Journal of Computer and System Sciences 124 (2022) 214–233

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

The tree-generative capacity of combinatory categorial grammars ☆

Marco Kuhlmann [a,1,*], Andreas Maletti [b], Lena Katharina Schiffer [b,2]

[a] Department of Computer and Information Science, Linköping University, SE-581 83 Linköping, Sweden
[b] Faculty of Mathematics and Computer Science, Universität Leipzig, P.O. box 100 920, D-04009 Leipzig, Germany

ARTICLE INFO

Article history:
Received 11 September 2020
Received in revised form 23 August 2021
Accepted 29 October 2021
Available online 10 November 2021

Keywords:
Combinatory categorial grammar
Regular tree language
Linear context-free tree language

ABSTRACT

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

© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

1. Introduction

Categorial grammars [1] were introduced alongside the phrase-structure grammars (regular, context-free, context-sensitive grammars, etc.) of the Chomsky hierarchy [2] inspired by classical notions from proof theory [3,4]. Combinatory Categorial Grammar (CCG) [5,6] is an extension following the approach of combinatory logic [7,8]. CCG received considerable attention in theoretical computer science, culminating in the proofs of its mild context-sensitivity, as well as its equivalence to several other established grammar formalisms [9]. It has since become a widely applied formalism in computational linguistics [10,11].

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 adjacent categories can be combined to eventually obtain a (binary) derivation tree. The mentioned equivalence result due to Vijay-Shanker and Weir [9] shows that CCG, tree-adjoining grammar (TAG) [12] as well as linear indexed grammar [13] are weakly equivalent, 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 lexicon entries for the empty word. Modern variants of CCG disfavor rule restrictions and the obtained *pure* CCGs are strictly less expressive than TAG [14] unless unbounded generalized composition rules are permitted, in which case they are strictly

☆ This contribution is an extended and revised version of M. Kuhlmann, A. Maletti, and L.K. Schiffer. "The Tree-Generative Capacity of Combinatory Categorial Grammars." *Foundations of Software Technology and Theoretical Computer Science*, LIPIcs 150, 44:1–44:14, 2019.

* Corresponding author.
E-mail addresses: marco.kuhlmann@liu.se (M. Kuhlmann), maletti@informatik.uni-leipzig.de (A. Maletti), schiffer@informatik.uni-leipzig.de (L.K. Schiffer).

[1] Supported by the Centre for Industrial IT (CENIIT), grant 15.02.
[2] Supported by the German Research Foundation (DFG) Research Training Group GRK 1763 'Quantitative Logics and Automata'.

https://doi.org/10.1016/j.jcss.2021.10.005
0022-0000/© 2021 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

more expressive than TAG [15]. Indeed, CCG with unbounded composition rules, rule restrictions as well as ε-entries in the lexicon is TURING-complete [16].

The mentioned studies examine the string (or weak) generative capacity of CCG, but already [15] asks for the tree (or strong) generative capacity or, more specifically, the expressiveness of the languages of CCG derivation trees [17]. The relation between the strong and the weak generative capacity of CCG is not fully understood. For example, KOLLER and KUHLMANN [18] show that CCG and TAG are incomparable when considered as generators of *dependency trees*, a type of linguistically motivated tree structures that explicitly model syntactic dependencies between words in a sentence. However, the exact characterization of what sets of dependency trees can and cannot be generated by CCG remains unknown. In the present contribution, we answer the original question and characterize the tree languages generated by CCGs, and relate them to the standard notions of regular [17,19] and context-free tree languages [20,21]. A tree language $\mathcal{F}$ is *generated* 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 [22], who studied the strong generative capacity of LAMBEK-style categorial grammars [23].

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 [24]). In general, we allow rule restrictions that further constrain the categories which 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 (Theorem 29) is that the tree languages generated by CCGs can also be generated by simple monadic context-free tree grammar (sCFTG), where *simple* means that the productions are linear and nondeleting. For CCG without rule restrictions, this inclusion is proper since not even all regular tree languages [17] are generated by these CCGs (Theorem 31). 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 13). This is an alternative proof for an analogous result in [25, Theorem 1.1], where the focus is on classical categorial grammars, which are pure CCGs without composition operations. Finally, if we limit the permitted composition operators to first degree, then exactly the regular tree languages are generated (Theorem 19).

We will briefly sketch the ideas behind our proofs. CCGs without composition operations can generate exactly those regular tree languages where for each node there exists a short path of bounded length to a leaf. Intuitively, application rules shorten a category on the way from the leaf upwards in the direction of the root. This is why the maximal category length in the lexicon, which contains the categories labeling leaves, puts a bound on this path length. In our construction, we consider decompositions of trees into these short paths and compile a lexicon containing categories modeling these paths, that can then be assembled appropriately through the CCG operations. As for the result on CCGs allowing composition rules of first degree, we construct a CCG that uses a certain finite set of categories, such that given a set of states of a tree automaton, for each transition that could be present in a tree automaton using these, there are corresponding categories that could be combined via rules. The rule restrictions of the CCG then control which of these rules are actually permitted, with the aim to simulate only valid transitions of the given tree automaton. Finally, for the main result, given a CCG allowing composition rules of arbitrary degree, we construct an sCFTG that generates the *rule trees* of the CCG. These represent its derivation trees and have the same shape, while using only a finite set of symbols. The nonterminals of the sCFTG represent either categories or CCG rules. The derivation takes place by gradually extending what we call a *spine* and by using branching productions to start new spines that are attached to superordinated spines. Nonterminals can only be replaced by terminal symbols when they represent either categories present in the lexicon or rules permitted by the CCG rule system.

2. Preliminaries

We denote the set of nonnegative integers by $\mathbb{N}$ and for every $k \in \mathbb{N}$ let $[k] = \{i \in \mathbb{N} \mid 1 \leq i \leq k\}$ as well as $\mathbb{Z}_k = \{i \in \mathbb{N} \mid i < k\}$. 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 (possibly infinite) 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 $\mathcal{L} \cdot \mathcal{L}' = \{w \cdot w' \mid w \in \mathcal{L},\ w' \in \mathcal{L}'\}$. We assume familiarity with the context-free languages [26] and the concept of context-free grammar (CFG) that can be used to describe them. 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.[3] 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 $a \in \Sigma_0$, (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 readability. Every subset $\mathcal{F} \subseteq T_{\Sigma_2,\Sigma_1}(\Sigma_0)$ is a *tree language*. The mapping

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

pos: $T_{\Sigma_2,\Sigma_1}(\Sigma_0) \to \mathcal{P}_+([2]^*)$ assigning positions to a tree is defined by (i) $\mathrm{pos}(a) = \{\varepsilon\}$ for all $a \in \Sigma_0$, (ii) $\mathrm{pos}(n(t)) = \{\varepsilon\} \cup \{1w \mid w \in \mathrm{pos}(t)\}$ 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 $t_1, t_2 \in T_{\Sigma_2,\Sigma_1}(\Sigma_0)$,

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

We let $\mathrm{leaves}(t) = \{w \in \mathrm{pos}(t) \mid w1 \notin \mathrm{pos}(t)\}$ 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. Let the *yield* of a tree be given by $\mathrm{yield}\colon T_{\Sigma_2,\Sigma_1}(\Sigma_0) \to \Sigma_0^+$ with $\mathrm{yield}(a) = a$ for all $a \in \Sigma_0$ and $\mathrm{yield}\big(c(t_1,\dots,t_k)\big) = \mathrm{yield}(t_1)\cdots\mathrm{yield}(t_k)$ for all $k \in [2]$, $c \in \Sigma_k$, and $t_1,\dots,t_k \in T_{\Sigma_2,\Sigma_1}(\Sigma_0)$. 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 $w = \mathrm{pos}_\square(C)$. A *relabeling* is a mapping $\rho\colon (\Sigma_2 \cup \Sigma_1 \cup \Sigma_0) \to \mathcal{P}_+(\Delta)$ for some alphabet Δ.[4] 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

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

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* (sCFTG) [20,21] is a tuple $G = (N, \Sigma, I, P)$ 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

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

The grammar is called *monadic* because there are only nullary and unary nonterminals; *simple* means that the productions are linear and nondeleting, i.e., if the left side of a production is in N_1, the special symbol $\square$ (marking the new position of the subtree of the nonterminal) appears 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 [27] 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 $\mathcal{T}(G)$ generated by G is $\mathcal{T}(G) = \{t \in T_{\Sigma_2,\emptyset}(\Sigma_0) \mid \exists n_0 \in I\colon n_0 \Rightarrow_G^+ t\}$, where $\Rightarrow_G^+$ is the transitive closure of $\Rightarrow_G$. The tree languages generated by sCFTGs form a subset of context-free tree languages, and a tree language $\mathcal{T}$ is *regular* if and only if there exists an RTG G such that $\mathcal{T} = \mathcal{T}(G)$. A detailed introduction to trees and tree languages can be found in [17]. The string language generated by G is $\mathcal{L}(G) = \{\mathrm{yield}(t) \mid t \in \mathcal{T}(G)\}$.

Example 1. The RTG $G_1 = (N, \Sigma, I, P)$ with $N = N_0 = I = \{s\}$, $\Sigma = \Sigma_2 \cup \Sigma_0$ with $\Sigma_2 = \{\sigma\}$ and $\Sigma_0 = \{\alpha, \beta\}$, and $P = \{s \to \sigma(\alpha, \sigma(s, \beta)),\ s \to \sigma(\alpha, \beta)\}$ generates the string language $\mathcal{L}(G_1) = \{\alpha^n\beta^n \mid n \geq 1\}$. Clearly, the only nonterminal s is nullary (since G_1 is an RTG) and thus occurs only as leaf in the right-hand sides of productions, which is similar to right-linearity for CFGs (Fig. 1).

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

Example 2. The sCFTG $G_2 = (N, \Sigma, I, P)$ with $N = N_1 \cup N_0$ with $N_1 = \{n\}$ and $N_0 = I = \{s\}$, $\Sigma = \Sigma_2 \cup \Sigma_0$ with $\Sigma_2 = \{\sigma\}$ and $\Sigma_0 = \{\alpha, \beta, \gamma\}$, and

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

generates the string language $\mathcal{L}(G_2) = \{\alpha^n\beta^n\gamma^n \mid n \geq 1\}$. The placeholder $\square$, which indicates the new position of the subtree under the unary nonterminal symbol n, appears exactly once on the right-hand sides of the productions with left-hand side n (as required for an sCFTG) (Fig. 1).

[4] We require that each input symbol can be relabeled.

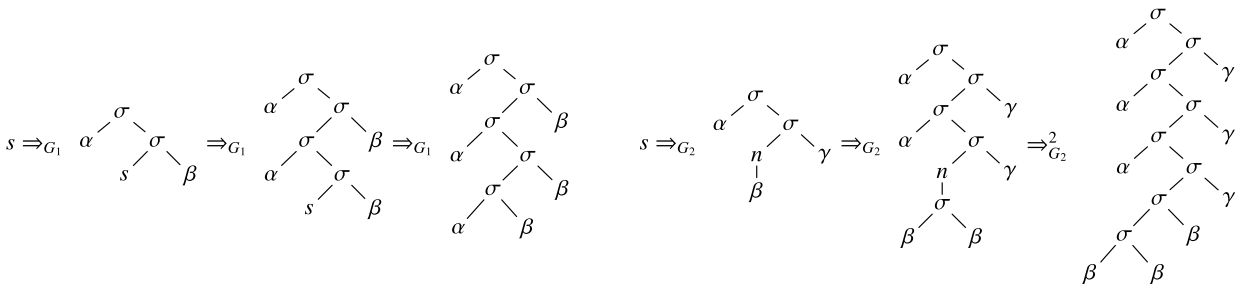

Fig. 1. Derivations using the RTG G_1 (left) and the sCFTG G_2 (right) of Examples 1 and 2, respectively.

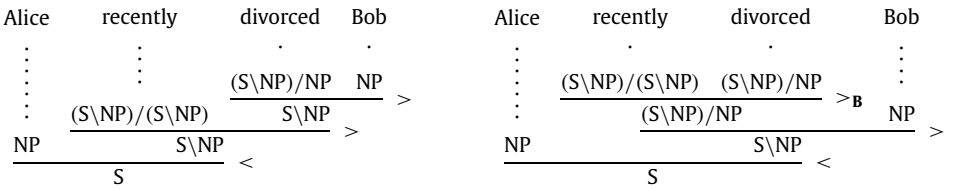

Fig. 2. Two sample derivations.

3. Combinatory categorial grammars

Combinatory categorial grammars (CCGs) extend the classical categorial grammars of Ajdukiewicz and Bar-Hillel [28] (AB-grammars) by rules inspired by combinatory logic [24]. Here, as in most of the formal work on CCGs, we restrict our attention to the rules of application and composition, which are based on the **B**-combinator.

3.1. Informal introduction

Like other categorial grammars, a CCG grammar has two components: a *lexicon*, which specifies the syntactic types or *categories* of individual words; and a set of *rules*, which specify how to derive the categories of phrases and sentences from the categories of their constituent parts. These two components together give rise to *derivations* such as the ones shown in Fig. 2. We draw derivation trees according to the standard conventions for CCGs, where the root is at the bottom. The dotted lines visualize the word–category mapping implemented by the lexicon. Categories can either be primitive, such as S ("sentence") or NP ("noun phrase"), or complex, such as (S\NP)/NP or S\NP. These complex categories denote function types. More specifically, an object with a complex category of the general form X/Y or $X\backslash Y$ takes an argument of category Y and returns an object of category X. When two categories are combined, the category that takes an argument is called *primary input category* and the one that provides said argument is called *secondary input category.* The forward slash specifies that the argument should appear to the right; the backward slash specifies that it should appear to the left. In Fig. 2, the left derivation combines categories using rules of functional application, as in AB-grammars: forward application (denoted by $>$) and backward application (denoted by $<$). The right derivation showcases forward composition (denoted by $>_{\mathbf{B}}$), where the category (S\NP)/(S\NP) for the lexical item *recently* combines with the category (S\NP)/NP for *divorced*. As in function composition, the argument /NP of the secondary input category gets appended to that part of the primary input category which remains when the provided argument /(S\NP) has been removed, resulting in the new category (S\NP)/NP. This category can then be combined with NP to the right. Higher-order versions of composition are able to transfer more than one outermost argument of the secondary input category to another category.

