

From Double Pushout Grammars to Hypergraph Lambek Grammars With and Without Exponential Modality

Tikhon Pshenitsyn

Department of Mathematical Logic and Theory of Algorithms, Faculty of Mathematics and Mechanics *
Lomonosov Moscow State University, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation

ptihon at yandex.ru

We study how to relate well-known hypergraph grammars based on the double pushout (DPO) approach and grammars over the hypergraph Lambek calculus HL (called HL-grammars). It turns out that DPO rules can be naturally encoded by types of HL using methods similar to those used by Kanazawa for multiplicative-exponential linear logic. In order to generalize his reasonings we extend the hypergraph Lambek calculus by adding the exponential modality, which results in a new calculus HMEL₀; then we prove that any DPO grammar can be converted into an equivalent HMEL₀-grammar. We also define the conjunctive Kleene star, which behaves similarly to this exponential modality, and establish a similar result. If we add neither the exponential modality nor the conjunctive Kleene star to HL, then we can still use the same encoding and show that any DPO grammar with a linear restriction on the length of derivations can be converted into an equivalent HL-grammar.

1 Introduction

In this paper, we aim to relate two kinds of graph grammars: double-pushout (DPO) hypergraph grammars and hypergraph Lambek grammars (HL-grammars).

DPO hypergraph grammars are one of the most well-known kinds of graph grammars, which were introduced in 1973 [2]. They are designed to generalize unrestricted Chomsky formal grammars from strings to graphs. Recall that a production in a formal grammar of the form $\alpha \Rightarrow \beta$ allows one to replace a substring α in any string γ by a string β . A production of a DPO hypergraph grammar, in turn, can be presented in the form $L \Rightarrow R$ where L and R are two hypergraphs. The procedure of replacing a hypergraph by another hypergraph, however, needs further clarification; this is done by using the double pushout approach, which is widely used in the field of graph grammars.

The hypergraph Lambek calculus HL and hypergraph Lambek grammars are novel approaches described in [13, 15]. They are based on logical grounds: HL generalizes the Lambek calculus introduced in [8]. The Lambek calculus L is a substructural logic of intuitionistic logic, and it is originally designed to model the syntax of natural languages. The hypergraph Lambek calculus HL inherits the main principles of L, its structural and model-theoretic properties. Besides, HL forms the basis for hypergraph Lambek grammars (HL-grammars). An HL-grammar is defined by an assignment of a finite number of types (i.e. formulas) of HL to symbols of an alphabet. With some simplifications, the mechanism of these grammars can be described as follows: in order to check that a hypergraph H is generated by an HL-grammar, one must replace each symbol in H by one of the types corresponding to it (which results in a hypergraph labeled by types) and to check that the resulting structure is derivable from axioms by rules of HL.

*The study was funded by RFBR, project N20-01-00670; the Interdisciplinary Scientific and Educational School of Moscow University “Brain, Cognitive Systems, Artificial Intelligence”; the Theoretical Physics and Mathematics Advancement Foundation “BASIS”.

Our objective is to figure out what class of hypergraph languages HL-grammars generate. In particular, it is clearly important to compare them with widely studied DPO grammars. The following question arises: can we convert each DPO grammar into an HL-grammar generating the same language? A simple complexity argument shows that the answer is negative: DPO grammars are Turing complete while the membership problem for HL-grammars is decidable and even NP-complete. Nevertheless, it turns out that there is a simple way to naturally encode any DPO rule as a type of HL. This encoding is essentially the same as the encoding that Kanazawa defined for translating unrestricted Chomsky grammars into grammars over the multiplicative-exponential Lambek calculus [4]. In the hypergraph case, we can also add the exponential modality $!$ to HL obtaining the hypergraph multiplicative-exponential Lambek calculus; in this paper, we define it in a somehow restricted way, which helps us to prove the cut elimination theorem for it, yet sufficient for our purposes. The new calculus is denoted as HMEL_0 . Then we can show that any DPO grammar can be converted into an HMEL_0 -grammar. In fact, apparently HMEL_0 -grammars are equivalent to DPO grammars; however, in this paper, we prove only one half of this statement, so the other one formally remains a claim.

However, our main goal is still to study the hypergraph Lambek calculus itself without the exponential modality. It turns out that we can establish a nice relationship using the same encoding as for HMEL_0 -grammars; to do this, we need to restrict DPO grammars. Namely, let us equip a DPO grammar Gr with a number $c \in \mathbb{N}$ and let us consider the language of hypergraphs H generated by Gr such that the length of a derivation of H in Gr is less than or equal to $c \cdot |E_H|$ where $|E_H|$ is the number of hyperedges in H . That is, we impose the linear restriction on the length of a derivation of a hypergraph with respect to the number of its hyperedges. It turns out that each such language can be generated by an HL-grammar. In fact, the converse statement (from HL-grammars to linearly restricted DPO grammars) holds as well, which is, however, a matter of another article to be written.

In the first version of this paper presented at TERMGRAPH 2022 in Haifa, we introduced an extension of HL by the *conjunctive Kleene star* instead of the exponential modality. The conjunctive Kleene star behaves similar to the exponential modality $!$ (or, more precisely, to the soft subexponential as in [5]). However, the conjunctive Kleene star has a different motivation and its axiomatization involves, in particular, the omega rule (see Section 5). Such an operation is quite peculiar and weird, and it received little attention in the literature (however, it is mentioned e.g. in [10] when considering *storage operators*). At the workshop discussion, I was advised to try to use the exponential modality to achieve the same goals, which is more common and well studied. Then, when we started studying the existing literature on the multiplicative-exponential Lambek calculus for strings, we found the article [4] by Kanazawa, where a similar construction is used to the one developed in the current work. So, in this final version of the paper, we do not use the conjunctive Kleene star in the constructions of grammars but the exponential modality. However, we retain the definition of the former in Section 5 in order not to change the content of the paper too much.

The paper is organized as follows. Section 2 contains definitions of all formalisms and notions of interest (hypergraphs, hyperedge replacement, DPO grammars, HL, HL-grammars) along with some intuition behind the hypergraph Lambek calculus. In Section 3, we define the hypergraph multiplicative-exponential Lambek calculus HMEL_0 and generalize the idea from [4] to prove that DPO grammars can be converted into equivalent HMEL_0 -grammars. In Section 4, we impose the linear restriction on DPO grammars and present the translation procedure from restricted grammars to HL-grammars. Section 5 contains the definition of the conjunctive Kleene star and some remarks concerning it (although this operation is not used anymore in transformations). In Section 6, we conclude.

