International Journal of Foundations of Computer Science
© World Scientific Publishing Company

SUBREGULAR CATEGORIAL GRAMMARS

MÁRTON MAKRAI*
Computer and Automation Research Institute, Hungarian Academy of Sciences,
H-1111 Budapest, Kende u 13-17, HUNGARY
makrai@sztaki.hu
http://budling.hu/˜makrai

Received (Day Month Year)
Accepted (Day Month Year)
Communicated by (xxxxxxxxxx)

Categorial grammars, though formally equivalent to context-free grammars, are particularly important in natural language processing, where real-time parsability and learnability are major concerns. In this paper we restrict categorial grammars so that the accepted language class turns out to be that of the regular languages and two classes in the subregular hierarchy, non-counting and strictly k-locally testable languages.

Keywords: categorial grammar; regular languages; non-counting languages; strictly k-locally testable languages

1991 Mathematics Subject Classification: 68Q45, 68T50

Introduction

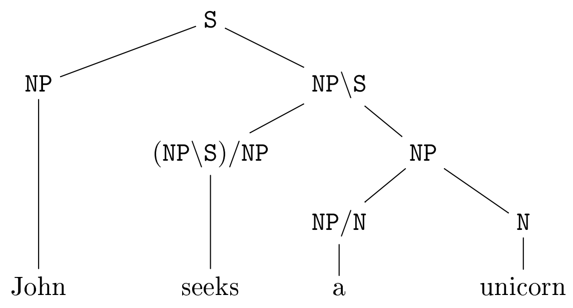
Categorial grammar (CG) is a family of formalisms mainly used in natural language syntax. In their original form [1] CGs are equivalent in generative capacity to context-free grammars. In this paper we restrict categorial grammars so that they accept languages in some narrower language classes distinguished by better parsability and learnability properties.

Categorial grammars are mainly used in natural language syntax organized according to the view that syntactic constituents should generally combine as functions or according to a function-argument relationship. Unlike context-free grammars, categorial grammars are lexicalized, meaning that only a small number of (language-independent) rules are employed, and all other syntactic phenomena derive from the lexical entries of specific words.

The basic idea, going back to Kazimierz Ajdukiewicz [1] and Yehoshua Bar-Hillel [3], is to distinguish between *primitive* and *derived* syntactic *categories*. Typical

*This research was supported by Hungarian Scientific Research Fund “OTKA” No. 82333 (Semantics-based language technology).

Figure 1. A categorial grammar parse tree.



primitive categories are that of a noun (N), a noun phrase (NP e. g. *John*) or a sentence (S). Determiners like *the* or *a* on the other hand, are derived categories NP/N which can be thought of as functions taking a noun phrase as argument and returning sentence (S), so a multiplication-like operation, the *cancellation* $\mathtt{NP/N} \cdot \mathtt{N} \Rightarrow \mathtt{N}$ can be done. Unidirectional categorial grammars with only one kind of fraction / will be defined in Section 1.2.1. In a bidirectional categorial grammar left fractions like intransitive verbs (verbs that have no object, category NP\S) cancel from the left ($\mathtt{NP}{\cdot}\mathtt{NP\backslash S} \Rightarrow \mathtt{S}$) and we also have right fractions like the already discussed NP/N. The bidirectional case will be briefly discussed in 3.1. See Figure 1.

Categorial grammars are used for deciding which category, if any, a sequence of vocabulary items will have.

In 1959, C. Gaifman proved that bidirectional categorial grammars were equivalent in *generative capacity* with Chomsky's type-2 grammars, the context-free grammars, and this result deflected attention from categorial grammars. The proof was later simplified by E. Shamir [4]. From the linguist's point of view, categorial grammar remains interesting because it is a lexicalized framework.

Opinions on the complexity of human languages vary greatly. Hauser, Chomsky and Fitch [10] argue that human languages, in contrast to animal communication, are *recursive*, a notion corresponding to (central) self-embedding in the grammar (for the definition, see section 2.1.1).This view on the boundary between animal and human communication, according to which natural languages are not regular, are highly contested in the linguistics literature, especially by computational linguists, see Rogers and Pullum [21]. So while theoreticians emphasize the comlexity of human language as opposed to animal communication, computational linguists seek for simple solutions of the problems in natural language processing. The study of subregular families takes on added importance in light of this controversy, because may authors feel that both human and animal communication are subregular. Karlsson [13] reveals that center-embedding in language is restricted to a depth of three, which means that there is no recursion in the strict, formal sense. From the computational linguistics point of view, on the other hand, the class of regular languages owes its importance to the fact that it allows real-time recognition.

In this paper first we introduce a variant of CGs accepting all and only regular languages. Next we discuss restriction to the class of *star-free languages* [18], and finally we turn to *strictly k-local(ly testable) languages.* It is conjectured that natural languages are star-free [15]. Locally testable languages are important from the perspective of machine learning, as they can be learned in the sense of Gold [9].

The paper is organized as follows. The preliminaries are discussed in Section 1. We assume the reader to be familiar with automata but not with categorial grammars. We introduce regular, star-free, and strictly k-local languages. Section 2 contains the main results. We make different restrictions on categorial grammars and discuss which language classes the formalisms so obtained will accept. Finally, in Section 3 we show how the standard calculus of regular languages can be used to compute cancellations, the key operations in categorial grammars. Our results complement those of Kanazawa [12], who defined k-valued CGs and showed them to be learnable.

1. Preliminaries

Throughout the paper the reader is assumed to be familiar with formal language theory. For an introduction and standard notation of automata theory (such as *alphabet*, *language*, *regular* language, *deterministic* automaton, *minimal* automaton, *direct product* of automata, *accessible* and *coaccessible* states in an automaton) see [22].

