

Grammars over the Lambek Calculus with Permutation: Recognizing Power and Connection to Branching Vector Addition Systems with States

Tikhon Pshenitsyn

*Department of Mathematical Logic and Theory of Algorithms,
Lomonosov Moscow State University, GSP-1, Leninskie Gory,
Moscow, 119991, Russia.

Corresponding author(s). E-mail(s): ptihon@yandex.ru;

Abstract

In [Van Benthem, 1991] it is proved that all permutation closures of context-free languages can be generated by grammars over the Lambek calculus with the permutation rule (**LP**-grammars); however, to our best knowledge, it is not established whether converse holds or not. In this paper, we show that **LP**-grammars are equivalent to linearly-restricted branching vector addition systems with states and with additional memory (shortly, IBVASSAM), which are modified branching vector addition systems with states. Then an example of such an IBVASSAM is presented, which generates a non-semilinear set of vectors; this yields that **LP**-grammars generate more than permutation closures of context-free languages. Moreover, equivalence of **LP**-grammars and IBVASSAM allows us to present a normal form for **LP**-grammars and, as a consequence, prove that **LP**-grammars are equivalent to **LP**-grammars without product.

Keywords: Lambek calculus, categorial grammar, formal language, vector addition system, branching vector addition system with states

1 Introduction

In the formal grammar theory, there are two families of approaches, which are, in some sense, opposed to each other: generative grammars and categorial grammars. Generative grammars are rule-based: a string belongs to a language generated by a grammar if and only if this string can be produced from a fixed start object of the grammar using rewriting rules specified in it. Categorial grammars work in an opposite way in the sense that a grammar does not produce a string step-by-step but rather takes a whole string in the first place and *proves* that it belongs to its language. Hence to define any particular kind of categorial grammars we must specify what proof mechanism is used in its core.

One of prominent kinds of categorial grammars is Lambek categorial grammars. They are based on the Lambek calculus L, which is a logic designed to model syntax of natural languages [1]. This is a substructural logic of intuitionistic logic; namely, it is obtained from the latter by dropping structural rules of weakening, contraction, and permutation. In the Lambek calculus, types (i.e. formulas) are built from atomic ones using three operations: the left division \backslash , the right division $/$, and the product \cdot . Two divisions correspond to the implication in the intuitionistic logic; two variants of the implication arise since the order of types matters in this substructural logic. Following [1] we consider *sequents* as provable objects in L, which are structures of the form $A_1, \dots, A_n \rightarrow B$ where $n > 0$ and A_i, B are types. Finally, we define a Lambek grammar as a finite correspondence between symbols of an alphabet and types of L; besides, in a grammar, we must fix some *distinguished type* S . Then a string $w = a_1 \dots a_n$ belongs to the language generated by such a grammar if and only if we can replace each symbol a_i by a type T_i corresponding to it in such a way that the sequent $T_1, \dots, T_n \rightarrow S$ is derivable in L.

One of the famous results proved in [2] is that Lambek grammars generate only context-free languages (the converse, i.e., that each context-free language without the empty word is generated by some Lambek grammar, was proved in [3] in 1960). The proof uses several delicate tricks involving free-group interpretations and so-called binary-reduction lemma.

After the seminal work [1] of Lambek, numerous modifications and extensions of L have been introduced for different purposes. For each such modification one can define a corresponding class of categorial grammars in the same way as Lambek grammars are defined and then study what class of languages new grammars generate. In particular, it is interesting to check if the ideas behind the theorem proved by Pentus in [2] fit in a new class of categorial grammars. This work follows this agenda: we are going to investigate what languages can be generated by categorial grammars based on the Lambek calculus with the permutation rule (denoted as LP). This calculus is obtained from L by allowing one to freely change the order of types in left-hand sides of sequents. It is used and studied in, e.g., [4, 5]. In particular, in [5], it is proved that LP-grammars generate all permutation closures of context-free languages; however, the converse statement was neither proved nor disproved

in that paper. Stepan L. Kuznetsov [6] introduced this problem to me conjecturing that there is a counterexample to the converse statement. To my best knowledge, this question is still open; for instance, in [7, p. 230] the question of existence of a Pentus-like proof for LP is mentioned as an open one.

In this paper, we answer this question and show that LP-grammars generate some languages that are not permutation closures of context-free languages (hence confirming Kuznetsov's conjecture). This is done by introducing an equivalent formalism called *linearly-restricted branching vector addition systems with states and additional memory (LBVASSAM)*¹. We prove that LBVASSAM generate exactly Parikh images of languages generated by LP-grammars; the proof is inspired by a proof of the fact that double-pushout hypergraph grammars with a linear restriction on length of derivations can be embedded in hypergraph Lambek grammars [8]. In the latter work, we deal with quite a general formalism extending the Lambek calculus; nicely, working with it provided us with methods applicable to LP-grammars as well.

The transformation of an LP-grammar into an equivalent LBVASSAM and vice versa presented in this paper also allows us to prove some nice facts about LP-grammars. For example, we can prove that LP-grammars are equivalent to LP(/)-grammars, i.e. grammars based on the Lambek calculus with permutation and with division only. Another observation is that LP-grammars are equivalent to LP-grammars of order 2, i.e. to grammars with the maximal nesting depth of divisions being equal to 2.

In Section 2, we introduce some preliminary notions and formally define the Lambek calculus with permutation LP along with LP-grammars. In Section 3, we define linearly-restricted branching vector addition systems with states and additional memory. In Section 4, we prove the equivalence theorem.

2 Preliminaries

Let us start with clarifying some notation used in the remainder of the paper.

\mathbb{N} contains 0. Σ^* is the set of all strings over the alphabet Σ (including the empty string ε). $\mathcal{M}(\Sigma)$ is the set of all finite nonempty multisets with elements from Σ . In this paper, we call a language any subset of $\mathcal{M}(\Sigma)$. The length $|w|$ of a multiset w is its cardinality; by $|w|_a$ we denote the number of occurrences of an element a in w . The size $|v|$ of a vector $v = (v_1, \dots, v_k) \in \mathbb{N}^k$ equals $v_1 + \dots + v_k$. By e_i we denote the standard-basis vector $(0, \dots, 0, 1, 0, \dots, 0)$ where 1 stands at the i -th position.

If $\Sigma = \{a_1, \dots, a_k\}$ is a finite alphabet (with a fixed enumeration of symbols from 1 up to k), then Parikh image of a multiset $w \in \mathcal{M}(\Sigma)$ is defined as $\pi(w) = (|w|_{a_1}, \dots, |w|_{a_k})$. This definition is generalized to languages in an obvious way: $\pi(L) = \{\pi(w) \mid w \in L\}$ ($L \subseteq \mathcal{M}(\Sigma)$). We can also consider inverse Parikh image: $\pi^{-1}(V) = \{w \in \mathcal{M}(\Sigma) \mid \pi(w) \in V\}$.

¹We apologize to the reader for such long abbreviations.

2.1 Lambek Calculus With Permutation

In this section, we define the Lambek calculus with permutation LP in the Gentzen style. We fix a countable set of *primitive types* Pr and define the set of *types* as follows: $Tp := Pr \mid Tp/Tp \mid Tp \cdot Tp$ (in contrast to the Lambek calculus without permutation, we do not need to introduce the left division \backslash here). A *sequent* is a structure of the form $A_1, \dots, A_n \rightarrow B$ where $n > 0$, and A_i, B are types. The sequence A_1, \dots, A_n is called an *antecedent*, and B is called a *succedent*.

