The Tree-Generative Capacity of Combinatory Categorial Grammars

Marco Kuhlmann
Dept. of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden
marco.kuhlmann@liu.se

Andreas Maletti
Institute for Computer Science, Universität Leipzig, P.O. box 100 920, D-04009 Leipzig, Germany
maletti@informatik.uni-leipzig.de

Lena Katharina Schiffer
Institute for Computer Science, Universität Leipzig, P.O. box 100 920, D-04009 Leipzig, Germany
schiffer@informatik.uni-leipzig.de

Abstract

The generative capacity of combinatory categorial grammars as acceptors of tree languages is investigated. It is demonstrated that the such obtained tree languages can also be generated by simple monadic context-free tree grammars. However, the subclass of pure combinatory categorial grammars cannot even accept all regular tree languages. Additionally, the tree languages accepted by combinatory categorial grammars with limited rule degrees are characterized: If only application rules are allowed, then they can accept only a proper subset of the regular tree languages, whereas they can accept exactly the regular tree languages once first degree composition rules are permitted.

2012 ACM Subject Classification Theory of computation; Theory of computation → Formal languages and automata theory; Theory of computation → Tree languages; Theory of computation → Grammars and context-free languages; Computing methodologies; Computing methodologies → Artificial intelligence; Computing methodologies → Natural language processing

Keywords and phrases Combinatory Categorial Grammar, Regular Tree Language, Linear Context-free Tree Language

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2019.44

Funding *Marco Kuhlmann*: Supported by the Centre for Industrial IT (CENIIT), grant 15.02.
Lena Katharina Schiffer: Supported by the German Research Foundation (DFG) Research Training Group GRK 1763 "Quantitative Logics and Automata".

1 Introduction

Categorial grammars [5] were introduced alongside the phrase-structure grammars (regular, context-free, context-sensitive grammars, etc.) of the Chomsky hierarchy [6] inspired by classical notions from proof theory [1, 3]. Combinatory Categorial Grammar (CCG) [23, 24] is an extension following the approach of combinatory logic [22, 7]. CCG received considerable attention in theoretical computer science culminating in the proofs of its mild context-sensitivity, which in particular, requires efficient parsing [27], as well as its equivalence to several other established grammar formalisms [28]. It has since become a widely applied formalism in computational linguistics [18, 17].

The basis for CCG is provided by a lexicon and a rule system. The lexicon assigns syntactic categories to the symbols of the input and the rule system describes how adjoining categories can be combined to eventually obtain a (binary) derivation tree. The mentioned equivalence result due to Vijay-Shanker and Weir [28] shows that CCG, Tree-Adjoining Grammar (TAG) [12] as well as linear indexed grammars [11] are equivalent in expressive power, which establishes that they generate the same string languages. However, the used construction depends on the ability to restrict the combination rules and to include entries for the empty word in the lexicon. Modern variants of CCG disfavor rule restrictions and the obtained *pure* CCG are strictly less expressive than TAG [14] unless unbounded generalized composition rules are permitted, in which case they are strictly more expressive than TAG [26]. Indeed, CCG with unbounded composition rules, rule restrictions as well as ε-entries in the lexicon are Turing-complete [15].

The mentioned studies examine the string (or weak) generative capacity of CCG, but already [26] asks for the tree (or strong) generative capacity or, more specifically, the expressiveness of the tree languages of CCG derivation trees [10]. Koller and Kuhlmann [13] show that CCG and TAG generate incomparable classes of *dependency trees*. In this contribution, we answer the original question and characterize the tree languages accepted by CCGs, and relate them to the standard notions of regular [9, 10] and context-free tree languages [19, 20]. A tree language $\mathcal{F}$ is *accepted* by a CCG G if $\mathcal{F}$ is obtained as a relabeling of the derivation tree language of G. Our work therefore is similar in spirit to that of Tiede [25], who studied the strong generative capacity of Lambek-style categorial grammars [16].

In the variant of CCG we investigate, the rule system is finite and includes only application and composition operators (i.e. rules based on the **B**-combinator of combinatory logic [8]). In general, we allow rule restrictions that further constrain the categories the rules can be applied to. Notice that our results concern only binary trees since the derivation trees of CCGs are binary. Our main result is that the tree languages accepted by CCGs can also be generated by simple monadic context-free tree grammars (Theorem 20). For CCG without rule restrictions this inclusion is proper since not even all regular tree languages [10] are accepted by these CCGs (Theorem 22). In addition, we show that CCGs without composition operations, which are weakly equivalent to (ε-free) context-free grammars, generate a strict subclass of the regular tree languages that does not even include all local tree languages (Theorem 9). Finally, if we limit the permitted composition operators to first degree, then exactly the regular tree languages are accepted (Theorem 14).

2 Preliminaries

We denote the set of nonnegative integers by $\mathbb{N}$ and let $[k] = \{i \in \mathbb{N} \mid 1 \leq i \leq k\}$ for every $k \in \mathbb{N}$. The power-set (i.e. set of all subsets) of a set A is $\mathcal{P}(A) = \{A' \mid A' \subseteq A\}$, and $\mathcal{P}_+(A) = \mathcal{P}(A) \setminus \{\emptyset\}$ contains all nonempty subsets. As usual, an alphabet is a finite set of symbols. The monoid $(\Sigma^*, \cdot, \varepsilon)$ consists of all strings (i.e. sequences) over a set Σ together with concatenation $\cdot$ and the empty string ε. We often write concatenation by juxtaposition. The length of a string $w \in \Sigma^*$ (i.e. the number of components in the sequence) is denoted by $|w|$. Any set $\mathcal{L} \subseteq \Sigma^*$ is a language, and the languages form a monoid $(\mathcal{P}(\Sigma^*), \cdot, \{\varepsilon\})$ with concatenation lifted to languages by $\mathcal{L} \cdot \mathcal{L}' = \{w \cdot w' \mid w \in \mathcal{L},\, w' \in \mathcal{L}'\}$. Every mapping $f \colon \Sigma \to \Delta^*$ [respectively, $f \colon \Sigma \to \mathcal{P}(\Delta^*)$] extends uniquely to a monoid homomorphism $f' \colon \Sigma^* \to \Delta^*$ [respectively, $f' \colon \Sigma^* \to \mathcal{P}(\Delta^*)$]. We will not distinguish the mapping f and its induced homomorphism f', but rather use f for both.

Given two sets A and A', a relation from A to A' is a subset $\rho \subseteq A \times A'$. The inverse of ρ is $\rho^{-1} = \{(a', a) \mid (a, a') \in \rho\}$, and for every $B \subseteq A$, we let $\rho(B) = \{a' \mid \exists b \in B \colon (b, a') \in \rho\}$. The relation $\rho \subseteq A \times A'$ can also be understood as a mapping $\widehat{\rho} \colon A \to \mathcal{P}(A')$ with $\widehat{\rho}(a) = \rho(\{a\})$ for all $a \in A$. We will not distinguish these two representations.

We build binary trees over the set Σ_2 of binary internal symbols, the alphabet Σ_1 of unary internal symbols, and the alphabet Σ_0 of leaf symbols.[1] Formally, the set $T_{\Sigma_2,\Sigma_1}(\Sigma_0)$ of binary (Σ_2, Σ_1)-trees indexed by Σ_0 is the smallest set T such that (i) $a \in T$ for all $a \in \Sigma_0$,

[1] We explicitly allow an infinite set of internal binary symbols.