2 Preliminaries

Σ^* is the set of strings over an alphabet Σ including the empty word Λ ; if R is a relation, then R^* is its transitive reflexive closure. Each function $f : \Sigma \rightarrow \Delta$ can be extended to a homomorphism $f : \Sigma^* \rightarrow \Delta^*$. By $w(i)$ we denote the i -th symbol of $w \in \Sigma^*$, and by $|w|$ we denote the number of symbols in w .

Let $[n]$ denote the set $\{1, 2, \dots, n\}$ (and $[0] := \emptyset$ accordingly).

Given a set of *labels* Σ along with a *rank function* $rk : \Sigma \rightarrow \mathbb{N}$, a *hypergraph* G over Σ is a tuple $G = \langle V_G, E_G, att_G, lab_G, ext_G \rangle$ where V_G is a finite set of *nodes*, E_G is a finite set of *hyperedges*, $att_G : E_G \rightarrow V_G^*$ assigns a string (understand it as an ordered multiset) of *attachment nodes* to each hyperedge, $lab_G : E_G \rightarrow \Sigma$ labels each hyperedge by some element of Σ in such a way that $rk(lab_G(e)) = |att_G(e)|$ whenever $e \in E_G$, and $ext_G \in V_G^*$ is a string of *external nodes*. Hypergraphs are always considered up to isomorphism. The set of all hypergraphs with labels from Σ is denoted by $\mathcal{H}(\Sigma)$. Note that we allow attachment nodes of a hyperedge as well as external nodes to coincide. The *rank function* rk_G (or rk , if G is clear) is defined as follows: $rk_G(e) := |att_G(e)|$. Besides, $rk(G) := |ext_G|$.

In drawings of hypergraphs, black circles correspond to nodes, labeled rectangles correspond to hyperedges, att is represented by numbered lines, and external nodes are represented by numbers in parentheses (round brackets). If a hyperedge has exactly two attachment nodes, then it is depicted by a labeled arrow that goes from the first attachment node to the second one.

A *handle* a^\bullet is a hypergraph $a^\bullet = \langle [n], [1], att, lab, 1 \dots n \rangle$ where $att(1) = 1 \dots n$ and $lab(1) = a$ ($a \in \Sigma$, $rk(a) = n$). A hypergraph a° is of the form $\langle [n], [1], att, lab, \Lambda \rangle$ where att, lab are as in the definition of a^\bullet . A hypergraph $D[k] = \langle [k], \emptyset, \emptyset, \emptyset, \Lambda \rangle$ is called *discrete* ($k \in \mathbb{N}$).

Given a hypergraph H and a function $f : E_H \rightarrow \Sigma$, a *relabeling* $f(H)$ is the hypergraph $f(H) = \langle V_H, E_H, att_H, f, ext_H \rangle$. It is required that $rk_H(e) = rk(f(e))$ for $e \in E_H$.

The *replacement of a hyperedge* e_0 in G by a hypergraph H (such that $rk(e_0) = rk(H)$) is done as follows: (1) remove e_0 from G ; (2) insert an isomorphic copy of H (H and G must consist of disjoint sets of nodes and hyperedges); (3) for each $i = 1, \dots, rk(e_0)$, fuse the i -th external node of H with the i -th attachment node of e_0 . The result is denoted as $G[e_0/H]$. It is well known that if several hyperedges of a hypergraph are replaced by other hypergraphs, then the result does not depend on the order of the replacements; moreover the result is not changed, if replacements are done simultaneously [1]. The following notation is in use: if e_1, \dots, e_k are distinct hyperedges of a hypergraph H and they are simultaneously replaced by hypergraphs H_1, \dots, H_k resp., then the result is denoted $H[e_1/H_1, \dots, e_k/H_k]$.

In a case where a hypergraph G does not have external nodes ($ext_G = \Lambda$) let us call it *zero rank*. If at least one of the hypergraphs H_1, H_2 is zero rank, then one can define their *disjoint union* $H_1 + H_2$ as the hypergraph $\langle V_{H_1} \sqcup V_{H_2}, E_{H_1} \sqcup E_{H_2}, att, lab, ext \rangle$ such that $att|_{H_i} = att_{H_i}$, $lab|_{H_i} = lab_{H_i}$ ($i = 1, 2$), and $ext = ext_{H_i}$ if H_i is not zero rank and $ext = \Lambda$ otherwise; that is, we just put these hypergraphs together without fusing any nodes or hyperedges. The disjoint union of a zero-rank hypergraph H with itself k times is denoted by $k \cdot H$.

2.1 DPO Grammars

Given two hypergraphs G and H , a *morphism* $f : G \rightarrow H$ is a pair of functions $f_V : V_G \rightarrow V_H$, $f_E : E_G \rightarrow E_H$ such that $f_V(att_G(e)) = att_H(f_E(e))$, $lab_H(f_E(e)) = lab_G(e)$ for all $e \in E_G$, and $f_V(ext_G) = ext_H$.

Let I, G_1, G_2 be zero-rank hypergraphs with morphisms $\varphi_i : I \rightarrow G_i$, $i = 1, 2$. Let \equiv_V be the smallest equivalence relation on the disjoint union $V_{G_1} \sqcup V_{G_2}$ that satisfies $\varphi_1(v) \equiv \varphi_2(v)$ for $v \in V_I$; a relation \equiv_E is defined similarly on $E_{G_1} \sqcup E_{G_2}$. $\langle x \rangle$ denotes the equivalence class of x with respect to \equiv_V if x is a node, and with respect to \equiv_E if x is a hyperedge. The *gluing* of G_1 and G_2 over I denoted as $G_1 +_{\varphi_1, \varphi_2} G_2$ is a

hypergraph G such that $V_G = (V_{G_1} \sqcup V_{G_2}) / \equiv_V$, $E_G = (E_{G_1} \sqcup E_{G_2}) / \equiv_E$; given $\langle e \rangle \in E_G$ with $rk(e) = k$, if $e \in E_{G_i}$ for some $i = 1, 2$, then $att_G(\langle e \rangle) = \langle att_{G_i}(e)(1) \rangle \dots \langle att_{G_i}(e)(k) \rangle$ and $lab_G(\langle e \rangle) = lab_{G_i}(e)$. This is a well-defined notion taken from [6] (where it is defined for graphs rather than for hypergraphs). There, the authors state that the gluing of two graphs is a pushout in the category of graphs. In this paper, we do not work within the categorical approach, so we stick to the set-theoretic definition.