1.1. *Automata, grammars, and languages*

$\mathcal{A} = (Q, \Sigma, \delta, i, F)$ will denote a (possibly) nondeterministic finite-state automaton with a finite set of *states* Q, an *alphabet* Σ, a *transition function* $\delta \subseteq Q \times \Sigma \times Q$, an *initial state* $i \in Q$ and a set of *final states* $F \subseteq Q$.

Context-free grammars will be denoted by $G = (N, T, S, P)$ where N and T is the alphabet of *nonterminals* and *terminals* respectively, $S \in N$ is the start (sentence) symbol and $P \subseteq N \times (N \cup T)^*$ contains the production rules. Elements of P are written as $X \Rightarrow w$. A *regular* grammar is a context-free grammar (N, T, S, P) such that all the production rules in P are of one of the following forms $A \Rightarrow a$, $A \Rightarrow aB$ and $A \Rightarrow \varepsilon$ with $A, B \in N$ and $a \in T$.

We will see that categorial grammars do not accept the *empty string* ε. A language L with $\varepsilon \notin L$ will be called *ε-free.*

We say that two strings $v_1, v_2 \in \Sigma^*$ are syntactically congruent with respect to L iff for every string $u, w \in \Sigma^*$ the equivalence $uv_1w \in L \iff uv_2w \in L$ holds true. This fact will be denoted by $v_1 \sim_L v_2$.

1.2. *Categorial grammars*

1.2.1. *Definition of categorial grammars*

We now define special cases of categorial grammars as rewriting systems. We take the following definitions of rewriting systems from [5].

We require an infinite set of *primitive category variables* C_i, and a finite set of *primitive category constants* collected in $\mathcal{C}_p$. In an *unidirectional* categorial grammar there are binary function symbols / for *division* and $\cdot$ for *concatenation*. The set of *terms* is defined inductively. Variables and primitive categories are terms. If t_1, t_2 are terms, and f is a function symbol, then the string $t_1 f t_2$ is a term. A *category* is a term not containing $\cdot$. The set of categories is denoted by $\mathcal{C}$.

A substring of a term t that is a term itself is a *subterm* of t.

Replacing the subterm u in t by another term results in a term. In particular, one can replace every occurrence of a variable C_i in t by a term t_i. For the sake of brevity, we equip formal variables with a conventional ordering, and we write the result of several such global replacements $C_1 \mapsto t_1$, performed independently as $t(t_1, t_2, \ldots, t_k)$. Replacements allow for empty occurrences.

A *rewrite rule* is an ordered pair of terms written in the form $u \Rightarrow v$. An *instance* of a rewrite rule results from making an identical substitution into variables on both sides: $u(t_1, t_2, \ldots, t_k) \Rightarrow v(t_1, t_2, \ldots, t_k)$. Let $U \Rightarrow V$ be an instance of a rewrite rule, and let U occur in term t; say, $t = t_1 U t_2$. Then it is permitted to rewrite t into $r = t_1 V t_2$.

We have three variants of category systems with respect to which rewrite rules are used. The rule of *application* are common in all the category system defined here:

$$C_1/C_2{\cdot}C_2 \Rightarrow C_1 \tag{2}$$

Associativity of $\cdot$ is enforced by

$$C_1{\cdot}(C_2{\cdot}C_3) \Rightarrow (C_1{\cdot}C_2){\cdot}C_3 \tag{3}$$

We will see in Section 1.2.2 why we do not use the converse as well. We ensure that applications can be done "through brackets" as well by

$$(C_1{\cdot}C_2/C_3){\cdot}C_3 \Rightarrow C_1{\cdot}C_2 \tag{4}$$

In an *application category system* we have rules (2)–(4).

In a *composition category system* we have rules (2)–(4) and the following ones.

$$C_1/C_2{\cdot}C_2/C_3 \Rightarrow C_1/C_3 \tag{5}$$

$$(C_1{\cdot}C_2/C_3){\cdot}C_3/C_4 \Rightarrow C_1{\cdot}C_2/C_4 \tag{6}$$

(5) is called *composition*. Rule (6) is to rule (5) as (4) is to (2). We note that rules called *lifting* and *division* in the literature are not needed here.

A *unidirectional application (respectively composition) categorial grammar* is a tuple $\mathcal{G} = (V, \mathcal{C}, S, f)$, where

- V is a finite set, that of the *vocabulary items*,
- $\mathcal{C}$ is the set of categories in an unidirectional application (respectively composition) category system R,
- $S \in \mathcal{C}_p$ is a distinguished primitive category in $\mathcal{C}$ called the *sentence symbol* and
- $f : V \to 2^{\mathcal{C}}$ is the *assignment function.* We call $f(s)$ the *signature* of symbol s.

(Note that we prefer to denote the set of symbols in an automaton by Σ and the set of vocabulary items in a categorial grammar by V but there's no essential difference and sometimes we even do it the other way round.) We denote transitive-reflexive closure of the rewriting relation of R by $\Rightarrow^*$.

We extend f to strings $w = a_1a_2 \dots a_k$ over V. Then $f(w)$ is the set of categories C, such that there exist categories $C_i \in f(a_i)$ with $C_1 \cdot C_2 \cdots C_k \Rightarrow^* C$. Furthermore, w is *accepted* by $\mathcal{G}$, if $S \in f(w)$. The language $L(\mathcal{G})$ accepted by $\mathcal{G}$ is the set of strings accepted by $\mathcal{G}$.