(ii) $n(t) \in T$ for all $n \in \Sigma_1$ and $t \in T$, and (iii) $c(t_1, t_2) \in T$ for all $c \in \Sigma_2$ and $t_1, t_2 \in T$. We use graphical representations of trees to increase the readability. Every subset $\mathcal{F} \subseteq T_{\Sigma_2,\Sigma_1}(\Sigma_0)$ is a tree language. The mapping $\mathrm{pos}\colon T_{\Sigma_2,\Sigma_1}(\Sigma_0) \to \mathcal{P}_+(\{1,2\}^*)$ assigning positions to a tree is defined by (i) $\mathrm{pos}(a) = \{\varepsilon\}$ for all $a \in \Sigma_0$, (ii) $\mathrm{pos}(n(t)) = \{\varepsilon\} \cup \{1 \cdot w \mid w \in \mathrm{pos}(t)\}$ for all $n \in \Sigma_1$ and $t \in T_{\Sigma_2,\Sigma_1}(\Sigma_0)$, and (iii) for all $c \in \Sigma_2$ and $t_1, t_2 \in T_{\Sigma_2,\Sigma_1}(\Sigma_0)$,

$$\mathrm{pos}\big(c(t_1, t_2)\big) = \big\{\varepsilon\big\} \cup \big\{1 \cdot w \mid w \in \mathrm{pos}(t_1)\big\} \cup \big\{2 \cdot w \mid w \in \mathrm{pos}(t_2)\big\} \enspace .$$

We let $\mathrm{leaves}(t) = \{w \in \mathrm{pos}(t) \mid w \cdot 1 \notin \mathrm{pos}(t)\}$ be the set of leaf positions in t, and $\mathrm{ht}(t) = \max_{w \in \mathrm{leaves}(t)} |w|$ be the height of the tree t. The subtree of t at position $w \in \mathrm{pos}(t)$ is denoted by $t|_w$, and the label of t at position w is denoted by $t(w)$. Moreover, $t[t']_w$ denotes the tree obtained from t by replacing the subtree at position w by the tree $t' \in T_{\Sigma_2,\Sigma_1}(\Sigma_0)$. Given $\Delta \subseteq \Sigma_2 \cup \Sigma_1 \cup \Sigma_0$, let $\mathrm{pos}_\Delta(t) = \{w \in \mathrm{pos}(t) \mid t(w) \in \Delta\}$. We simply write $\mathrm{pos}_\delta(t)$ instead of $\mathrm{pos}_{\{\delta\}}(t)$.

We reserve the use of the symbol $\square$. The set $C_{\Sigma_2,\Sigma_1}(\Sigma_0)$ of *contexts* contains all trees of $T_{\Sigma_2,\Sigma_1}(\Sigma_0 \cup \{\square\})$, in which the special symbol $\square$ occurs exactly once. Let $C \in C_{\Sigma_2,\Sigma_1}(\Sigma_0)$. Since $\mathrm{pos}_\square(C)$ contains one element, we often identify it with its only element. To save space, we write tC for $C[t]_w$, where $w = \mathrm{pos}_\square(C)$.[2]

A *relabeling* is a mapping $\rho\colon (\Sigma_2 \cup \Sigma_1 \cup \Sigma_0) \to \mathcal{P}_+(\Delta)$ for some alphabet Δ.[3] It induces a mapping $\widehat{\rho}\colon T_{\Sigma_2,\Sigma_1}(\Sigma_0) \to \mathcal{P}_+(T_{\Delta,\Delta}(\Delta))$ for every $t \in T_{\Sigma_2,\Sigma_1}(\Sigma_0)$ by

$$\widehat{\rho}(t) = \big\{u \in T_{\Delta,\Delta}(\Delta) \mid \mathrm{pos}(u) = \mathrm{pos}(t),\, \forall w \in \mathrm{pos}(u)\colon u(w) \in \rho\big(t(w)\big)\big\} \enspace .$$

In the following, we again do not distinguish between the relabeling ρ and its induced mapping $\widehat{\rho}$ on trees. A *simple (monadic) context-free tree grammar* [19, 20] (sCFTG) is a system $G = (N, \Sigma, I, P)$ such that (i) $N = N_1 \cup N_0$, where N_1 and N_0 are alphabets of unary and nullary *nonterminals*, respectively, (ii) $\Sigma = \Sigma_2 \cup \Sigma_0$, where Σ_2 and Σ_0 are alphabets of internal and leaf *terminal symbols*, respectively, such that $N \cap \Sigma = \emptyset$, (iii) $I \subseteq N_0$ are nullary *start nonterminals*, and (iv) P is a finite set of *productions* such that

$$P \subseteq \big(N_0 \times T_{\Sigma_2,N_1}(\Sigma_0 \cup N_0)\big) \cup \big(N_1 \times C_{\Sigma_2,N_1}(\Sigma_0 \cup N_0)\big) \enspace .$$

The grammar is called monadic, because there are only nullary and unary nonterminals; simple means that the rules are linear and nondeleting, so all subtrees of a nonterminal on the left side of a rule have to appear exactly once on the right side. If $N_1 = \emptyset$, then G is a *regular tree grammar* (RTG). We write productions (n, r) as $n \to r$. Next, we define the rewrite semantics [2] for the sCFTG G. For arbitrary $\xi, \zeta \in T_{\Sigma_2,N_1}(\Sigma_0 \cup N_0)$ and positions $w \in \mathrm{pos}(\xi)$ we let $\xi \Rightarrow_{G,w} \zeta$ if there exists a production $n \to r \in P$ such that

- $\xi|_w = n$ and $\zeta = \xi[r]_w$ with $n \in N_0$, or
- $\xi|_w = n(\xi')$ and $\zeta = \xi[\xi' r]_w$ with $n \in N_1$ and $\xi' \in T_{\Sigma_2,N_1}(\Sigma_0 \cup N_0)$.

We write $\xi \Rightarrow_G \zeta$ if there exists $w \in \mathrm{pos}(\xi)$ such that $\xi \Rightarrow_{G,w} \zeta$. The tree language $\mathcal{F}(G)$ generated by G is $\mathcal{F}(G) = \{t \in T_{\Sigma_2,\emptyset}(\Sigma_0) \mid \exists n_0 \in I\colon n_0 \Rightarrow_G^+ t\}$, where $\Rightarrow_G^+$ is the transitive closure of $\Rightarrow_G$. The tree languages generated by sCFTGs are *context-free*,[4] and a tree language $\mathcal{F}$ is *regular* if and only if there exists an RTG G such that $\mathcal{F} = \mathcal{F}(G)$. A detailed introduction to trees and tree languages can be found in [10].

[2] This order tC is beneficial for arguments C (see Section 3).
[3] We require that each input symbol can be relabeled.
[4] Note that this is not an equivalence. There are context-free tree languages that are not generated by any sCFTG.

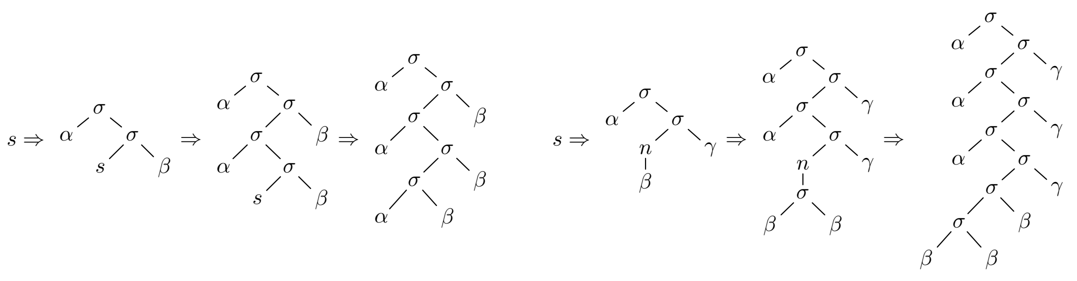

Figure 1 Derivations using the RTG G_1 (left) and the sCFTG G_2 (right) of Examples 1 and 2, respectively.

▶ **Example 1.** The regular tree grammar $G_1 = (N, \Sigma, I, P)$ with $N = N_0 = I = \{s\}$, $\Sigma_2 = \{\sigma\}$, $\Sigma_0 = \{\alpha, \beta\}$, and $P = \{s \to \sigma(\alpha, \sigma(s, \beta)),\ s \to \sigma(\alpha, \beta)\}$ generates the leaf language $\{\alpha^n \beta^n \mid n \geq 1\}$. Note that because it is an RTG, all nonterminals are nullary and thus leaves, which is similar to the property of right-linearity that can be encountered in CFGs.