Note that, if I is discrete, then the gluing procedure can be represented as replacement:

Proposition 1. *Let $I = D[k]$ and let G_i, φ_i be as above. Let $G'_1 = \langle V_{G_1}, E_{G_1} \sqcup \{e_0\}, att_{G'_1}, lab_{G'_1} \rangle$ where $att_{G'_1}(e) = att_{G_1}(e)$, $lab_{G'_1}(e) = lab_{G_1}(e)$ for $e \in E_{G_1}$, and $att_{G'_1}(e_0) = \varphi_1(1) \dots \varphi_1(k)$ (the label of e_0 does not matter). Let $G'_2 = \langle V_{G_2}, E_{G_2}, att_{G_2}, lab_{G_2}, \varphi_2(1) \dots \varphi_2(k) \rangle$. Then $G_1 +_{\varphi_1, \varphi_2} G_2 = G'_1[e_0/G'_2]$.*

This proposition immediately follows from the definitions of gluing and replacement.

A hypergraph grammar rule over a set of labels C is of the form $r = (L \xrightarrow{\varphi_L} I \xrightarrow{\varphi_R} R)$ where $L, I, R \in \mathcal{H}(C)$ are zero rank and φ_L, φ_R are morphisms. A hypergraph G is transformed into H via r if there is a zero-rank hypergraph C and a morphism $\psi : I \rightarrow C$ such that $G \cong C +_{\psi, \varphi_L} L$, $H \cong C +_{\psi, \varphi_R} R$ [6] (\cong means that hypergraphs are isomorphic). Categorically, this can be expressed by a double pushout diagram:

$$\begin{array}{ccccc} L & \xleftarrow{\varphi_L} & I & \xrightarrow{\varphi_R} & R \\ \downarrow m & & \downarrow \psi & & \downarrow n \\ G & \xleftarrow{\eta_L} & C & \xrightarrow{\eta_R} & H \end{array}$$

This transformation is denoted as $G \xRightarrow[r]{\psi} H$ or simply as $G \Rightarrow H$.

NB! Hereinafter, we consider only hypergraph rules with I being discrete. This does not substantially restrict the formalism.

Example 1. Consider the following DPO rule ρ :

$$\rho = \left(\begin{array}{c} \begin{array}{ccc} & \{1\} & \\ & \bullet & \\ \{2\} & \swarrow l & \searrow r & \bullet & \{3\} \end{array} & \leftarrow D[3] & \rightarrow & \begin{array}{ccc} & \{1\}_1 & \boxed{f} \\ & \bullet & \\ \{2\}_2 & \swarrow 1 & \searrow 3 & \bullet & \{3\} \\ & \boxed{t} & \end{array} \end{array} \right)$$

Note that both the leftmost and the rightmost hypergraphs are zero rank (no external nodes in round brackets); numbers in curly brackets represent images of nodes of the interface hypergraph $D[3]$ (i.e. $\varphi_L(1)$ is the node with the superscript $\{1\}$ in the leftmost hypergraph and so on).

The definition of a hypergraph grammar rule and Proposition 1 imply that the rule application of $G \xRightarrow[r]{\psi} H$ for $r = (L \xrightarrow{\varphi_L} D[k] \xrightarrow{\varphi_R} R)$ consists of an inverse replacement $G = C'[e_0/L'] \Leftarrow C'$ and of a replacement $C' \Rightarrow C'[e_0/R'] = H$. Here C is as in the definition of a hypergraph rule application, and C', e_0, L' and R' are defined in the same way as G'_1, e_0 and G'_2 in Proposition 1; in particular, $L' = \langle V_L, E_L, att_L, lab_L, \varphi_L(1) \dots \varphi_L(k) \rangle$, $R' = \langle V_R, E_R, att_R, lab_R, \varphi_R(1) \dots \varphi_R(k) \rangle$.

An application of a hypergraph rule can be extended to cases where G and H are not zero rank. Indeed, we can say that G is transformed into H via r if there is a hypergraph C' with a distinguished hyperedge (call it e_0) such that $C'[e_0/L'] \cong G$ and $C'[e_0/R'] \cong H$. Hereinafter we will use this extended definition. Clearly, if $G \Rightarrow H$, then $rk(G) = rk(H)$.

A DPO hypergraph grammar HGr is of the form $\langle N, \Sigma, P, Z \rangle$ where N, Σ are disjoint finite alphabets of *nonterminal* and *terminal* labels resp., P is a finite set of hypergraph grammar rules over $N \cup \Sigma$ of the form $L \xrightarrow{\varphi_L} D[k] \xrightarrow{\varphi_R} R$, and Z is a *start hypergraph*. The language $L(HGr)$ generated by HGr is the set of all hypergraphs $H \in \mathcal{H}(\Sigma)$ such that $Z \Rightarrow^* H$. Note that we can assume without loss of generality that $Z = S^\bullet$ for $S \in N$.

2.2 Hypergraph Lambek Calculus and Hypergraph Lambek Grammars

Now let us define the hypergraph Lambek calculus. Details concerning its motivation can be found in [13, 15]; we also provide a motivation relating the hypergraph Lambek calculus with linear logic in the next subsection.

Let us fix a set Pr of *primitive types* along with a function $rk : Pr \rightarrow \mathbb{N}$; we require that for each $k \in \mathbb{N}$ there are countably many $p \in Pr$ such that $rk(p) = k$. Besides, we fix a countable set of labels $\$n, n \in \mathbb{N}$ and set $rk(\$n) = n$; let us agree that these labels do not belong to any other set considered in the definition of the calculus. Then the set of *types* Tp is defined inductively as follows:

1. All primitive types are types.
2. Let $N \in Tp$ be a type, and let D be a hypergraph such that labels of all of its hyperedges, except for one, are from Tp , and one of them equals $\$d$ for some d ; let also $rk(N) = rk(D)$. Then $N \div D$ is also a type such that $rk(N \div D) := d$. The hyperedge of D labeled by $\$d$ is denoted by $e_D^\$$.
3. If M is a hypergraph labeled by types from Tp , then $\times(M)$ is also a type, and $rk(\times(M)) := rk(M)$.