1.2.2. *Convergence*

Categorial grammars (either application or composition) are *noetherian* in the sense that there do not exist infinitely many terms t_i such that

$$t_0 \Rightarrow t_1 \Rightarrow t_2 \Rightarrow \cdots \Rightarrow t_i \Rightarrow \dots$$

Indeed, all the rules (2)–(6) except for (3) decrease the length of the string (starting, of course, from a finite length), and (3) is a re-bracketing in one direction.

It is *confluent*, i. e. whenever w, t_1 and t_2 are terms such that $w \Rightarrow^* t_1$ and $w \Rightarrow^* t_2$, then a term z exists such that $t_1 \Rightarrow^* z$ and $t_2 \Rightarrow^* z$. A rewrite system that is both noetherian and confluent is *convergent* which means that each equivalence class E of terms under the equivalence relation generated by $\Rightarrow$ contains a canonical representative $n_E \in E$. It is the only term in its equivalence class that rewrites no further: there exists no w such that $n_E \Rightarrow w$. We say that n_E is in *normal form.* By applying any rewrite rule that applies, in any order, every term is transformed into its normal form in finitely many steps.

1.3. ***Subregular languages***

Now we turn to the subregular languages. In the following subsections we will define the star-free and the strictly k-local class respectively following Rogers and Pullum [21].

1.3.1. *Star-free languages*

The greatest well-studied subregular class is the star-free one. We characterize its members in terms of regular expressions, automata, and set-theoretically as well.

Star-free languages are the languages definable by regular expressions which may employ complement but not Kleene-closure.

Given a finite-state automaton $\mathcal{A} = (Q, \Sigma, \delta, i, F)$, some states $Q' \subseteq Q$ and an input string $w \in \Sigma^*$, we say that w induces a permutation on Q' if the state mapping induced by w i.e. $\delta_w|_{Q'} : q \mapsto \delta(q, w)$ is a permutation of Q'. ($\delta_w|_{Q'}$ denotes the restriction of δ_w to Q'.) This means that $\delta_w(q)$ is defined for every $q \in Q'$ and no two different states get mapped to the same state. In this context we call a set of states $Q' \subseteq Q$ *nontrivial* if $|Q'| \geq 2$. Permutations of nontrivial sets will also be called nontrivial. $\mathcal{A}$ is *counter-free* if no string induces a permutation on a nontrivial subset of Q. The term counter-free refers to the fact that the automata cannot do counting modulo some integer. Membership in star-free languages cannot depend on the number of times some factor (subword) occurs modulo some fixed value.

Last but not least, a language L over Σ is *non-counting*, that is, there is some $n > 0$ such that, for all strings u, v, w over Σ, if $uv^n w$ occurs in L then $uv^{n+i}w$, for all $i \geq 1$, occurs in L as well.

Lemma 1 ([18]) *For a language L the following properties are equivalent:*

- *L is star-free.*
- *The minimal automaton of L is counter-free.*
- *L is non-counting.*

1.3.2. *Strictly k-local languages*

We recall the basics of strictly k-local languages from [21]. The base of the subregular hierarchy is the class of *strictly local* languages, those that can be distinguished simply on the basis of which symbols occur adjacently.

A strictly k-local description is a set of k-factors (called k-grams in natural language processing), length k sequences of symbols over the alphabet augmented with start and end symbols. The language such a description defines is the set of strings which include only k-factors from the set. Formally, given a length k, the set of *k-factors* of a string w is:

$$F_k(w) = \begin{cases} \{y \mid w = xyz, \quad x, y, z \in \Sigma^*, |y| = k\} & \text{if } |w| > k, \\ \{w\} & \text{otherwise.} \end{cases}$$

The set of *k-factors* of a language L is the set of k-factors of the strings that it contains:

$$F_k(L) = \bigcup_{w \in L} F_k(w).$$

A *strictly k-local description* $\mathcal{D}$ is a set of k-factors from the alphabet Σ, possibly starting with $\rtimes$ and/or ending with $\ltimes$ where $\rtimes$ and $\ltimes$ are endmarkers not occurring in Σ:

$$\mathcal{D} \subseteq F_k(\{\rtimes\}\Sigma^*\{\ltimes\})$$

A string w satisfies description $\mathcal{D}$ iff the augmented string $\rtimes w \ltimes$ includes only k-factors given in $\mathcal{D}$. $L(\mathcal{D})$, the language defined by $\mathcal{D}$, is given as

$$L(\mathcal{D}) = \{w \in \Sigma^* \mid F_k(\rtimes w \ltimes) \subseteq \mathcal{D}\}.$$

A language that can be defined by a strictly k-local description is *strictly k-local*. If it is strictly k-local for some k, it is *strictly local*.

Example 2. *Let us see an example.* [21] *has shown that the language* $\{(\boldsymbol{ding\ dong})^n \mid n \geq 1\}$ *is a strictly* 2*-local language by giving strictly* 2*-local description*

$$\{\rtimes \boldsymbol{ding}, \boldsymbol{ding\ dong}, \boldsymbol{dong\ ding}, \boldsymbol{dong}\ltimes\}.$$

The characteristic property of strictly local languages is the following

Lemma 3 (Suffix Substitution Closure [18, 17]) *A language L is strictly k-local iff whenever there is a string x of length $k-1$ and strings u_1, v_1, u_2 and v_2, such that $u_1xv_1 \in L$ and $u_2xv_2 \in L$ then $u_1xv_2 \in L$ as well.*

2. Subregular categorial grammars

Categorial grammars are in the general case equivalent with context-free grammars. In the simplified proof of this fact Shamir [4] actually used some restriction of the set of permitted categories that do not decrease the generative capacity. Now we seek restrictions on categorial grammars that will make the class of languages accepted by such grammars one of the narrower classes introduced in Section 1.3.