Two important facts concerning the regular tree languages are that they properly include the derivation tree languages of CFGs and that their leaf languages are exactly the context-free languages.

▶ **Example 2.** The sCFTG $G_2 = (N, \Sigma, I, P)$ with $N = N_0 = I = \{s\}$, $N_2 = \{n\}$, $\Sigma_2 = \{\sigma\}$, $\Sigma_0 = \{\alpha, \beta, \gamma\}$ and

$$P = \big\{s \to \sigma(\alpha, \sigma(\beta, \gamma)),\ s \to \sigma(\alpha, \sigma(n(\beta), \gamma)),$$
$$n \to \sigma(\alpha, \sigma(n(\sigma(\square, \beta)), \gamma)),\ n \to \sigma(\alpha, \sigma(\sigma(\square, \beta), \gamma))\big\}$$

generates the leaf language $\{\alpha^n \beta^n \gamma^n \mid n \geq 1\}$. Since G_2 is simple, the placeholder $\square$, which indicates the new position of the subtree under the unary nonterminal symbol n, appears exactly once on the right side of the respective rules.

3 Combinatory Categorial Grammars

Combinatory categorial grammars (CCGs) extend the classical categorial grammars of Ajdukiewicz and Bar-Hillel [4] by rules inspired by combinatory logic [8]. Here, as in most of the formal work on CCGs, we restrict our attention to the rules of composition, which are based on the **B**-combinator.

Let A be an alphabet, and let $\mathcal{C}(A) = T_{S,\emptyset}(A)$, where $S = \{/, \backslash\}$ is the set of slashes. The elements of $\mathcal{C}(A)$ are called *categories* (over A), of which the elements of $A \subseteq \mathcal{C}(A)$ are *atomic*. We write categories using infix notation, omitting unnecessary parentheses based on the convention that slashes are left-associative. Thus every category takes the form $c = a|_1 c_1 \cdots |_k c_k$ where $a \in A$, $|_i \in S$, and $c_i \in \mathcal{C}(A)$, for all $i \in [k]$. The atomic category a is called the *target* of c and the slash–argument pairs $|_i c_i$ are called the *arguments* of c. If $c_i \in A$ for all $i \in [k]$, we call c a *first-order category*. The set of all first-order categories over A is denoted by $\mathcal{C}_f(A)$. The number k is called the *arity* of c. Note that, from the tree perspective, the sequence of arguments is a context $\alpha = \square|_1 c_1 \cdots |_k c_k$. The number k is the *length* of α; we write it as $|\alpha|$. We let $\mathcal{A}(A) \subseteq C_{S,\emptyset}(A)$ be the set of all argument contexts (over A). Finally, for every $k \in \mathbb{N}$, we let $\mathcal{C}(A, k) = \{\, c \in \mathcal{C}(A) \mid \mathrm{arity}(c) \leq k \,\}$ and $\mathcal{A}(A, k) = \{\, \alpha \in \mathcal{A}(A) \mid |\alpha| \leq k \,\}$.

Intuitively, a category c/c' can be combined with a category c' to its right to become c; similarly, a category $c\backslash c'$ can be combined with c' to its left. Formally, given an alphabet A and $k \in \mathbb{N}$, a *rule of degree k over A* takes one of two possible forms [28]:

$$ax/c,\; c|_1c_1 \cdots |_kc_k \to ax|_1c_1 \cdots |_kc_k \qquad \text{(forward rule)}$$
$$c|_1c_1 \cdots |_kc_k,\; ax\backslash c \to ax|_1c_1 \cdots |_kc_k \qquad \text{(backward rule)}$$

where $a \in A$, $c \in \mathcal{C}(A) \cup \{y\}$, and $|_i \in S$ and $c_i \in \mathcal{C}(A) \cup \{y_i\}$ for every $i \in [k]$. The category $ax|c$ with $| \in \{/, \backslash\}$ is called the *primary input category* and the other category $c|_1c_1 \cdots |_kc_k$ is the *secondary input category* of the rule. The categories $c, c_1, \ldots, c_k$ can thus be either concrete categories from $\mathcal{C}(A)$ or a category variable $\{y, y_1, \ldots, y_k\}$ that will match each category from $\mathcal{C}(A)$. Similarly, the argument context variable x will match each argument context of $\mathcal{A}(A)$. We let $\mathcal{R}(A)$ be the set of all rules over A, and for every $k \in \mathbb{N}$ let $\mathcal{R}(A, k)$ be the finite set of all generic (i.e. always using variables instead of concrete categories) rules over A with degree at most k. Rules of degree 0 are called *application rules*, whereas rules of higher degree are called *composition rules*. A *rule system* is a pair $\Pi = (A, R)$ consisting of an alphabet A and a finite set $R \subseteq \mathcal{R}(A)$ of rules over A. A *ground instance* of a rule r is obtained by substituting concrete categories for the variables $\{y, y_1, \ldots\}$ and a concrete argument context for the variable x in r. The set of all ground instances of Π induces a relation $\to_\Pi \subseteq \mathcal{C}(A)^2 \times \mathcal{C}(A)$, which extends to a relation $\Rightarrow_\Pi \subseteq \mathcal{C}(A)^* \times \mathcal{C}(A)^*$ by $\Rightarrow_\Pi = \{\, (\varphi\, c\, c'\, \psi,\; \varphi\, c''\, \psi) \mid \varphi, \psi \in \mathcal{C}(A)^*;\; c, c' \to_\Pi c'' \,\}$.

▶ **Example 3.** Consider the rule $r = Dx/D,\; D/E\backslash C \to Dx/E\backslash C$, where $\{C, D, E\}$ are atoms and x is an argument context variable. A possible ground instance of this rule is $D/C/E/D,\; D/E\backslash C \to D/C/E/E\backslash C$, where x was replaced by the argument context $\square/C/E$. The primary input category $c_1 = D/C/E/D$ has target D and arguments $/C$, $/E$, and $\backslash D$. As c_1 takes three atomic categories as arguments, it is a first-order category and $\mathit{arity}(c_1) = 3$. The rule degree is determined by the number of arguments replacing the last argument of the primary input category, so r has degree $k = 2$. Note that $D/C/E/D$ is short for $((D/C)/E)/D$, which is different from $(D/C)/(E/D)$. The rules $r' = Dx/(E/D),\; E/D\backslash C \to Dx\backslash C$ and $r'' = Dx/(E/D),\; E/D\backslash(C/C) \to Dx\backslash(C/C)$ both have first degree.

▶ **Definition 4** ([28]). *A* combinatory categorial grammar *(CCG) is a tuple $G = (\Sigma, A, R, I, L)$ consisting of an alphabet Σ of* input symbols*, a* rule system *(A, R), a set $I \subseteq A$ of* initial categories*, and a finite relation $L \subseteq \Sigma \times \mathcal{C}(A)$ called* lexicon*. It is a k-*CCG *[resp.* pure k-CCG*], for $k \in \mathbb{N}$, if each $r \in R$ has degree at most k [resp. if $R = \mathcal{R}(A, k)$].*

▶ **Example 5.** The classical categorial grammars of Ajdukiewicz and Bar-Hillel [4], which are also called *AB-grammars*, are 0-CCGs. However, the term 0-CCG is more general since as opposed to AB-grammars, they allow rule restrictions (i.e. they are not necessarily pure). As a concrete example, let $G_3 = (\Sigma, A, \mathcal{R}(A, 0), I, L)$ be the CCG given by the input alphabet $\Sigma = \{c, d\}$, the atomic categories $A = \{C, D\}$, the set of initial categories $I = \{C\}$, and the lexicon L with $L(c) = \{C/D,\; C/D/C\}$ and $L(d) = \{D\}$. Clearly, it is a 0-CCG. For a slightly more involved example containing rule restrictions, see Example 15.