3.2. Formal definition

In this article, we formalize categories as term trees. Let A be an alphabet, and let $C(A) = T_{S,\emptyset}(A)$, where $S = \{/, \backslash\}$ is the set of slashes. The elements of $C(A)$ are *categories* (over A), of which the elements of $A \subseteq C(A)$ 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 $c_i \in C(A)$ for all $i \in [k]$. The atomic category a is called the *target* of c and the slash–category pairs $|_i c_i$ are called the *arguments* of c. Let $\mathrm{argcats}(c) = \{c_i \mid i \in [k]\}$ be the set of all categories used as arguments of category c. If $\mathrm{argcats}(c) \subseteq A$, we call c a *first-order category*. The set of all first-order categories over A is denoted by $C_0(A)$. The number k is called the *arity* of c. Note that, from the tree perspective, the sequence of arguments is a context $\alpha = \Box|_1 c_1 \cdots |_k c_k$. 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 $C(A,k) = \{c \in C(A) \mid \mathrm{arity}(c) \leq k\}$ and $\mathcal{A}(A,k) = \{\alpha \in \mathcal{A}(A) \mid k \geq |\alpha|\}$.

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 [9]:

$$\text{forward rule:} \quad \frac{ax/c \quad c|_1c_1 \cdots |_kc_k}{ax|_1c_1 \cdots |_kc_k} \qquad \text{backward rule:} \quad \frac{c|_1c_1 \cdots |_kc_k \quad ax\backslash c}{ax|_1c_1 \cdots |_kc_k}$$

where $a \in A$, $c \in C(A) \cup \{y\}$, and $|_i \in S$ and $c_i \in C(A) \cup \{y_i\}$ for every $i \in [k]$. The category variables $y, y_1, \ldots, y_k$ can match any category of $C(A)$, while the argument context variable x can match any argument context of $\mathcal{A}(A)$. Note that the variable x occurs in every rule in the category $ax|c$ with $| \in \{/, \backslash\}$, which is called the *primary input category*. The other category $c|_1c_1 \cdots |_kc_k$ is the *secondary input category* of the rule, and the category variables $y, y_1, \ldots, y_k$ may, but need not, occur in it. Each category $c, c_1, \ldots, c_k$ can thus be either a concrete category of $C(A)$ or a category variable if no restriction is intended. Indeed, the category variables $y, y_1, \ldots, y_k$ only offer a more succinct rule description in the notion of combinatory categorial grammar that we consider since only the finitely many instantiations of categories that occur in the lexicon will ever yield useful rules (see Proposition 12). Since we concern ourselves exclusively with the expressive power, we often assume, without loss of generality, that all secondary input categories of rules are concrete categories of $C(A)$; i.e., the rules contain no category variables. The argument context variable x will match each argument context of $\mathcal{A}(A)$, so in the primary input category we can only restrict the target and the last argument. 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 $\frac{\cdot \quad \cdot}{\cdot}\Pi \subseteq C(A)^2 \times C(A)$, which extends to a relation $\Rightarrow_\Pi \subseteq C(A)^* \times C(A)^*$ by $\Rightarrow_\Pi = \bigcup_{\varphi,\psi \in C(A)^*} \{(\varphi\, c\, c'\, \psi,\ \varphi\, c''\, \psi) \mid \frac{c \quad c'}{c''}\Pi\}$.

Example 3. Consider the forward rule $r = \frac{Dx/D \quad D/E\backslash C}{Dx/E\backslash C}$, where $\{C, D, E\}$ are atoms. A possible ground instance of this rule is $\frac{D/C/E/D \quad D/E\backslash C}{D/C/E/E\backslash C}$, 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 $/D$. As c_1 takes three atomic categories as arguments, it is a first-order category and $arity(c_1) = 3$. 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 latter is a higher-order category. The rules $\frac{Dx/(E/D) \quad E/D\backslash C}{Dx\backslash C}$ and $\frac{Dx/(E/D) \quad E/D\backslash(C/C)}{Dx\backslash(C/C)}$ both have degree 1.

Definition 4 *([9]).* 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 $L \subseteq \Sigma \times C(A)$ 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 $R = \mathcal{R}(A, k)$].

In a pure k-CCG the rule system contains all rules up to degree k — without exception. Thus, as long as the degree limit is respected, all instances of forward and backward rules can be applied.

Example 5. The classical categorial grammars of Ajdukiewicz and Bar-Hillel [28] (AB-grammars) are 0-CCGs. However, 0-CCGs are more general since they allow rule restrictions, whereas AB-grammars are 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 $I = \{C\}$ of initial categories, and the lexicon L with $L(c) = \{C/D,\ C/D/C\}$ and $L(d) = \{D\}$. Clearly, G_3 is a 0-CCG. For a slightly more involved example containing rule restrictions, we refer to Example 21.

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

$$C(G) = \{\varphi \in C(A)^* \mid \exists a_0 \in I\colon \varphi \Rightarrow_\Pi^* a_0\} \qquad \text{and} \qquad \mathcal{L}(G) = L^{-1}(C(G)) \ .$$

A tree $t \in T_{C(A),\emptyset}(L(\Sigma))$ is a *derivation tree of G* if $\frac{t(w1) \quad t(w2)}{t(w)}\Pi$ for every $w \in \mathrm{pos}(t) \setminus \mathrm{leaves}(t)$. The set of all such trees is denoted by $\mathcal{D}(G)$. Note that it is not required that $t(\varepsilon) \in I$.

The grammar of Example 5 generates $\mathcal{L}(G_3) = \{c^i d^i \mid i \geq 1\}$, which is context-free but not regular. A derivation tree for the string *ccdd* is shown in Fig. 3. Note that in this and the following derivations, we do no longer explicitly label the rule instances with their names ("forward application", "backward composition", etc.), as we did in Fig. 2. Overall, G_3 generates the category sequences $C(G_3) = \{(C/D/C)^{i-1} \cdot (C/D) \cdot D^i \mid i \geq 1\}$.

The string language generated by a CCG is obtained by relabeling the leaf categories of the derivation trees using the lexicon. For the generated tree language we similarly allow a relabeling to avoid the restriction to the particular symbols of $C(A)$. However, since the set $C(A)$ is infinite, we restrict the possible relabelings such that they only depend on the target and the last argument of a given category.

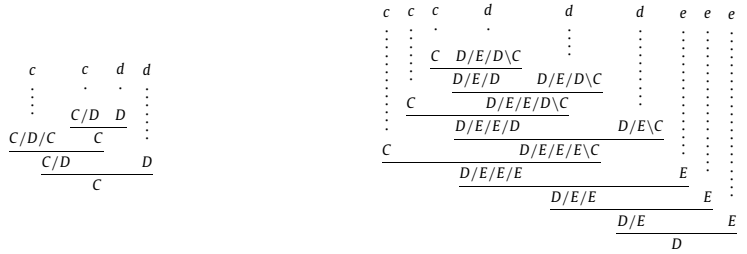

Fig. 3. Derivations using the AB-grammar G_3 (left) and the CCG G_4 (right) of Examples 5 and 21, respectively.

Definition 7. Let $G = (\Sigma, A, R, I, L)$ be a CCG and $\rho\colon C(A) \to \mathcal{P}_+(\Delta)$ a mapping. If $\rho(b) = \rho(b')$ for all atoms $a \in A$, slashes $| \in S$, categories $c \in C(A)$, and ground instances b and b' of $ax|c$, then ρ is called *category relabeling*. Together, G and ρ *generate* the tree language $\mathcal{F}_\rho(G) = \bigcup_{d \in \mathcal{D}(G),\, d(\varepsilon) \in I} \rho(d)$. A tree language $\mathcal{F} \subseteq T_{\Delta,\emptyset}(\Delta)$ is *generatable by* the CCG G if there exists a category relabeling $\rho'\colon C(A) \to \mathcal{P}_+(\Delta)$ such that $\mathcal{F} = \mathcal{F}_{\rho'}(G)$.

Because $L(\Sigma)$ is finite, there exists $k \in \mathbb{N}$ such that $L(\Sigma) \subseteq C(A, k)$. The least such integer k is called the *arity of* L and denoted by $\mathrm{arity}(L)$; i.e., $\mathrm{arity}(L) = \max\{\mathrm{arity}(c) \mid c \in L(\Sigma)\}$. If $L = \emptyset$, then we let $\mathrm{arity}(L) = 0$. Further, let $\mathrm{argcats}(L) = \bigcup_{c \in L(\Sigma)} \mathrm{argcats}(c)$ be the set of all categories used as arguments of lexical entries.

4. 0-CCGs

In this section we characterize the tree languages generatable by 0-CCGs. This has already been investigated by Buszkowski with a focus on classical categorial grammars [25, Theorem 1.1]. We present an alternative proof for this result. 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 $\mathrm{arity}(L)$. Thus, the derivation trees are built over a finite set of symbols.

Theorem 8 *(see [28] and [22, Proposition 3.25]). The string languages generated 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 tree languages generatable by* G *are regular.*

Proof. It is rather easy to show that every 0-CCG generates a context-free language.[5] It is considerably more complicated to show that every ε-free context-free language can be generated by some 0-CCG. A Greibach-like normal form for context-free grammars was developed by [28] for this purpose. By [22, Proposition 3.25] the tree language $\mathcal{D}(G)$ is regular. Moreover, the regular tree languages are closed under relabelings [19, Theorem 2.4.16] and intersection [19, Theorem 2.4.2],[6] so also $\mathcal{F}_\rho(G)$ is regular for every category relabeling ρ. □

To characterize the tree languages generatable by 0-CCGs, we need to introduce an additional structural property of the derivation tree language $\mathcal{D}(G)$ and the generatable tree languages. Roughly speaking, the *min-height* $\mathrm{mht}(t)$ 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 $\mathrm{mht}\colon T_{\Sigma_2,\emptyset}(\Sigma_0) \to \mathbb{N}$ be such that $\mathrm{mht}(a) = 0$ and

$$\mathrm{mht}\big(c(t_1, t_2)\big) = 1 + \min\big(\mathrm{mht}(t_1), \mathrm{mht}(t_2)\big)$$

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 $\mathrm{mht}(t|_w) \leq h$ for every $w \in \mathrm{pos}(t)$. 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 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. The property 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 $\rho\colon (\Sigma_2 \cup \Sigma_0) \to \mathcal{P}_+(\Delta)$.

Example 9. Let us reconsider the 0-CCG G_3 of Example 5. The derivation tree language $\mathcal{D}(G_3)$ is universally mht-bounded by 1 (see Fig. 3). The tree $\gamma(\alpha, \gamma(\gamma(\alpha, \alpha), \gamma(\alpha, \alpha)))$ also has min-height 1, but is only universally mht-bounded by 2, since the subtree $\gamma(\gamma(\alpha, \alpha), \gamma(\alpha, \alpha))$ has min-height 2.

[5] This also follows from the following fact about its derivation tree language.
[6] The intersection can be used to restrict $\mathcal{D}(G)$ to those trees whose root symbol belongs to I.

It turns out that exactly the universally mht-bounded regular tree languages are generatable by 0-CCGs. We already observed that the tree languages generatable by 0-CCGs are regular, but for the converse we have to exploit the universal mht-bound. We first show that indeed the tree languages generatable by 0-CCGs are universally mht-bounded.

Lemma 10. *The tree language $\mathcal{D}(G)$ is universally* mht*-bounded by* arity(L).

Proof. We first prove that $\mathrm{mht}(t) \leq \mathrm{arity}(L) - \mathrm{arity}(t(\varepsilon))$ for every $t \in \mathcal{D}(G)$ by induction on t. For $t = c \in L(\Sigma)$, we have $\mathrm{arity}(c) \leq \mathrm{arity}(L)$, so we obtain $\mathrm{mht}(c) = 0 \leq \mathrm{arity}(L) - \mathrm{arity}(c)$, which completes the induction base. In the induction step, let $t = c(t_1, t_2)$ with $c \in C(A)$ and $t_1, t_2 \in \mathcal{D}(G)$. Since $R \subseteq \mathcal{R}(A, 0)$ we can only use application rules. Hence, $\mathrm{arity}(c) \leq \max(\mathrm{arity}(t(1)), \mathrm{arity}(t(2))) - 1$, which we call (†). By the induction hypothesis (IH) we have $\mathrm{mht}(t_1) \leq \mathrm{arity}(L) - \mathrm{arity}(t(1))$ and $\mathrm{mht}(t_2) \leq \mathrm{arity}(L) - \mathrm{arity}(t(2))$. Thus, we obtain

$$\begin{aligned}
\mathrm{mht}\big(c(t_1,t_2)\big) &= 1 + \min\Big(\mathrm{mht}(t_1), \mathrm{mht}(t_2)\Big) && \overset{\text{(IH)}}{\leq} 1 + \min\Big(\mathrm{arity}(L) - \mathrm{arity}\big(t(1)\big), \mathrm{arity}(L) - \mathrm{arity}\big(t(2)\big)\Big) \\
&= 1 + \mathrm{arity}(L) - \max\Big(\mathrm{arity}\big(t(1)\big), \mathrm{arity}\big(t(2)\big)\Big) && \overset{(\dagger)}{\leq} 1 + \mathrm{arity}(L) - \big(\mathrm{arity}(c) + 1\big) \\
&= \mathrm{arity}(L) - \mathrm{arity}\big(t(\varepsilon)\big)
\end{aligned}$$

as required. This completes the induction. Now let $t \in \mathcal{D}(G)$ and $w \in \mathrm{pos}(t)$. Then $t|_w \in \mathcal{D}(G)$ is a derivation tree, and thus $\mathrm{mht}(t|_w) \leq \mathrm{arity}(L)$ by the auxiliary statement, which proves that t is universally mht-bounded by $\mathrm{arity}(L)$. □

Since the universal mht-bound is a structural property, we can transfer it from the derivation trees to the relabeled trees. The following corollary is a direct consequence of Lemma 10.

Corollary 11. *There exist regular tree languages $\mathcal{F} \subseteq T_{\Sigma_2,\emptyset}(\Sigma_0)$ that are not generatable by any* 0*-CCG.*

Proof. The tree language $\mathcal{F} = T_{\Sigma_2,\emptyset}(\Sigma_0)$ for non-empty alphabets Σ_2 and Σ_0 is not generatable by any 0-CCG since it is not universally mht-bounded. □

We have thus established that the tree languages generated by 0-CCGs are regular and universally mht-bounded. We note that this result does not concern weak generative capacity. In particular, every (binary) regular tree language can be converted into a universally mht-bounded one that yields the same strings; this implies that a formalism that is able to generate all universally mht-bounded regular tree languages will still be weakly equivalent to the full class of regular tree languages, and therefore, to context-free (string) grammars.

The remainder of this section will be devoted to the second half of the proof of Theorem 8. More precisely, we will show that every universally mht-bounded regular tree language can be generated by a 0-CCG. The construction uses the universal mht-bound, which yields short paths to a leaf. We utilize those paths 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 input 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. This idea is illustrated in Fig. 4. Consider for example the rightmost run, which is r_3. In the CCG derivation tree, the leaf at the bottom of this run is a primary input category that has two arguments, which store besides r_3 the labels of nodes on the spine and their siblings. Following the spine upwards, these arguments are removed through application rules, until only the target remains. It stores r_1, which is the run that r_3 gets combined with.