The only axiom of LP is $A \rightarrow A$. There are five inference rules:

$$\frac{\Pi \rightarrow A \quad \Gamma, B \rightarrow C}{\Gamma, B/A, \Pi \rightarrow C} (\diagup \rightarrow) \quad \frac{\Pi, A \rightarrow B}{\Pi \rightarrow B/A} (\rightarrow \diagup)$$

$$\frac{\Gamma, A, B \rightarrow C}{\Gamma, A \cdot B \rightarrow C} (\cdot \rightarrow) \quad \frac{\Pi \rightarrow A \quad \Psi \rightarrow B}{\Pi, \Psi \rightarrow A \cdot B} (\rightarrow \cdot)$$

$$\frac{\Gamma, A, B, \Delta \rightarrow C}{\Gamma, B, A, \Delta \rightarrow C} (\text{perm})$$

Here small Latin letters p, q, r, \dots represent primitive types; capital Latin letters A, B, C, \dots represent types; capital Greek letters Γ, Δ, \dots represent sequences of types (and Π, Ψ denote nonempty sequences). The notation $\mathcal{K} \vdash \Pi \rightarrow A$ means that the sequent $\Pi \rightarrow A$ is derivable in a calculus \mathcal{K} .

Furtermore $A \times k$ is a shorthand notation for $\underbrace{A, \dots, A}_{k \text{ times}}$, and $A^k := \underbrace{A \cdot \dots \cdot A}_{k \text{ times}}$.

Remark 1 The rule (perm) enables us to freely change order of types in antecedents. Taking this into account we can regard antecedents of sequents as multisets rather than as strings. Hence let us hereinafter denote by a_1, \dots, a_n a multiset $\{a_1, \dots, a_n\}$ (where a_i are types or symbols of an alphabet); in the same spirit, when we write Γ, Δ for multisets Γ and Δ , this means their union $\Gamma \cup \Delta$.

The Lambek calculus with permutation can be restricted to the calculus without the product \cdot , i.e. we can consider a fragment of LP with division only. We will denote this fragment as LP(\diagup).

Definition 1 The *length* of types and sequents is defined as follows:

1. $|p| = 1$;
2. $|A \circ B| = |A| + |B| + 1$, $\circ \in \{\cdot, /\}$;
3. $|A_1, \dots, A_n \rightarrow B| = |A_1| + \dots + |A_n| + |B|$.

The *depth* of a type A without products is defined as follows:

1. $d(p) = 0$, $p \in Pr$;
2. $d(A/B) = \max\{d(A), d(B) + 1\}$.

The cut rule is admissible in LP (i.e. everything that can be derived using it can also be derived without it):

$$\frac{\Pi \rightarrow A \quad \Gamma, A \rightarrow B}{\Gamma, \Pi \rightarrow B} \text{ (cut)}$$

This implies that the following rules are also admissible:

$$\frac{\Gamma, A \cdot B \rightarrow C}{\Gamma, A, B \rightarrow C} (\cdot \rightarrow)^{-1} \quad \frac{\Pi \rightarrow B/A}{\Pi, A \rightarrow B} (\rightarrow /)^{-1}$$

Indeed, the first rule is in fact an application of the cut rule to the sequents $A, B \rightarrow A \cdot B$ and $\Gamma, A \cdot B \rightarrow C$; the second rule is an application of the cut rule to the sequents $\Pi \rightarrow B/A$ and $B/A, A \rightarrow B$. The above two rules are inverted $(\cdot \rightarrow)$ and $(\rightarrow /)$; i.e. we proved that the latter rules are invertible.

Now let us formulate the definition of LP-grammars.

Definition 2 An LP-grammar is a tuple $G = \langle \Sigma, S, \triangleright \rangle$ where Σ is a finite *alphabet*, $S \in Tp$ is a *distinguished type*, and $\triangleright \subseteq \Sigma \times Tp$ is a finite binary relation between symbols of the alphabet and types (in other words, one assigns several types to each element of Σ). The language $L(G)$ generated by G is the set of multisets $a_1, \dots, a_n \in \mathcal{M}(\Sigma)$ such that there exist types T_1, \dots, T_n of LP, for which it holds that:

1. $a_i \triangleright T_i$ ($i = 1, \dots, n$);
2. $LP \vdash T_1, \dots, T_n \rightarrow S$.

Definition 3 We denote by $Tp(G)$ the set of all types involved in G (including S); more formally, $Tp(G) := \{T \mid \exists a : a \triangleright T\} \cup \{S\}$. Let us also inductively define the set $STp^+(G)$ of *positive subtypes* of G and the set $STp^-(G)$ of *negative subtypes* of G as follows:

- If there exists a such that $a \triangleright T$, then $T \in STp^-(G)$;
- $S \in STp^+(G)$;
- If $A/B \in STp^\pm(G)$, then $A \in STp^\pm(G)$ and $B \in STp^\mp(G)$ (here \pm and \mp are either $+$ and $-$ resp. or $-$ and $+$ resp.);
- If $A \cdot B \in STp^\pm(G)$, then both A and B are in $STp^\pm(G)$.

The set $STp(G)$ of all subtypes of G is simply the union $STp^+(G) \cup STp^-(G)$.

It is clear that whenever we consider a derivation of a sequent as in Definition 2, negative subtypes of G may appear within it (not as a proper subtype but as a type) only in antecedents of sequents while positive subtypes may appear only in their succedents (proof is by the induction on the length of a derivation).

3 IBVASSAM

In this section, we aim to define a new formalism called linearly-restricted branching vector addition system with states and additional memory. This kind of systems is based on branching vector addition systems with states (BVASS) defined in [9], which in turn extend vector addition systems. The latter systems are introduced in [10]; they represent a very natural and simple formalism, which is equivalent to well-known Petri nets. Countless modifications of vector addition systems are considered in the literature: VASP, VASS, AVASS, BVASS, EBVASS, EVASS, PVASS etc. These extensions are used for different purposes; it should be noted that one of them is proving undecidability of linear logic and its fragments [11, 12]. Speaking of branching vector addition systems with states [9] (BVASS), they are developed as a natural extension of both vector addition systems and Parikh images of context-free grammars. Here is the formal definition of BVASS:

Definition 4 A *branching vector addition system with states (BVASS)* is a tuple $G = \langle Q, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, s, K \rangle$ where

1. $K \in \mathbb{N}$ is the *dimension of G_s* ;
2. Q is a finite set of *states*;
3. \mathcal{P}_0 is a finite set of *axioms* of the form $q(\nu)$ where $q \in Q, \nu \in \mathbb{N}^K$;
4. \mathcal{P}_1 is a finite set of *unary rules* of the form $p(x + \delta_2) \leftarrow q(x + \delta_1)$ where $p, q \in Q, \delta_1, \delta_2 \in \mathbb{N}^K$;
5. \mathcal{P}_2 is a finite set of *binary rules* of the form $p(x + y) \leftarrow q(x), r(y)$ where $p, q, r \in Q$;
6. $s \in Q$ is a distinguished *accepting state*.

A formula $p(v)$ (let us call it a *fact*) for $p \in Q, v \in \mathbb{N}^K$ is *derivable in such a BVASS G* if one of the following holds (an inductive definition):

- $p(v) \in \mathcal{P}_0$;
- $v = w + \delta_2$ where $w \in \mathbb{N}^K, p(x + \delta_2) \leftarrow q(x + \delta_1) \in \mathcal{P}_1$, and $q(w + \delta_1)$ is derivable in G ;
- $v = u + w$ where $u, w \in \mathbb{N}^K, p(x + y) \leftarrow q(x), r(y) \in \mathcal{P}_2$, and $q(u), r(w)$ are derivable in G .

