gives derivations $\mathtt{h}^*_j \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} \widehat{s}_j$ for $j \in [p]$ such that $q_{\mathrm{sel}}[\widehat{s}_{j_r}] \Rightarrow^*_{td'} t_r$ for every $r \in [l]$. Now let $s_0 \cdot (\mathtt{a}_1, \ldots, \mathtt{a}_k) \in L_{\mathtt{g}}$ be as above such that $t_0 \cdot (\mathtt{h}_{j_1}, \ldots, \mathtt{h}_{j_l}) \in td(s_0 \cdot (\mathtt{a}_1, \ldots, \mathtt{a}_k))$, and let $s = s_0 \cdot (\#_{\mathtt{a}_1}[\widehat{s}_1, \ldots, \widehat{s}_p], \ldots, \#_{\mathtt{a}_k}[\widehat{s}_1, \ldots, \widehat{s}_p])$. Then $\mathtt{g} \Rightarrow_{\mathcal{N}',\mathrm{IO}} s_0 \cdot (\mathtt{a}_1, \ldots, \mathtt{a}_k) \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s_0 \cdot (\#_{\mathtt{a}_1}[\mathtt{h}^*_1, \ldots, \mathtt{h}^*_p], \ldots, \#_{\mathtt{a}_k}[\mathtt{h}^*_1, \ldots, \mathtt{h}^*_p]) \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s$ and so $\mathtt{g} \Rightarrow^*_{\mathcal{N}',\mathrm{IO}} s$. Moreover, $q_0[s] \Rightarrow^*_{td'} t_0 \cdot (q_{\mathrm{sel}}[\widehat{s}_{j_1}], \ldots, q_{\mathrm{sel}}[\widehat{s}_{j_l}]) \Rightarrow^*_{td'} t_0 \cdot (t_1, \ldots, t_l) = t$ and so $t \in td'(s)$.

This proves that $L_{\mathrm{T}}(\mathcal{N}) = td'(L_{\mathrm{T}}(\mathcal{N}'))$. ∎

Theorem 8.19 The class $\mathrm{TBY}^*(\mathrm{REG})$ is closed under DEL.

Proof By Lemma 8.8 and the fact that $\mathrm{TBY}^*(\mathrm{REG})$ is, by definition, closed under tree homomorphisms and YIELD, it suffices to show that it is closed under rDEL. We do this by proving that, in fact, even $\mathrm{TBY}^n(\mathrm{REG})$ is closed under rDEL, for $n \in \mathbb{N}$. For $n = 0$, this is just the fact that REG is closed under rDEL by Lemma 8.7 (for $C = \mathrm{FIN}$). It implies, by Lemma 8.18, that $\mathrm{TD}^*(\mathrm{REG})$ is closed under rDEL. For the inductive step, let $C = \mathrm{TD}^*(\mathrm{TBY}^n(\mathrm{REG}))$ (which equals $\mathrm{TBY}^n(\mathrm{REG})$ for $n > 0$) be closed under rDEL. Then we get

$$\begin{aligned}
&\mathrm{rDEL}(\mathrm{TBY}^{n+1}(\mathrm{REG})) \\
&= \mathrm{rDEL}(\mathrm{TD}^*(\mathrm{YIELD}(C))) \\
&\subseteq \mathrm{TD}^*(\mathrm{rDEL}(\mathrm{YIELD}(C))) && \text{(by Lemma 8.18)} \\
&\subseteq \mathrm{TD}^*(\mathrm{SET}(\mathrm{YIELD}(\mathrm{rDEL}(C)))) && \text{(by Lemma 8.12)} \\
&\subseteq \mathrm{TD}^*(\mathrm{YIELD}(C)) \\
&= \mathrm{TBY}^{n+1}(\mathrm{REG}),
\end{aligned}$$

as claimed. ∎

By the above proof, $\mathrm{TBY}^n(\mathrm{REG})$ is closed under rDEL, for every $n \in \mathbb{N}$. Also, $\mathrm{TBY}^n(\mathrm{REG})$ is closed under linear tree homomorphisms – by definition for $n > 0$, and by [Tha73, Theorem 10] for $n = 0$. Hence Lemma 8.8 implies the following more precise result: $\mathrm{DEL}(\mathrm{TBY}^n(\mathrm{REG})) \subseteq \mathrm{TBY}^{n+1}(\mathrm{REG})$ for every $n \in \mathbb{N}$.