We restate the following proposition [9, Lemma 3.1], which will be needed in the proof of Theorem 13.

Proposition 12 *(see [9, Lemma 3.1]). Let $\Pi = (A, R)$ be a rule system, and let*

$$c_1 \cdots c_k \Rightarrow_\Pi^* c'_1 \cdots c'_{k'}$$

for some categories $c_1, \ldots, c_k, c'_1, \ldots, c'_{k'} \in C(A)$. Then $k' \leq k$ and for each $i \in [k']$ the category c'_i is of the form $c'_i = a_i|_1 c''_1 \cdots |_\ell c''_\ell$, where a is the target of c_j for some $j \in [k]$ and for each $m \in [\ell]$ the argument $|_m c''_m$ is an argument of the category c_{j_m} for some $j_m \in [k]$.

Theorem 13. *Let $\mathcal{F} \subseteq T_{\Sigma_2,\emptyset}(\Sigma_0)$ be a tree language. Then the following are equivalent:*

- *$\mathcal{F}$ is generatable by some* 0*-CCG.*
- *$\mathcal{F}$ is generatable by some pure* 0*-CCG.*
- *$\mathcal{F}$ is regular and universally* mht*-bounded.*

Proof. The inclusion of the tree languages generatable by 0-CCG in the regular and universally mht-bounded tree languages is trivially true by Theorem 8 and Lemma 10. The step from the second to the first statement is also immediately apparent

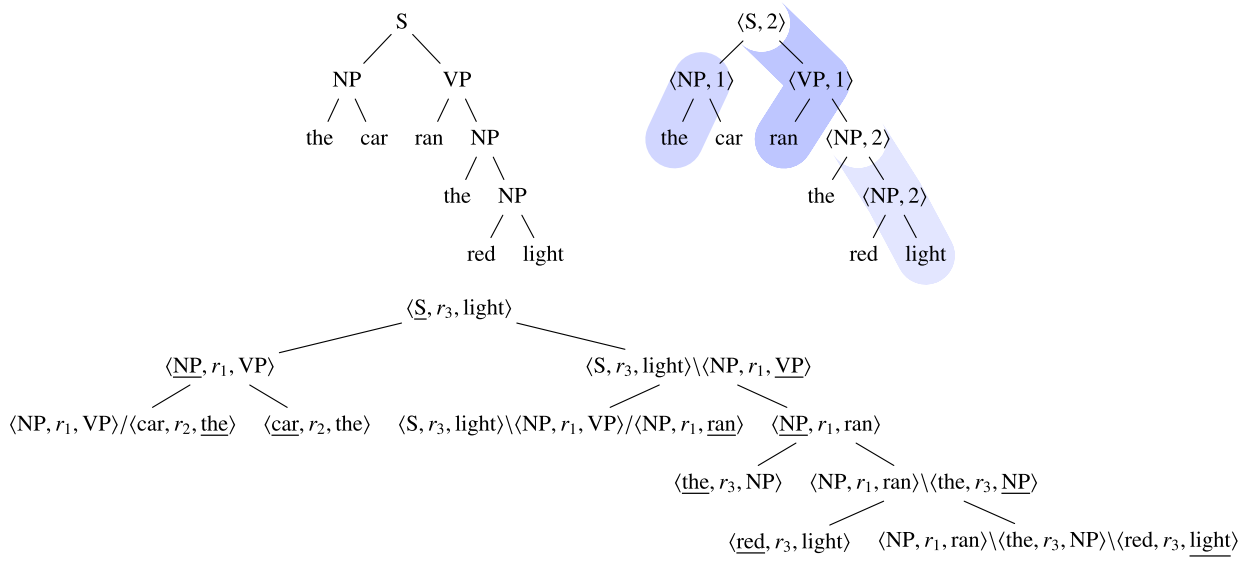

Fig. 4. Decomposition into spinal runs and the corresponding derivation tree of the 0-CCG with the symbol resulting after relabeling underlined.

since each pure 0-CCG is a 0-CCG. In the following, we will show the remaining step: that each regular, universally mht-bounded tree language $\mathcal{F}$ is generated by some pure 0-CCG.

Let $h \in \mathbb{N}$ be such that $\mathcal{F}$ is universally mht-bounded by h. Since $\mathcal{F}$ is regular, there exists a context-free grammar $G = (N, \Gamma, I, P)$ and a relabeling ρ [19, Theorem 2.9.5] such that $\mathcal{F} = \{\rho(t) \mid t \in \mathcal{D}(G),\ t(\varepsilon) \in I\}$, where $\mathcal{D}(G)$ consists of the usual derivation trees of the CFG G. Without loss of generality, we can assume that N does not contain useless nonterminals.[7] Since the composition of two relabelings (seen as relations) is again a relabeling, it is sufficient to prove that there exists a pure 0-CCG G' and a relabeling ρ' such that $\mathcal{F}_{\rho'}(G') = \{t \in \mathcal{D}(G) \mid t(\varepsilon) \in I\}$. We observe that $\mathcal{D}(G)$ is also universally mht-bounded by h.

First we will annotate the derivation trees with directions, where 1 indicates left and 2 indicates right. For every $r \in T_{N\times[2],\emptyset}(\Gamma \cup N)$ we let $r[0] = r$ if $r \in \Gamma \cup N$ and $r[0] = n$ if $r = \langle n, \delta\rangle(r_1, r_2)$ for some $n \in N$, $\delta \in [2]$, and $r_1, r_2 \in T_{N\times[2],\emptyset}(\Gamma \cup N)$. The set $\mathrm{SR}(G)$ of *spinal runs of* G is the smallest set $R \subseteq T_{N\times[2],\emptyset}(\Gamma \cup N)$ such that (i) $\Gamma \subseteq R$, and (ii) $\langle n, \delta\rangle(r_1, r_2) \in R$ for every $n \in N$, $\delta \in [2]$, $r_{3-\delta} \in \Gamma \cup N$, and $r_\delta \in R$ with $(n \to r_1[0] \cdot r_2[0]) \in P$. The direction $\delta \in [2]$ indicates which successor continues the spine. Since $\mathcal{D}(G)$ is universally mht-bounded by h, we are interested only in those spines of length at most h, so we let $\mathrm{SR} = \{r \in \mathrm{SR}(G) \mid \mathrm{ht}(r) \le h\}$. Clearly, SR is finite, so we can build the atomic categories $A = (\Gamma \cup N) \times \mathrm{SR} \times (\Gamma \cup N)$. Each atomic category thus stores two symbols, which are required for the relabeling, and a spinal run to enforce consistency. Next, we construct the argument tree $\arg(r', r'')$, which is an element of $\mathcal{A}(A, h)$, and define additional direct spine access $r'[i] \in \Gamma \cup N$ for all spinal runs $r', r'' \in \mathrm{SR}$, and $i \in [\mathrm{ht}(r')]$ as follows:

- if $r' \in \Gamma$, then $\arg(r', r'') = \square$,
- if $r' = \langle n, \delta\rangle(r_1, r_2)$ for some $n \in N$, $\delta \in [2]$, $r_{3-\delta} \in \Gamma \cup N$, and $r_\delta \in \mathrm{SR}$, then[8]

$$\arg(r', r'') = \arg(r_\delta, r'')\Big[\square \mid \langle r_{3-\delta}, r'', r_\delta[0]\rangle\Big] \qquad \text{and} \qquad r'[i] = r_\delta[i-1]\ ,$$

  where $\mid = /$ if $\delta = 1$ and $\mid = \backslash$ otherwise.

The base "$\mathrm{base}(r')$" of r' is simply $r'[\mathrm{ht}(r')]$. Note that $\mathrm{base}(r') \in \Gamma$. The notion of spinal runs and the construction of the argument tree are illustrated in Fig. 5.

Next, we construct the pure 0-CCG $G' = (\Gamma, A, \mathcal{R}(A, 0), I \times \mathrm{SR} \times (\Gamma \cup N), L)$ with

$$L(\gamma) = \Big\{\arg(r, r)\Big[\langle r[0], r', g\rangle\Big] \mid r, r' \in \mathrm{SR},\ \mathrm{base}(r) = \gamma,\ g \in \Gamma \cup N\Big\}$$

for every $\gamma \in \Gamma$. It is clear that all categories of $L(\Gamma)$ are left-spinal (i.e., all right children are leaves — see Fig. 5). Moreover, all right children are atomic categories. Together with the fact that we can only use application rules, we obtain that all categories that can occur in derivations of $\mathcal{D}(G')$ must be subtrees of the categories in $L(\Gamma)$. Consequently, let $C \subseteq C(A)$ be

[7] We require that for every nonterminal $n \in N$ there exist strings $w, w_1, w_2 \in \Gamma^*$ and an initial nonterminal $n_0 \in I$ such that $n_0 \Rightarrow_G^* w_1 n w_2 \Rightarrow_G^* w$.

[8] For better readability, we write $\mid(\square, c)$ using the infix notation $\square \mid c$.

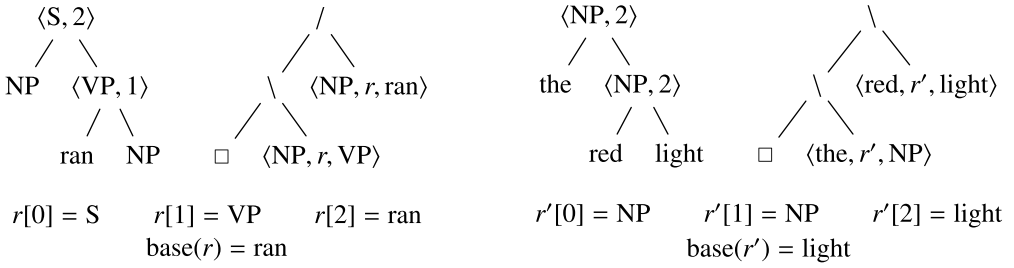

Fig. 5. Two spinal runs r (left) and r' (right) together with their argument trees $\arg(r, r)$ and $\arg(r', r')$ and their bases $\mathrm{base}(r)$ and $\mathrm{base}(r')$.

that subset of all categories. Moreover, if the right child of a category $c \in C$ is decorated with the spinal run r, then the category c is a subtree of a category built from $\arg(r, r)$ in the definition of L. Finally, we need to construct the missing category relabeling ρ'. To avoid complication, we only define it on the categories of C, since its definition on the remaining categories is irrelevant as those other categories cannot occur in derivations of $\mathcal{D}(G')$. Let $\rho' \colon C \to \mathcal{P}_+(\Gamma \cup N)$ be given by $\rho'(\langle g, r, g' \rangle) = \{g\}$ and $\rho'(c \mid \langle g, r, g' \rangle) = \{g'\}$ for all $\langle g, r, g' \rangle \in A \cap C$, $c \in C$, and $| \in S$. Note that ρ is trivially a category relabeling.

It remains to prove that $\mathcal{F}_{\rho'}(G') = \{t \in \mathcal{D}(G) \mid t(\varepsilon) \in I\}$. We first prove the auxiliary statement $\rho'(\mathcal{D}(G')) = \mathcal{D}(G)$.

We start with the direction $\rho'(\mathcal{D}(G')) \subseteq \mathcal{D}(G)$ using induction. In the induction base we have $c \in L(\Gamma)$. Now we distinguish two cases: If $c = \langle g, r', g' \rangle \in A$ is atomic, then $\rho'(c) = g = \mathrm{base}(r) \in \Gamma$ for some $r \in \mathrm{SR}$ by construction of the lexicon. Otherwise $c = c' \mid \langle g, r', g' \rangle$ for some $c' \in C$, $| \in S$, and $\langle g, r', g' \rangle \in A$. Moreover, c was obtained from the argument tree $\arg(r, r)$ for some $r \in \mathrm{SR}$ by substitution. In this case $\rho'(c) = g' = \mathrm{base}(r) \in \Gamma$ by the definition of "arg". Consequently, we have $\rho'(c) \in \Gamma$ and $\Gamma \subseteq \mathcal{D}(G)$, which completes the induction base. In the induction step, let $d = g(d_1, d_2)$ for some $g \in \Gamma \cup N$ and $d_1, d_2 \in \rho'(\mathcal{D}(G'))$. Consequently, there exist $c \in C$ and $t_1, t_2 \in \mathcal{D}(G')$ such that $c(t_1, t_2) \in \mathcal{D}(G')$ and $\rho'(c(t_1, t_2)) = g(d_1, d_2)$. Moreover, $d_1, d_2 \in \mathcal{D}(G)$ by the induction hypothesis. It remains to prove that $g \to d_1(\varepsilon) \cdot d_2(\varepsilon) \in P$, and thus $g(d_1, d_2) \in \mathcal{D}(G)$. We already remarked that only left-spinal categories can occur in $\mathcal{D}(G')$, hence $\{t_1(\varepsilon), t_2(\varepsilon)\} = \{c|a, a\}$ for some $| \in S$ and $a \in A$. Moreover, let $a = \langle g_1, r, g_2 \rangle$ for some $g_1, g_2 \in \Gamma \cup N$ and $r \in \mathrm{SR}$. We assume that $t_1(\varepsilon) = a$ and $t_2(\varepsilon) = c\backslash a$. The remaining case, in which $t_1(\varepsilon) = c/a$ and $t_2(\varepsilon) = a$, is analogous. By the definition of ρ', we obtain that $d_1(\varepsilon) = g_1$ and $d_2(\varepsilon) = g_2$. Moreover, we already remarked that $c\backslash a$ must be a subtree of a category in $L(\Gamma)$. More precisely, it must be a subtree of the category $\arg(r, r)[\langle r[0], r', \overline{g} \rangle]$ for some $r' \in \mathrm{SR}$ and $\overline{g} \in \Gamma \cup N$ because we have the spinal run r annotated to a right child. Clearly, the left-spinal property makes it easy for us to locate the required subtree.