The language $L(G)$ generated by such a BVASS G consists of vectors $v \in \mathbb{N}^K$ such that $s(v)$ is derivable in G .

Definition 5 The *size of a derivation* is the total number of axioms and rule applications occurring in it.

To introduce IBVASSAM, we make two changes in the definition of BVASS:

1. We allow one to use “additional memory”; this means that main objects in such systems are still vectors $v \in \mathbb{N}^K$ but in the end of a derivation, i.e. when we obtain a fact $s(v)$, we cut off a part of v (in other words, we

project this vector onto a subspace \mathbb{N}^k for $k \leq K$). Moreover, we require that at the end of a derivation there are only zeroes in the part of v we cut off; hence we have a zero test, which, however, can be used only as the last step of a derivation.

2. We limit the size of a derivation of a vector v in BVASS by requiring that it must not exceed $C|v|$ for some constant C (we call this a *linear restriction*). That is, the size of a derivation is bounded by a linear function of the size of the resulting vector. Note that this restriction immediately makes the reachability problem decidable and even places it in NP since in order to check that a vector v is derivable, we only need to check all possible derivations of $s(v)$ of the length $\leq C|v|$.

Let us formally introduce both modifications. Given a vector $v \in \mathbb{N}^k$ and $K \geq k$, we denote by $\iota_K(v)$ the vector $v' \in \mathbb{N}^K$ such that $v'_i = v_i$ ($i = 1, \dots, k$) and $v'_i = 0$ for $i > k$.

Definition 6 A *branching vector addition system with states and additional memory (BVASSAM)* is a tuple $G = \langle Q, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, k, K \rangle$ where all the components except for k are defined as in Definition 4, and $0 \leq k \leq K$.

The language $L(G)$ generated by such a BVASSAM G consists of vectors $v \in \mathbb{N}^k$ such that $s(\iota_K(v))$ is derivable in $\langle Q, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, s, K \rangle$.

Definition 7 A *linearly-restricted BVASSAM* $G|_C$ (IBVASSAM) is a BVASSAM G equipped with a natural number C .

The language $L(G|_C)$ generated by this grammar consists of vectors $u \in \mathbb{N}^k$ such that there exists a derivation of $s(\iota_K(u))$ in G of the size not greater than $C \cdot |u|$.

4 Equivalence of linearly-restricted BVASSAM and LP-grammars

The main result of this paper is that IBVASSAM are equivalent to LP-grammars in the sense that a language L is generated by an LP-grammar if and only if its Parikh image $\pi(L)$ is generated by an IBVASSAM. The proof is split into two directions: the “if” direction is proved in Section 4.1 and the “only if” one is proved in Section 4.2.

4.1 From IBVASSAM to LP-Grammars

Construction 1 (an LP-grammar corresponding to a IBVASSAM) Given an IBVASSAM $G = G|_C = \langle Q, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, s, k, K \rangle|_C$, our aim is to construct an LP-grammar generating $\pi^{-1}(L(G))$ (we assume that it is over the alphabet $\Sigma = \{a_1, \dots, a_k\}$). Let us introduce several preliminary constructions:

1. We consider all states from Q as primitive types; besides, we introduce new primitive types g_1, \dots, g_K, f (g_i correspond to standard-basis vectors e_i in \mathbb{N}^K , and f stands for “finish!”).

Let us agree on the following notation: if $v \in \mathbb{N}^K$ is a natural-valued vector,

8 *Grammars over LP: Recognizing Power and Connection to BVASS*

then $g^v := g_1^{v_1} \cdot \dots \cdot g_K^{v_K}$ and $g \times v := g_1 \times v_1, \dots, g_K \times v_K$. Hence g^v consists of g_i -s combined using the product \cdot while $g \times v$ is a multiset consisting of g_i -s.

2. For each $\varphi_0 = q(\nu) \in \mathcal{P}_0$ we define a type $T(\varphi_0) := f/g^\nu/q$;
3. For each $\varphi_1 = q(x + \delta_2) \leftarrow p(x + \delta_1) \in \mathcal{P}_1$ we define a type $T(\varphi_1) := (p \cdot g^{\delta_1})/g^{\delta_2}/q$;
4. For each $\varphi_2 = q(x + y) \leftarrow p(x), r(y) \in \mathcal{P}_2$ we define a type $T(\varphi_2) := f/(f/r)/(f/p)/q$.
5. Let \mathcal{P} denote $\mathcal{P}_0 \cup \mathcal{P}_1 \cup \mathcal{P}_2$.

Now we define an LP-grammar itself, which we denote as $\text{LPG}(G)$: $\text{LPG}(G) := \langle \Sigma, f/s, \triangleright \rangle$ where $a_i \triangleright A$ iff $A = g_i \cdot T(\varphi_1) \cdot \dots \cdot T(\varphi_j)$ for some $0 \leq j \leq C$ such that all φ_l are from \mathcal{P} (i here changes from 1 to k).

The main idea of the construction is that we encode each axiom and rule φ of an IBVASSAM by a type $T(\varphi)$ and then “attach” these types to a primitive type g_i .

Remark 2 Formally, the definition of A^l is not correct if $l = 0$ (what is the product of A zero times?); consequently, g^v can also be undefined. A canonical way of understanding A^0 is $A^0 = 1$; however, we do not have the unit in LP. Nevertheless, let us notice that in Construction 1 types of the form g^v appear only in certain positions, namely, within types of the form $f/g^\nu/q$ and $(p \cdot g^{\delta_1})/g^{\delta_2}/q$. This suggests the following treatment of the problematic cases:

- If $v_{i_1} \neq 0, \dots, v_{i_j} \neq 0$ ($j \geq 1, i_1 < \dots < i_j$) and $v_t = 0$ for $t \notin \{i_1, \dots, i_j\}$, then we define g^v as $g_{i_1}^{v_{i_1}} \cdot \dots \cdot g_{i_j}^{v_{i_j}}$.
- If $\nu = 0$ in the type $f/g^\nu/q$, then we simply consider a type of the form f/q instead.
- If $\delta_1 = 0$ in the type $(p \cdot g^{\delta_1})/g^{\delta_2}/q$, then we simply consider a type of the form $p/g^{\delta_2}/q$ instead.
- If $\delta_2 = 0$ in the type $(p \cdot g^{\delta_1})/g^{\delta_2}/q$, then we simply consider a type of the form $(p \cdot g^{\delta_1})/q$ instead. Finally, if both $\delta_1 = \delta_2 = 0$, then we use the type p/q .

Our goal is to prove the following theorem:

Theorem 1 *Let $G = G'|_C$ be an IBVASSAM. Then $L(G) = \pi(L(\text{LPG}(G)))$.*

Let us prove several lemmas first.

Lemma 1 *If $\text{LP} \vdash B_1, \dots, B_m \rightarrow q$ where B_i are from $\text{STp}(\text{LPG}(G))$ and $q \in Q$, then $n = 1$ and $B_1 = q$.*

Proof Consider a derivation of $B_1, \dots, B_m \rightarrow q$ and find an occurrence of the axiom $q \rightarrow q$ such that q in its succedent corresponds to q in the succedent of the resulting sequent. Now, let us assume the contrary: if the length of a derivation is greater than 0 (equivalently, if at least one rule is applied), then some rule is applied to $q \rightarrow q$ as well. Let us consider all the cases, which rule it might be:

Cases $(\rightarrow /)$ and $(\rightarrow \cdot)$ are impossible because at the end of the derivation q stands alone in the antecedent, it is not a subtype of a larger type.

Case $(/ \rightarrow)$: if this rule was applied to $q \rightarrow q$, a type of the form q/D would appear in the antecedent of the conclusion. However, there are no types of such form in $STp(LPG(G))$.