9 Conclusion

Several directions of research remain.

In view of the MW-like result of Theorem 5.6 it would be natural to generalize delegation networks in such a way that the tree generators $\gamma_{\mathtt{g}}$ are turned into jungle generators, i.e., $L(\gamma_{\mathtt{g}}) \subseteq \mathrm{J}_{\Sigma_{\mathcal{N}},k}$ if $\mathtt{g}$ has rank k. In Definition 2.4, the first case of the definition of $\mathrm{iterate}_{\mathcal{N}}(\sigma)(\mathtt{f})$ would thus evaluate jungles rather than trees. By Lemma 5.3 this does indeed yield a generalization of delegation networks. We guess that, with the definition of $L_{\mathrm{J}}(\mathcal{N})$ adapted in the obvious way, Theorem 5.6 still holds. We also guess that this generalization does not really extend the generating power of delegation networks. In particular, we

think that for every CFJG G of which the initial nonterminal has rank 0, and for every Π-interpretation $(\mathbb{A}, \pi)$, there exists a finitary delegation network $\mathcal{N} = (\mathcal{G}, \Pi, \Gamma, \mathsf{g}_0, \mathbb{A}, \pi)$ such that $L(\mathcal{N}) = \pi(L(G))$.

A question left open by Theorem 5.6 is whether $L_{\mathrm{J}}(\mathcal{N}) = L(\mathcal{N}')$ for a delegation network $\mathcal{N}'$ obtained from $\mathcal{N}$ by changing its interpretation $(\mathbb{A}, \pi)$ into some $(\mathbb{A}', \pi')$, similar to the fact that $L_{\mathrm{T}}(\mathcal{N}) = L(\mathcal{N}_{\mathrm{free}})$ (see Section 6) and similar to the result of [MW67]. We guess that such an interpretation $(\mathbb{A}', \pi')$ does not exist. Maybe a more general notion of interpretation can be used to obtain a similar result.

Let us repeat some open questions mentioned before. Is $\mathrm{DEL}^n(\mathrm{FIN})$ a proper subset of $\mathrm{DEL}^{n+1}(\mathrm{FIN})$ for every $n \in \mathbb{N}$? Are $\mathrm{DEL}^n(\mathrm{FIN})$ and $\mathrm{DEL}^*(\mathrm{FIN})$ closed under intersection with regular tree languages? More generally, what are the closure properties of $\mathrm{DEL}^*(\mathrm{FIN})$?

In addition to studying these open theoretical questions, an implementation of (many-sorted) delegation networks should be made. Such an implementation may combine the ideas of the system Treebag [DK00, DK01] (see also `http://www.cs.umu.se/~drewes/treebag`), which implements tree-based generation, with the evaluation strategy sketched in the Introduction and after Theorem 3.5. In this way, a very general core system would be obtained, to which implementations of any desired classes of tree generators and interpretations may be added. To increase efficiency, such a system should be able to work in a parallel and perhaps distributed manner. While the implementation of the theoretical concept as such is rather straightforward, there are two major practical challenges. The first is to handle nondeterminism in an appropriate way, including cyclic delegation which may easily lead to infinite recursion. The second challenge is to design and implement a suitable user interface that makes it possible to interact with the delegation network and its individual generators in a convenient way. Such an interface should be based on an abstract interaction model which can easily be instantiated for various combinations of tree generators and interpretations. Again, the ideas realized in Treebag can be used as a starting point, but a much more flexible model must be developed.

References

[AG68] Michael A. Arbib and Yehoshafat Give'on. Algebra automata I: Parallel programming as a prolegomena to the categorical approach. *Information and Control*, 12:331–345, 1968.

[Asv78] Peter R.J. Asveld. Iterated context-independent rewriting – an algebraic approach to families of languages. PhD thesis, Univ. of Twente, 1978.