Now we distinguish two cases according to the definition of ρ'. If c is atomic, then $c = \langle r[0], r', \overline{g} \rangle$ and $r[0] = g$ by the definition of ρ'. By the construction of the argument "$\arg(r, r)$" we have

$$r(\varepsilon) = \langle r[0], 2 \rangle = \langle g, 2 \rangle \qquad \text{and} \qquad r(1) = g_1 = d_1(\varepsilon) \qquad \text{and} \qquad r(2) = \langle r[1], \delta \rangle = \langle g_2, \delta \rangle = \langle d_2(\varepsilon), \delta \rangle$$

for some $\delta \in [2]$. Since $r \in \mathrm{SR}$ we have $r[0] \to r(1) \cdot r[1] \in P$, which yields $g \to d_1(\varepsilon) \cdot d_2(\varepsilon) \in P$ as desired with the help of the equations above. In the remaining case c is not atomic. Let $c = c' \mid \langle g', r, g'' \rangle$ for some $c' \in C$, $| \in S$, and $g', g'' \in \Gamma \cup N$. The definition of ρ' yields that $g'' = g$. Since 'arg' reverses the order (see Fig. 5), our subtree $c\backslash a$ corresponds to an initial fragment of r. Thus, let $r = C[r'']$ with $C \in C_{N \times [2], \emptyset}(\Gamma \cup N)$ and $r'' \in \mathrm{SR}$ such that $c\backslash a = \arg(C[r''(\varepsilon)], r)[\langle r[0], r', \overline{g} \rangle]$. Let $w = \mathrm{pos}_{\square}(C)$. Since we have at least two arguments in $c\backslash a$, the definition of 'arg' yields $|w| \geq 2$, so let $w = w'\delta_1'\delta_2'$ with $w' \in [2]^*$ and $\delta_1', \delta_2' \in [2]$. Then the last two arguments are constructed by $\square \mid \langle g', r, g'' \rangle \setminus \langle g_1, r, g_2 \rangle = \arg(C|_{w'}[r''(\varepsilon)], r)$ and thus

$$r(w'\delta_1') = \langle g'', \delta_2' \rangle = \langle g, 2 \rangle \qquad \text{and} \qquad r(w'\delta_1' 1) = g_1 = d_1(\varepsilon) \qquad \text{and} \qquad r(w'\delta_1' 2) = \langle g_2, \delta'' \rangle = \langle d_2(\varepsilon), \delta'' \rangle$$

for some $\delta'' \in [2]$. Since $r \in \mathrm{SR}$ we have $r|_{w'\delta_1'}[0] \to r|_{w'\delta_1'}(1) \cdot r|_{w'\delta_1'}[1] \in P$, which together with the equalities above yields the existence of the production $g \to d_1(\varepsilon) \cdot d_2(\varepsilon) \in P$ as required. Hence $\rho'(\mathcal{D}(G')) \subseteq \mathcal{D}(G)$.

For the converse inclusion $\mathcal{D}(G) \subseteq \rho'(\mathcal{D}(G'))$ we first prove an auxiliary statement. Let $t \in \mathcal{D}(G')$ be a derivation with $\mathrm{arity}(t(\varepsilon)) = 0$ (i.e., that terminates in an atomic category). Moreover, let $t(\varepsilon) = \langle g, r, \overline{g} \rangle$ for some $g, \overline{g} \in \Gamma \cup N$ and $r \in \mathrm{SR}$. Then for every $r' \in \mathrm{SR}$ and $g' \in \Gamma \cup N$ there exists a derivation $t_{r', g'} \in \mathcal{D}(G')$ such that $t_{r', g'}(\varepsilon) = \langle g, r', g' \rangle$ and $\rho'(t_{r', g'}) = \rho'(t)$. In other words, in any derivation with an atomic category at the root we can adjust the derivation such that the root label contains any desired spinal run $r' \in \mathrm{SR}$ and third component $g' \in \Gamma \cup N$. The resulting tree is still a derivation and relabels to the same tree as t. This statement is very easy to prove using Proposition 12, which shows that $t(\varepsilon)$ is the target of a category of $L(\Gamma)$. However, by the construction of $L(\Gamma)$ those targets always allow each spinal run r' as second component and each g' as third component. A detailed proof is left to the reader.

We return to the main proof that $\mathcal{D}(G) \subseteq \rho'(\mathcal{D}(G'))$. We rather prove the stronger statement that

$$\mathcal{D}\big(G\big) \subseteq \rho'(\mathcal{D}_0(G')) \qquad \text{with} \qquad \mathcal{D}_0(G') = \big\{t \in \mathcal{D}(G') \mid \mathrm{arity}\big(t(\varepsilon)\big) = 0\big\} \tag{†}$$

by induction. This means that we restrict ourselves on the right-hand side to those derivations $\mathcal{D}_0(G')$ that finish in an atomic category. In the induction base, let $u = \gamma \in \Gamma$. Then $\langle \gamma, r, g\rangle \in L(\gamma) \cap \mathcal{D}_0(G')$ is atomic for every $r \in \mathrm{SR}$ and $g \in \Gamma \cup N$, and thus $u \in \rho'(\mathcal{D}_0(G'))$ by the definition of ρ'. In the induction step, we have $u \notin \Gamma$ and the desired property $u' \in \rho'(\mathcal{D}_0(G'))$ is true for all proper subtrees u' of u. For every $u' \in \mathcal{D}(G)$ and $w \in \mathrm{pos}(u')$ such that $u'(w) \in \Gamma$ we construct a spinal run $r_{w,u'}$ as follows:

- If $w = \varepsilon$, then $r_{w,u'} = u'$.
- If $w = \delta w'$ and $u' = n(u'_1, u'_2)$ for some $\delta \in [2]$, $w' \in \mathrm{pos}(u'_\delta)$, $n \in N$, and $u'_1, u'_2 \in \mathcal{D}(G)$, then the spinal run $r_{w,u'}$ is given by $r_{w,u'}(\varepsilon) = \langle n, \delta\rangle$, $r_{w,u'}|_\delta = r_{w',u'_\delta}$, and $r_{w,u'}|_{3-\delta} = u'_{3-\delta}(\varepsilon)$.

It is easily checked that $r_{w,u'}$ is a spinal run of G (i.e., $r_{w,u'} \in \mathrm{SR}(G)$) and $\mathrm{ht}(r_{w,u'}) = |w|$. Now we return to the tree $u \in \mathcal{D}(G)$. Since u is universally mht-bounded by h, there exists a position $w \in \mathrm{pos}(u)$ such that $|w| \le h$ and $u(w) \in \Gamma$. In other words, we select a short path w to a leaf arbitrarily. More precisely, let $w = \delta_1 \cdots \delta_\ell$ with $\delta_1, \ldots, \delta_\ell \in [2]$. Consequently, the spinal run $r = r_{w,u} \in \mathrm{SR}(G)$ has height $\ell \le h$, which yields that $r \in \mathrm{SR}$. We first deal with the positions outside the spine. For every $i \in [\ell]$, let $\overline{\delta_i} = 3 - \delta_i$, so we have $\overline{\delta_i} = 1$ if $\delta_i = 2$, and $\overline{\delta_i} = 2$ if $\delta_i = 1$. Moreover, we define the positions $\overline{w_i} = \delta_1 \cdots \delta_{i-1}\overline{\delta_i}$, which refer to the positions outside the spine of r. Similarly, for every $0 \le i \le \ell$, let $w_i = \delta_1 \cdots \delta_i$ be the i-th position on the spine of r. Trivially, $r(\overline{w_i}) = u(\overline{w_i})$ for all $i \in [\ell]$ by the construction of $r = r_{w,u}$. By the induction hypothesis, for every $i \in [\ell]$ we know that $u|_{\overline{w_i}} \in \rho'(\mathcal{D}_0(G'))$ and together with the auxiliary statement we obtain that there exists a derivation $t_i \in \mathcal{D}_0(G')$ such that $u|_{\overline{w_i}} \in \rho'(t_i)$ and $t_i(\varepsilon) = \langle u(\overline{w_i}), r, u(w_i)\rangle$. By construction, we have $r[i] = u(w_i)$ for all $0 \le i \le \ell$. In particular, $\mathrm{base}(r) = u(w)$. Let $r' \in \mathrm{SR}$ be an arbitrary spinal run and $g' \in \Gamma \cup N$. We consider the category $c = \arg(r, r)[\langle r[0], r', g'\rangle]$, which is contained in $L(u(w))$ by construction of L because $\mathrm{base}(r) = u(w)$. More precisely, let $c = \langle r[0], r', g'\rangle|_1 a_1 |_2 \cdots |_\ell a_\ell$ for some $|_1, \ldots, |_\ell \in S$ and $a_1, \ldots, a_\ell \in A$. Note that the categories $a_1, \ldots, a_\ell$ are atomic because all relevant categories are left-spinal. By the construction of c we know for every $i \in [\ell]$ that (i) $|_i = /$ if and only if $\delta_i = 1$, and (ii) $a_i = t_i(\varepsilon)$. Now we can construct the required derivation of $\mathcal{D}_0(G')$ by combining this category c with the subderivations $t_i \in \mathcal{D}_0(G')$. For every $i \in \mathbb{Z}_\ell$ we let

$$t'_\ell = c \qquad \text{and} \qquad t'_i(\varepsilon) = \langle r[0], r', g'\rangle|_1 a_1 |_2 \cdots |_i a_i \qquad t'_i|_{\delta_{i+1}} = t'_{i+1} \qquad t'_i|_{\overline{\delta_{i+1}}} = t_{i+1} \ .$$

Finally, we set $t' = t'_0$. A straightforward check shows that $t' \in \mathcal{D}_0(G')$. It remains to show that $u \in \rho'(t')$. Obviously, $\mathrm{pos}(u) = \mathrm{pos}(t')$, so we need to show that $u(w) \in \rho'(t'(w))$ for every $w \in \mathrm{pos}(u)$. If $w = \overline{w_i}w'$ for some $i \in [\ell]$ and $w' \in \mathrm{pos}(u|_{\overline{w_i}})$, then this is trivially true because $t'|_{\overline{w_i}} = t_i$ and we already observed that $u|_{\overline{w_i}} \in \rho'(t_i)$. Consequently, we only need to prove the property for all the prefixes w_i (with $i \in \mathbb{Z}_{\ell+1}$) of w. Let $i \in \mathbb{Z}_{\ell+1}$. By the construction of t' we have $t'(w_i) = \langle r[0], r', g'\rangle|_1 a_1 |_2 \cdots |_i a_i$. For $i = 0$, we thus obtain $\rho'(\langle r[0], r', g'\rangle) = \{r[0]\} = \{u(\varepsilon)\}$. For all $i \in [\ell]$ we have $\rho'(t'(w_i)) = \{u(w_i)\}$ since $a_i = t_i(\varepsilon) = \langle u(\overline{w_i}), r, u(w_i)\rangle$. Consequently, we established the more general statement (†) and $\mathcal{D}(G) \subseteq \rho'(\mathcal{D}(G'))$.

We proved the two main statements $\rho'(\mathcal{D}(G')) = \mathcal{D}(G)$ and $\rho'(\mathcal{D}_0(G')) = \mathcal{D}(G)$. With the help of the latter statement, we can now reason as follows:

$$\mathcal{F}_{\rho'}(G') = \{\rho'(t) \mid t \in \mathcal{D}_0(G'),\, t(\varepsilon) \in I \times \mathrm{SR} \times (\Gamma \cup N)\} = \{t \in \mathcal{D}(G) \mid t(\varepsilon) \in I\} \ ,$$

which concludes the proof. □

The good closure properties of regular tree languages allow us to derive a number of closure results for the tree languages generatable by 0-CCGs (see Table 1). We have seen that, while classical 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 classical 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. [29] showing that context-free grammars are not closed under strong lexicalization, meaning that there are context-free grammars such that no lexicalized grammar[9] generates the same derivation tree language.

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 derivation tree languages have at most one additional argument after the category consumed by the composition. We will prove that 1-CCGs generates exactly the regular tree languages by showing inclusion in both directions. Regarding the first direction, it has already been reasoned in the literature [30,31] that the derivation trees of 1-CCGs can be simulated by CFGs.

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

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

closure \ class	regular tree languages = tree languages generatable by 1-CCGs	tree languages generatable by 0-CCGs
union	✓	✓
intersection	✓	✓
complement	✓	✗
relabeling	✓	✓
α-concatenation [17]	✓	✓
α-iteration [17]	✓	✗

Lemma 14 *(see [30, Proposition 4] and [31, Section 3.1]). For each 1-CCG G the derivations $\mathcal{D}(G)$ and the generated tree language are regular.*

Proof. The derivation tree language $\mathcal{D}(G)$ contains only a finite number of arguments. Furthermore, there exist only finitely many secondary input arguments, since the degree of the rules is limited [31]. A rule of degree 1 only replaces the last argument of the primary input category by another argument. As a consequence, the arity of the primary input category cannot increase through composition. So we have only a finite number of categories and can use the same construction that was used in [22, Proposition 3.25] for 0-CCGs to show that $\mathcal{D}(G)$ is regular. Analogous to the proof for 0-CCGs, $\mathcal{F}_\rho(G)$ is regular for every relabeling ρ as well since regular tree languages are closed under relabelings [19, Theorem 2.4.16] and intersection [19, Theorem 2.4.2]. □

The following lemma establishes a normal form for regular tree grammars that is easily achieved using standard techniques. The construction is illustrated in Example 16. An RTG (N, Σ, I, P) is a *tree automaton* (TA) if for each production $(n \to r) \in P$ there exist $\sigma \in \Sigma$ and $n', n'' \in N$ such that $r = \sigma$ or $r = \sigma(n', n'')$.

Lemma 15. *For each RTG there exist a TA $G' = (\mathbb{Z}_m, \Sigma, I', P')$ that accepts the same tree language and a mapping $\pi : \mathbb{Z}_m \to \Sigma$ such that every nonterminal $n \in \mathbb{Z}_m$ generates a uniquely defined terminal symbol $\pi(n)$; i.e., for all $n \in \mathbb{Z}_m$ and $t \in T_\Sigma$ with $n \Rightarrow^+_{G'} t$ we have $t(\varepsilon) = \pi(n)$.*

Proof. For each RTG there exists an equivalent TA $G = (N, \Sigma, I, P)$ by [19, Theorem 2.3.6]. Given a TA G in which a nonterminal n can produce terminals σ_1 and σ_2 with $\sigma_1 \neq \sigma_2$, we can construct an equivalent TA G' by creating copies n_{σ_1} and n_{σ_2} of n. Productions with n on the left-hand side like $n \to \sigma(n', n'')$ and $n \to \sigma$ with $\sigma \in \{\sigma_1, \sigma_2\}$ are replaced by $n_\sigma \to \sigma(n', n'')$ and $n_\sigma \to \sigma$, respectively. Productions with n on the right-hand side [e.g., $n' \to \sigma(n, n'')$] are replaced by one copy of the production for each copy of n [e.g., $n' \to \sigma(n_{\sigma_1}, n'')$ and $n' \to \sigma(n_{\sigma_2}, n'')$]. Each copy n_σ of a start nonterminal $n \in I$ becomes part of the new set of start nonterminals. The nonterminal set $\mathbb{Z}_m$ is obtained by applying a bijection $\pi : N' \to \mathbb{Z}_{|N'|}$, where N' contains all copied and unmodified nonterminals of N. It is easy to see that G' generates the same tree language as G. □

Example 16. Let $G = (N, \Sigma, I, P)$ be the TA with $N = \{s, a, b, c\}$, $\Sigma = \{\sigma, \tau\}$, $I = \{s\}$, and

$$P = \big\{ s \to \sigma(a, b),\ a \to \sigma(b, c),\ a \to \tau,\ b \to \sigma,\ c \to \sigma \big\} .$$

Nonterminal a can produce the terminal symbol σ or τ, so our intermediate TA $G' = (N', \Sigma, I, P')$ has nonterminals $N' = \{s, a_\sigma, a_\tau, b, c\}$, the production $a \to \sigma(b, c)$ has been replaced by $a_\sigma \to \sigma(b, c)$, and $a \to \tau$ has been replaced by $a_\tau \to \tau$. Instead of $s \to \sigma(a, b)$, the two copies $s \to \sigma(a_\sigma, b)$ and $s \to \sigma(a_\tau, b)$ are contained in P'. After mapping N' to $\mathbb{Z}_5$, we obtain productions $P'' = \{0 \to \sigma(1, 3),\ 0 \to \sigma(2, 3),\ 1 \to \sigma(3, 4),\ 2 \to \tau,\ 3 \to \sigma,\ 4 \to \sigma\}$.