▶ **Definition 6.** *A combinatory categorial grammar $G = (\Sigma, A, R, I, L)$* accepts *the category sequences $\mathcal{C}(G) \subseteq \mathcal{C}(A)^*$ and the string language $\mathcal{L}(G) \subseteq \Sigma^*$, where*

$$\mathcal{C}(G) = \{\varphi \in \mathcal{C}(A)^* \mid \exists a_0 \in I \colon \varphi \Rightarrow^*_{(A,R)} a_0\} \qquad \textit{and} \qquad \mathcal{L}(G) = L^{-1}(\mathcal{C}(G)) \;.$$

A tree $t \in T_{\mathcal{C}(A),\emptyset}(L(\Sigma))$ is a derivation tree *of G if $t(w \cdot 1),\; t(w \cdot 2) \to_{(A,R)} t(w)$ for every $w \in \mathrm{pos}(t) \setminus \mathrm{leaves}(t)$. The set of all such trees is denoted by $\mathcal{D}(G)$.*

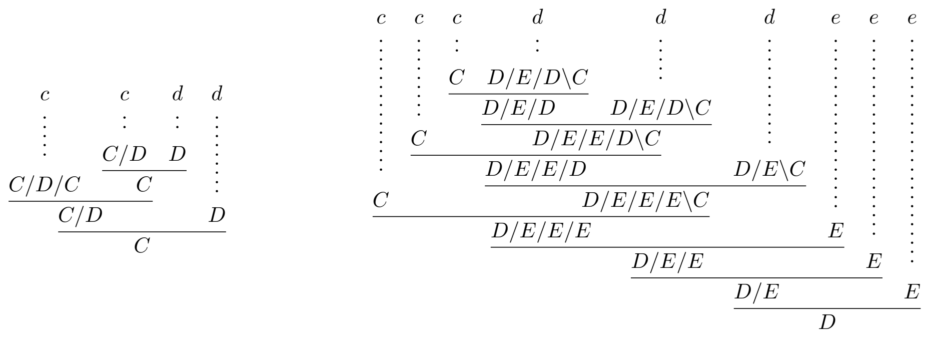

Figure 2 Derivations using the AB-grammar G_3 (left) and the CCG G_4 (right) of Examples 5 and 15, respectively.

The grammar of Example 5 accepts $\mathcal{L}(G_3) = \{c^i d^i \mid i \geq 1\}$, which is context-free but not regular. A derivation tree for the string *ccdd* is shown in Figure 2. We draw derivation trees according to the standard conventions for CCGs, so the root is drawn at the bottom. The dotted lines visualize the input symbol–category mapping implemented by the lexicon. Overall, G_3 accepts the category sequences $\mathcal{C}(G_3) = \{(C/D/C)^{i-1} \cdot (C/D) \cdot D^i \mid i \geq 1\}$.

The language accepted by a CCG is obtained by relabeling the leaf categories of the derivation trees using the lexicon. For the accepted tree language we similarly allow a relabeling to avoid the restriction to the particular symbols of $\mathcal{C}(A)$.

▶ **Definition 7.** *Let $G = (\Sigma, A, R, I, L)$ be a CCG and $\rho \colon \mathcal{C}(A) \to \mathcal{P}_+(\Delta)$ be a relabeling. They* accept *the tree language $\mathcal{F}_\rho(G) = \{\rho(d) \in T_{\Delta,\emptyset}(\Delta) \mid d \in \mathcal{D}(G),\ d(\varepsilon) \in I\}$. A tree language $\mathcal{F} \subseteq T_{\Delta,\emptyset}(\Delta)$ is* acceptable *by the CCG G if there exists a relabeling $\rho' \colon \mathcal{C}(A) \to \mathcal{P}_+(\Delta)$ such that $\mathcal{F} = \mathcal{F}_{\rho'}(G)$.*

Because $L(\Sigma)$ is finite, there exists $k \in \mathbb{N}$ such that $L(\Sigma) \subseteq \mathcal{C}(A, k)$. The least such integer k is called the *arity of L* and denoted by $\operatorname{arity}(L)$; i.e. $\operatorname{arity}(L) = \max\{\operatorname{arity}(c) \mid c \in L(\Sigma)\}$. If $L = \emptyset$, then we let $\operatorname{arity}(L) = 0$.

4 0-CCGs

Let $G = (\Sigma, A, R, I, L)$ be a 0-CCG. An important property of 0-CCGs is that each category that occurs in a derivation tree has arity at most $\operatorname{arity}(L)$. Thus, derivation trees are built over a finite set of symbols.

▶ **Theorem 8** (see [4] and [25, Proposition 3.25])**.** *The string languages accepted by 0-CCGs are exactly the ε-free context-free languages. Moreover, for each 0-CCG G the derivation tree language $\mathcal{D}(G)$ and the accepted tree languages are regular.*

To characterize the tree languages accepted by 0-CCGs, we need to introduce an additional structural property of the derivation tree language $\mathcal{D}(G)$ and the acceptable tree languages. Roughly speaking, the *min-height* $\operatorname{mht}(t)$ of a tree t is the minimal length of a path from the root to a leaf. Recall that the height coincides with the maximal length of those paths. For all alphabets Σ_2 and Σ_0, let $\operatorname{mht} \colon T_{\Sigma_2,\emptyset}(\Sigma_0) \to \mathbb{N}$ be such that $\operatorname{mht}(a) = 0$ and $\operatorname{mht}(c(t_1, t_2)) = 1 + \min(\operatorname{mht}(t_1), \operatorname{mht}(t_2))$ for all $a \in \Sigma_0$, $c \in \Sigma_2$, and $t_1, t_2 \in T_{\Sigma_2,\emptyset}(\Sigma_0)$. A tree $t \in T_{\Sigma_2,\emptyset}(\Sigma_0)$ is *universally* mht-*bounded by* $h \in \mathbb{N}$ if $\operatorname{mht}(t|_w) \leq h$ for every $w \in \operatorname{pos}(t)$. Finally, a tree language $\mathcal{F} \subseteq T_{\Sigma_2,\emptyset}(\Sigma_0)$ is *universally* mht-*bounded by* h if every $t \in \mathcal{F}$ is

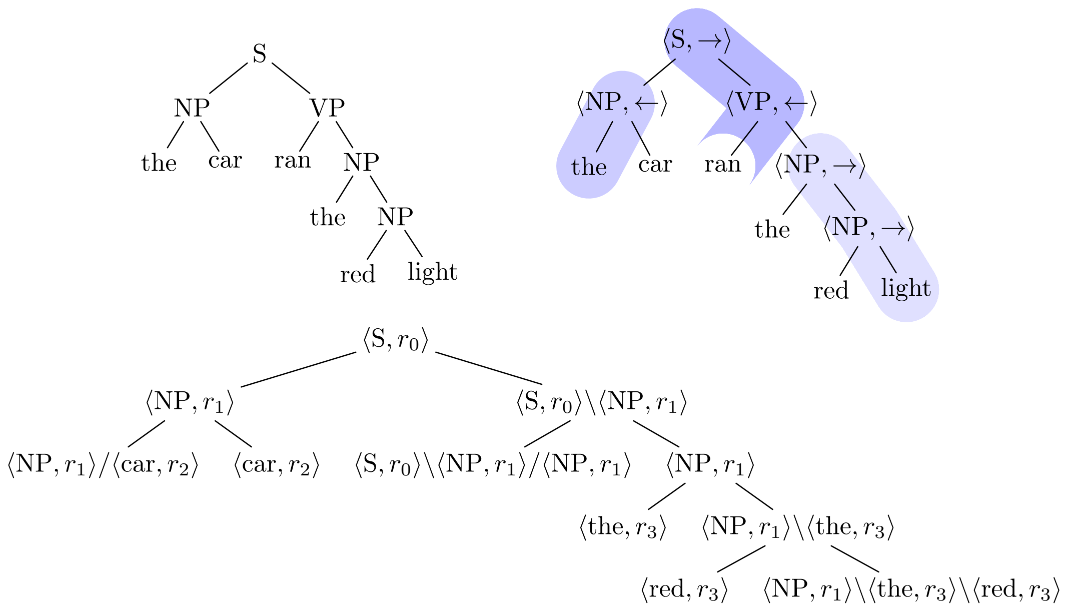

Figure 3 Decomposition into spinal runs and the corresponding derivation tree of the 0-CCG.

universally mht-bounded by h, and it is *universally* mht-*bounded* if there exists $h \in \mathbb{N}$ such that it is universally mht-bounded by h. Note that "universally mht-bounded" is a purely structural property of a tree as it only depends on the shape of the tree and is completely agnostic about the node labels. It is thus preserved by the application of a relabeling. Consequently, $\rho(\mathcal{F})$ is universally mht-bounded by h if and only if $\mathcal{F}$ is universally mht-bounded by h for every tree language $\mathcal{F} \subseteq T_{\Sigma_2,\emptyset}(\Sigma_0)$ and relabeling $\rho\colon (\Sigma_2 \cup \Sigma_0) \to \mathcal{P}_+(\Delta)$.