[BC87] Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewriting. *Mathematical Systems Theory*, 20:83–127, 1987.

[CDG+02] Hubert Comon, Max Dauchet, Rémi Gilleron, Florent Jacquemard, Denis Lugiez, Sophie Tison, and Marc Tommasi. *Tree Automata Techniques and Applications*, 2002. Internet publication available at `http://www.grappa.univ-lille3.fr/tata`.

[CE12] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic – A Language-Theoretic Approach.* Cambridge University Press, 2012.

[CR93] Andrea Corradini and Francesca Rossi. Hyperedge replacement jungle rewriting for term-rewriting systems and logic programming. *Theoretical Computer Science*, 109:7–48, 1993.

[Dam82] Werner Damm. The IO- and OI-hierarchies. *Theoretical Computer Science*, 20:95–208, 1982.

[DE98] Frank Drewes and Joost Engelfriet. Decidability of the finiteness of ranges of tree transductions. *Information and Computation*, 145:1–50, 1998.

[DHK97] Frank Drewes, Annegret Habel, and Hans-Jörg Kreowski. Hyperedge replacement graph grammars. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Vol. 1: Foundations*, chapter 2, pages 95–162. World Scientific, Singapore, 1997.

[DK00] Frank Drewes and Peter Knirsch. Treebag – a short presentation. In M. Nagl and A. Schürr, editors, *Proceedings of the Applications of Graph Transformation with Industrial Relevance (AGTIVE'99)*, volume 1779 of Lecture Notes in Computer Science, pages 411–417. Springer, 2000.

[DK01] Frank Drewes and Renate Klempien-Hinrichs. Treebag. In S. Yu and A. Păun, editors, *Proceedings of the 5th International Conference on Implementation and Application of Automata (CIAA'00)*, volume 2088 of Lecture Notes in Computer Science, pages 329–330. Springer, 2001.

[Dre00] Frank Drewes. Tree-based picture generation. *Theoretical Computer Science*, 246:1–51, 2000.

[Dre01] Frank Drewes. Tree-based generation of languages of fractals. *Theoretical Computer Science*, 262:377–414, 2001.

[Dre06] Frank Drewes. *Grammatical Picture Generation – A Tree-Based Approach.* Texts in Theoretical Computer Science. An EATCS Series. Springer, 2006.

[Dre07a] Frank Drewes. Delegation networks. Report UMINF 07.04, Umeå University, 2007.

[Dre07b] Frank Drewes. From tree-based generation to delegation networks. In S. Bozapalidis and G. Rahonis, editors, *Proc. 2nd International Conference on Algebraic Informatics (CAI'07)*, volume 4728 of Lecture Notes in Computer Science, pages 48–72. Springer, 2007.

[EH92] Joost Engelfriet and Linda Heyker. Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29:161–210, 1992.

[EM02] Joost Engelfriet and Sebastian Maneth. Output string languages of compositions of macro tree transducers. *Journal of Computer and System Sciences*, 64:350–395, 2002.

[Eng97] Joost Engelfriet. Context-free graph grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages.* Vol. 3: *Beyond Words*, chapter 3, pages 125–213. Springer, 1997.

[ERS80] Joost Engelfriet, Grzegorz Rozenberg, and Giora Slutzki. Tree transducers, L systems, and two-way machines. *Journal of Computer and System Sciences*, 20:150–202, 1980.

[ES77] Joost Engelfriet and Erik Meineche Schmidt. IO and OI. I. *Journal of Computer and System Sciences*, 15:328–353, 1977.

[ES78] Joost Engelfriet and Erik Meineche Schmidt. IO and OI. II. *Journal of Computer and System Sciences*, 16:67–99, 1978.

[ES79] Joost Engelfriet and Giora Slutzki. Bounded nesting in macro grammars. *Information and Control*, 42:157–193, 1979.

[EV85] Joost Engelfriet and Heiko Vogler. Macro tree transducers. *Journal of Computer and System Sciences*, 31:71–146, 1985.

[EV94] Joost Engelfriet and Heiko Vogler. The translation power of top-down tree-to-graph transducers. *Journal of Computer and System Sciences*, 49:258–305, 1994.