Given a TA $G = (\mathbb{Z}_m, \Sigma, I, P)$ in the normal form of Lemma 15 with mapping $\pi : \mathbb{Z}_m \to \Sigma$, 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 category relabeling $\rho : C(A) \to \mathbb{Z}_m$ such that $\mathcal{F}(G) = \mathcal{F}_{\pi \circ \rho}(G')$. Because ρ maps from categories to nonterminals, but $\mathcal{F}(G)$ is labeled by terminal symbols, we use $\pi : \mathbb{Z}_m \to \Sigma$ to map from nonterminals to terminals. Given a production $n \to \sigma(n_1, n_2) \in P$ and a category relabeling $\rho : C(A) \to \mathbb{Z}_m$, we need categories $c_1 \in \rho^{-1}(n_1)$ and $c_2 \in \rho^{-1}(n_2)$ for each category $c \in \rho^{-1}(n)$ such that $\frac{c_1 \quad c_2}{c}$ 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 in 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 Fig. 6 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 necessary to obtain an atomic initial category. Based on these observations, we construct a 1-CCG G' with m^2 atoms in the following way:

	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

	$\langle 0,0\rangle$	$\langle 0,1\rangle$	$\langle 0,2\rangle$	$\langle 1,0\rangle$	$\langle 1,1\rangle$	$\langle 1,2\rangle$	$\langle 2,0\rangle$	$\langle 2,1\rangle$	$\langle 2,2\rangle$
$\langle 0,0\rangle$	0	1	2	0	1	2	0	1	2
$\langle 0,1\rangle$	0	1	2	0	1	2	0	1	2
$\langle 0,2\rangle$	0	1	2	0	1	2	0	1	2
$\langle 1,0\rangle$	1	2	0	1	2	0	1	2	0
$\langle 1,1\rangle$	1	2	0	1	2	0	1	2	0
$\langle 1,2\rangle$	1	2	0	1	2	0	1	2	0
$\langle 2,0\rangle$	2	0	1	2	0	1	2	0	1
$\langle 2,1\rangle$	2	0	1	2	0	1	2	0	1
$\langle 2,2\rangle$	2	0	1	2	0	1	2	0	1

Fig. 6. 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 a/a' is the result of four different forward compositions combining a/a'' and a''/a' (with 4 choices for a''). The relabeling matrix (right) shows a 1-CCG with nine atomic categories after relabeling using category relabeling $\rho_G : C(\mathbb{Z}_3^2) \to \mathbb{Z}_3$, obtained from a TA G with 3 nonterminals by applying Definition 17. Suppose we want to find two categories projected to nonterminals $\langle g, h\rangle = \langle 0, 2\rangle$ whose composition yields $\langle i, j\rangle/\langle i', j'\rangle = \langle 0, 1\rangle/\langle 0, 1\rangle$. These are categories $\langle 0, 1\rangle/\langle 1, 0\rangle$ and $\langle 1, 0\rangle/\langle 0, 1\rangle$ since $\langle k, l\rangle = \langle h - j' \bmod 3, g - i \bmod 3\rangle = \langle 2 - 1 \bmod 3, 0 - 0 \bmod 3\rangle = \langle 1, 0\rangle$.

Definition 17. Given a TA $G = (\mathbb{Z}_m, \Sigma, I, P)$ in the normal form of Lemma 15, we construct the 1-CCG

$$C_G = \left(\Sigma_0,\ \mathbb{Z}_m^2,\ R,\ \rho_G^{-1}(I) \cap \mathbb{Z}_m^2,\ L\right)$$

and the category relabeling $\rho_G : C(\mathbb{Z}_m^2) \to \mathbb{Z}_m$ such that

$$R = \bigcup_{\substack{\sigma \in \Sigma_2 \\ a,b,c \in \mathbb{Z}_m^2}} \left(\left\{ \frac{ax/b \quad b}{ax} \;\middle|\; \rho(a) \to \sigma\big(\rho(a/b), \rho(b)\big) \in P \right\} \cup \left\{ \frac{ax/b \quad b/c}{ax/c} \;\middle|\; \rho(a/c) \to \sigma\big(\rho(a/b), \rho(b/c)\big) \in P \right\} \right),$$

$$L(\alpha) = \bigcup_{a,b \in \mathbb{Z}_m^2} \left(\Big\{ a \;\Big|\; \rho(a) \to \alpha \in P \Big\} \cup \Big\{ a/b \;\Big|\; \rho(a/b) \to \alpha \in P \Big\} \right) \quad \text{for all } \alpha \in \Sigma_0 \ ,$$

and $\rho\big(\langle i, j\rangle\big) = i$ as well as $\rho\big(\langle i, j\rangle/\langle i', j'\rangle\big) = i + j' \bmod m$ for all $i, i', j, j' \in \mathbb{Z}_m$. The relabeling on all other categories is irrelevant.

Lemma 18. *Every regular tree language $\mathcal{F}$ is generatable by a 1-CCG.*

Proof. By definition there exists an RTG G such that $\mathcal{F}(G) = \mathcal{F}$. Moreover, by Lemma 15 there exists an equivalent TA $G' = (\mathbb{Z}_m, \Sigma, I, P)$ and mapping $\pi : \mathbb{Z}_m \to \Sigma$ with the properties mentioned in Lemma 15. We show that the 1-CCG $C_{G'} = (\Sigma_0, \mathbb{Z}_m^2, R, \rho_{G'}^{-1}(I) \cap \mathbb{Z}_m^2, L)$ of Definition 17 generates the tree language $\mathcal{F} = \mathcal{F}(G')$ using the category relabeling $\pi \circ \rho_{G'}$. We achieve this by arguing that $\mathcal{D}(G') = \rho_{G'}(\mathcal{D}(C_{G'}))$, which by the choice $\rho_{G'}^{-1}(I) \cap \mathbb{Z}_m^2$ of initial categories and the definition of $\rho_{G'}$ already proves the main statement. The category $\langle i, j\rangle/\langle i', j'\rangle$ is the result of the forward composition of $\langle i, j\rangle/\langle k, l\rangle$ and $\langle k, l\rangle/\langle i', j'\rangle$, where $i, i', j, j', k, l \in \mathbb{Z}_m$. Fig. 6 illustrates the category relabeling $\rho_{G'}$ by means of a *relabeling matrix*, which is a matrix indexed by atoms a and a' with entries indicating the relabeling $\rho_{G'}(a/a')$. The row and column labels of this matrix 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 relabeling matrix contains all m nonterminals. Relabeling in this manner ensures that all pairs $\langle g, h\rangle$ of nonterminals are covered by each result category a/a'; i.e., we can find an atom a'' such that a/a'' relabels to g and a''/a' relabels to h and their composition would yield a/a' as required. Formally, given a category $\langle i, j\rangle/\langle i', j'\rangle$ and an ordered pair $\langle g, h\rangle$ of nonterminals, we need to verify that there exist $k, l \in \mathbb{Z}_m$ with $\rho_{G'}(\langle i, j\rangle/\langle k, l\rangle) = g$ and $\rho_{G'}(\langle k, l\rangle/\langle i', j'\rangle) = h$. 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 $\langle i, j\rangle$ and nonterminals $g, h \in \mathbb{Z}_m$, we want to find a category $\langle i, j\rangle/\langle k, l\rangle$ and an atom $\langle k, l\rangle$ such that $\rho_{G'}(\langle i, j\rangle/\langle k, l\rangle) = g$ and $\rho_{G'}(\langle k, l\rangle) = h$. From the definition of the relabeling we have $\rho_{G'}(\langle k, l\rangle) = k$, so $k = h$ and $l = g - i \bmod m$. It is straightforward to show that each derivation of $C_{G'}$ relabels (via $\rho_{G'}$) to a derivation of G' due to the definition of R. For the converse, suppose that we would like to simulate a production $n \to \sigma(g, h) \in P$ and we have already settled on category a/a' for n. We already argued that we can always find suitable preimages a/a'' and a''/a' that relabel to g and h, respectively. So for every derivation $d \in \mathcal{D}(G')$ we can find a derivation of $C_{G'}$ that relabels to d. Due to the fact that the categories occurring in derivation trees of $C_{G'}$ cannot have higher order or arity greater than 1, they never leave the relevant domain of $\rho_{G'}$. □

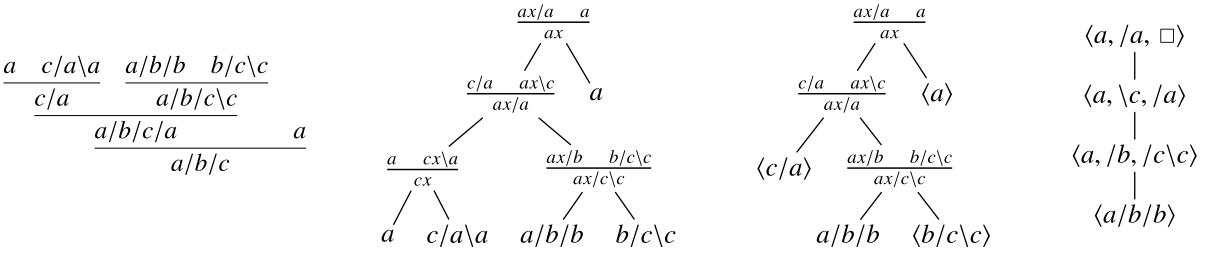

Fig. 7. CCG derivation tree (without lexical entries), corresponding rule tree t, spinal(t), and its encoding enc(t).

Theorem 19. *The tree languages generatable by* 1*-CCGs are exactly the regular tree languages.*

Because the yield languages of the regular tree languages are exactly the context-free languages, we obtain the following generalization of Theorem 8. Note that 0-CCGs are included in the set of 1-CCGs.

Corollary 20. *The string languages generated by* 1*-CCGs are exactly the ε-free context-free languages. Moreover, for each* 1*-CCG G the derivation tree language $\mathcal{D}(G)$ and the tree languages generatable by G are regular.*

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 in the derivation trees $\mathcal{D}(G)$ for a CCG G. Let us illustrate the problem first.

Example 21. Let $G_4 = (\Sigma, A, R, \{D\}, L)$ be the 3-CCG given by the alphabet $\Sigma = \{c, d, e\}$, the atoms $A = \{C, D, E\}$, 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

$$\frac{Dx/D \quad D/E/D\backslash C}{Dx/E/D\backslash C} \qquad \frac{Dx/E \quad E}{Dx} \qquad \frac{Dx/D \quad D/E\backslash C}{Dx/E\backslash C} \qquad \frac{C \quad Dx\backslash C}{Dx}$$

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

Since classical tree language theory only handles finitely many labels, we switch to a different representation and consider *rule trees*. The idea is to label the internal nodes of these trees not by categories, but by the rules that are applied at them to obtain the respective categories, whereas the leaves are still labeled by lexical categories. To keep the presentation simple, we assume, without loss of generality, that all secondary input categories of rules in R are concrete categories of $C(A)$; i.e., we disallow rules with category variables. We introduce the following shorthands. We let $\mathrm{T} = T_{R,\emptyset}(L(\Sigma))$ be the set of all potential rule trees (see Definition 22), and for all alphabets N_1 and N_0 we let $\mathrm{SF}(N_1, N_0) = T_{R,N_1}(L(\Sigma) \cup N_0)$ be the sentential forms of a sCFTG with unary nonterminals N_1 and nullary nonterminals N_0.

Definition 22. 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 $\mathrm{cat}_G \colon \mathrm{T} \to C(A)$ is the partial mapping that is inductively defined by

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

and is undefined for all other cases. The set of all rule trees of G is denoted by $\mathcal{R}(G)$.

Through rule trees, the (well-formed) derivation trees of a CCG are encoded in a natural way while 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. An example rule tree alongside the corresponding CCG derivation tree is depicted in Fig. 7.

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 number of categories. Let $k \in \mathbb{N}$ be the maximal arity of a category in

$$I \cup L(\Sigma) \cup \left\{ c\gamma \;\middle|\; \frac{ax/c \quad c\gamma}{ax\gamma} \in R \right\} \cup \left\{ c\gamma \;\middle|\; \frac{c\gamma \quad ax\backslash c}{ax\gamma} \in R \right\} ,$$

i.e., the maximal arity of the categories that occur as initial category, in the lexicon, or as the secondary input category of a rule of R. Further, let $C_L(A,k) = \{c \in C(A,k) \mid \operatorname{argcats}(c) \subseteq \operatorname{argcats}(L)\}$. The sets $C_L(A)$, $\mathcal{A}_L(A,k)$, and $\mathcal{A}_L(A)$ are defined analogously. The derivation trees in $\mathcal{D}(G)$ can only contain categories whose arguments already appear as arguments of lexical entries since every argument in an output category already appears in one of the input categories [9, Lemma 3.1]. Thus, we can restrict ourselves to categories using these arguments.[10] Roughly speaking, the constructed sCFTG will use the categories $C_L(A,k)$ as nullary nonterminals and tuples $\langle a, |c, \gamma\rangle$ consisting of an atomic category $a \in A$, a single argument $|c$ with $| \in S$ and $c \in C_L(A,k)$, and an argument tree $\gamma \in \mathcal{A}_L(A,k)$ as unary nonterminals. Recall that we write substitutions $\alpha[t]$ as $t\alpha$ for $\alpha \in \mathcal{A}(A)$ and $t \in C(A) \cup \mathcal{A}(A)$.