Let us reconsider the 0-CCG G_3 of Example 5. The set $\mathcal{D}(G_3)$ is universally mht-bounded by 1 (see Figure 2). It turns out that exactly the universally mht-bounded regular tree languages are acceptable by 0-CCGs. We already observed that the tree languages acceptable by 0-CCGs are regular, but for the converse we have to exploit the universal mht-bound. We utilize those short paths to a leaf to decompose the tree into spines, which are short paths in the tree that lead from a node to a leaf and are never longer than the universal min-height. The primary categories for the applications are placed along those spines and each spine terminates in an atomic category that can be combined with the category from another spine. The idea of the construction is illustrated in Figure 3. This close relation and the good closure properties of regular tree languages allow us to derive a number of closure results for the tree languages acceptable by 0-CCGs (see Table 1).

▶ **Theorem 9.** *Let $\mathcal{F} \subseteq T_{\Sigma_2,\emptyset}(\Sigma_0)$ be a tree language. Then $\mathcal{F}$ is acceptable by some* 0*-CCG if and only if it is regular and universally* mht*-bounded.*

We have seen that, while basic categorial grammars and context-free grammars are weakly equivalent, they are not strongly equivalent when considered as tree-generating devices. More specifically, the class of derivation tree languages of basic categorial grammars are a proper subclass of the class of local tree languages (i.e. derivation tree languages of context-free grammars). This result is similar to a result by Schabes et al. [21] showing that context-free grammars are not closed under strong lexicalization, meaning that there are context-free grammars such that no lexicalized grammar[5] generates the same derivation tree language.

[5] A CFG is called lexicalized if every production contains a terminal symbol.

Table 1 Closure properties of the tree languages acceptable by 0-CCGs and 1-CCGs.

closure \ class	regular tree languages = tree languages acceptable by 1-CCGs	tree languages acceptable by 0-CCGs
union	✓	✓
intersection	✓	✓
complement	✓	✗
relabeling	✓	✓
α-concatenation [10]	✓	✓
α-iteration [10]	✓	✗

5 1-CCGs

In this section, we will consider 1-CCGs, which allow rules of degree at most 1. Thus, the secondary input categories appearing in their derivation tree languages have at most one additional argument after the category consumed by the composition. We will show that the 1-CCGs accept exactly the regular tree languages by showing inclusion in both directions.

▶ **Lemma 10.** *For each 1-CCG G the derivations $\mathcal{D}(G)$ and the accepted tree language are regular.*

The following lemma establishes a normal form for regular tree grammars that is easily achieved using standard techniques. For every $m \in \mathbb{N}$, let $\mathbb{Z}_m = \{i \in \mathbb{N} \mid 0 \leq i < m\}$.

▶ **Lemma 11.** *For each RTG G there exists an equivalent RTG $G' = (\mathbb{Z}_m, \Sigma, I', P')$ in normal form, in which every nonterminal $n \in \mathbb{Z}_m$ generates a uniquely defined terminal symbol σ_n; i.e. for all $n \in \mathbb{Z}_m$ there exists $\sigma_n \in \Sigma$ such that $t(\varepsilon) = \sigma_n$ for all $t \in T_\Sigma$ with $n \Rightarrow^+_{G'} t$.*

Given an RTG $G = (\mathbb{Z}_m, \Sigma, I, P)$ in the normal form of Lemma 11, we are allowed to regard only the nonterminals of G when constructing an equivalent 1-CCG. Our goal is to find a 1-CCG $G' = (\Sigma', A, R, I', L)$ and a projection $\pi : \mathcal{C}(A) \to \mathbb{Z}_m$ such that $\mathcal{F}(G) = \mathcal{F}_{\pi' \circ \pi}(G')$. Because π maps from categories to nonterminals, but $\mathcal{F}(G)$ is labeled by terminal symbols, we need the projection $\pi' : \mathbb{Z}_m \to \Sigma$ to map from nonterminals to terminals. This function is well-defined due to the constraint on G, that each nonterminal generates a single terminal.

Given a production $n \to \sigma(n_1, n_2) \in P$ and a projection $\pi : \mathcal{C}(A) \to \mathbb{Z}_m$, we need categories $c_1 \in \pi^{-1}(n_1)$ and $c_2 \in \pi^{-1}(n_2)$ for each category $c \in \pi^{-1}(n)$ such that $c_1, c_2 \to c$ is a valid ground instance of a rule in R. This ensures that each category can be derived by the composition of two categories mapped to matching nonterminals. We only regard first-order categories with at most one argument due to the restriction on 1-CCGs. Starting from any nonterminal, the productions P allow the derivation of at most all ordered pairs of nonterminals. The number of ordered pairs $\mathbb{Z}_m^2$ increases quadratically in m, whereas the number of different composition input pairs resulting in a fixed category increases only linearly in $|A|$. The *category matrix* depicted in Figure 4 illustrates that a first-order category with one argument is the result of the forward compositions of $|A|$ different category pairs. In addition to composition rules, application rules are neccessary to obtain an atomic initial category. Based on these observations, we construct a 1-CCG G' with m^2 atoms in the following way.

▶ **Definition 12.** *Given an RTG $G = (\mathbb{Z}_m, \Sigma, I, P)$ in the normal form of Lemma 11, we construct the* 1*-CCG $G' = (\{x\}, \mathbb{Z}_m^2, R, \pi^{-1}(I) \cap \mathbb{Z}_m^2, L)$ with*

$$\begin{aligned} R &= \{a/b,\, b \to a \mid \pi(a) \to \sigma\bigl(\pi(a/b), \pi(b)\bigr) \in P,\, \sigma \in \Sigma_2,\, a, b \in \mathbb{Z}_m^2\} \\ &\quad \cup \{a/b,\, b/c \to a/c \mid \pi(a/c) \to \sigma\bigl(\pi(a/b), \pi(b/c)\bigr) \in P,\, \sigma \in \Sigma_2,\, a, b, c \in \mathbb{Z}_m^2\} \\ L &= \{(x, a) \mid a \in \mathcal{C}(A, 1) \cap \mathcal{C}_f(A),\, \pi(a) \to \alpha \in P,\, \alpha \in \Sigma_0,\, a \in \mathbb{Z}_m^2\} \end{aligned}$$

where the mapping $\pi \colon (\mathbb{Z}_m^2 \cup \{n/n' \mid n, n' \in \mathbb{Z}_m^2\}) \to \mathbb{Z}_m$ is given by $\pi\bigl((i,j)\bigr) = i$ and $\pi\bigl((i,j)/(i',j')\bigr) = i + j' \bmod m$ for all $i, i', j, j' \in \mathbb{Z}_m$.