In the following three sections we define variants of categorial grammars equivalent to the regular, star-free, and strictly k-local class respectively

2.1. *Regular categorial grammars*

In the following two sections we give categorial grammars for regular languages. First we paraphrase a characterization of regular grammars among context-free ones due to Chomsky [7] in terms of categorial grammars. After that we characterize regular categorial grammars with restrictions on the category set.

2.1.1. *Not self-embedding categorial grammars*

Now we paraphrase a characterization of regular grammars among context-free ones due to Chomsky [7] in terms of categorial grammars. We define self-embedding categorial grammars in a way analogous to self-embedding context-free grammars.

A context-free grammar $G = (N, T, S, P)$ is *self-embedding* iff $X \Rightarrow^* PXQ$ for some $X \in N$, $P, Q \in (N \cup T)^+$. A context-free language L is self-embedding iff all context-free grammars generating L are self-embedding.

Definition 4. *A categorial grammar* $(V, \mathcal{C}, S, f)$ *is* self-embedding *if there exists a string* $w = a_1 a_2 .... a_k \in V^*$*,* $C_i \in f(a_i)$*, and an index* $1 < j < k$ *such that* $C = C_j$ *and* $C_1 \cdot C_2 \cdots C_k$ *cancels to* C_j*.*

Note that given a self-embedding grammar (generative or categorial) it does not follow that the corresponding language is properly context-free (i.e. not regular).

Theorem 5. *An* ε*-free language is regular iff it is accepted by an application categorial grammar which is* not *self-embedding.*

Proof. First assume that L is regular. A language is regular iff it is context-free and not self-embedding [7] so there exists a context-free grammar G generating L that is not self-embedding. [4] shows that for any context-free grammar there exists an equivalent categorial grammar. Consider the categorial grammar $(V, \mathcal{C}, S, f)$ corresponding to G according to [4]. Notice that every category C is assigned to at most one symbol.

To see that $(V, \mathcal{C}, S, f)$ preserves the non-self-embedding property, recall that in the proof by [4], signatures are extended to nonterminals of the context-free grammar. With this extension we get that given nonterminals A_i $(i = 0, \ldots, r)$ and categories $C_i \in f(A_i)$, cancellation $C_1 \ldots C_r \Rightarrow^* C_0$ implies $A_0 \Rightarrow^* A_1, \ldots, A_r$, so $(V, \mathcal{C}, S, f)$ is not self-embedding.

Conversely, given a categorial grammar that is not self-embedding, the context-free grammar constructed to it according to [4] will also be not self-embedding. □

2.1.2. *Characterizing regular CGs by restrictions on the category set*

Now we give a second characterization of regular categorial grammars by restrictions on the category set.

Definition 6. *A* (right) regular categorial grammar *is a composition categorial grammar* $(V, \mathcal{C}, S, f)$*, in which for any symbol* $a \in V$*, elements of* $f(a)$ *are of the form* C_1 *or* C_1/C_2*, where* C_1 *and* C_2 *are* primitive *categories.*

No categorial grammar accepts the empty string. However the following proposition can be stated.

Proposition 7. *An* ε*-free language* L *is accepted by a right regular categorial grammar iff it is regular.*

Proof. Consider the bijection between right regular categorial grammars of the form $\mathcal{G} = (V, \mathcal{C}, S, f)$ with *primitive* categories $\mathcal{C}_p$ and the right regular grammars $\mathcal{R} = (\mathcal{C}_p, V, S, P)$ where

$$A \to a \in P \ \text{ iff } \ A \in f(a) \quad \text{and}$$
$$A \to aB \in P \ \text{ iff } \ A/B \in f(a)$$

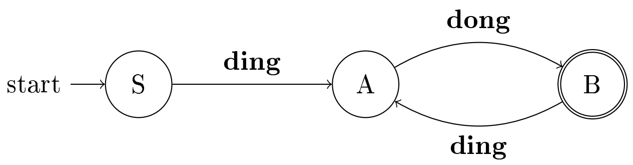


Figure 2. The minimal automaton of language $\{(\textbf{ding dong})^n \mid n \geq 1\}$

Note that primitive categories in $\mathcal{G}$ correspond to nonterminals in $\mathcal{R}$. Clearly $\mathcal{R}$ generates the same strings that $\mathcal{G}$ accepts. □

Example 8. *Consider the automaton in Figure 2 that accepts the language* $\{(\textbf{\textit{ding dong}})^n \mid n \geq 1\}$. *(We remind the reader that no categorial grammar accepts the empty string, so* $(\textbf{\textit{ding dong}})^*$ *is not an appropriate example.) The corresponding regular categorial grammar is* $(V, \mathcal{C}, S, f)$ *with*

- $V = \{\textbf{\textit{ding}}, \textbf{\textit{dong}}\}$
- $\mathcal{C} = \{S, A, B\}$
- $f(\textbf{\textit{ding}}) = \{S/A, B/A\}$, $f(\textbf{\textit{dong}}) = \{A/B, A\}$

The string ***ding dong ding dong*** *is assigned* $S/A \cdot A/B \cdot B/C \cdot C \Rightarrow^* S$ *while* ***ding ding*** *is not accepted.*

Remark 9. *The bijection between right regular categorial grammars and the right regular grammars above extends to finite-state automaton in the standard way. Here primitive categories in* $\mathcal{G}$ *correspond to states.*

2.2. *Star-free variants of categorial grammars*

Now we turn to star-free languages. In the following two sections we restate the definition and the automata-theoretic characterization of this class in terms of categorial grammars.