Definition 23. We construct the sCFTG $G' = (N_1 \cup N_0, R \cup L(\Sigma), I', P)$ with

- $N_1 = \{\langle a, |c, \gamma\rangle \mid a \in A,\ | \in S,\ c \in C_L(A,k),\ \gamma \in \mathcal{A}_L(A,k)\}$ and $N_0 = \{\langle c\rangle \mid c \in C_L(A,k)\}$,
- $I' = \{\langle a_0\rangle \mid a_0 \in I\}$, and
- the following set $P = P_1 \cup P_2 \cup P_3 \cup P_4 \cup P_5$ of productions with

$$P_1 = \big\{\langle c\rangle \to c \mid c \in L(\Sigma)\big\} \tag{1}$$
$$P_2 = \big\{\langle a, /c, \gamma\rangle \to s\big(\square, \langle c\gamma\rangle\big) \mid s = \big(ax/c,\ c\gamma \to ax\gamma\big) \in R\big\} \tag{2}$$
$$P_3 = \big\{\langle a, \backslash c, \gamma\rangle \to s\big(\langle c\gamma\rangle, \square\big) \mid s = \big(c\gamma,\ ax\backslash c \to ax\gamma\big) \in R\} \tag{3}$$
$$P_4 = \big\{\langle a\alpha\gamma\rangle \to \langle a, |c, \gamma\rangle\big(\langle a\alpha|c\rangle\big) \mid a \in A,\ \alpha, \gamma \in \mathcal{A}_L(A),\ | \in S,\ c \in C_L(A,k),\ |\alpha| < k,\ |\alpha\gamma| \le k\big\} \tag{4}$$
$$P_5 = \big\{\langle a, |c, \gamma\rangle \to \langle a, |'c', \square\rangle\big(\langle a, |c, \gamma|'c'\rangle(\square)\big) \mid a \in A,\ |, |' \in S,\ c, c' \in C_L(A,k),\ \gamma \in \mathcal{A}_L(A, k-1)\big\} \tag{5}$$

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 24. $\mathcal{F}(G') \subseteq \mathcal{R}(G)$

Proof. We will start with an auxiliary statement. For all sentential forms $\xi \in \mathrm{SF}(N_1, N_0)$ and $t \in \mathrm{T}$ such that $\xi \Rightarrow_{G'}^{+} t$, we prove that

(i) if $\xi(\varepsilon) = \langle c\rangle \in N_0$, then $\mathrm{cat}_G(t) = c$, and
(ii) if $\xi(\varepsilon) = \langle a, |c, \gamma\rangle \in N_1$, $\xi|_1 \in \mathrm{T}$, and $\mathrm{cat}_G(\xi|_1) = a\alpha|c$ with $\alpha \in \mathcal{A}_L(A)$, then $\mathrm{cat}_G(t) = a\alpha\gamma$.

Note that we do not make any statement for trees ξ such that $\xi(\varepsilon) \notin N_0 \cup N_1$. For the remaining trees, we prove this statement by induction on the length of the derivation. We have $\xi \Rightarrow_{G',\varepsilon} \zeta \Rightarrow_{G'}^{\ell} t$ for some $\zeta \in \mathrm{SF}(N_1, N_0)$ and $\ell \in \mathbb{N}$, where we applied the first derivation step at the root and $\Rightarrow_{G'}^{\ell}$ is the ℓ-fold composition of $\Rightarrow_{G'}$ with itself. We distinguish five cases based on the production $p \in P$ used at the root in the first derivation step:

(1) If $p = \langle c\rangle \to c \in P_1$ is a production of type (1), then $c \in L(\Sigma)$, $\xi = \langle c\rangle$, and $\zeta = t = c$. Since $c \in L(\Sigma)$, we trivially have $\mathrm{cat}_G(t) = c$, which proves statement (i).[11]
(2) If $p = \langle a, /c, \gamma\rangle \to s(\square, \langle c\gamma\rangle) \in P_2$ is a production of type (2), then we have $\xi(\varepsilon) = \langle a, /c, \gamma\rangle$ and we only need to prove statement (ii). Consequently, let $\xi|_1 \in \mathrm{T}$ and $\mathrm{cat}_G(\xi|_1) = a\alpha/c$ for some $\alpha \in \mathcal{A}_L(A)$. From the derivation $\zeta = s(\xi|_1, \langle c\gamma\rangle) \Rightarrow_{G'}^{\ell} t$, we can conclude that $\langle c\gamma\rangle \Rightarrow_{G'}^{\ell'} t|_2$ for some $1 \le \ell' < \ell$. The latter yields $\mathrm{cat}_G(t|_2) = c\gamma$ by the induction hypothesis. From the facts $s = \frac{ax/c \quad c\gamma}{ax\gamma} \in R$, $\mathrm{cat}_G(t|_1) = a\alpha/c$, and $\mathrm{cat}_G(t|_2) = c\gamma$, we conclude that $\mathrm{cat}_G(t) = a\alpha\gamma$.
(3) If $p = \langle a, \backslash c, \gamma\rangle \to s(\langle c\gamma\rangle, \square) \in P_3$ is a production of type (3), then we need to prove statement (ii), which can be done in the same way as in the previous case (2).
(4) If $p = \langle a\alpha\gamma\rangle \to \langle a, |c, \gamma\rangle\big(\langle a\alpha|c\rangle\big) \in P_4$ is a production of type (4), then we need to prove statement (i). By context-freeness, we can rearrange the derivation $\zeta \Rightarrow_{G'}^{\ell} t$ such that

$$\zeta = \langle a, |c, \gamma\rangle\big(\langle a\alpha|c\rangle\big) \Rightarrow_{G'}^{\ell'} \langle a, |c, \gamma\rangle\big(t'\big) \Rightarrow_{G'}^{\ell''} t$$

for some $t' \in \mathrm{T}$ and $\ell', \ell'' \ge 1$ such that $\ell = \ell' + \ell''$. Consequently, we have a subderivation $\langle a\alpha|c\rangle \Rightarrow_{G'}^{\ell'} t'$, from which we conclude that $\mathrm{cat}_G(t') = a\alpha|c$ by the induction hypothesis. Now we established the preconditions of statement (ii) for the subderivation $\langle a, |c, \gamma\rangle\big(t'\big) \Rightarrow_{G'}^{\ell''} t$, so $\mathrm{cat}_G(t) = a\alpha\gamma$ by the induction hypothesis.

[10] This restriction is required for the sCFTG to have a finite set of nonterminals and was missing in the previous version of this contribution [32].
[11] This also proves statement (ii) in this case since its precondition is not fulfilled. We omit such obvious observations in the next cases.

(5) If $p = \langle a, |c, \gamma\rangle \to \langle a, |'c', \square\rangle\big(\langle a, |c, \gamma|'c'\rangle(\square)\big) \in P_5$ is a production of type (5), then we need to prove statement (ii). Let $\xi|_1 \in \mathrm{T}$ and $\mathrm{cat}_G(\xi|_1) = a\alpha|c$ for some $\alpha \in \mathcal{A}_L(A)$. We can again reorder the derivation $\zeta \Rightarrow^{\ell}_{G'} t$ such that

$$\zeta = \langle a, |'c', \square\rangle\big(\langle a, |c, \gamma|'c'\rangle(\xi|_1)\big) \Rightarrow^{\ell'}_{G'} \langle a, |'c', \square\rangle\big(t'\big) \Rightarrow^{\ell''}_{G'} t$$

for some $t' \in \mathrm{T}$ and $\ell', \ell'' \geq 1$ such that $\ell = \ell' + \ell''$. In the first subderivation we find $\langle a, |c, \gamma|'c'\rangle(\xi|_1) \Rightarrow^{\ell'}_{G'} t'$. Since $\xi|_1 \in \mathrm{T}$ and $\mathrm{cat}_G(\xi|_1) = a\alpha|c$, we meet the requirements of statement (ii), obtaining $\mathrm{cat}_G(t') = a\alpha\gamma|'c'$ by the induction hypothesis. Once more we now have established that $t' \in \mathrm{T}$ and $\mathrm{cat}_G(t') = a\alpha\gamma|'c'$, so we satisfy the requirements of statement (ii) of the induction hypothesis applied to the second subderivation $\langle a, |'c', \beta\rangle\big(t'\big) \Rightarrow^{\ell''}_{G'} t$. Consequently, $\mathrm{cat}_G(t) = a\alpha\gamma$.

This completes the proof of the auxiliary statement. We can apply the auxiliary statement to derive that $\mathrm{cat}_G(t) = a_0 \in I$ for all $t \in \mathrm{T}$ and $\langle a_0\rangle \in I'$ such that $\langle a_0\rangle \Rightarrow^{+}_{G'} t$. Consequently, $\mathcal{F}(G') \subseteq \mathcal{R}(G)$. □

For the converse, we decompose and encode rule trees $\mathcal{R}(G)$ in a more compact manner. These encodings will help us to structure the derivation of rule trees, as we can use them as components and intermediate steps of the complete derivation. First, we translate a rule tree into its *primary spine form*. For all $a \in A$, $c \in C_L(A, k)$, $\gamma \in \mathcal{A}_L(A, k)$, and $t_1, t_2 \in \mathrm{T}$ we let

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

Clearly, $\mathrm{spinal}\colon \mathrm{T} \to T_{R,\emptyset}(L(\Sigma) \cup N_0)$. An example is shown in Fig. 7. Additionally, we encode rule trees using only the nonterminals of G'. To this end, we define a mapping $\mathrm{enc}\colon \mathrm{T} \to T_{\emptyset,N_1}(N_0)$. For all $a \in A$, $c \in C_L(A, k)$, $\gamma \in \mathcal{A}_L(A, k)$, and $t_1, t_2 \in \mathrm{T}$ we let

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

This encoding is also demonstrated in Fig. 7. We show that $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^{*}_{G'} \mathrm{enc}(t) \Rightarrow^{*}_{G'} \mathrm{spinal}(t)$ for every $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A, k)$ in the following sequence of lemmas. We begin by proving the second, easier part.

Lemma 25. $\mathrm{enc}(t) \Rightarrow^{*}_{G'} \mathrm{spinal}(t)$ *for every* $t \in \mathrm{T}$ *with* $\mathrm{cat}_G(t) \in C_L(A, k)$.

Proof. The proof is by induction on t. In the induction base, we have $t \in L(\Sigma)$ and thus $t \in C_L(A, k)$. Hence $\mathrm{enc}(t) = \langle t\rangle$ and $\mathrm{spinal}(t) = t$. Since $t \in L(\Sigma)$ we can apply a production of type (1) to obtain $\mathrm{enc}(t) = \langle t\rangle \Rightarrow_{G'} t = \mathrm{spinal}(t)$ as desired. In the induction step, let $t = s(t_1, t_2)$ with $s = \frac{ax/c \quad c\gamma}{ax\gamma} \in R$. The case of a backward composition is analogous. Then $\mathrm{enc}(t) = \langle a, /c, \gamma\rangle(\mathrm{enc}(t_1))$ and $\mathrm{spinal}(t) = s(\mathrm{spinal}(t_1), \langle c\gamma\rangle)$. Consequently,

$$\mathrm{enc}(t) = \langle a, /c, \gamma\rangle\big(\mathrm{enc}(t_1)\big) \Rightarrow^{*}_{G'} \langle a, /c, \gamma\rangle\big(\mathrm{spinal}(t_1)\big) \Rightarrow^{*}_{G'} s\big(\mathrm{spinal}(t_1), \langle c\gamma\rangle\big) = \mathrm{spinal}(t) \ ,$$

where we used the induction hypothesis in the first step and then a production of type (2). □

We now turn to the first part, that is showing that $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^{*}_{G'} \mathrm{enc}(t)$ for every $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A, k)$. This will be dealt with in the next two lemmas. For this purpose, we need to introduce some additional notation. As usual, let A be a finite set of atomic categories. For all argument trees $\alpha, \alpha' \in \mathcal{A}_L(A)$ we write $\alpha \sqsubseteq \alpha'$ if there exists a position $w \in \mathrm{pos}(\alpha') \cap \{1\}^*$ such that $\alpha = \alpha'|_w$ (i.e., α is a subtree of α' that is located on the left spine). In other words, if $\alpha \sqsubseteq \alpha'$ and $\alpha' = a|_1 c_1 \cdots |_\ell c_\ell$, then $\alpha = a|_1 c_1 \cdots |_i c_i$ for some $i \leq \ell$. Additionally, we simplify our notation for encodings. Recall the sets N_1 and N_0 of nonterminals from Definition 23. We let $\mathrm{Enc} = T_{\emptyset,N_1}(N_0)$ be the set of encodings. Encodings are essentially strings, so $\mathrm{pos}(e) \subseteq \{1\}^*$ for all $e \in \mathrm{Enc}$. Instead of a position 1^k we simply write just k in an encoding. Moreover, we write $|e|$ instead of $|\mathrm{pos}(e)|$. In other words, we identify the set $\mathrm{pos}(e)$ with the corresponding set $\mathbb{Z}_{|e|}$ of nonnegative integers.

Let $e \in \mathrm{Enc}$ be an encoding with positions $\mathrm{pos}(e) = \mathbb{Z}_{\ell+1}$ such that $e(\ell) = \langle a\alpha\rangle$ and $e(i) = \langle a_i, |_i c_i, \gamma_i\rangle$ for all $i < \ell$. It is *consistent* if

- $a = a_i$ for all $i < \ell$, and
- there exist $\alpha_0, \dots, \alpha_{\ell-1}$ such that $\alpha = \alpha_{\ell-1}|_{\ell-1} c_{\ell-1}$ and $\alpha_i \gamma_i = \alpha_{i-1}|_{i-1} c_{i-1}$ for all $i \in [\ell - 1]$.

For such a consistent encoding, we have $\mathrm{cat}_G(e|_\ell) = a\alpha$ and $\mathrm{cat}_G(e|_i) = a\alpha_i\gamma_i$ for all $i < \ell$. Now, let $e, e' \in \mathrm{Enc}$ be two encodings. We write $e \prec e'$ if there exist positions $i \in \mathrm{pos}(e)$ and $i' \in \mathrm{pos}(e')$ with $i \le i'$ such that

- $e(j) = \langle a_j, |_j c_j, \Box\rangle$ for all $j < i$ (i.e., all labels of e at positions $j < i$ have $\Box$ in the third component), and
- $e|_i = \langle a, |c, \gamma\rangle(\overline{e})$ and $e'|_{i'} = \langle a, |c, \gamma'\rangle(\overline{e})$ with $\gamma \sqsubset \gamma'$ (i.e., the subtrees $e|_i$ and $e'|_{i'}$ coincide except for the third components γ and γ' of the labels at the roots, for which we have that γ is a strict subtree on the left spine of γ').

It is clear that $\prec$ is a strict partial order (irreflexive and transitive) on encodings. Indeed the positions i and i' that demonstrate $e \prec e'$ are unique. Finally, we let $f\colon \mathrm{Enc}^2 \to \mathbb{Z}$ be such that $f(e, e') = |e'| - |e|$ for all $e, e' \in \mathrm{Enc}$. It is obvious that $f(e, e') \in \mathbb{N}$ provided that $e \prec e'$.