▶ **Lemma 13.** *For each RTG G there exists a 1-CCG G' accepting the tree language $\mathcal{F}(G)$.*

Proof. We have to establish that the 1-CCG $G' = (\{x\}, \mathbb{Z}_m^2, R, \pi^{-1}(I), L)$ of Definition 12 accepts the tree language $\mathcal{F}(G)$ of RTG $G = (\mathbb{Z}_m, \Sigma, I, P)$ using the relabeling $\pi' \circ \pi$. The category $c = (i,j)/(i',j')$ is the result of the forward composition of $(i,j)/(k,l)$ and $(k,l)/(i',j')$, where $i, i', j, j', k, l \in \mathbb{Z}_m$. Figure 4 illustrates the projection π by means of a *projection matrix*, which is a category matrix with categories replaced according to the projection. Row and column labels follow lexicographic order. When we slice it evenly into blocks of size $m \times m$, we can observe that the entries in the rows cycle through the nonterminals, whereas in a single column, each block has only a single nonterminal in all m entries. This is because the value of j' changes in every entry, whereas the value of i changes only every m entries. Nonetheless, a complete column of the whole projection matrix contains all m nonterminals. Relabeling in this manner ensures that all pairs of nonterminals are covered by arbitrary pairs of row and column. These are determined by the result category.

Given a category $(i,j)/(i',j')$ and an ordered pair (g,h) of nonterminals, we need to verify that there exist $k, l \in \mathbb{Z}_m$ with $\pi((i,j)/(k,l)) = g$ and $\pi((k,l)/(i',j')) = h$. Since $g = (i{+}l) \bmod m$ and $h = (k{+}j') \bmod m$, we obtain $l = (g{-}i) \bmod m$ and $k = (h{-}j') \bmod m$. Furthermore, given an arbitrary atom (i,j) and nonterminals $g, h \in \mathbb{Z}_m$, we want to find a category $(i,j)/(k,l)$ and an atom (k,l) such that $\pi((i,j)/(k,l)) = g$ and $\pi((k,l)) = h$. From the definition of the projection, $\pi((k,l)) = k$, so we have $k = h$ and $l = (g - i) \bmod m$.

We relabel $\mathcal{F}(G')$ using $\pi' \circ \pi$ as described above. Due to the fact that the categories occurring in derivation trees of G' cannot have higher order or arity greater than 1, they never leave the domain of π. As a result, we were able to construct a 1-CCG accepting the tree language $\mathcal{F}(G)$. Thus, for each regular tree language, we can construct a 1-CCG accepting it. ◀

▶ **Theorem 14.** *The tree languages accepted by 1-CCGs are exactly the regular tree languages.*

6 Inclusion in the Context-Free Tree Languages

In this section, we want to relate the derivation tree languages of CCGs to the context-free tree languages. However, this is complicated by the presence of potentially infinitely many categories. Let us illustrate the problem first.

▶ **Example 15.** Let $G_4 = (\Sigma, A, R, \{D\}, L)$ be the 3-CCG given by the alphabet $\Sigma = \{c, d, e\}$, the atomic categories $A = \{C, D, E\}$, the lexicon L with $L(c) = \{C\}$, $L(d) = \{D/E\backslash C, D/E/D\backslash C\}$, $L(e) = \{E\}$, and the rule set R consisting of the rules

$$\begin{aligned} &Dx/D,\, D/E/D\backslash C \to Dx/E/D\backslash C \qquad && Dx/E,\, E \to Dx \\ &Dx/D,\, D/E\backslash C \to Dx/E\backslash C && C,\, Dx\backslash C \to Dx \end{aligned}$$

	a_0	a_1	a_2	a_3
a_0	a_0/a_0	a_0/a_1	a_0/a_2	a_0/a_3
a_1	a_1/a_0	a_1/a_1	a_1/a_2	a_1/a_3
a_2	a_2/a_0	a_2/a_1	a_2/a_2	a_2/a_3
a_3	a_3/a_0	a_3/a_1	a_3/a_2	a_3/a_3

	$(0,0)$	$(0,1)$	$(0,2)$	$(1,0)$	$(1,1)$	$(1,2)$	$(2,0)$	$(2,1)$	$(2,2)$
$(0,0)$	0	1	2	0	1	2	0	1	2
$(0,1)$	0	1	2	0	1	2	0	1	2
$(0,2)$	0	1	2	0	1	2	0	1	2
$(1,0)$	1	2	0	1	2	0	1	2	0
$(1,1)$	1	2	0	1	2	0	1	2	0
$(1,2)$	1	2	0	1	2	0	1	2	0
$(2,0)$	2	0	1	2	0	1	2	0	1
$(2,1)$	2	0	1	2	0	1	2	0	1
$(2,2)$	2	0	1	2	0	1	2	0	1

Figure 4 The category matrix (left) contains all first-order categories of arity 1 with only forward slashes in a CCG with four atoms. Each category is the result of the forward composition of a category taken from the same row and one from the same column, respectively. The i-th entry of each row can be combined with the i-th entry of each column. Thus, each category is the result of four different forward compositions. The projection matrix (right) shows a 1-CCG with nine atomic categories after relabeling using projection $\pi\colon \mathcal{C}(A) \to \mathbb{Z}_3$, obtained from an RTG with three nonterminals by applying Definition 12. Suppose we want to find two categories projected to nonterminals $(g,h) = (0,2)$ whose composition yields $(i,j)/(i',j') = (0,1)/(0,1)$. These are categories $(0,1)/(1,0)$ and $(1,0)/(0,1)$ since $(k,l) = ((h-j') \bmod 3, (g-i) \bmod 3) = ((2-1) \bmod 3, (0-0) \bmod 3) = (1,0)$.

where $x \in \mathcal{A}(A)$. From a few sample derivation trees (see Figure 2) we can convince ourselves that G_4 accepts the string language $\mathcal{L}(G_4) = \{\, c^i d^i e^i \mid i \geq 1 \,\}$, which shows that 3-CCGs can accept non-context-free string languages. In addition, the derivation trees $\mathcal{D}(G_4)$ contain infinitely many categories as labels.

Since the classical tree language theory only handles finitely many labels, we switch to a different representation and consider *rule trees*. To simplify the notation, we introduce the following shorthands. We let $\mathrm{T} = T_{R,\emptyset}\big(L(\Sigma)\big)$ be the set of all potential rule trees (see Definition 16), and for all alphabets N_1 and N_0 we let $\mathrm{SF}(N_1, N_0) = T_{R,N_1}\big(L(\Sigma) \cup N_0\big)$ be the forms of a sCFTG with unary nonterminals N_1 and nullary nonterminals N_0.

▶ **Definition 16.** *Let $G = (\Sigma, A, R, I, L)$ be a CCG. A tree $t \in \mathrm{T}$ is a* rule tree *of G if $\mathrm{cat}_G(t) \in I$, where $\mathrm{cat}_G \colon \mathrm{T} \to \mathcal{C}(A)$ is the partial function that is inductively defined by*

- *$\mathrm{cat}_G(c) = c$ for all $c \in L(\Sigma)$,*
- *$\mathrm{cat}_G\big((ax/c,\, c\gamma \to ax\gamma)(t_1, t_2)\big) = a\alpha\gamma$ for all trees $t_1, t_2 \in \mathrm{T}$ such that $\mathrm{cat}_G(t_1) = a\alpha/c$ and $\mathrm{cat}_G(t_2) = c\gamma$, and*
- *$\mathrm{cat}_G\big((c\gamma,\, ax\backslash c \to ax\gamma)(t_1, t_2)\big) = a\alpha\gamma$ for all trees $t_1, t_2 \in \mathrm{T}$ such that $\mathrm{cat}_G(t_1) = c\gamma$ and $\mathrm{cat}_G(t_2) = a\alpha\backslash c$.*

The set of all rule trees of G is denoted by $\mathcal{R}(G)$.

The rule trees provide a natural encoding of the (successful) derivation trees of a CCG using only finitely many labels. More precisely, there is an (obvious) bijection between the derivation trees $\mathcal{D}(G)$ and the domain of the function cat_G.