Example 5 contains an exemplar of a type (note that \times binds stronger than \div). A *sequent* is a structure of the form $H \rightarrow A$ where H is a hypergraph labeled by types (called the *antecedent* of the sequent), and A is a type (called the *succedent*) such that $rk(H) = rk(A)$.

The hypergraph Lambek calculus HL derives hypergraph sequents. The only axiom of HL is of the form $A^\bullet \rightarrow A$ where $A \in Tp$. There are four inference rules of HL:

$$\frac{H[e/N^\bullet] \rightarrow A \quad H_1 \rightarrow lab_D(d_1) \quad \dots \quad H_k \rightarrow lab_D(d_k)}{H \left[e/D[e_D^\$/N \div D]^\bullet, d_1/H_1, \dots, d_k/H_k \right] \rightarrow A} (\div \rightarrow) \quad \frac{D[e_D^\$/F] \rightarrow N}{F \rightarrow N \div D} (\rightarrow \div)$$

$$\frac{H_1 \rightarrow lab_M(m_1) \quad \dots \quad H_l \rightarrow lab_M(m_l)}{M[m_1/H_1, \dots, m_l/H_l] \rightarrow \times(M)} (\rightarrow \times) \quad \frac{H[e/M] \rightarrow A}{H[e/(\times(M))^\bullet] \rightarrow A} (\times \rightarrow)$$

Here $N \div D, \times(M)$ are types; $e \in E_H$; $E_D = \{e_D^\$, d_1, \dots, d_k\}$, $E_M = \{m_1, \dots, m_l\}$. In each rule presented above, the sequents above the line are called *premises*, and the sequent below the line is called the *conclusion*. A hypergraph sequent $H \rightarrow A$ is said to be *derivable in HL* (denoted by $HL \vdash H \rightarrow A$) if it can be obtained from axioms of HL by applications of rules of HL. A corresponding sequence of rule applications is called a *derivation*. An example of a derivation is given in Example 6.

Inference rules of the hypergraph Lambek calculus are defined as transformations operating on hypergraph sequents. All the rules are defined through replacement; besides, after an application of each rule a new type appears either in the antecedent or in the succedent of a sequent. Let us take a closer look at two particular rules, namely, at $(\div \rightarrow)$ and $(\times \rightarrow)$. The rule $(\div \rightarrow)$ is organized as follows: given a sequent $H[e/N^\bullet] \rightarrow A$ (note that $H[e/N^\bullet]$ is structurally the same hypergraph H , the replacement only changes the label of e) and sequents $H_i \rightarrow lab_D(d_i)$ for $i = 1, \dots, k$, we replace e in H by D , then relabel the $\$d$ -labeled hyperedge by the type $(N \div D)$, and then replace each d_i by the corresponding antecedent H_i ($i = 1, \dots, k$). Hence, this rule essentially consists of several replacements. In contrast, the rule $(\times \rightarrow)$ performs a transformation inverse to replacement: if one has a hypergraph $G[e/M]$ in the antecedent, then he/she can “compress” its subhypergraph M into a single hyperedge e labeled by the type $\times(M)$.

Remark 1. The rules $(\times \rightarrow)$ and $(\rightarrow \div)$ are invertible in HL. This means that:

1. If a sequent of the form $H[e/(\times(M))^\bullet] \rightarrow A$ is derivable in HL, then so is $H[e/M] \rightarrow A$.
2. If a sequent of the form $F \rightarrow N \div D$ is derivable in HL, then so is $D[e_D^\$/F] \rightarrow N$.

Here all the notation is the same as for the rules $(\rightarrow \div)$ and $(\times \rightarrow)$. This can be proved using the cut elimination theorem; the theorem and its proof can be found in [15, Proposition 1].

An HL-grammar is a tuple $HLGr = \langle \Sigma, S, \triangleright \rangle$ where Σ is an alphabet along with a rank function $rk : \Sigma \rightarrow \mathbb{N}$; $S \in Tp$ is a distinguished type; $\triangleright \subseteq \Sigma \times Tp$ is a finite binary relation such that $a \triangleright T$ implies $rk(a) = rk(T)$. The language $L(HLGr)$ generated by an HL-grammar $HLGr = \langle \Sigma, S, \triangleright \rangle$ is the set of all hypergraphs $G \in \mathcal{H}(\Sigma)$ for which a function $f_G : E_G \rightarrow Tp$ exists such that:

1. $lab_G(e) \triangleright f_G(e)$ whenever $e \in E_G$;
2. $HL \vdash f_G(G) \rightarrow S$ (recall that $f_G(G)$ is a relabeling of G by means of f_G).

2.3 Some Insights Into HL

In this subsection, we would like to outline the relationship between the Lambek calculus, HL and linear logic in order to provide some intuition for the hypergraph Lambek calculus. This can be done using ideas from [11]. In that paper, the authors introduce *first-order multiplicative intuitionistic linear logic* MILL1 and show that the Lambek calculus can be embedded in it (as well as its variants like the Lambek calculus with permutation, the nonassociative Lambek calculus etc.). This calculus turns out to be closely related to HL as we are going to show.

The content of this subsection is not necessary to understand the main technical results of the paper, so if the reader would like to skip it, we advise him/her to proceed with Section 3.