2.2.1. *CG-style definition of star-free languages*

We define non-counting categorial grammars as follows.

Definition 10. *A* non-counting categorial grammar *is a regular categorial grammar* $(V, \mathcal{C}, S, f)$ *such that for every string* $w \in V^*$ *there exists a natural number* n *such that*

$$f(w^n) = f(w^{n+1}). \tag{13}$$

Example 11. *We take an example from* [21], *the language* L *over the alphabet* $V = \{a, b, c\}$ *in which any* c *is preceded by some* b. L *is given by the automaton in Figure 3.*

The corresponding categorial grammar is $(V, \mathcal{C}, S, f)$ *where:*

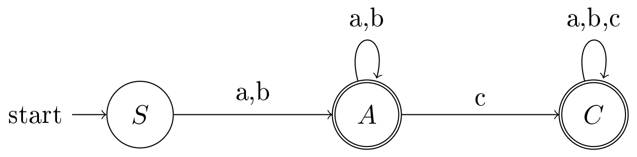


Figure 3. The minimal automaton of language L in Example 11

- $\mathcal{C} = \{S, A, C\}$
- $f(a) = f(b) = \{S/A, A/A, C/C, S, A, C\}$, $f(c) = \{A/C, C/C, A, C\}$

An example derivation:

$$f(abbaa) \ni S/S \cdot S/C \cdot C/C \cdot C/C \cdot C \Rightarrow^* S$$

Now we compute signatures.

$$\begin{aligned} f(aa) &= f(a) \\ f(ab) = f(ba) = f(bb) = f(bc) &= f(b) \\ f(ac) = f(ca) = f(cb) = f(cc) &= f(c) \end{aligned}$$

We see that $f(s^2) = f(s)$ for every $s \in V$. Furthermore $f(w^2) = f(w)$ for every $w \in V^*$ since signatures of 2-letter strings (and thus those of longer strings as well) coincide with those of single symbols. So 1 is a uniform n such that $f(w^n) = f(w^{n+1})$ i. e. this language is non-counting.

Now we state the main theorem in this section.

Theorem 12. *An ε-free language is non-counting iff it is accepted by a non-counting categorial grammar.*

This theorem is a consequence of the following Proposition.

Proposition 13. *Given a categorial grammar $\mathcal{G} = (V, \mathcal{C}, S, f)$ accepting a language L and strings $w_1, w_2 \in V^*$, the implication*

$$f(w_1) = f(w_2) \implies w_1 \sim_L w_2 \tag{15}$$

holds, where $\sim_L$ denotes the syntactic congruence with respect to L. If $\mathcal{G}$ is the categorial grammar constructed from the minimal automaton $\mathcal{A} = (Q, V, \delta, i, F)$ of L according to Proposition 7, then the converse of (15) also holds.

Proof. (15) is obvious. For the converse, recall that states of $\mathcal{A}$ can be identified with primitive categories in $\mathcal{G}$. For any string $x \in V^*$ and primitive categories D_1, D_2 the equivalence

$$D_1/D_2 \in f(x) \iff \delta(D_1, x) = D_2 \tag{16}$$

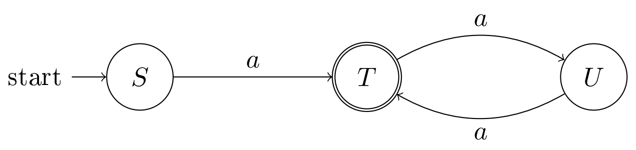


Figure 4. A counting automaton

holds.

On the other hand, given $w_1 \sim_L w_2$, the strings w_1 and w_2 induce the same state mapping in $\mathcal{A}$. $\mathcal{A}$ has exactly one final state q_f. Given (16), it remains to be proved that for every *primitive* category C, the containments $C \in f(w_1)$ and $C \in f(w_2)$ are equivalent. Indeed, $C \in f(w_1) \iff \delta(C, w_1) = q_f \iff \delta(C, w_2) = q_f \iff C \in f(w_2)$. $\square$

2.2.2. *Automata-theoretic properties in terms of CGs*

Now we turn to an other formalism for star-free languages. This time categorial grammars will be restricted following automata-theoretic considerations. Before giving the definition, consider the "counting" (i. e. not non-counting) language $\left(a^2\right)^* a = \{a^{2k+1} \mid k = 0, 1\dots\}$ with the automaton in Figure 4 and categorial grammar $\mathcal{G} = (V, \mathcal{C}, S, f)$ where $V = \{a\}$, $\mathcal{C} = \{S, T, U\}$ and $f(a) = \{S/T, T/U, U/T, S, U\}$. Here a permutes ("interchanges") states T and U.

We define permutation-free categorial grammars as follows.

Definition 14. *Given a categorial grammar $\mathcal{G} = (V, \mathcal{C}, S, f)$, distinct categories $C_1, C_2, C_3 \dots C_k$ and a string w over V we say that w induces the (cyclic) permutation $(C_1, C_2, C_3 \dots, C_k)$ in $\mathcal{G}$, if*

$$C_1/C_2, C_2/C_3, \dots, C_k/C_1 \in f(w).$$

A permutation-free categorial grammar *is a regular categorial grammar, in which no string induces a permutation $(C_1, C_2, C_3 \dots, C_k)$ with $k > 1$.*

Returning to our example, we see that a induces the permutation (S, A) in the grammar $\mathcal{G}$ accepting the counting language $\left(a^2\right)^*$, so $\mathcal{G}$ is not permutation-free. In the following we show that this properties is necessary. We first present an auxiliary statement.