Lemma 26. *Let $e, e' \in Enc$ be consistent encodings with $\mathrm{cat}_G(e) = \mathrm{cat}_G(e')$ such that $e \prec e'$. Then $e \Rightarrow^+_{G'} e'$.*

Proof. Let $i \le i'$ be the unique positions required to show that $e \prec e'$, and $e|_i = \langle a, |c, \gamma\rangle(\overline{e})$ and $e'|_{i'} = \langle a, |c, \gamma'\rangle(\overline{e})$. Also let $a\beta = \mathrm{cat}_G(e) = \mathrm{cat}_G(e')$ and $a\alpha|c = \mathrm{cat}_G(\overline{e})$. Then obviously $\mathrm{cat}_G(e|_i) = a\alpha\gamma$. Moreover, $i = |\alpha\gamma| - |\beta|$ because the third component is $\Box$ for all labels at positions strictly smaller than i, which yields that $\mathrm{cat}_G(e|_{i-j}) = (a\alpha\gamma)|_j$ for all $j \le i$. Similarly, we have $\mathrm{cat}_G(e') = a\alpha\gamma'$ with $|\alpha\gamma'| > |\alpha\gamma|$. Since we can only remove a single argument in each step, we obtain that $|\alpha\gamma'| - |\beta| \le i'$.

We now prove the statement by induction on $f(e, e')$. In the induction base, we assume that $f(e, e') = |e'| - |e| = 0$. Consequently, we have $i = i'$ and $|\alpha\gamma'| - |\beta| \le i' = i = |\alpha\gamma| - |\beta| < |\alpha\gamma'| - |\beta|$, which is a contradiction. Hence this case cannot occur.

In the induction step, let $f(e, e') = |e'| - |e| > 0$. Consequently, $i < i'$. Let $\gamma' = \gamma|'c'\gamma''$ for some $|' \in S$, $c' \in C_L(A, k)$, and $\gamma'' \in \mathcal{A}_L(A, k)$. Using an application of a production of type (5) we obtain

$$e|_i = \langle a, |c, \gamma\rangle(\overline{e}) \Rightarrow_{G'} \langle a, |'c', \Box\rangle\big(\langle a, |c, \gamma|'c'\rangle(\overline{e})\big) = \hat{e} \ .$$

Let $e'' = e[\hat{e}]_i$, which immediately yields $|e''| > |e|$. Then e'' is again a consistent encoding because $\mathrm{cat}_G(\hat{e}) = a\alpha\gamma$, which is also the category of the replaced subtree $e|_i$. Consequently, the newly constructed encoding e'' has the same category as e and e'. Next we prove that $e'' \preceq e'$. If $e'' = e'$, then trivially $e'' \preceq e'$. Thus, let $e'' \ne e'$. Since $e''|_{i+2} = e'|_{i'+1}$, let $j'' \le i + 1$ be the largest integer such that $e''(j'') \ne e'(j')$, where $j' = i' - (i+1) + j''$. Clearly, such an integer j'' must exist because $e'' \ne e'$. Now we prove by case analysis on $j'' \le i + 1$ that $e'' \preceq e'$.

- If $j'' = i+1$, then $j' = i'$. Since the labels of e'' at $j'' = i+1$ and of e' at $j' = i'$ differ, although their first two components are the same, we obtain that $\gamma|'c' \ne \gamma' = \gamma|'c'\gamma''$ and thus $\gamma'' \ne \Box$. Then trivially $e'' \prec e'$ using the positions $j'' = i + 1$ and $j' = i'$, for which we know $i + 1 \le i'$ and all labels of e'' at strict prefixes of $i + 1$ have $\Box$ in the third component (for all positions strictly smaller than i this is true since the labels of e'' and e coincide and for i it is true by the definition of e'').
- Otherwise, we have $j'' < i+1$. Let $e''|_{j''} = \langle a, |''c'', \Box\rangle(h'')$ and $e'|_{j'} = \langle a, |'''c''', \alpha'\rangle(h')$. Obviously, we have $h'' = h'$ because we selected the maximal j''. Consequently, we also have $|'''c''' = |''c''$ by consistency. Since the labels are different, we must have $\Box \sqsubset \alpha'$. Then $e'' \prec e'$ using the positions j'' and j', for which we know that $j'' = j' - i' + (i + 1) \le j'$ because $i + 1 \le i'$. Moreover, all labels of e'' at strict prefixes of $j'' \le i$ have $\Box$ in third component since those labels coincide in e'' and e.

Hence $e'' \preceq e'$. Now we return to the main statement. If $e'' = e'$, then clearly $e \Rightarrow^+_{G'} e'' = e'$. Otherwise, we have consistent encodings e'' and e' with $\mathrm{cat}_G(e'') = \mathrm{cat}_G(e')$ such that $e'' \prec e'$. Since $|e''| > |e|$, we additionally have $f(e'', e') = |e'| - |e''| < |e'| - |e| = f(e, e')$. Consequently, we can apply the induction hypothesis to e'' and e' and obtain that $e'' \Rightarrow^+_{G'} e'$, which together with $e \Rightarrow_{G'} e''$ yields $e \Rightarrow^+_{G'} e'$ as desired. □

It should be obvious that for every $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A)$ the encoding $\mathrm{enc}(t)$ is consistent and has the same category $\mathrm{cat}_G(t)$. For the proof of $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^*_{G'} \mathrm{enc}(t)$ for all $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A, k)$, we define another mapping $\mathrm{enc}'\colon \mathrm{T} \to T_{\emptyset, N_1}(N_0)$. For all $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A, k)$, we let

$$\mathrm{enc}'(t) = \langle \mathrm{cat}_G(t)\rangle \ .$$

Otherwise, for all $a \in A$, $c \in C_L(A, k)$, $\gamma \in \mathcal{A}_L(A, k)$, and $t_1, t_2 \in \mathrm{T}$, we let

$$\mathrm{enc}'\Big(\frac{ax/c \quad c\gamma}{ax\gamma}(t_1, t_2)\Big) = \langle a, /c, \gamma\rangle\big(\mathrm{enc}'(t_1)\big)$$

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

Fig. 8. Derivation of the encoding.

$$\mathrm{enc}'\Big(\frac{c\gamma \quad a x\backslash c}{a x\gamma}(t_1,t_2)\Big) = \langle a,\backslash c,\gamma\rangle\big(\mathrm{enc}'(t_2)\big) \ .$$

Similarly, for every $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A)$ the encoding $\mathrm{enc}'(t)$ is consistent and has the same category $\mathrm{cat}_G(t)$. Note how this is different from the previous encoding $\mathrm{enc}(t)$. While before, we only added a nullary nonterminal if the input consisted of a single node $c \in C_L(A,k)$, we now compress a complete subtree whose category is in $C_L(A,k)$ to a nullary nonterminal. In the following, we will show that we can derive the original encoding from this shortened variant.

Lemma 27. $\mathrm{enc}'(t) \Rightarrow^*_{G'} \mathrm{enc}(t)$ *for every* $t \in \mathrm{T}$ *with* $\mathrm{cat}_G(t) \in C_L(A)$.

Proof. We prove it by induction on t. In the induction base, we let $t \in L(\Sigma)$. Consequently, $t = \mathrm{cat}_G(t) \in C_L(A,k)$ and $\mathrm{enc}'(t) = \langle \mathrm{cat}_G(t)\rangle = \mathrm{enc}(t)$, which proves the induction base. In the induction step, let $t = r(t_1,t_2)$ for some rule $r \in R$ and $t_1, t_2 \in \mathrm{T}$. We only consider forward compositions. Thus, let $r = \frac{a x/c \quad c\gamma}{a x\gamma}$. By assumption $\mathrm{cat}_G(t) \in C_L(A)$, so let $\mathrm{cat}_G(t) = a\alpha\gamma$ for some $\alpha \in \mathcal{A}_L(A)$. Now we distinguish three cases depending on the arities of the categories of t and t_1:

- Suppose that $\mathrm{cat}_G(t) = a\alpha\gamma \notin C(A,k)$. Then

  $$\mathrm{enc}'(t) = \langle a,/c,\gamma\rangle\big(\mathrm{enc}'(t_1)\big) \Rightarrow^*_{G'} \langle a,/c,\gamma\rangle\big(\mathrm{enc}(t_1)\big) = \mathrm{enc}(t) \ ,$$

  where we used the induction hypothesis applied to t_1.
- Now suppose that $\mathrm{cat}_G(t) = a\alpha\gamma \in C_L(A,k)$ and for the subtree t_1 suppose that $\mathrm{cat}_G(t_1) = a\alpha/c \notin C(A,k)$. Let $\langle a\beta|c'\rangle$ be the leaf of $\mathrm{enc}'(t_1)$. Note that the categories of subtrees of $\mathrm{enc}'(t_1)$ for all strictly smaller positions are not in $C(A,k)$. Consequently, we have $a\beta \sqsubset a\alpha\gamma$, so let $\gamma' \in \mathcal{A}_L(A,k)$ be such that $a\beta\gamma' = a\alpha\gamma$. In addition,

  $$\mathrm{enc}'(t) = \langle a\alpha\gamma\rangle \Rightarrow_{G'} \langle a,|c',\gamma'\rangle\big(\langle a\beta|c'\rangle\big) = e \ .$$

  Clearly, $e = \langle a,|c',\gamma'\rangle(\langle a\beta|c'\rangle)$ is consistent and has category $a\alpha\gamma$, which is also true for $e' = \langle a,/c,\gamma\rangle(\mathrm{enc}'(t_1))$. Next we show that $e \prec e'$. Let $\ell = |e'| - 1$. Obviously, $e|_1 = \langle a\beta|c'\rangle = e'|_\ell$ and $e'_{\ell-1} = \langle a,|c',\gamma''\rangle$ for some $\gamma'' \in \mathcal{A}_L(A,k)$ because e' is consistent. It remains to prove that $\gamma' \sqsubset \gamma''$, which is true because for all non-zero positions strictly smaller than $\ell - 1$ the encoding has subtrees with categories that are not in $C(A,k)$. Hence $a\beta\gamma' = a\alpha\gamma \in C_L(A,k)$ must be a subtree on the left spine of all those categories. Consequently, we have $e \prec e'$ for these consistent encodings with $\mathrm{cat}_G(e) = \mathrm{cat}_G(e')$, so we use Lemma 26 to conclude that $e \Rightarrow^+_{G'} e'$. Thus, in summary we have

  $$\mathrm{enc}'(t) = \langle a\alpha\gamma\rangle \Rightarrow_{G'} \langle a,|c',\gamma'\rangle\big(\langle a\beta|c'\rangle\big) = e \Rightarrow^+_{G'} e' = \langle a,/c,\gamma\rangle\big(\mathrm{enc}'(t_1)\big) \Rightarrow^*_{G'} \langle a,/c,\gamma\rangle\big(\mathrm{enc}(t_1)\big) = \mathrm{enc}(t) \ .$$

- Finally, suppose that $\mathrm{cat}_G(t) = a\alpha\gamma \in C_L(A,k)$ and for t_1 suppose that $\mathrm{cat}_G(t_1) = a\alpha/c \in C_L(A,k)$. Then

  $$\begin{aligned}\mathrm{enc}'(t) = \langle a\alpha\gamma\rangle &\Rightarrow_{G'} \langle a,/c,\gamma\rangle\big(\langle a\alpha/c\rangle\big) = \langle a,/c,\gamma\rangle\big(\mathrm{enc}'(t_1)\big)\\ &\Rightarrow^*_{G'} \langle a,/c,\gamma\rangle\big(\mathrm{enc}(t_1)\big) = \mathrm{enc}(t) \ .\end{aligned}$$

Thus, we have proved the statement. A particular case is the desired derivation $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^*_{G'} \mathrm{enc}(t)$ for every $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A,k)$. □

An example of a derivation of the form $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^*_{G'} \mathrm{enc}(t)$ is presented in Fig. 8. Combining Lemmata 25 and 27, we can conclude that $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^*_{G'} \mathrm{enc}(t) \Rightarrow^*_{G'} \mathrm{spinal}(t)$ holds for every $t \in \mathrm{T}$ with $\mathrm{cat}_G(t) \in C_L(A,k)$. This will prove extremely useful in the following.

Lemma 28. $\mathcal{R}(G) \subseteq \mathcal{F}(G')$

Proof. We first prove by induction the auxiliary statement that for every rule tree $t \in \mathrm{T}$ such that $\mathrm{cat}_G(t) \in C_L(A,k)$ we have $\langle \mathrm{cat}_G(t)\rangle \Rightarrow^*_{G'} t$. In the induction base we have $t \in L(\Sigma)$, which also yields $t = \mathrm{cat}_G(t) \in C_L(A,k)$, and thus $\langle t\rangle \Rightarrow_{G'} t$

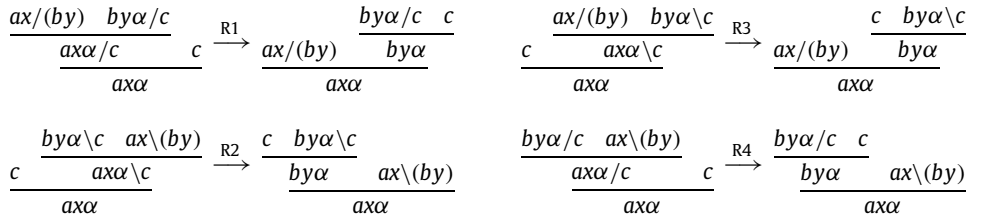

Fig. 9. Rule schemes of [14] with $a, b \in A$, $x, y, \alpha \in \mathcal{A}(A)$, and $c \in C(A)$.

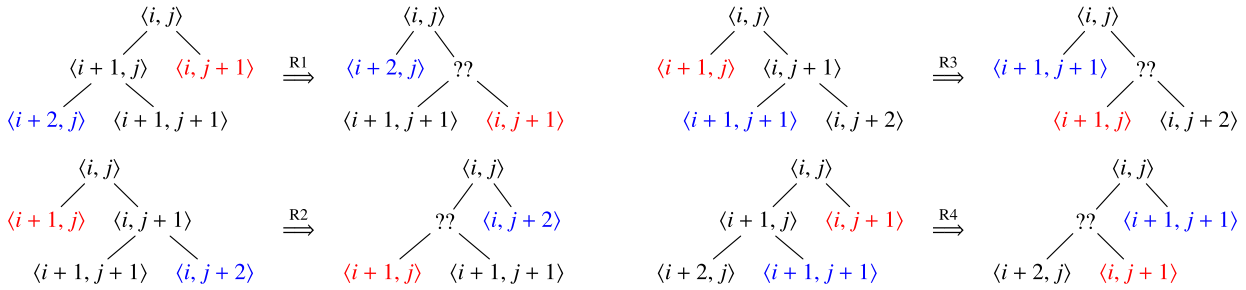

Fig. 10. Rule schemes of [14] with the relabeling indicated based on the input tree. The roots of equal subtrees occurring at the wrong position are marked in red and blue. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

by a production of type (1). In the induction step, we use Lemmata 25 and 27 to obtain $\langle c \rangle \Rightarrow_{G'}^* \mathrm{spinal}(t)$, and we apply the induction hypothesis to the nullary nonterminals present in $\mathrm{spinal}(t)$ to conclude the proof of the auxiliary statement. With the help of the auxiliary statement, we immediately obtain that for every rule tree $t \in \mathcal{R}(G)$ we have $\langle \mathrm{cat}_G(t) \rangle \Rightarrow_{G'}^* t$ because $\mathrm{cat}_G(t) \in I$. Moreover, $\langle \mathrm{cat}_G(t) \rangle \in I'$ and hence $t \in \mathcal{L}(G')$. □

Theorem 29. *The rule tree language $\mathcal{R}(G)$ of a CCG G can be generated by an sCFTG.*

7. Proper inclusion for pure CCGs

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

Example 30. Let us consider the CFG $G_{\mathrm{ex}} = (N, \Gamma, \langle 0, 0 \rangle, P)$ with the nonterminals $N = \{\langle i, j \rangle \mid i, j \in \mathbb{Z}_3\}$, the terminals $\Gamma = \{\alpha\}$, and the set P of productions containing 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 13 already shows that the tree language $\mathcal{D}(G_{\mathrm{ex}})$ is not generatable by any 0-CCG. Similarly, it is impossible to generate $\mathcal{D}(G_{\mathrm{ex}})$ 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. The transformation schemes are depicted in Fig. 9. Due to the absence of rule restrictions in pure CCGs, the applicability of these transformations cannot be prevented.