The Lambek calculus is a propositional logic, which is usually presented in the Gentzen style, i.e. as a sequent calculus. The use of sequent calculi is convenient, because it is easier to check derivability using them than using a Hilbert-style axiom-centered calculus. Besides, if we consider e.g. the classical propositional calculus in the Gentzen style, then we have a nice division of its rules into two parts: there are purely logical rules describing behaviour of logical operations like conjunction, disjunction, impication, or negation, and there are structural rules like weakening, contraction, or permutation. For example, in classical logic the rule of weakening is of the following form:

$$\frac{\Gamma \rightarrow A}{\Gamma, B \rightarrow A}$$

It is understood as follows: if one can prove A from assumptions Γ , then he/she can also prove it from Γ, B . Clearly, such a rule is not concerned with a particular logical operation but it tells us something about the sequent behaviour itself. By dropping some or all the structural rules, one obtains *substructural logics*, which turn out to be useful in computer science, linguistics and other branches of science. The Lambek calculus is one of them; it includes neither weakening nor contraction nor even permutation. Consequently, the order of assumptions in the left-hand side of a sequent matters. This gives rise to a non-commutative version of conjunction called the product and to two its residues called the left division and the right division. Formally, types (formulas) of L are built from primitive ones Pr using these three operations denoted as \cdot , \backslash , and $/$ resp. A sequent is a structure of the form $A_1, \dots, A_n \rightarrow A$ where $n > 0$ and A_i, A are types. The only axiom of L is of the form $A \rightarrow A$ for each type A . There are six rules:

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

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

Here capital Latin letters denote types, and capital Greek letters denote sequences of types (and Π is nonempty).

Example 2. The sequent $p, q/r, r/s \rightarrow (p \cdot q)/s$ is derivable in L (where p, q, r , and s are primitive types):

$$\frac{\frac{\frac{p \rightarrow p \quad q \rightarrow q}{p, q \rightarrow p \cdot q} (\rightarrow \cdot) \quad \frac{r \rightarrow r \quad s \rightarrow s}{r/s, s \rightarrow r} (/ \rightarrow)}{\frac{p, q/r, r/s, s \rightarrow p \cdot q}{p, q/r, r/s \rightarrow (p \cdot q)/s} (\rightarrow /)}$$

There are many modifications of the Lambek calculus designed for linguistic and logical purposes: the multimodal Lambek calculus, the displacement calculus, the nonassociative Lambek calculus etc. Besides, there are calculi generalizing and unifying these modifications, i.e. they represent the latter in a uniform setting. One of such generalizations is the hypergraph Lambek calculus. Another one is first-order intuitionistic linear logic [11], or, more precisely, its multiplicative fragment MILL1. Its language includes individual variables x_0, x_1, \dots , individual constants c_0, c_1, \dots , functional symbols of different arities, the binary connectives \otimes and \multimap , and the quantifiers \forall and \exists . A *term* is an application of a functional symbol to a list of variables and constants. *Formulas* of MILL1 are built from terms using \otimes , \multimap , and the quantifiers as in any first-order logic. For example, $\forall x. \exists y. (s(x, y) \otimes t(x, y, c_0))$ is a formula. A *sequent* is of the form $\Gamma \rightarrow A$ where Γ is a *multiset* of formulas (i.e. it is not ordered) and A is a formula. The only axiom scheme of MILL1 is $A \rightarrow A$ for all formulas A . The inference rules are as follows:

$$\frac{\Gamma, A, B \rightarrow C}{\Gamma, A \otimes B \rightarrow C} (\otimes \rightarrow) \quad \frac{\Gamma \rightarrow A \quad \Delta \rightarrow B}{\Gamma, \Delta \rightarrow A \otimes B} (\rightarrow \otimes) \quad \frac{\Delta \rightarrow A \quad \Gamma, B \rightarrow C}{\Gamma, \Delta, A \multimap B \rightarrow C} (\multimap \rightarrow) \quad \frac{\Gamma, A \rightarrow B}{\Gamma \rightarrow A \multimap B} (\rightarrow \multimap)$$

$$\frac{\Gamma, A \rightarrow C}{\Gamma, \exists x. A \rightarrow C} (\exists \rightarrow) \quad \frac{\Gamma \rightarrow A[x := e]}{\Gamma \rightarrow \exists x. A} (\rightarrow \exists) \quad \frac{\Gamma, A[x := e] \rightarrow C}{\Gamma, \forall x. A \rightarrow C} (\forall \rightarrow) \quad \frac{\Gamma \rightarrow A}{\Gamma \rightarrow \forall x. A} (\rightarrow \forall)$$

Here e is an arbitrary constant or variable of our choice (either present in Γ or in A or a fresh one). It is required that in the rules $(\exists \rightarrow)$ and $(\rightarrow \forall)$ the variable x does not occur freely in Γ or C .

In [11], the translation of L into MILL1 is discovered such that each type of L is transformed into a formula of MILL1 with two free variables:

1. $\|p\|^{x,y} = p(x, y)$;
2. $\|A/B\|^{x,y} = \forall z. \|B\|^{y,z} \multimap \|A\|^{x,z}$;
3. $\|B \setminus A\|^{x,y} = \forall z. \|B\|^{z,x} \multimap \|A\|^{z,y}$;
4. $\|A \cdot B\|^{x,y} = \exists z. \|A\|^{x,z} \otimes \|B\|^{z,y}$.

Finally, a sequent $A_1, \dots, A_n \rightarrow B$ of L is translated into the sequent $\|A_1\|^{c_0, c_1}, \dots, \|A_n\|^{c_{n-1}, c_n} \rightarrow \|B\|^{c_0, c_n}$ where c_0, c_1, \dots, c_n are distinct constants. It is proved [11] that a sequent is derivable in L if and only if its translation is derivable in MILL1.

Example 3. The types from the sequent presented in Example 2 are translated into MILL1 as follows:

1. $\|p\|^{x,y} = p(x, y)$;
2. $\|q/r\|^{x,y} = \forall z. r(y, z) \multimap q(x, z)$;
3. $\|r/s\|^{x,y} = \forall z. s(y, z) \multimap r(x, z)$;
4. $\|(p \cdot q)/s\|^{x,y} = \forall z. (s(y, z) \multimap \exists t. p(x, t) \otimes q(t, z))$.

The sequent from Example 2 is translated as follows:

$$p(c_0, c_1), \forall z. r(c_2, z) \multimap q(c_1, z), \forall z. s(c_3, z) \multimap r(c_2, z) \rightarrow \forall z. (s(c_3, z) \multimap \exists t. p(c_0, t) \otimes q(t, z)) \quad (1)$$

It is an exercise to verify that the latter sequent is derivable in MILL1.