[FV98] Zoltán Fülöp and Heiko Vogler. *Syntax-Directed Semantics: Formal Models Based on Tree Transducers.* Springer, 1998.

[FV09] Zoltán Fülöp and Heiko Vogler. Weighted tree automata and tree transducers. In Werner Kuich, Manfred Droste, and Heiko Vogler, editors, *Handbook of Weighted Automata*, chapter 9, pages 313–403. Springer, 2009.

[GS84] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[GS97] Ferenc Gécseg and Magnus Steinby. Tree languages. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages.* Vol. 3: *Beyond Words*, chapter 1, pages 1–68. Springer, 1997.

[Hab92] Annegret Habel. *Hyperedge Replacement: Grammars and Languages*, volume 643 of *Lecture Notes in Computer Science*. Springer, 1992.

[HK87] Annegret Habel and Hans-Jörg Kreowski. May we introduce to you: Hyperedge replacement. In *Proceedings of the Third Intl. Workshop on Graph Grammars and Their Application to Computer Science*, volume 291 of Lecture Notes in Computer Science, pages 15–26. Springer, 1987.

[HKP91] Annegret Habel, Hans-Jörg Kreowski, and D. Plump. Jungle evaluation. *Fundamenta Informaticae*, 15:37–60, 1991.

[HP91] B. Hoffmann and D. Plump. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 1991.

[KM06] Stephan Kepser and Uwe Mönnich. Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science*, 354:82–97, 2006.

[KR11] Stephan Kepser and Jim Rogers. The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. *Journal of Logic, Language, and Information*, 20:361–384, 2011.

[Lee74] Jan van Leeuwen. A generalisation of Parikh's theorem in formal language theory. In J. Loeckx, editor, *Proc. 2nd Intl. Colloquium on Automata, Languages, and Programming (ICALP 1974)*, volume 14 of Lecture Notes in Computer Science, pages 17–26. Springer, 1974.

[LMSS12] Markus Lohrey, Sebastian Maneth, and Manfred Schmidt-Schausz. Parameter reduction and automata evaluation for grammar-compressed trees. *Journal of Computer and System Sciences*, 78:1651–1669, 2012.

[Mai74] Tom S. E. Maibaum. A generalized approach to formal languages. *Journal of Computer and System Sciences*, 8:409–502, 1974.

[ME12] Andreas Maletti and Joost Engelfriet. Strong lexicalization of tree adjoining grammars. In *Proc. 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 506–515, 2012. Available from `http://www.aclweb.org/anthology/P12-1053`.

[MW67] Jorge Mezei and Jesse B. Wright. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.

[NP92] Maurice Nivat and Andreas Podelski, editors. *Tree Automata and Languages*. Elsevier, Amsterdam, 1992.

[Plu99] Detlef Plump. Term graph rewriting. In H. Ehrig, G. Engels, H.-J. Kreowski, and G. Rozenberg, editors, *Handbook of Graph Grammars and Computing by Graph Transformation,* Vol. II: *Applications, Languages, and Tools*, chapter 1, pages 3–61. World Scientific, Singapore, 1999. Available also from `http://www.termgraph.org.uk`.

[Sch11] Daniel Schwencke. A category theoretic view of nondeterministic recursive program schemes. In Marc Bezem, editor, *Proc. 20th Conf. on Computer Science Logic (CSL 2011)*, pages 496–511, 2011. Available from http://drops.dagstuhl.de/opus/volltexte/2011/3252/.

[SO07] Heiko Stamer and Friedrich Otto. Restarting tree automata and linear context-free tree languages. In Symeon Bozapalidis and George Rahonis, editors, *Proc. 2nd International Conference on Algebraic Informatics (CAI'07)*, volume 4728 of *Lecture Notes in Computer Science*, pages 275–289, 2007.

[Tar55] Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5:285–309, 1955.

[Tha73] James W. Thatcher. Tree automata: an informal survey. In A.V. Aho, editor, *Currents in the Theory of Computing*, pages 143–172. Prentice Hall, Englewood Cliffs, NJ, 1973.