The rule $(\cdot \rightarrow)$ cannot be applied in such a case because we have only one type (namely, q) in the antecedent. \square

We also need a similar lemma for sequents with elements of the form g_i in the succedent. To formulate it, let us firstly introduce a function \mathcal{D} , which takes a multiset of types of LP and returns another multiset of types: $\mathcal{D}(p) = p$ ($p \in Pr$), $\mathcal{D}(A/B) = A/B$, $\mathcal{D}(A \cdot B) = \mathcal{D}(A), \mathcal{D}(B)$, $\mathcal{D}(\Gamma, \Delta) = \mathcal{D}(\Gamma), \mathcal{D}(\Delta)$. Informally, we replace all outermost products \cdot in a multiset of types by commas.

Lemma 2 *If $LP \vdash B_1, \dots, B_m \rightarrow g^u$ where B_i are from $STp(LPG(G))$ and $\vec{0} \neq u \in \mathbb{N}^K$, then $\mathcal{D}(B_1, \dots, B_m)$ and $g \times u$ are equal as multisets.*

Proof The proof is by induction on the length of a derivation of the sequent. The base case is trivial since in such a case we have a sequent of the form $g_i \rightarrow g_i$ for some i . To prove the induction step let us consider the last rule applied. Again, there are several cases:

Case $(\rightarrow /)$ is impossible since there are no divisions in g^u .

Case $(/ \rightarrow)$: then the last rule application has the form

$$\frac{\Gamma, E \rightarrow g^u \quad \Delta \rightarrow B}{\Gamma, \Delta, E/B \rightarrow g^u} (/ \rightarrow)$$

Applying the induction hypothesis we conclude that E must be a product of several primitive types of the form g_i . However, there are no types of the form E/B with such E in $STp(LPG(G))$ (indeed, for each type C/D from $STp(LPG(G))$ it is the case that C includes either f or some $q \in Q$). This allows us to draw a conclusion that the last rule application cannot be of the form $(/ \rightarrow)$.

For the case $(\cdot \rightarrow)$ the last rule application is of the form

$$\frac{\Gamma, E_1, E_2 \rightarrow g^u}{\Gamma, E_1 \cdot E_2 \rightarrow g^u} (\cdot \rightarrow)$$

It suffices to notice that the function \mathcal{D} returns the same output for both antecedents; the induction hypothesis completes the proof for this case.

Case $(\rightarrow \cdot)$: the last rule application must be of the form

$$\frac{\Gamma_1 \rightarrow g^{u_1} \quad \Gamma_2 \rightarrow g^{u_2}}{\Gamma_1, \Gamma_2 \rightarrow g^u}$$

where $u_1 + u_2 = u$. Applying the induction hypothesis to the premises, we immediately succeed. \square

The following lemma is crucial:

Lemma 3 *A sequent of the form $g \times u, T(\psi_1), \dots, T(\psi_m), t \rightarrow f$ ($u \in \mathbb{N}^K$, $m \in \mathbb{N}$, $\psi_i \in \mathcal{P}$, $t \in Q$) is derivable if and only if $t(u)$ has a derivation in G such that all the occurrences of axioms and rules in it are exactly those of ψ_1, \dots, ψ_m .*

Proof There are two directions in this lemma, and we are going to prove both of them by a straightforward induction on the length of a derivation in a corresponding formalism.

Let us start with “only if”. The base case is trivial since the sequent of interest cannot be derived if $n = 0$ since it is not an axiom (we have t in the antecedent and $f \neq t$ in the succedent), and it cannot be obtained by applying any rule (because it includes only primitive types).

We proceed with the induction step by considering the last rule application in the derivation. Let $T(\psi_1)$ be the type that appears as the result of this application. There are three cases:

Case 1: $\psi_1 = q(\nu) \in \mathcal{P}_0$. Then the final steps of the derivation must be of the form

$$\frac{\frac{e \times u_2, f, \Theta_2 \rightarrow f \quad g \times u_1, \Theta_1 \rightarrow g^\nu}{g \times u, f/g^\nu, T(\psi_2), \dots, T(\psi_m) \rightarrow f} \quad q \rightarrow q}{g \times u, f/g^\nu/q, T(\psi_2), \dots, T(\psi_m), t \rightarrow f} (/ \rightarrow)$$

Here $g \times u_1, g \times u_2 = g \times u$ (i.e. $u_1 + u_2 = u$) and $\Theta_1, \Theta_2 = T(\psi_2), \dots, T(\psi_m)$ (as multisets, i.e., up to the order of types). Indeed, since $T(\psi_1) = f/g^\nu/q$ appears after the last rule application, there has to be a premise with q in the succedent; according to Lemma 1 this premise must be of the form $q \rightarrow q$. Since the only type from Q in the antecedent of the sequent $g \times u, f/g^\nu/q, T(\psi_2), \dots, T(\psi_m), t \rightarrow f$ is t , we conclude that $q = t$.

The sequent $g \times u, f/g^\nu, T(\psi_2), \dots, T(\psi_m) \rightarrow f$ can be obtained only by an application of $(/ \rightarrow)$. However, any of $T(\psi_i)$ ($i \in \{2, \dots, m\}$) cannot be the major type of this rule because each of them is of the form A/r where A is a type and $r \in Q$; if it was major, then it would be the case that r is present in the antecedent of $g \times u, f/g^\nu, T(\psi_2), \dots, T(\psi_m) \rightarrow f$, which does not hold. Consequently, f/g^ν must be major, thus the next step of the derivation is as shown above.

Now, let us examine the sequent $g \times u_2, f, \Theta_2 \rightarrow f$. We claim that it must be an axiom. Otherwise, it appears as the result of some rule application; more precisely, it must be an instance of $(/ \rightarrow)$. However, we notice that all the types with division in this sequent are of the form A/r for some $r \in Q$; if the last rule application was $(/ \rightarrow)$, then it would be of the form (according to Lemma 1):

$$\frac{\dots \rightarrow f \quad r \rightarrow r}{g \times u_2, f, \Theta_2 \rightarrow f}$$

This would imply that r is present in $g \times u_2, f, \Theta_2$, which is not the case. Concluding we obtain that $g \times u_2$ and Θ_2 are empty (in what follows that $u_2 = \vec{0}$).

Now it suffices to apply Lemma 2 to $g \times u_1, \Theta_1 \rightarrow g^\nu$, which allows us to draw a conclusion that $g \times u_1 = g \times \nu$ and Θ_1 is empty. Finally, we have that $u = u_1 = \nu$, $q = t$ and hence u is derivable in G : $t(u) = q(\nu)$ is an axiom.

Case 2: $\psi_1 = q(x + \delta_2) \leftarrow p(x + \delta_1) \in \mathcal{P}_1$. Then the last steps of the derivation must be of the form

$$\frac{\frac{g \times u_1, g \times \delta_1, \Theta_1, p \rightarrow f}{g \times u_1, p \cdot g^{\delta_1}, \Theta_1 \rightarrow f} (\cdot \rightarrow) \quad \frac{g \times u_2, \Theta_2 \rightarrow g^{\delta_2}}{g \times u, (p \cdot g^{\delta_1})/g^{\delta_2}, T(\psi_2), \dots, T(\psi_m) \rightarrow f} (/ \rightarrow)_2}{g \times u, (p \cdot g^{\delta_1})/g^{\delta_2}/q, T(\psi_2), \dots, T(\psi_m), t \rightarrow f} q \rightarrow q (/ \rightarrow)_1$$

Here $g \times u_1 + g \times u_2 = g \times u$ (equivalently, $u_1 + u_2 = u$) and $T(\psi_2), \dots, T(\psi_m) = \Theta_1, \Theta_2$ (up to the order of types). Reasonings concerning the last two steps of the derivation (which are numbered using subscript 1 and 2) are the same as in the previous case; they imply that $q = t$, $u_2 = \delta_2$, and Θ_2 is empty. Besides, we force using the rule $(\cdot \rightarrow)$, which is possible due to admissibility of the rule $(\cdot \rightarrow)^{-1}$; in fact, in the above derivation, $(\cdot \rightarrow)$ is used multiple times in a row, since we eliminate all the products in the type $p \cdot g^{\delta_1}$ and obtain a multiset of types $g \times \delta_1, p$.