An important observation is that, although MILL1 is a commutative logic, that is, the order of formulas in antecedents of sequents can be freely changed, we can still embed a noncommutative logic like L in it. The trick is that we preserve the order of types in L by using constants c_0, \dots, c_n : they fix the linear structure of an original sequent from L.

In [15], we prove that L can be embedded in HL. E.g. the sequent $p, q/r, r/s \rightarrow (p \cdot q)/s$ is translated into the following hypergraph sequent:

$$(1) \bullet \xrightarrow{p} \bullet \xrightarrow{tr_L(q/r)} \bullet \xrightarrow{tr_L(r/s)} \bullet (2) \rightarrow tr_L((p \cdot q)/s) \quad (2)$$

Here tr_L is a translation from L into HL, which we are not going to describe in this paper. Comparing the embeddings of L to MILL1 and to HL we observe a number of correspondences: constants c_i ($i = 0, 1, 2, 3$) correspond to the nodes of the graph in the antecedent of (2); the formula $p(c_0, c_1)$ corresponds to a p -labeled edge going from the node c_0 to the node c_1 ; constants c_0, c_2 correspond to the external nodes of the graph. We can also compare the rules of HL and the rules of MILL1 and observe strong similarity. In fact, these observations can be generalized: we claim that the hypergraph Lambek calculus can be embedded in MILL1 (see an example below). However, formally introducing this embedding and proving its correctness should be a matter of another paper; moreover, this is not relevant for our further considerations in this work. Recall that the main goal is to provide an intuition concerning how HL is organized and how to add the exponential modality to HL.

Example 4. The type $DPO(\rho)$ from Example 5 can be translated into the following formula of MILL1: $\forall x. \forall y. \forall z. (f(x) \otimes t(x, y, z)) \multimap (l(x, y) \otimes r(x, z))$. The sequent of HL, which appears at the end of the derivation from Example 6, can be translated into the following MILL1 sequent:

$$\begin{aligned} f(c_0), t(c_0, c_1, c_2), p(c_1, c_2), \forall x. \forall y. \forall z. (f(x) \otimes t(x, y, z)) \multimap (l(x, y) \otimes r(x, z)) \\ \rightarrow \exists x. \exists y. \exists z. l(x, y) \otimes r(x, z) \otimes p(y, z) \end{aligned}$$

Having this in mind, we would like to proceed with extending HL by the exponential.

3 The Hypergraph Multiplicative-Exponential Lambek Calculus

In this section, we show a way of extending HL with the exponential modality $!$ resulting in the *hypergraph multiplicative exponential Lambek calculus* $HME\mathcal{L}_0$ (we will explain this 0 subscript later). After doing this we show how to convert any DPO grammar into an equivalent $HME\mathcal{L}_0$ -grammar. The conversion procedure is similar to that presented in [4]: there, each unrestricted Chomsky grammar is converted into an equivalent grammar over the multiplicative-exponential Lambek calculus.

In Section 2.3, we introduced the rules for the fragment of intuitionistic linear logic. Now, let us look at the full logic ILL1, which includes exponentials. Formulas of ILL1 are built using \otimes , \multimap , $\mathbf{1}$, $\&$, \oplus , $!$ and the quantifiers \forall and \exists . The rules of ILL1 include those of MILL1 and those defining the new connectives. Let us focus on the rules for the $!$ modality:

$$\frac{\Gamma, A \rightarrow B}{\Gamma, !A \rightarrow B} (! \rightarrow) \quad \frac{!A_1, \dots, !A_n \rightarrow B}{!A_1, \dots, !A_n \rightarrow !B} (\rightarrow !)$$

$$\frac{\Gamma \rightarrow B}{\Gamma, !A \rightarrow B} (w) \quad \frac{\Gamma, !A, !A \rightarrow B}{\Gamma, !A \rightarrow B} (c)$$

Our aim is to transfer these rules to HL. It would be great to introduce the exponential modality added to HL unrestrictedly, that is, to be able to consider the type $!A$ for each type A . However, this faces certain difficulties if we expect the resulting calculus to be well-behaving, namely, to enjoy the cut

elimination theorem. Unfortunately, for now we have not invented a general treatment of exponentials in the hypergraph calculus in such a way that this theorem holds, so this remains an open question. However, if we restrict ourselves and allow one to consider the type $!A$ only if A has the rank 0 ($rk(A) = 0$), then it turns out that such a calculus can be well defined; in particular, it satisfies the cut elimination theorem.

The rules for the exponential modality we suggest to add to HL are the following ones:

$$\frac{H + A^\bullet \rightarrow B}{H + (!A)^\bullet \rightarrow B} (! \rightarrow) \quad \frac{\sum_{i=1}^n (!A_i)^\bullet \rightarrow A}{\sum_{i=1}^n (!A_i)^\bullet \rightarrow !A} (\rightarrow!) \quad \frac{H \rightarrow B}{H + (!A)^\bullet \rightarrow B} (w) \quad \frac{H + 2 \cdot (!A)^\bullet \rightarrow B}{H + (!A)^\bullet \rightarrow B} (c) \quad (3)$$

Here $rk(!A_i) = rk(A_i) = rk(!A) = rk(A) - 0$. The summation symbol stands for multiple disjoint union. Clearly, these rules generalize their string counterparts considered earlier for ILL1. Note that if $rk(X) = 0$, then X^\bullet is a zero-rank hyperedge “floating” as a separate component of an antecedent hypergraph.

Definition 1. The *hypergraph multiplicative-exponential Lambek calculus* HMEL_0 is defined similarly to HL but in the definition of types Tp we additionally say that, if A is a type such that $rk(A) = 0$, then $!A$ is a type as well such that $rk(!A) = 0$. We add the rules from (3) to the rules of HL.

HMEL_0 -grammars are defined in the same way as HL-grammars but based on HMEL_0 .