Lemma 15. *Given a counter-free automaton $\mathcal{A} = (Q, \Sigma, \delta, i, F)$, the deterministic automaton $\mathcal{A}_d = (Q_d, \Sigma, \delta_d, i_d, F_d)$ obtained from $\mathcal{A}$ by the subset construction (See* [8]*) is also counter-free .*

Proof of the lemma. States of $\mathcal{A}_d$ can be identified with subsets of Q. Let $Q(q_d) \subseteq Q$ denote the subset corresponding to $q_d \in Q_d$. Suppose that there exists a nontrivial subset $R_d \subseteq Q_d$ and input string $w \in \Sigma^*$ such that $\delta_w|_{R_d} : q \mapsto \delta(q, w)$ is a permutation of R_d and let

$$R = \bigcup_{r_d \in R_d} Q(r_d).$$

$R \subseteq Q$ is clearly nontrivial. Given that R is finite, to conclude that $\delta_w|_R : q \mapsto \delta(q, w)$ is a permutation, it suffices to show that $\delta_w|_R$ is surjective. Indeed, for every $r \in R$ there exists a $r_d \in R_d$ such that $r \in Q(r_d)$. Being $\delta_w|_{Q'}$ a permutation, there exists $s_d \in R_d$, such that $\delta_d(s_d, w) = r_d$, which by construction of $\mathcal{A}_d$ means that there exists a $s \in Q(s_d) \subseteq R$, such that $\delta(s, w) = r$. □

Theorem 16. *An ε-free language L is non-counting iff it is accepted by a permutation-free categorial grammar.*

Proof. Recall that a language L is non-counting iff its minimal automaton is counter-free (See Lemma 1 in Section 1.3.1). Consider the correspondence between automata $\mathcal{A}$ and regular categorial grammars $\mathcal{G}$ introduced in Proposition 7. Remember that states of $\mathcal{A}$ can be identified with primitive categories in $\mathcal{G}$. It follows from (16) that any string x induces a nontrivial *cyclic* permutation in $\mathcal{R}$ iff it does in $\mathcal{A}$. Thus any non-counting language L is accepted by a permutation-free categorial grammar.

Conversely, given a permutation-free categorial grammar $\mathcal{G}$, no string induces a nontrivial cyclic permutation in automaton $\mathcal{A}_\mathcal{G} = (\mathcal{C}_p, V, \delta, i, F)$. It is also true that no string induces any nontrivial permutation in $\mathcal{A}_\mathcal{G}$. Indeed, suppose by contraposition that a string w induces a permutation π on some nontrivial subset $Q' \subseteq Q$. It can be seen from the disjunct cycle decomposition on π that w also induces a nontrivial cyclic a permutation on some subset $Q'' \subseteq Q'$. We have that $\mathcal{A}_\mathcal{G}$ is permutation-free. It follows from Lemma 15 that the subset construction of the minimal automaton preserves the permutation-free property, in other words, that the minimal automaton recognizing the language L that $\mathcal{G}$ accepts is counter-free, thus L is non-counting. □

2.3. *Strictly k-local categorial grammars*

Finally we turn to strictly k-local languages. We define strictly k-local categorial grammars in the following way.

Definition 17. *A strictly k-local categorial grammar is a regular categorial grammar $(V, \mathcal{C}, S, f)$ in which for every $w \in V^{k-1}$ the implication*

$$C_1/C_2, C_3/C_4 \in f(w) \implies C_2 = C_4 \tag{19}$$

holds.

Theorem 18. *A ε-free language is accepted by a strictly k-local categorial grammar iff it is strictly k-local.*

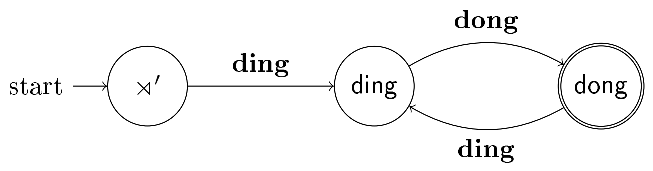


Figure 5. The automaton corresponding to language $\{(\mathbf{ding\ dong})^n \mid n \geq 1\}$

Proof. Let L be the language accepted by a strictly k-local categorial grammar $\mathcal{G}$. It suffices to show that whenever there is a string w of length $k-1$ and strings u_1, v_1, u_2 and v_2 , such that $u_1wv_1 \in L$ and $u_2wv_2 \in L$ then $u_1wv_2 \in L$ as well. $\mathcal{G}$ is regular, so it follows from $u_1wv_1, u_2wv_2 \in L$ that there exist primitive categories C_i and D_i such that $C_i \in f(v_i)$, $D_i/C_i \in f(w)$ and $S/Di \in f(u_i)$. It follows from (19) that $C_1 = C_2$. Consequently u_1wv_2 is assigned the category $S/D_1 \cdot D_1/C_1 \cdot C_1 = S/D_1 \cdot D_1/C_2 \cdot C_2$ which cancels to S, so $u_1wv_2 \in L$.