Finally, we apply the induction hypothesis to the sequent $g \times u_1, g \times \delta_1, \Theta_1, p \rightarrow f$ and obtain that $p(u_1 + \delta_1)$ has a derivation in G such that axioms and rules used in it are exactly ψ_2, \dots, ψ_m . It remains to notice that $g \times u = g \times (\delta_2 + u_1)$ and $t(u) = q(\delta_2 + u_1)$ can be derived from $p(\delta_1 + u_1)$ using ψ_1 . This concludes the proof.

Case 3. $\psi_1 = q(x + y) \leftarrow p(x), r(y)$. Then the last steps of the derivation must be of the form

$$\frac{\frac{g \times u_3, f, \Theta_3 \rightarrow f}{g \times u_2, g \times u_3, f/(f/r), \Theta_2, \Theta_3 \rightarrow f} (\rightarrow /) \quad \frac{g \times u_2, \Theta_2, r \rightarrow f}{g \times u_2, \Theta_2 \rightarrow f/r} (\rightarrow /)}{\frac{g \times u_1, \Theta_1, p \rightarrow f}{g \times u_1, \Theta_1 \rightarrow f/p} (\rightarrow /)} (/ \rightarrow)_3 \quad \frac{g \times u_1, \Theta_1, p \rightarrow f}{g \times u_1, \Theta_1 \rightarrow f/p} (\rightarrow /)}{\frac{g \times u, f/(f/r)/(f/p), T(\psi_2), \dots, T(\psi_m) \rightarrow f}{g \times u, f/(f/r)/(f/p)/q, T(\psi_2), \dots, T(\psi_m), t \rightarrow f} q \rightarrow q (/ \rightarrow)_2} (/ \rightarrow)_1$$

Here $g \times u = g \times u_1 + g \times u_2 + g \times u_3$ (i.e. $u_1 + u_2 + u_3 = u$) and $\Theta_1, \Theta_2, \Theta_3 = T(\psi_2), \dots, T(\psi_m)$; the applications of $(/ \rightarrow)$ are numbered in order to refer to them. Reasonings for this case are similar to those for Cases 1 and 2. The main observation is that only the type $f/(f/r)/(f/p)$ (the type $f/(f/r)$) can be major in a rule application with number 2 (number 3 resp.) because all other types in the antecedent are either primitive or of the form A/x where $x \in Q$; however, there is no primitive type $x \in Q$ in the antecedent. The rule applications of $(\rightarrow /)$ are forced using invertibility of $(\rightarrow /)$. The sequent $g \times u_3, f, \Theta_3 \rightarrow f$ can be derivable only if it is axiom, i.e. if $g \times u_3, \Theta_3$ are empty. Finally, we apply the induction hypothesis to $g \times u_2, \Theta_2, r \rightarrow f$ and $g \times u_1, \Theta_1, p \rightarrow f$ and conclude that $r(u_2)$ and $p(u_1)$ can be derived in G with the multiset of axioms and rules used in total in both derivations equal to ψ_2, \dots, ψ_m . Finally we apply the rule ψ_1 and come up with $q(u_1 + u_2) = q(u)$ as desired.

Speaking of the “if” direction, we need to transform a derivation of $t(u)$ in G into a derivation of $g \times u, T(\psi_1), \dots, T(\psi_m), t \rightarrow f$ where ψ_1, \dots, ψ_m are all the axiom and rule occurrences in the derivation of $t(u)$. This is done straightforwardly by induction on m ; in fact, each axiom or rule application in G is remodeled in LP in the same way as shown above (Cases 1-3). \square

We are ready to prove Theorem 1.

Proof (of Theorem 1) Let $w = w_1, \dots, w_m$ belong to $L(\text{LPG}(G))$. This is the case if and only if $\text{LP} \vdash A_1, \dots, A_m \rightarrow f/s$ where $w_i \triangleright A_i$ ($i = 1, \dots, m$) for some types A_i .

Each type A_i in the antecedent of this sequent is either primitive or is a product of several types (cf. Construction 1).

We claim that $w \in L(\text{LPG}(G))$ if and only if the following sequent is derivable for some $n \leq C|w| = Cm$ and some $\varphi_i \in \mathcal{P}$:

$$g \times v, T(\varphi_1), \dots, T(\varphi_n), s \rightarrow f \quad (1)$$

Here $v = \iota_K(\pi(w))$. To prove the “only if” direction of the claim we use $(\cdot \rightarrow)^{-1}$ in the antecedent of $A_1, \dots, A_m \rightarrow f/s$ and thus release all types of the form $T(\varphi)$, which were combined together within types A_1, \dots, A_m , hence obtaining the sequent of the form $g \times v, T(\varphi_1), \dots, T(\varphi_n) \rightarrow f/s$. Finally, we move s from the succedent to the antecedent using $(\rightarrow /)^{-1}$. The restriction on n arises because there is at most C types of the form $T(\varphi)$ combined within each A_i , and the total number of types A_i equals $|w| = m$; therefore, the total number of types is less than or equal to Cm . To prove the “if” direction it suffices to find suitable types A_1, \dots, A_m such that $w_i \triangleright A_i$ and $\text{LP} \vdash A_1, \dots, A_m \rightarrow f/s$. Here they are:

1. $A_i = (g \times \iota_K(\pi(w_i))) \cdot T(\varphi_{j_1}) \cdot \dots \cdot T(\varphi_{j_2})$ for $i = 1, \dots, \lfloor n/C \rfloor$ where $j_1 = C(i-1) + 1$ and $j_2 = Ci$.
2. $A_i = (g \times \iota_K(\pi(w_i))) \cdot T(\varphi_{j_1}) \cdot \dots \cdot T(\varphi_{j_2})$ for $i = \lfloor n/C \rfloor + 1$ where $j_1 = C\lfloor n/C \rfloor + 1$ and $j_2 = n$. It is required that $n/C \notin \mathbb{N}$, otherwise, $A_i = g \times \pi(w_i)$.
3. $A_i = g \times \iota_K(\pi(w_i))$ for $i > \lfloor n/C \rfloor + 1$.

Note here that $g \times \pi(w_i)$ is a single primitive type since w_i is one symbol. According to Construction 1 $w_i \triangleright A_i$; finally, note that $A_1, \dots, A_m \rightarrow f/s$ is derivable from (1) by using $(\cdot \rightarrow)$ several times and $(\rightarrow /)$. The claim is proved.