The subscript 0 corresponds to the restriction on putting $!$ only on types with rank 0. As we mentioned, it allows us to prove the following theorem:

Theorem 1. *If $\text{HMEL}_0 \vdash H \rightarrow A$ and $\text{HMEL}_0 \vdash G[e_0/A^\bullet] \rightarrow B$, then $\text{HMEL}_0 \vdash G[e_0/H] \rightarrow B$.*

In other words, this theorem states that we can add the cut rule introduced below to the list of rules without enlarging the set of derivable sequents:

$$\frac{H \rightarrow A \quad G[e_0/A^\bullet] \rightarrow B}{G[e_0/H] \rightarrow B} (\text{cut})$$

Proof. We prove that both the cut rule and the *mix rule*, which we introduce below, can be added to HMEL_0 without affecting the set of derivable sequents:

$$\frac{H \rightarrow !C \quad G' + (!C) \cdot n \rightarrow B}{G' + H \rightarrow B} (\text{mix})$$

Here H is zero rank. Such a rule is commonly introduced to prove the cut elimination theorem for variants of linear logic including weakening (w) and contraction (c). In fact, the whole proof resembles that from [9], and it is typical for substructural logics with exponentials like HMEL_0 . Let us denote $!C$ by A and both $G' + (!C) \cdot n$ and $G[e_0/A^\bullet]$ by F for the sake of uniformity.

The proof is done by nested induction: the outer one is on the size of A (counted as the total number of primitive types and of symbols \times , \div and $!$ in the construction of A), and the inner one is on the sum of lengths of the derivations of $H \rightarrow A$ and $F \rightarrow B$. We need to consider several cases depending on the last rule applied in the derivation of $H \rightarrow A$ or in that of $F \rightarrow B$. An important notion is that of *the major type*: given a derivation of some sequent, the major type is the type that appears in the sequent after the last step of the derivation. E.g., in Example 2, the major type is $(p \cdot q)/s$, and in Example 6, it is $\text{DPO}(\rho)$.

The first group of cases is where $H \rightarrow A$ or $F \rightarrow B$ is an axiom; then the statement of the theorem becomes trivial.

The second group is where A from the succedent of $H \rightarrow A$ is not major in the derivation of $H \rightarrow A$. Then the last rule is concerned with H somehow, and it does not affect A . Thus we can repeat the same

step but when H replaces e within G , and after that apply the induction hypothesis. More formally, let the last step of the derivation of $H \rightarrow A$ be as follows:

$$\frac{H' \rightarrow A \quad \mathcal{P}}{H \rightarrow A}$$

Here \mathcal{P} are some additional premises. Then we apply the induction hypothesis to $H' \rightarrow A$ and $F \rightarrow B$ and thus conclude that $G[e_0/H'] \rightarrow B$ ($G' + H' \rightarrow B$ for the mix rule) is derivable. Finally, we perform the following step:

$$\frac{G[e_0/H'] \rightarrow B \quad \mathcal{P}}{G[e_0/H] \rightarrow B}$$

This completes the proof for this group of cases. From now on, we assume that A is major in $H \rightarrow A$.

The third group of cases is where the distinguished occurrence of A (or all the n distinguished occurrences of $A = !C$ if we consider (mix)) is not major in the derivation of $F \rightarrow B$ and the last rule in its derivation is not $(\rightarrow !)$. Then similar reasonings to those for the second group can be applied.

The fourth group is where A is major in both $H \rightarrow A$ and $F \rightarrow B$ but A is not of the form $!C$ (it is possible only for the cut rule). Then the reasonings are the same as for HL, see [15, Appendix A].

In the remaining cases $A = !C$ (thus the cut rule becomes an instance of the mix rule) is major in both $H \rightarrow A$ and $F \rightarrow B$; we also need to cover the case where the last rule in the derivation of $F \rightarrow B$ is $(\rightarrow !)$. In all these cases, the last rule in the derivation of $H \rightarrow A$ is $(\rightarrow !)$, consequently, the sequent $H \rightarrow C$ is derivable and $H = \sum_{i=1}^n (!A_i)^\bullet$ for some A_i such that $rk(A_i) = 0$. The following cases are possible depending on the last rule applied in the derivation of $F \rightarrow B$:

Case I. The last rule application is $(! \rightarrow)$. Then it must be of the form:

$$\frac{G' + (!C) \cdot (n-1) + C^\bullet \rightarrow B}{G' + (!C) \cdot n \rightarrow B} (! \rightarrow)$$

Using the induction hypothesis, we apply the mix rule to $H \rightarrow A$ and $G' + A \cdot (n-1) + C^\bullet \rightarrow B$ concluding that $G' + H + C^\bullet \rightarrow B$ is derivable. Then we again apply the induction hypothesis to the sequents $H \rightarrow C$ and $G' + H + C^\bullet \rightarrow B$ (note that the size of C is less than that of A so the first induction parameter decreases) coming up with the sequent $G' + H + H \rightarrow B$. Finally, one applies the contraction rule (c) n times thus contracting two H 's into a single H .

Case II. The last rule application is (w). Then it must be of the form:

$$\frac{G' + (!C) \cdot (n-1) \rightarrow B}{G' + (!C) \cdot n \rightarrow B} (! \rightarrow)$$

It remains to apply the induction hypothesis to $H \rightarrow A$ and $G' + (!C) \cdot (n-1) \rightarrow B$.

Case III. The last rule application is (c). The reasonings are similar to the previous case.

Case IV. The last rule application is $(\rightarrow !)$. In what follows, $G' = \sum_{i=1}^m (!D_i)^\bullet$ for some types D_i , $B = !D$ and $F \rightarrow D$ is derivable. Let us apply the induction hypothesis to $H \rightarrow A$ and $F \rightarrow D$ thus concluding that $\sum_{i=1}^m (!B_i)^\bullet + \sum_{i=1}^n (!A_i)^\bullet \rightarrow D$ is derivable. Finally, we apply the rule $(\rightarrow !)$, which results in $G' + H \rightarrow B$ as desired. \square

As a consequence of the cut elimination theorem, we notice that Remark 1 holds for HMEL_0 as well.