In the following, let $G = (\Sigma, A, R, I, L)$ be a CCG. Our goal is to construct an sCFTG that generates exactly the rule tree language $\mathcal{R}(G)$. To this end, we first need to limit the arity of the categories. Let $k \in \mathbb{N}$ be the maximal arity of a category in

$$I \cup L(\Sigma) \cup \{c\gamma \mid ax/c,\, c\gamma \to ax\gamma \in R\} \cup \{c\gamma \mid c\gamma,\, ax\backslash c \to ax\gamma \in R\}\,,$$

i.e. the maximal arity of the categories that occur in the lexicon, as initial category, or as the secondary premise of a rule of R. Roughly speaking, the constructed sCFTG will use the categories $\mathcal{C}(A, k)$ as nullary nonterminals and tuples $\langle a, |c, \gamma\rangle$ consisting of an

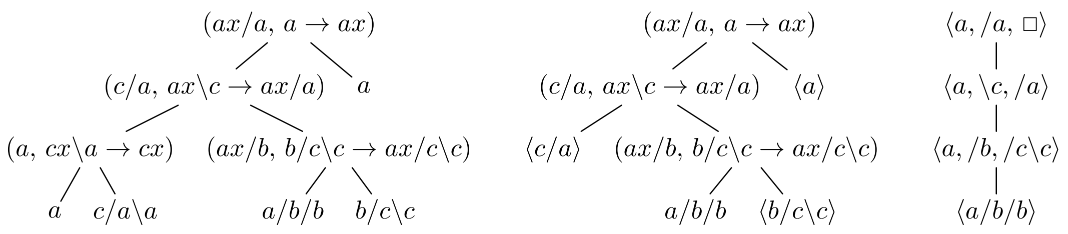
$(ax/a,\ a \to ax)$
$(c/a,\ ax\backslash c \to ax/a)$
a
$(a,\ cx\backslash a \to cx)$
$(ax/b,\ b/c\backslash c \to ax/c\backslash c)$
a
$c/a\backslash a$
$a/b/b$
$b/c\backslash c$
$(ax/a,\ a \to ax)$
$(c/a,\ ax\backslash c \to ax/a)$
$\langle a\rangle$
$\langle c/a\rangle$
$(ax/b,\ b/c\backslash c \to ax/c\backslash c)$
$a/b/b$
$\langle b/c\backslash c\rangle$
$\langle a, /a, \square\rangle$
$\langle a, \backslash c, /a\rangle$
$\langle a, /b, /c\backslash c\rangle$
$\langle a/b/b\rangle$

Figure 5 Rule tree t (without lexical entries), spinal(t), and its encoding enc(t).

atomic category $a \in A$, a single argument $|c$ with $| \in S$ and $c \in \mathcal{C}(A,k)$, and an argument tree $\gamma \in \mathcal{A}(A,k)$ as unary nonterminals. Recall that we write substitutions $\alpha[t]$ as $t\alpha$ for $\alpha \in \mathcal{A}(A)$ and $t \in \mathcal{C}(A) \cup \mathcal{A}(A)$.

▶ **Definition 17.** *We construct the sCFTG $G' = (N_1 \cup N_0, R \cup L(\Sigma), I', P)$ with*

- $N_1 = \{\langle a, |c, \gamma\rangle \mid a \in A,\ | \in S,\ c \in \mathcal{C}(A,k),\ \gamma \in \mathcal{A}(A,k)\}$ *and* $N_0 = \{\langle c\rangle \mid c \in \mathcal{C}(A,k)\}$,
- $I' = \{\langle a_0\rangle \mid a_0 \in I\}$, *and*
- *the following set P of productions*

$$\begin{aligned}
P = \big\{&\langle c\rangle \to c \mid c \in L(\Sigma)\big\} \cup && (1)\\
&\big\{\langle a, /c, \gamma\rangle \to s\big(\square, \langle c\gamma\rangle\big) \mid s = \big(ax/c,\ c\gamma \to ax\gamma\big) \in R\big\} \cup && (2)\\
&\big\{\langle a, \backslash c, \gamma\rangle \to s\big(\langle c\gamma\rangle, \square\big) \mid s = \big(c\gamma,\ ax\backslash c \to ax\gamma\big) \in R\big\} \cup && (3)\\
&\big\{\langle a\alpha\gamma\rangle \to \langle a, |c, \gamma\rangle\big(\langle a\alpha|c\rangle\big) \mid a \in A,\ \alpha, \gamma \in \mathcal{A}(A),\ | \in S,\ c \in \mathcal{C}(A,k),\\
&\qquad |\alpha| < k,\ |\alpha\gamma| \le k\big\} \cup && (4)\\
&\big\{\langle a, |c, \gamma\rangle \to \langle a, |'c', \square\rangle\big(\langle a, |c, \gamma|'c'\rangle(\square)\big)\\
&\qquad \mid a \in A,\ |, |' \in S,\ c, c' \in \mathcal{C}(A,k),\ \gamma \in \mathcal{A}(A, k-1)\big\} && (5)
\end{aligned}$$

We still have to establish that G' indeed generates exactly $\mathcal{R}(G)$. This will be achieved by showing both inclusions in the next chain of lemmas.

▶ **Lemma 18.** $\mathcal{F}(G') \subseteq \mathcal{R}(G)$.

For the converse, we decompose and encode rule trees $\mathcal{R}(G)$ in a more compact manner. First, we translate a rule tree into its *primary spine form*. For all $a \in A$, $c \in \mathcal{C}(A,k)$, $\gamma \in \mathcal{A}(A,k)$, and $t_1, t_2 \in \mathrm{T}$ we let

$$\begin{aligned}
\mathrm{spinal}\big(c\big) &= c\\
\mathrm{spinal}\big((ax/c,\ c\gamma \to ax\gamma)(t_1, t_2)\big) &= (ax/c,\ c\gamma \to ax\gamma)\big(\mathrm{spinal}(t_1), \langle c\rangle\big)\\
\mathrm{spinal}\big((c\gamma,\ ax\backslash c \to ax\gamma)(t_1, t_2)\big) &= (c\gamma,\ ax\backslash c \to ax\gamma)\big(\langle c\rangle, \mathrm{spinal}(t_2)\big)\ .
\end{aligned}$$

Clearly, spinal: $\mathrm{T} \to T_{R,\emptyset}(L(\Sigma) \cup N_0)$. An example is shown in Figure 5. Additionally, we encode rule trees using only the nonterminals of G' [i.e. a tree of $T_{\emptyset,N_1}(N_0)$]. To this end, we define a mapping enc: $\mathrm{T} \to T_{\emptyset,N_1}(N_0)$. For all $a \in A$, $c \in \mathcal{C}(A,k)$, $\gamma \in \mathcal{A}(A,k)$, and $t_1, t_2 \in \mathrm{T}$ we let

$$\begin{aligned}
\mathrm{enc}(c) &= \langle c\rangle\\
\mathrm{enc}\big((ax/c,\ c\gamma \to ax\gamma)(t_1, t_2)\big) &= \langle a, /c, \gamma\rangle\big(\mathrm{enc}(t_1)\big)\\
\mathrm{enc}\big((c\gamma,\ ax\backslash c \to ax\gamma)(t_1, t_2)\big) &= \langle a, \backslash c, \gamma\rangle\big(\mathrm{enc}(t_2)\big)\ .
\end{aligned}$$

This encoding is also demonstrated in Figure 5.

$$\langle a/b/c\rangle \Rightarrow_{G'} \begin{array}{c}\langle a, /b, /c\rangle\\ |\\ \langle a/b/b\rangle\end{array} \Rightarrow_{G'} \begin{array}{c}\langle a, \backslash c, \square\rangle\\ |\\ \langle a, /b, /c\backslash c\rangle\\ |\\ \langle a/b/b\rangle\end{array} \Rightarrow_{G'} \begin{array}{c}\langle a, /a, \square\rangle\\ |\\ \langle a, \backslash c, /a\rangle\\ |\\ \langle a, /b, /c\backslash c\rangle\\ |\\ \langle a/b/b\rangle\end{array}$$

Figure 6 Derivation of the encoding.

▶ **Lemma 19.** $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^*_{G'} \mathrm{enc}(t) \Rightarrow^*_{G'} \mathrm{spinal}(t)$ *for every* $t \in \mathrm{T}$ *with* $\mathrm{cat}_G(t) \in \mathcal{C}(A, k)$, *and hence* $\mathcal{R}(G) \subseteq \mathcal{F}(G')$.