Now it remains to apply Lemma 3, which implies that (1) is derivable for some $\varphi_1, \dots, \varphi_n \in \mathcal{P}$ if and only if $s(v)$ has a derivation in G of the size n . The latter is equivalent to the statement that $\pi(w) \in L(G)$. Summarizing all the steps we obtain that $w \in L(\text{LPG}(G))$ if and only if $\pi(w) \in L(G)$; this is the statement of the theorem. \square

4.2 From LP-Grammars to IBVASSAM

Construction 2 (an IBVASSAM corresponding to an LP-grammar) Assume we are given an LP-grammar $G = \langle \Sigma, S, \triangleright \rangle$, $\Sigma = \{a_1, \dots, a_k\}$. We construct an equivalent IBVASSAM $\text{IBAM}(G) := \langle Q, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, S, k, K \rangle|_F$ as follows:

1. $Q := STp^+(G) \cup \{C/A \mid C \in STp^+(G), A \in STp^-(G)\}$.
2. K equals $|STp^-(G)| + |\Sigma|$, and k equals $|\Sigma|$. Hereinafter we fix a bijection $\text{ind} : STp^-(G) \rightarrow \{k+1, \dots, K\}$.
3. \mathcal{P}_0 consists of axioms $p(e_{\text{ind}(p)})$ for $p \in Pr \cap STp^-(G)$.
4. \mathcal{P}_1 consists of the following rules:
 - (a) $C(x + e_{\text{ind}(A \cdot B)}) \leftarrow C(x + e_{\text{ind}(A)} + e_{\text{ind}(B)})$ for $A \cdot B \in STp^-(G)$, $C \in STp^+(G)$;
 - (b) $(A/B)(x) \leftarrow A(x + e_{\text{ind}(B)})$ for $A/B \in Q$;
 - (c) $A(x + e_{\text{ind}(B)}) \leftarrow (A/B)(x)$ for $A/B \in Q$;
 - (d) $S(x + e_i) \leftarrow S(x + e_{\text{ind}(A)})$ for A such that $a_i \triangleright A$.
5. \mathcal{P}_2 consists of the following rules:
 - (a) $(A \cdot B)(x + y) \leftarrow A(x), B(y)$ for $A \cdot B \in STp^+(G)$;
 - (b) $(C/(A/B))(x + y) \leftarrow (C/A)(x), B(y)$ for $A/B \in STp^-(G)$, $C \in STp^+(G)$.

6. S , which is a distinguished type in G , is also a distinguished state in the new grammar.
7. $F = 7 \cdot \max_{A \in Tp(G)} |A|$.

The main result regarding this construction is the following theorem:

Theorem 2 *Let $G = \langle \Sigma, S, \triangleright \rangle$ be an LP-grammar. Then $\pi(L(G)) = L(\text{IBAM}(G))$.*

In the below lemmas we use the same notation as in Construction 2.

Given a vector $u \in \mathbb{N}^K$ such that $u_1 = \dots = u_k = 0$, we denote by $\mathcal{A}(u)$ the multiset of types $A_1, \dots, A_{|u|}$ such that all $A_i \in STp^-(G)$ and for each $A \in STp^-(G)$ it holds that $|\{i \mid A_i = A\}| = u_{\text{ind}(A)}$.

Lemma 4 *Let $C \in STp^+(G)$ and let $u \in \mathbb{N}^K$ such that $u_1 = \dots = u_k = 0$. Then $\text{LP} \vdash \mathcal{A}(u) \rightarrow C$ if and only if $C(u)$ has a derivation in $\text{IBAM}(G)$. Besides, there exists a derivation of $C(u)$ of a size not greater than $3|\mathcal{A}(u) \rightarrow C|$.*

Proof We start with the “if” direction. It is proved by induction on the length of a derivation, and the proof is straightforward: the derivation of $C(u)$ can be directly represented as a derivation in LP. Let us consider two not so trivial cases of the possible last rule application in the derivation of $C(u)$:

Case 1. The last rule applied is $C(x + e_{\text{ind}(B)}) \leftarrow (C/B)(x)$, i.e. $u = u' + e_{\text{ind}(B)}$ and $(C/B)(u')$ is derivable. By the induction hypothesis, the sequent $\mathcal{A}(u') \rightarrow C/B$ is derivable. Using $(\rightarrow /)^{-1}$ we obtain that $\text{LP} \vdash \mathcal{A}(u'), B \rightarrow C$. It remains to observe that $\mathcal{A}(u'), B = \mathcal{A}(u)$.

Case 2. The last rule applied is $(D/(A/B))(x + y) \leftarrow (D/A)(x), B(y)$, i.e. $C = D/(A/B)$ and $u = u_1 + u_2$ for $(D/A)(u_1)$ and $B(u_2)$ being derivable. By the induction hypothesis, sequents $\mathcal{A}(u_1) \rightarrow D/A$ and $\mathcal{A}(u_2) \rightarrow B$ are derivable. The sequent $\mathcal{A}(u_1), \mathcal{A}(u_2) \rightarrow C$ can be derived as follows:

$$\frac{\frac{\mathcal{A}(u_1) \rightarrow D/A \quad \mathcal{A}(u_2) \rightarrow B}{\mathcal{A}(u_1), \mathcal{A}(u_2) \rightarrow (D/A) \cdot B} (\rightarrow \cdot) \quad (D/A) \cdot B \rightarrow D/(A/B)}{\mathcal{A}(u_1), \mathcal{A}(u_2) \rightarrow D/(A/B)} (\text{cut})$$

It is straightforward to check that $\text{LP} \vdash (D/A) \cdot B \rightarrow D/(A/B)$.

The remaining rules are easier to consider because they do not require using the cut rule or invertibility of $(\cdot \rightarrow)$ and $(\rightarrow /)$. Note that the rule 44d cannot be applied in the derivation of $C(u)$ because $u_1 = \dots = u_k = 0$.

The other direction is more interesting because we also need to control the size of a derivation. The proof, however, is still straightforward and it is still proved by induction on the length of a derivation (in LP). The base case is where $\mathcal{A}(u) \rightarrow C$ is an axiom (say, it is of the form $p \rightarrow p$). Then $p(e_{\text{ind}(p)})$ is an axiom in $\text{IBAM}(G)$, and hence $u = e_{\text{ind}(p)}$ has a derivation of the size 1.

Now let us prove the induction step in a usual manner by considering possible last rule applications. The only interesting case is the one when $(/ \rightarrow)$ is applied:

$$\frac{\Gamma, A, \Delta \rightarrow C \quad \Psi \rightarrow B}{\Gamma, A/B, \Psi, \Delta \rightarrow C}$$

Here $\Gamma, A/B, \Psi, \Delta = \mathcal{A}(u)$. There exist u_1 and u_2 such that $\Gamma, A, \Delta = \mathcal{A}(u_1)$ and $\Psi = \mathcal{A}(u_2)$; moreover, we know that $u = u_1 + u_2 - e_{\text{ind}(A)} + e_{\text{ind}(A/B)}$. Using the induction hypothesis we conclude that $C(u_1)$ and $B(u_2)$ have derivations of sizes not greater than $3|\mathcal{A}(u_1) \rightarrow C|$ and $3|\mathcal{A}(u_2) \rightarrow B|$ resp. Finally, we do the following three steps in the derivation:

1. $(C/A)(u_1 - e_{\text{ind}(A)}) \leftarrow C(u_1)$;
2. $(C/(A/B))(u_1 + u_2 - e_{\text{ind}(A)}) \leftarrow (C/A)(u_1 - e_{\text{ind}(A)}), B(u_2)$;
3. $C(u_1 + u_2 - e_{\text{ind}(A)} + e_{\text{ind}(A/B)}) \leftarrow (C/(A/B))(u_1 + u_2 - e_{\text{ind}(A)})$.