Given a k-local language, consider the strictly $k-1$-local description $\mathcal{D} = F_{k-1}(\rtimes L \ltimes)$ and the (typically not fully specified) automaton $\mathcal{A} = (\mathcal{D}, V, \delta, \rtimes, F)$ where

$$\delta(x, a_k) = \begin{cases} xa_k & \text{if } |x| < k-1 \text{ and } xa_k \in \mathcal{D} \\ a_2a_3 \cdots a_k & \text{if } x = a_1a_2 \cdots a_{k-1} \text{ and } a_2a_3 \cdots a_k \in \mathcal{D} \end{cases}$$
$$F = \{a_1a_2 \cdots a_{k-1} \mid a_2 \cdots a_{k-1} \ltimes \in \mathcal{D}\}$$

An example is given below. $\mathcal{A}$ is in state x iff the longest suffix s of the input with $|s| \leq k-1$ is x. Consider the categorial grammar $\mathcal{G} = (V, \mathcal{C}, S, f)$ corresponding to $\mathcal{A}$ according to Proposition 7. Elements of $\mathcal{D}$ are identified with primitive categories of $\mathcal{G}$. To show (19), let w be of length $k-1$ and $C_1/C_2, C_3/C_4 \in f(w)$. Then by (16) we have $\delta(C_1, w) = C_2, \delta(C_3, w) = C_4$ so $C_2 = w = C_4$. □

Example 19. *The automaton corresponding to the strictly* 2*-local language*

$$\{(\boldsymbol{ding\ dong})^n \mid n \geq 1\}$$

that we have seen in Example 2 is in Figure 5.

The corresponding categorial grammar is $(V, \mathcal{C}, S, f)$ *where* $V = \{\boldsymbol{ding}, \boldsymbol{dong}\}$, $\mathcal{C} = \{S, \mathsf{ding}, \mathsf{dong}\}$, $S = \rtimes'$, *and* $f(\boldsymbol{ding}) = \{\rtimes'/\mathsf{ding}, \mathsf{dong}/\mathsf{ding}\}, f(\boldsymbol{dong}) = \{\mathsf{ding}/\mathsf{dong}, \mathsf{ding}\}$. *States are identified by* 1*-length suffixes and vocabulary items correspond to symbols of the automaton. Derived categories correspond to transitions while* $\mathsf{ding} \in f(\boldsymbol{dong})$ *corresponds to the fact that there is a transition labeled* ***dong*** *from state* ding *to the accepting state. It is accidental that there is a unique accepting state. The existence of such a grammar is what we have sated in Theorem 18.*

3. Categorial grammars by automata calculus

In this section we show how the standard calculus of regular languages can be used to compute cancellations. Implementations of these operations will be discussed in

3.1.

In the preceding section we represented regular categorial grammars as finite state automata. In this section we represent (concatenations of) categories as finite state automata. We emphasize the difference. In the preceding section symbols of the automata were vocabulary items of the categorial grammars. (Primitive) categories were represented as states. In this section states of the automata do not correspond to anything meaningful in the grammar.

Let the alphabet Σ contain the primitive categories, the operation symbol / and parens. Concatenation will not be marked explicitly. This way a category can be viewed as a string over Σ, and the signature of a vocabulary item can be viewed as a language over Σ. Languages so obtained will also be called *signature*s.

Let start with some language theoretic preliminaries. Given languages A and B the *quotient*s $A^{-1}B$ and BA^{-1} are defined as

$$A^{-1}B = \{w \in \Sigma^* \mid \exists a \in A, aw \in B\}$$
$$BA^{-1} = \{w \in \Sigma^* \mid \exists a \in A, wa \in B\}$$

Furthermore, to simplify notation, given a signature S let $S^{()}$ denote $S \cup (S)$, elements of S in optional brackets.

The main operation on signatures is $\circ$. $S_1 \circ S_2$ is the set of strings corresponding to categories which can be obtained from categories corresponding to elements of S_1 and S_2 by *right application*

$$C_1/C_2{\cdot}C_2 \Rightarrow C_1$$

where the category used as the left (respectively right) argument of $\cdot$ has to be taken from S_1 (respectively S_2). Now we formulate how cancellations can be computed with regular operations and division.

Proposition 20. *Cancellations can be computed in the following way:*

$$S_1 \circ S_2 = S_1(/S_2^{()})^{-1}$$

$S^{()}$ and $\circ$ are obtained by combined use of restricted regular operations (more precisely union and concatenation) and division.

Now we turn to the automata operations corresponding to these language operations. Automata operations corresponding to union and concatenation are well-known. We therefore discuss division in detail. We will rely on the notation and propositions in Eilenberg [8].

Divisions and automata

Let $\mathcal{A} = (Q_1, \Sigma, \delta_1, i_1, F_1)$ and $\mathcal{B} = (Q_2, \Sigma, \delta_2, i_2, F_2)$ be deterministic automata recognizing languages A and B respectively We define the *division* operation on automata by

$$\mathcal{B}^{-1}\mathcal{A} = (Q_1, \Sigma, \delta_1, i_1 B, F_1)$$
$$\mathcal{A}\mathcal{B}^{-1} = (Q_1, \Sigma, \delta_1, i_1, F_1 B^{-1})$$

where

$$i_1B = \{\delta_1(i_1, b) \mid b \in B\}$$
$$F_1B^{-1} = \{q \in B_1 \mid \delta(q, b) \in F_1 \text{ with } b \in B, \}$$

Clearly $|\mathcal{B}^{-1}\mathcal{A}| = B^{-1}A$ and $|\mathcal{A}\mathcal{B}^{-1}| = AB^{-1}$.