▶ **Theorem 20.** *The rule trees* $\mathcal{R}(G)$ *of a CCG* G *can be generated by an sCFTG.*

7 Proper Inclusion for Pure CCGs

In this section, we show that there exist CFG derivation tree languages that cannot be accepted by any pure CCG. A CCG (Σ, A, R, I, L) is *pure* if $R = \mathcal{R}(A, k)$ for some $k \in \mathbb{N}$. In particular, this shows that the inclusion demonstrated in Section 6 is proper for pure CCGs. We start with our counterexample CFG. To make the text more readable, we assume henceforth that all computations with nonterminals are performed modulo 3.

▶ **Example 21.** Let us consider the CFG $G_{\mathrm{ex}} = (N, \Gamma, \langle 0, 0\rangle, P)$ with the nonterminals $N = \{\langle i, j\rangle \mid 0 \leq i, j \leq 2\}$, the terminals $\Gamma = \{\alpha\}$, and the set P of productions contains exactly $\langle i, j\rangle \to \langle i+1, j\rangle\ \langle i, j+1\rangle$ and $\langle i, j\rangle \to \alpha$ for every $\langle i, j\rangle \in N$. Clearly, the tree language $\mathcal{D}(G_{\mathrm{ex}})$ is not universally mht-bounded.

Theorem 9 already shows that the tree language $\mathcal{D}(G_{\mathrm{ex}})$ cannot be accepted by any 0-CCG. Similarly, it is impossible to accept $\mathcal{D}(G_{\mathrm{ex}})$ with a pure CCG. This follows from the transformation schemes of [14] that change the order of consecutive application and non-application operations, resulting in derivation trees with reordered subtrees and therefore with the wrong shape after relabeling. Due to the absence of rule restrictions in pure CCGs, the applicability of these transformations cannot be prevented.

▶ **Theorem 22.** *The tree language* $\mathcal{D}(G_{\mathrm{ex}})$ *cannot be accepted any pure CCG.*

8 Conclusion

We have shown that the tree languages accepted by CCGs with limited composition depth and rule restrictions are a subset of the tree languages generated by simple monadic context-free tree grammars. This inclusion is proper for pure CCGs (i.e. without rule restrictions). In addition, the tree languages accepted by 0-CCGs are a proper subset of regular tree languages, whereas those accepted by 1-CCGs are exactly the regular tree languages. While the step from 0-CCGs to 1-CCGs does not increase the weak generative capacity, the strong generative capacity increases. We also observe that there is no difference in expressivity for 0-CCGs between the pure and non-pure variants, while for higher rule degrees, pure CCGs are strictly weaker. The first statement follows from the fact that we are able to construct an equivalent pure 0-CCG for each mht-bounded regular tree language.

The construction used in the classical proof of weak equivalence between CCG and TAG [28] demonstrated that there is no difference in weak generative capacity between 2-CCGs and k-CCGs with $k > 2$ and that the inclusion of higher-order categories in the lexicon does not change weak generative capacity. However, as stated in the Introduction, this construction utilizes ε-entries, which are problematic from a computational point of view [15]. Future work should explore the relationship between TAG and CCG, and in particular the effects of higher rule degrees k (up to unlimited composition depth) and higher-order categories, in the absence of ε-entries. Another interesting direction is strong generative capacity: If for a given sCFTG a strongly equivalent CCG could be constructed (the inverse direction of what we showed in Theorem 20), this would characterize the tree-generative capacity of CCG exactly. Furthermore, the effect of the inclusion of additional rules on the expressivity should be studied.

References

1 Kazimierz Ajdukiewicz. Die syntaktische Konnexität. *Studia Philosophica*, 1:1–27, 1935.

2 Franz Baader and Tobias Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

3 Yehoshua Bar-Hillel. A quasi-arithmetical notation for syntactic description. *Language*, 29(1):47–58, 1953.

4 Yehoshua Bar-Hillel, Haim Gaifman, and Eli Shamir. On Categorial and Phrase Structure Grammars. In Yehoshua Bar-Hillel, editor, *Language and Information: Selected Essays on Their Theory and Application*, pages 99–115. Addison Wesley, 1964.

5 Yehoshua Bar-Hillel, Micha Perles, and Eli Shamir. On formal properties of simple phrase structure grammars. In Yehoshua Bar-Hillel, editor, *Language and Information: Selected Essays on their Theory and Application*, chapter 9, pages 116–150. Addison Wesley, 1964.

6 Noam Chomsky. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124, 1956.

7 Haskell B. Curry. Foundations of combinatorial logic. *American Journal of Mathematics*, 52(3):509–536, 1930.

8 Haskell B. Curry, Robert Feys, and William Craig. *Combinatory Logic.* Number 1 in Studies in Logic and the Foundations of Mathematics. North-Holland, 1958.

9 Ferenc Gécseg and Magnus Steinby. *Tree Automata.* Akadémiai Kiadó, Budapest, 1984. 2nd revision available at `arXiv:1509.06233`.

10 Ferenc Gécseg and Magnus Steinby. Tree Languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.

11 John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages and Computation.* Addison Wesley, 1979.

12 Aravind K. Joshi and Yves Schabes. Tree-Adjoining Grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Beyond Words*, volume 3 of *Handbook of Formal Languages*, pages 69–123. Springer, 1997.

13 Alexander Koller and Marco Kuhlmann. Dependency Trees and the Strong Generative Capacity of CCG. In *Proc. 12th EACL*, pages 460–468. ACL, 2009.

14 Marco Kuhlmann, Alexander Koller, and Giorgio Satta. Lexicalization and Generative Power in CCG. *Comput. Linguist.*, 41(2):187–219, 2015.

15 Marco Kuhlmann, Giorgio Satta, and Peter Jonsson. On the Complexity of CCG Parsing. *Comput. Linguist.*, 44(3):447–482, 2018.

16 Joachim Lambek. The mathematics of sentence structure. *Amer. Math. Monthly*, 65(3):154–170, 1958.

17 Kenton Lee, Mike Lewis, and Luke Zettlemoyer. Global neural CCG parsing with optimality guarantees. In *Proc. 2016 EMNLP*, pages 2366–2376. ACL, 2016.

18 Mike Lewis and Mark Steedman. Unsupervised induction of cross-lingual semantic relations. In *Proc. 2013 EMNLP*, pages 681–692. ACL, 2013.

19 William C. Rounds. Context-Free Grammars on Trees. In *Proc. 1st STOC*, pages 143–148. ACM, 1969.

20 William C. Rounds. Tree-Oriented Proofs of Some Theorems on Context-Free and Indexed Languages. In *Proc. 2nd STOC*, pages 109–116. ACM, 1970.

21 Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing Strategies with 'Lexicalized' Grammars: Application to Tree Adjoining Grammars. In *Proc. 12th CoLing*, pages 578–583, 1988.

22 Moses Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92(3–4):305–316, 1924.

23 Mark Steedman. *The Syntactic Process*. MIT Press, 2000.

24 Mark Steedman and Jason Baldridge. Combinatory Categorial Grammar. In Robert D. Borsley and Kersti Börjars, editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, chapter 5, pages 181–224. Blackwell, 2011.

25 Hans-Jörg Tiede. *Deductive Systems and Grammars: Proofs as Grammatical Structures*. PhD thesis, Indiana University, Bloomington, IN, USA, 1999.

26 Krishnamurti Vijay-Shanker and David J. Weir. Combinatory Categorial Grammars: Generative Power and Relationship to Linear Context-Free Rewriting Systems. In *Proc. 26th ACL*, pages 278–285. ACL, 1988.

27 Krishnamurti Vijay-Shanker and David J. Weir. Polynomial time parsing of combinatory categorial grammars. In *Proc. 28th ACL*, pages 1–8. ACL, 1990.

28 Krishnamurti Vijay-Shanker and David J. Weir. The Equivalence of Four Extensions of Context-Free Grammars. *Math. Systems Theory*, 27(6):511–546, 1994.