Theorem 31. *The tree language $\mathcal{D}(G_{\mathrm{ex}})$ is not generatable by any pure CCG.*

Proof. For the sake of a contradiction, suppose that there exists a pure CCG $G = (\Sigma, A, \mathcal{R}(A, k), I, L)$ and a (category) relabeling ρ such that $\mathcal{F}_\rho(G) = \mathcal{D}(G_{\mathrm{ex}})$. Let $t \in \mathcal{D}(G_{\mathrm{ex}})$ be such that $\mathrm{mht}(t) > \mathrm{arity}(L)$. Since $\mathcal{D}(G_{\mathrm{ex}})$ is not universally mht-bounded, such a tree exists. By Lemma 10, the CCG G has to use non-application operations to produce trees with the shape of t, so $k \geq 1$. Let $u \in \mathcal{D}(G)$ be such that $t \in \rho(u)$. Moreover, select the top-most (i.e., least with respect to the shortlex order) position $w \in \mathrm{pos}(u)$ at which an application rule is applied followed by a non-application operation. Such a position must exist since the rule applied at the root is always an application rule (because the initial category is atomic). This yields one of the cases listed left of the $\Rightarrow$-symbol in Fig. 9. We apply the such identified transformation rule of Fig. 9 that matches at w to obtain another tree $u' \in \mathcal{D}(G)$ [14]. However, as we illustrate in Fig. 10, each transformation rule leads to a derivation tree of the wrong shape (i.e., one that can be relabeled in an undesirable manner). Let us walk

Table 2
Summary of the expressive power of various types of CCGs, where CFL, TAL, RTL, and sCFTL are the context-free languages, tree-adjoining (string) languages, regular tree languages, and tree languages generated by sCFTG, respectively.

expr. power \ class	pure 0-CCG	0-CCG	pure 1-CCG	1-CCG	pure CCG	CCG
strings	$= \mathrm{CFL}$				$\subsetneq \mathrm{TAL}$	$= \mathrm{TAL}$
trees	$=$ mht-bounded RTL		$\subsetneq \mathrm{RTL}$	$= \mathrm{RTL}$	$\subsetneq \mathrm{sCFTL}$	$= \mathrm{sCFTL}$

through one case in detail. Suppose that the first transformation rule of Fig. 9 applies at position w. Moreover, assume that $u(w)$ relabels to $\langle i, j\rangle$ for some $i, j \in \mathbb{Z}_3$ because $t \in \rho(u) \subseteq \mathcal{D}(G_{\mathrm{ex}})$. Then we know that $u(w2)$ relabels to $\langle i, j+1\rangle$ because $\rho(u) \subseteq \mathcal{D}(G_{\mathrm{ex}})$. However, after applying the transformation (i.e., in the tree u'), the subtree $u|_{w2}$ now occurs at $u'|_{w22}$, which creates the wrong shape, so $\rho(u') \nsubseteq \mathcal{D}(G_{\mathrm{ex}})$ contradicting $\mathcal{F}_\rho(G) = \mathcal{D}(G_{\mathrm{ex}})$. □

8. Conclusion

Table 2 summarizes our main results and puts them into the context of related work on the expressive power of various types of CCGs, both as generators of strings and as generators of trees (weak and strong generative capacity). The weak equivalence of CFG and pure 0-CCG, which is CCG without rule restrictions and only application rules, is a classical result due to Bar-Hillel, Gaifman, and Shamir [28]. We showed that the tree languages generated by 0-CCGs are a proper subset of regular tree languages (Theorem 13, see also [25]), whereas those generated by 1-CCGs are exactly the regular tree languages (Theorem 19). While the step from 0-CCGs to 1-CCGs does not increase weak generative capacity (Corollaries 8 and 20), the strong generative capacity increases. We also observe that there is no difference in tree expressivity for 0-CCGs between the pure and non-pure variants (Theorem 13), while for higher rule degrees, pure CCGs are strictly weaker and cannot even generate all regular tree languages (Theorem 31). For weak generative capacity, this difference has already been investigated [14,31] and occurs for rule degrees 2 and higher.

The string languages generatable by pure 1-CCGs are, as for the non-pure variant, exactly the context-free languages. This statement deserves some further explanation. Although it follows from Corollary 20 that 1-CCGs cannot generate non-context-free languages, it is not immediately clear that they can generate all context-free languages. To demonstrate this, we employ the classical construction for pure 0-CCGs [28]. Given a CFG, a pure 0-CCG generating the same string language is constructed, such that lexicon entries contain only atomic arguments with leading backward slashes. As a consequence, each derivation tree generated by a CCG with this lexicon can use only backward rules, and all secondary input categories of application rules are atomic. Now consider some derivation tree of the pure 1-CCG with the same lexicon. As argued in the proof of Theorem 31, if a composition rule appears in the derivation tree, we can find a position where a composition is followed by an application. This is because the root category is always obtained through application. We transform the derivation tree by repeated use of rule scheme R2 (see Fig. 9) into a derivation tree that uses only application rules. Note that each use of the transformation rule eliminates a composition from the derivation tree, since y and α are empty due to the properties of the grammar. The transformation does not change the string that labels the leaves, since this particular rotation does not change the order of the three involved subtrees. This shows that the corresponding pure 0-CCG can generate the same string. Thus, the pure 1-CCG generates the same string language as the pure 0-CCG, and therefore the desired context-free language.

We now turn our attention back to the more general discussion. Our main result is that the tree languages generated by CCGs with limited composition depth and rule restrictions are a subset of the tree languages generated by simple monadic context-free tree grammars (Theorem 29). The size of the constructed grammar (Definition 23) is exponential in the maximum arity of categories that occur in the lexicon or as secondary input categories of the given CCG. As mentioned above, the inclusion is proper for pure CCGs (Theorem 31).

The construction used in the classical equivalence proof between CCG and TAG [9] demonstrated that there is no difference in string-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 [16]. Recently, it has been shown that for a given sCFTG a strongly equivalent CCG can be constructed [33]. This is the inverse direction of what we showed in Theorem 29 and, together with our result, proves strong equivalence of CCG and sCFTG, thus characterizing the tree-generative capacity of CCG exactly. It also proves strong equivalence of CCG and TAG because of the strong equivalence of sCFTG and TAG [34]. Further, from the construction is concluded that rule degree 2 and first-order categories suffice to give CCG its full expressive power on trees, and that ε-entries can be avoided. The translation of an sCFTG into a strongly equivalent CCG increases the grammar size only polynomially. However, to remove ε-entries from a CCG, it first has to be converted into an sCFTG, then they are trimmed from the sCFTG, and the result is converted back into a CCG, leading to an overall increase of the grammar size exponential in the maximum arity of categories occurring in the lexicon or as secondary input categories of the given grammar.

Finally, we would like to point out the structural similarity between the rules of the sCFTG in Definition 23 and the parsing algorithm for CCG proposed in [35]. This similarity is not coincidental: Both that parsing algorithm and the constructed sCFTG rely on very similar decompositions of CCG derivations into small, recombinable parts. More generally, we

believe that work on the formal aspects of CCG and work on its parsing can be very fruitful for each other. Future efforts in both areas could study practically relevant extensions of the classical CCG formalism that we considered in this article, and in particular the inclusion of the rules of type raising and substitution (respectively based on the **T** and **S** combinator of combinatory logic), which have so far been left out from much of the theoretical literature on CCG.

CRediT authorship contribution statement

Marco Kuhlmann: Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. **Andreas Maletti:** Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing. **Lena Katharina Schiffer:** Conceptualization, Methodology, Validation, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Y. Bar-Hillel, M. Perles, E. Shamir, On formal properties of simple phrase structure grammars, in: Y. Bar-Hillel (Ed.), Language and Information: Selected Essays on Their Theory and Application, Addison Wesley, 1964, pp. 116–150, Ch. 9.
[2] N. Chomsky, Three models for the description of language, IEEE Trans. Inf. Theory 2 (3) (1956) 113–124.
[3] K. Ajdukiewicz, Die syntaktische Konnexität, Stud. Philos. 1 (1935) 1–27.
[4] Y. Bar-Hillel, A quasi-arithmetical notation for syntactic description, Language 29 (1) (1953) 47–58.
[5] M. Steedman, The Syntactic Process, MIT Press, 2000.
[6] M. Steedman, J. Baldridge, Combinatory categorial grammar, in: R.D. Borsley, K. Börjars (Eds.), Non-Transformational Syntax: Formal and Explicit Models of Grammar, Blackwell, 2011, pp. 181–224, Ch. 5.
[7] M. Schönfinkel, Über die Bausteine der mathematischen Logik, Math. Ann. 92 (3–4) (1924) 305–316.
[8] H.B. Curry, Foundations of combinatorial logic, Am. J. Math. 52 (3) (1930) 509–536.
[9] K. Vijay-Shanker, D.J. Weir, The equivalence of four extensions of context-free grammars, Math. Syst. Theory 27 (6) (1994) 511–546.
[10] M. Lewis, M. Steedman, Unsupervised induction of cross-lingual semantic relations, in: Proc. 2013 EMNLP, ACL, 2013, pp. 681–692.
[11] K. Lee, M. Lewis, L. Zettlemoyer, Global neural CCG parsing with optimality guarantees, in: Proc. 2016 EMNLP, ACL, 2016, pp. 2366–2376.
[12] A.K. Joshi, Y. Schabes, Tree-adjoining grammars, in: G. Rozenberg, A. Salomaa (Eds.), Beyond Words, in: Handbook of Formal Languages, vol. 3, Springer, 1997, pp. 69–123.
[13] J.E. Hopcroft, J.D. Ullman, Introduction to Automata Theory, Languages and Computation, Addison Wesley, 1979.
[14] M. Kuhlmann, A. Koller, G. Satta, Lexicalization and generative power in CCG, Comput. Linguist. 41 (2) (2015) 187–219.
[15] K. Vijay-Shanker, D.J. Weir, Combinatory categorial grammars: generative power and relationship to linear context-free rewriting systems, in: Proc. 26th ACL, ACL, 1988, pp. 278–285.
[16] M. Kuhlmann, G. Satta, P. Jonsson, On the complexity of CCG parsing, Comput. Linguist. 44 (3) (2018) 447–482.
[17] F. Gécseg, M. Steinby, Tree languages, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 3, Springer, 1997, pp. 1–68, Ch. 1.
[18] A. Koller, M. Kuhlmann, Dependency trees and the strong generative capacity of CCG, in: Proc. 12th EACL, ACL, 2009, pp. 460–468.
[19] F. Gécseg, M. Steinby, Tree automata, Tech. Rep., arXiv:1509.06233, 2015.
[20] W.C. Rounds, Context-free grammars on trees, in: Proc. 1st STOC, ACM, 1969, pp. 143–148.
[21] W.C. Rounds, Tree-oriented proofs of some theorems on context-free and indexed languages, in: Proc. 2nd STOC, ACM, 1970, pp. 109–116.
[22] H.-J. Tiede, Deductive systems and grammars: Proofs as grammatical structures, Ph.D. thesis, Indiana University, Bloomington, IN, USA, 1999.
[23] J. Lambek, The mathematics of sentence structure, Am. Math. Mon. 65 (3) (1958) 154–170.
[24] H.B. Curry, R. Feys, W. Craig, Combinatory Logic, Studies in Logic and the Foundations of Mathematics, vol. 1, North-Holland, 1958.
[25] W. Buszkowski, Generative power of categorial grammars, in: Categorial Grammars and Natural Language Structures, 1988, pp. 69–94.
[26] J.-M. Autebert, J. Berstel, L. Boasson, Context-free languages and pushdown automata, in: G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages, vol. 1, Springer, 1997, pp. 111–174, Ch. 3.
[27] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.
[28] Y. Bar-Hillel, H. Gaifman, E. Shamir, On categorial and phrase structure grammars, in: Y. Bar-Hillel (Ed.), Language and Information: Selected Essays on Their Theory and Application, Addison Wesley, 1964, pp. 99–115.
[29] Y. Schabes, A. Abeillé, A.K. Joshi, Parsing strategies with 'lexicalized' grammars: application to tree adjoining grammars, in: Proc. 12th CoLing, 1988, pp. 578–583.
[30] T.A.D. Fowler, G. Penn, Accurate context-free parsing with combinatory categorial grammar, in: Proc. 48th ACL, ACL, 2010, pp. 335–344.
[31] M. Kuhlmann, A. Koller, G. Satta, The importance of rule restrictions in CCG, in: Proc. 48th ACL, ACL, 2010, pp. 534–543.
[32] M. Kuhlmann, A. Maletti, L. Schiffer, The tree-generative capacity of combinatory categorial grammars, in: Proc. Foundations of Software Technology and Theoretical Computer Science, in: LIPIcs, vol. 150, Schloss Dagstuhl — Leibniz-Zentrum für Informatik, 2019, pp. 44:1–44:14.
[33] L.K. Schiffer, A. Maletti, Strong equivalence of TAG and CCG, Trans. Assoc. Comput. Linguist. 9 (2021) 707–720.
[34] S. Kepser, J. Rogers, The equivalence of tree adjoining grammars and monadic linear context-free tree grammars, J. Log. Lang. Inf. 20 (3) (2011) 361–384.
[35] M. Kuhlmann, G. Satta, A new parsing algorithm for Combinatory Categorial Grammar, Trans. Assoc. Comput. Linguist. 2 (Oct 2014) 405–418.