This is a derivation of $C(u)$ of the size not greater than $3|\mathcal{A}(u_1) \rightarrow C| + 3|\mathcal{A}(u_2) \rightarrow B| + 3 = 3|\mathcal{A}(u) \rightarrow C|$. \square

Lemma 5 *Let $S(u)$ be derivable in $\text{IBAM}(G)$. If there is a step in its derivation of the form $S(v + e_i) \leftarrow S(v + e_{\text{ind}(A)})$ for A such that $a_i \triangleright A$, then we can make this step the last one in the derivation.*

Proof If we do not apply this rule in its original place, then a vector $v + e_{\text{ind}(A)}$ appears instead of $v + e_i$ in further steps of the derivation. This does not make the derivation incorrect because there are no rules decreasing the value of the i -th component (for $i \leq k$). Hence, at the end of the new derivation we obtain $S(u - e_i + e_{\text{ind}(A)})$. Applying the rule $S(x + e_i) \leftarrow S(x + e_{\text{ind}(A)})$ we complete the derivation. \square

Theorem 2 is proved as follows using the above lemmas:

Proof (of Theorem 2) Let us prove that, if $u \in L(\text{IBAM}(G))$, then $\pi^{-1}(u) \in L(G)$. A vector $u \in \mathbb{N}^k$ belongs to $L(\text{IBAM}(G))$ if and only if $S(v)$ has a derivation of the size not greater than $F|u|$ for $v = \iota_K(u)$. Applying Lemma 5 we can assume that rules of the form 44d are applied after all the remaining rules in the derivation of $S(v)$. In other words, there exists a vector v' such that $v'_1 = \dots = v'_k = 0$ and such that the derivation of $S(v)$ is decomposed into two parts:

1. We derive $S(v')$ without using rules of the form 44d;
2. We derive $S(v)$ from $S(v')$ using only rules 44d. Let us explicitly denote all the steps of this part of the derivation as follows: at the j -th step the rule $S(x + e_{i_j}) \leftarrow S(x + e_{\text{ind}(A_j)})$ is applied where $a_{i_j} \triangleright A_j$ and $j = 1, \dots, |v'| = |v|$.

Then proving the fact that $\pi^{-1}(u) \in L(G)$ is straightforward: if we replace each symbol a_{i_j} by a corresponding type A_j , then we obtain a multiset of types $A_1, \dots, A_{|v|} = \mathcal{A}(v')$. Lemma 4 yields that $\mathcal{A}(v') \rightarrow S$ is derivable, which completes the proof.

Conversely, let us prove that, if $\pi^{-1}(u) \in L(G)$, then $u \in L(\text{IBAM}(G))$. The former fact implies that there exist such $A_1, \dots, A_{|u|}$ that $a_{i_j} \triangleright A_j$ ($j = 1, \dots, |u|$) where $\pi^{-1}(u) = a_{i_1}, \dots, a_{i_{|u|}}$ and such that $\text{LP} \vdash A_1, \dots, A_{|u|} \rightarrow S$. There exists

a vector \tilde{v} such that $\mathcal{A}(\tilde{v}) = A_1, \dots, A_{|u|}$. Using the “only if” direction of Lemma 4 we conclude that $S(\tilde{v})$ has a derivation in $\text{IBAM}(G)$ of the size not greater than $3 \left| A_1, \dots, A_{|u|} \rightarrow S \right| = 3 \left(|A_1| + \dots + |A_{|u|}| + |S| \right) \leq 3 \cdot \max_{A \in T_P(G)} |A| \cdot (|u| + 1) \leq \left(6 \cdot \max_{A \in T_P(G)} |A| \right) |u| = F|u|$. Finally, we apply rules of the form 44d to $S(\tilde{v})$ $|u|$ times in such a way that the resulting fact is $S(\pi^{-1}(u))$. This finishes the proof. \square

Finally, we combine the results of Theorems 1 and 2 to obtain the following

Theorem 3 *A language is generated by an LP-grammar if and only if its Parikh image is generated by a linearly-restricted branching vector addition system with states and additional memory.*

4.3 Corollaries

Corollary 1 *There exists an LP-grammar generating the language of multisets*

$$\{a \times l, b \times n \mid 0 < n, 0 \leq l \leq n^2\}.$$

Proof We present an $\text{IBVASSAM } G = \langle Q, \mathcal{P}_0, \mathcal{P}_1, \mathcal{P}_2, s, k, K \rangle|_C$ generating the language $\{(n, l) \mid 0 < n, 0 \leq l \leq n^2\}$:

1. $K = 7, k = 2$. For our convenience we rename the standard-basis vectors as follows: $a := e_1, b := e_2, \alpha := e_3, \omega := e_4, \beta := e_5, \rho := e_6, \sigma := e_7$.
2. $Q = \{s\}$.
3. \mathcal{P}_0 includes only $s(\alpha)$.
4. \mathcal{P}_1 consists of unary rules
 - (a) $s(x + \alpha + b + \rho) \leftarrow s(x + \alpha)$,
 - (b) $s(x + \omega + b + \rho) \leftarrow s(x + \alpha)$,
 - (c) $s(x + \omega + \sigma + a) \leftarrow s(x + \omega + b)$,
 - (d) $s(x + \beta) \leftarrow s(x + \omega + \rho)$,
 - (e) $s(x + \beta + b) \leftarrow s(x + \beta + \sigma)$,
 - (f) $s(x + \omega) \leftarrow s(x + \beta)$,
 - (g) $s(x) \leftarrow s(x + \omega)$.
5. \mathcal{P}_2 is empty.
6. $C = 4$.

Let us say that a derivation in G is *completely typical* if it is of the following form for some $n > 0$:

1. We start with $s(\alpha)$ and apply the rule 44a $(n - 1)$ times. The result is

$$s(\alpha + (n - 1)(b + \rho)).$$

2. An application of 44b. The result is

$$s(\omega + nb + n\rho).$$

3. The following steps are done for $i = 1, \dots, n$. Let $x_1 := 0, y_1 := n$; then $s(\omega + nb + n\rho) = s(\omega + x_1 a + y_1 b + (n - y_1)\sigma + (n + 1 - i)\rho)$.

- (a) At the beginning of each iteration we have a fact of the form $s(\omega + x_i a + y_i b + (n - y_i)\sigma + (n + 1 - i)\rho)$.
 (b) We apply the rule 44c $l_i \leq y_i$ times. The result is

$$s(\omega + (x_i + l_i)a + (y_i - l_i)b + (n - y_i + l_i)\sigma + (n + 1 - i)\rho).$$

- (c) We apply the rule 44d. The result is

$$s(\beta + (x_i + l_i)a + (y_i - l_i)b + (n - y_i + l_i)\sigma + (n - i)\rho).$$

- (d) We apply the rule 44e $l'_i \leq n - y_i + l_i$ times. The result is

$$s(\beta + (x_i + l_i)a + (y_i - l_i + l'_i)b + (n - y_i + l_i - l'_i)\sigma + (n - i)\rho).$$

- (e) We apply the rule 44f. The result is

$$s(\omega + (x_i + l_i)a + (y_i - l_i + l'_i)b + (n - y_i + l_i - l'_i)\sigma + (n - i)\rho).$$

This is the last step of the iteration, so $x_{i+1} := x_i + l_i$, $y_{i+1} := y_i - l_i + l'_i$.

After all these steps we have the fact $s(\omega + x_{n+1}a + y_{n+1}b + (n - y_{n+1})\sigma)$.

4. The rule 44g is applied. The result is

$$s(x_{n+1}a + y_{n+1}b + (n - y_{n+1})\sigma).$$

We say that a derivation is *typical* if it is a beginning part of a completely typical derivation. Our claim is that each derivation in G is typical. This is straightforwardly proved by induction on the length of a derivation; the proof is a simple consideration of what rule can be next at each step of a completely typical derivation.