The sets i_1B and F_1B^{-1} can be defined in terms of automata as well. Let $\mathcal{A} \times \mathcal{B} = (Q, \Sigma, \delta, i, F)$ be direct product of automata $\mathcal{A}$ and $\mathcal{B}$ recognizing $A \cap B$. Then

$$i_1B = \{q \in Q_1 | \delta((i_1, i_2), w) = (q, f) \text{ with } w \in \Sigma^* \text{ and } f \in F_2\}.$$

In other words i_1B is the set of q's such that (q, f) is accessible in $\mathcal{A} \times \mathcal{B}$. F_1B^{-1} can be obtained by duality:

$$F_1B^{-1} = \{q \in Q_1 | \delta((q, i_2), w) = (f_1, f_2) \text{ with } w \in \Sigma^* \text{ and } f_j \in F_j\}$$

i.e. F_1B^{-1} is the set of q's such that (q, i_2) is coaccessible in $\mathcal{A} \times \mathcal{B}$.

3.1. *Implementation*

A kind of categorial grammar—namely bidirectional application categorial grammar—has been implemented in the functional programming language *Haskell*. In *bidirectional* categorial grammars we have a the binary function symbol \ in addition to / and · with the rule *left application*

$$C_2 \cdot C_2 \backslash C_1 \Rightarrow C_1,$$

the counterpart of (2). More details and the code itself can be found at

`http://www.math.bme.hu/~makraim/thesis/AutomataCalculus.hs`

Automata calculus has been implemented in Haskell earlier [24]. Our implementation differs from [24] it that it does not involve ε-moves, uses the standard `Set` data structure and lets automata have a set of initial states rather than restricting ourselves to a single start state.

4. Conclusion

We have defined regular, star-free and strictly k-local categorial grammars that accept all and only the languages belonging to the class of regular, star-free, and strictly k-local languages respectively. We have implemented a simple calculus for categories.

There are many more subregular language classes, such as locally testable languages [6, 25, 17], concatenations of (strictly) locally testable languages [14], locally threshold testable languages (in the strict sense) [23], (strictly) piecewise testable languages [20], neighborhood-distinct, left-to-right iterative, and right-to-left iterative languages [11], reversible languages in two different senses ([16, 2] and [19]), just to name a few. The connection of these to categorial grammars remains an open question.

Acknowledgements

I would like to thank András Kornai for constant help and encouragement, Mátyás Naszódi, László Kálmán, and Dávid Nemeskey for useful comments and Gergely Patai for his wholehearted help with Haskell.

This research was supported by Hungarian Scientific Research Fund "OTKA" No. 82333 (Semantics-based language technology).

Bibliography

[1] Kazimierz Ajdukiewicz. Die syntaktische konnexität. *Studia Philosophica*, 1:1–27, 1935.

[2] Dana Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.

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

[4] Yehoshua Bar-Hillel, Chaim Gaifman, and Eli Shamir. On categorial and phrase structure grammars. *Bulletin of the Research Council of Israel*, 9F:1–16, 1960.

[5] Tibor Beke. Categorification, term rewriting and the Knuth–Bendix procedure. *Journal of Pure and Applied Algebra (to appear)*, 2010.

[6] J.A. Brzozowski and Imre Simon. Characterizations of locally testable events. *Discrete Mathematics*, 4:243–271, 1973.

[7] Noam Chomsky. On certain formal properties of grammars. *Information and Control*, 2:137–167, 1959.

[8] Samuel Eilenberg. *Automata, Languages, and Machines*, volume A. Academic Press, 1974.

[9] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[10] Marc D. Hauser, Noam Chomsky, and W. Tecumseh Fitch. The faculty of language: What is it, who has it, and how did it evolve? *Science*, 298:1569–1579, 11 2002.

[11] Jeffrey Heinz. Learning phonotactic grammars from surface forms. In Donald Baumer, David Montero, and Michael Scanlon, editors, *Proceedings of the 25th West Coast Conference of Formal Linguistics*, 2006. University of Washington, Seattle.

[12] Makoto Kanazawa. Identification in the limit of categorial grammars. *Journal of Logic, Language and Information*, 5(2):115–155, 1996.

[13] Fred Karlsson. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 2007.

[14] Satoshi Kobayashi and Takashi Yokomori. Learning concatenations of locally testable languages from positive data, 1994.

[15] Andras Kornai. *Mathematical Linguistics*. Springer Verlag, 2008.

[16] R. McNaughton. The loop complexity of pure-group events. *Information and Control*, 11(1-2):167–176, 1967.

[17] Robert McNaughton. Algebraic decision procedures for local testability. *Mathematical Systems Theory*, 8(1):60–76, 1974.

[18] Robert McNaughton and Seymour Papert. *Counter-Free Automata*. MIT Press, 1971.

[19] Jean-Eric Pin. *On Reversible Automata*. LITP, Institut Blaise Pascal, 1992.

[20] James Rogers, Jeffrey Heinz, Gil Bailey, Matt Edlefsen, Molly Visscher, David Wellcome, and Sean Wibel. On languages piecewise testable in the strict sense. In Christian Ebert, Gerhard Jäger, and Jens Michaelis, editors, *The Mathematics of Language*, volume 6149 of *Lecture Notes in Artifical Intelligence*, pages 255–265. Springer, 2010.

[21] James Rogers and K. Pullum. Aural pattern recognition experiments and the subregular hierarchy, 2007.
[22] Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of formal languages.* Springer-Verlag New York, Inc., New York, NY, USA, 1997.
[23] José Ruiz, Salvador España Boquera, and Pedro García. Locally threshold testable languages in strict sense: Application to the inference problem. In *ICGI*, pages 150–161, 1998.
[24] Simon Thompson. Regular Expressions and Automata using Haskell. Technical Report 5-00, Computing Laboratory, University of Kent, January 2000.
[25] Zalcstein. Locally testable semigroups. *Semigroup Forum 5*, 1973.