Let $(n, l) \in L(G)$; equivalently, $s(la + nb)$ has a derivation in G of the size less than or equal to $4(n + l)$. This derivation is typical; in fact, it must be completely typical (since both ω and β disappear only at the last step of a completely typical derivation). Hence $l = x_{n+1} \leq n^2$ as desired.

Conversely, if $l \leq n^2$, then $s(la + nb)$ has a derivation in G of the size $\leq 4(n + l)$. Indeed, a derivation of interest is the completely typical one with parameters $l_i = n$ (for $i \leq \lfloor l/n \rfloor$), $l_i = l - \lfloor l/n \rfloor n$ (for $i = \lfloor l/n \rfloor + 1$), $l_i = 0$ (for $i > \lfloor l/n \rfloor + 1$); $l'_i = l_i$ (for all i). The size of this derivation can be computed by summing the number of rule applications at each stage: it equals $(n - 1) + 1 + 2l + 2n + 1 = 3n + 2l + 1 \leq 4(n + l)$. \square

This corollary gives a negative answer to the question of whether LP-grammars generate only permutation closures of context-free languages; indeed, the language $\text{PERM}(\{a^l b^n \mid 0 < n, 0 \leq l \leq n^2\})$ where $\text{PERM}(L) = \{a_{\sigma(1)} \dots a_{\sigma(m)} \mid m > 0, a_1 \dots a_m \in L, \sigma \in S_m\}$ is not a permutation closure of any context-free language (equivalently, it is not a permutation closure of a regular language). This can be proved by, e.g., using Theorem 4 from [13] (an iteration lemma for regular languages).

In the case of the Lambek calculus, the equivalence of Lambek grammars and context-free grammars allows one to show that Lambek grammars, in which we use only types of a very simple form (namely, either p , p/q , or $p/q/r$ for $p, q, r \in Pr$), are equivalent to all Lambek grammars; this result can be considered as a normal form for Lambek grammars. In the case of LP-grammars, Constructions 1 and 2 also enable one to establish a normal form for LP-grammars: namely, for each LP-grammar G the grammar $\text{LPG}(\text{IBAM}(G))$ is equivalent to G , and it uses types of a specific form. We can slightly modify Construction 1 to prove e.g. the following corollary:

Corollary 2 LP-grammars are equivalent to LP(/)-grammars.

Proof The modification of Construction 1 is as follows:

1. We define a function T' in the same way as T on \mathcal{P}_0 and \mathcal{P}_2 but we change the definition for \mathcal{P}_1 : $T'(\varphi_1) := (f/(f/(p \cdot g^{\delta_1}))) / g^{\delta_2} / q$;
2. We define the grammar $\text{LPG}'(G)$ as $\langle \Sigma, f/s, \triangleright' \rangle$ where $a_i \triangleright' B$ if and only if $B = f/(f/(s \cdot A))/s$ for $A = g_i \cdot T'(\varphi_1) \cdot \dots \cdot T'(\varphi_j)$ where $\varphi_l \in \mathcal{P}$ are some axioms and rules.

The proof of the fact that $L(G) = \pi(L(\text{LPG}'(G)))$ is similar to the proof of Theorem 1.

Note that $\text{LP} \vdash A/(B \cdot C) \rightarrow A/B/C$ and $\text{LP} \vdash A/B/C \rightarrow A/(B \cdot C)$. Consequently, in the above grammar, we can replace each type with the product under the division by a type without the product; i.e. we replace $(f/(f/(p \cdot g^{\delta_1}))) / g^{\delta_2} / q$ by $(f/(f/p/g^{\delta_1})) / g^{\delta_2} / q$ and $f/(f/(s \cdot A))/s$ by $f/(f/s/g_i/T'(\varphi_1)/\dots/T'(\varphi_j))/s$. Hence there exists an LP(/)-grammar equivalent to $\text{LPG}'(G)$. Let us denote it as $\text{LPG}''(G)$.

Given an LP-grammar G , the grammar $\text{LPG}''(\text{IBAM}(G))$ is equivalent to it, and it is an LP(/)-grammar. \square

5 Conclusion

We showed that LP-grammars are not context-free in the sense that they generate more than permutation closures of context-free languages. This result contrasts with that for Lambek grammars, which are context free [2]. We did this by establishing the equivalence of LP-grammars and IBVASSAM, which is yet another extension of vector addition systems.

Acknowledgments. I thank Stepan L. Kuznetsov for bringing my attention to this problem and for suggesting valuable ideas to explore.

Declarations

The study was supported by RFBR, project number 20-01-00670, by the Theoretical Physics and Mathematics Advancement Foundation “BASIS”, and by the Interdisciplinary Scientific and Educational School of Moscow University “Brain, Cognitive Systems, Artificial Intelligence”.

References

- [1] Lambek, J.: The mathematics of sentence structure. The American Mathematical Monthly **65**(3), 154–170 (1958)
- [2] Pentus, M.: Lambek grammars are context free. In: Proceedings of the Eighth Annual Symposium on Logic in Computer Science (LICS), pp. 429–433. IEEE Computer Society, Montreal, Canada (1993). <https://doi.org/10.1109/LICS.1993.287565>

- [3] Bar-Hillel, Y., Gaifman, H., Shamir, E.: On categorial and phrase structure grammars. *Bull. Res. Council. Israel* **9**, 1–6 (1960)
- [4] van Benthem, J.: The Semantics of Variety in Categorial Grammar. *Linguistic and Literary Studies in Eastern Europe*, vol. 25, pp. 37–55. John Benjamins, Amsterdam (1983)
- [5] van Benthem, J.: Language in action. *J. Philos. Log.* **20**(3), 225–263 (1991). <https://doi.org/10.1007/BF00250539>
- [6] Kuznetsov, S.L.: Personal communication
- [7] Valentín, O.: Theory of discontinuous Lambek calculus. PhD thesis, Universitat Autònoma de Barcelona. Departament de Filologia Catalana (2012)
- [8] Pshenitsyn, T.: Transformation of DPO Grammars Into Hypergraph Lambek Grammars With The Conjunctive Kleene Star. In: Grabmayer, C. (ed.) *Pre-proceedings of the 12th International Workshop on Computing with Terms and Graphs (TERMGRAPH 2022)*, Haifa, Israel (2022)
- [9] Verma, K.N., Goubault-Larrecq, J.: Karp-miller trees for a branching extension of VASS. *Discret. Math. Theor. Comput. Sci.* **7**(1), 217–230 (2005)
- [10] Karp, R.M., Miller, R.E.: Parallel program schemata. *Journal of Computer and System Sciences* **3**(2), 147–195 (1969). [https://doi.org/10.1016/S0022-0000\(69\)80011-5](https://doi.org/10.1016/S0022-0000(69)80011-5)
- [11] Lincoln, P., Mitchell, J.C., Scedrov, A., Shankar, N.: Decision problems for propositional linear logic. *Ann. Pure Appl. Log.* **56**(1-3), 239–311 (1992). [https://doi.org/10.1016/0168-0072\(92\)90075-B](https://doi.org/10.1016/0168-0072(92)90075-B)
- [12] Kanovich, M.I.: Petri nets, horn programs, linear logic and vector games. *Ann. Pure Appl. Log.* **75**(1-2), 107–135 (1995). [https://doi.org/10.1016/0168-0072\(94\)00060-G](https://doi.org/10.1016/0168-0072(94)00060-G)
- [13] Dömösi, P., Kudlek, M.: Strong iteration lemmata for regular, linear, context-free, and linear indexed languages. In: *FCT. Lecture Notes in Computer Science*, vol. 1684, pp. 226–233. Springer, ??? (1999)