Now our aim is to prove that HMEL_0 -grammars based on the new calculus are at least as expressive as DPO grammars. This can be done similarly to the proof of the fact that grammars over the multiplicative-exponential Lambek calculus are at least as expressive as unrestricted Chomsky grammars. Recall that

a rule in a Chomsky grammar is of the form $\alpha \Rightarrow \beta$ where α, β are two arbitrary strings of nonterminal and terminal symbols. The proof from [4] suggests converting each rule of the form $A_1 \dots A_n \Rightarrow B_1 \dots B_m$ (where A_i, B_j are nonterminal symbols) into the following type:

$$[\tau(A_1 \dots A_n \Rightarrow B_1 \dots B_m)] := (A_1 \dots A_n) / (B_1 \dots B_m).$$

Makoto Kanazawa proves in [4] that, given a finite set $R = \{r_1, \dots, r_k\}$ of rules of the above form, a string w of nonterminal symbols is derivable from the nonterminal symbol S using rules from R if and only if $\text{MELC} \vdash ![\tau(r_1)], \dots, ![\tau(r_k)], w \rightarrow S$ where all the nonterminal symbols are considered as primitive types and where all symbols in w are separated by commas. E.g. the rule $S \Rightarrow SSA$ allows one to produce $SSASA$ from S , hence $\text{MELC} \vdash !(S/(S \cdot S \cdot A)), S, S, A, S, A \rightarrow S$. MELC stands for the multiplicative-exponential Lambek calculus.

The same idea can be implemented for DPO rules. Firstly we need to slightly enhance DPO hypergraph grammars.

Construction 1. Given a DPO grammar $\langle N, \Sigma, P, S^\bullet \rangle$, we convert it into a grammar $\langle N', \Sigma, P', S^\bullet \rangle$, which we call *normalized*, as follows. For each $a \in \Sigma$ we introduce a new nonterminal label T_a with $rk(T_a) = rk(a)$; let $N' = N \sqcup \{T_a \mid a \in \Sigma\}$. Then for each $r = (L \xrightarrow{\varphi_L} D[k] \xrightarrow{\varphi_R} R) \in P$ we replace each terminal label a in L, R by T_a . Let us call such new rules *nonterminal* and denote the set of nonterminal rules as P_N . Finally, we add rules that allow one to replace T_a by a , i.e., rules of the form $(T_a^\circ \xrightarrow{\varphi_L} D[k] \xrightarrow{\varphi_R} a^\circ)$ where $rk(a) = k$, $\varphi_L(i) = \varphi_R(i) = i$ for $i = 1, \dots, k$ (here we use the notation of nodes as in the definitions of $D[k]$ and S°). These rules are called *terminal* and are denoted as P_T . Finally, $P' = P_N \cup P_T$.

Clearly, the normalized grammar generates the same language as the original one. Hereinafter we consider only normalized grammars.

Nonterminal rules can be converted into corresponding types of HL as follows:

Construction 2. Let us consider nonterminal labels of normalized grammars as primitive types (with the same rank function). If $r = (L \xrightarrow{\varphi_L} D[k] \xrightarrow{\varphi_R} R)$ is a nonterminal rule, then $\text{DPO}(r) := \times \left(\widehat{L} \right) \div \left(\widehat{R} + \$_0 \right)$ where $\widehat{L} = \langle V_L, E_L, att_L, lab_L, \varphi_L(1) \dots \varphi_L(k) \rangle$; $\widehat{R} = \langle V_R, E_R, att_R, lab_R, \varphi_R(1) \dots \varphi_R(k) \rangle$. Note that $\$_0$ is a separate hyperedge of rank 0 and that $rk(\text{DPO}(r)) = 0$.

Example 5. The nonterminal rule ρ from Example 1 is converted into the following type $\text{DPO}(\rho)$:

$$\text{DPO}(\rho) = \times \left(\begin{array}{c} \bullet \\ / \quad \backslash \\ (2) \quad (1) \quad (3) \end{array} \right) \div \left(\begin{array}{c} (1) \bullet \xrightarrow{1} \boxed{f} \\ | \\ (2) \bullet \xrightarrow{2} \boxed{t} \xrightarrow{3} \bullet (3) \\ \boxed{\$_0} \end{array} \right)$$

The main connection between r and $\text{DPO}(r)$ is shown in

Lemma 1. Let Y, Y' be two hypergraphs and let $Y \xRightarrow{r} Y'$ for $r \in P_N$. If $\text{HL} \vdash Y \rightarrow A$ for any type A , then $\text{HL} \vdash Y' + \text{DPO}(r)^\bullet \rightarrow A$ as well.

Lemma 2. Let $Y \xRightarrow{*} Y'$ in a normalized grammar $\langle N, \Sigma, P, S^\bullet \rangle$ by means of nonterminal rules where Y, Y' are hypergraphs. If $\text{HL} \vdash Y \rightarrow A$ for any type A , then $\text{HL} \vdash Y' + \sum_{r \in P_N} k_r \cdot \text{DPO}(r)^\bullet \rightarrow A$ for some $k_r \in \mathbb{N}$.

Lemma 3 (main). Let Y' be a hypergraph; let $\langle N, \Sigma, P, S^\bullet \rangle$ be a normalized grammar with the set P_N of nonterminal rules; let X be a nonterminal symbol. Then $X^\bullet \xRightarrow{*} Y'$ using rules from P_N if and only if $\text{HL} \vdash Y' + \sum_{r \in P_N} b_r \cdot (!\text{DPO}(r))^\bullet + \sum_{r \in P_N} k_r \cdot (\text{DPO}(r))^\bullet \rightarrow X$ for some $b_r, k_r \in \mathbb{N}$.

Lemma 1 is proved by straightforwardly applying $(\times \rightarrow)$ and then $(\div \rightarrow)$ to the sequent $Y \rightarrow A$ (below a representative example is provided). Lemma 2 directly follows from Lemma 1. Note that k_r in it is the number of applications of r in the derivation $Y \Rightarrow^* Y'$.

Therefore, a DPO derivation can be remodeled within HL but each rule application of r leaves a trace, namely, a floating hyperedge labeled by $\text{DPO}(r)$ in the antecedent. Using the exponential modality we can unify these hyperedges into a single one.

Example 6. The following derivation illustrates Lemma 1:

$$\begin{array}{c}
 \frac{l^\bullet \rightarrow l \quad r^\bullet \rightarrow r \quad p^\bullet \rightarrow p}{\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array}} (\rightarrow \times) \\
 \frac{\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \rightarrow \times \left(\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \right)}{\begin{array}{c} 1 \\ \bullet \\ \boxed{\times(\widehat{L})} \\ \bullet \\ 2 \quad 3 \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array}} (\times \rightarrow) \\
 \frac{\begin{array}{c} 1 \\ \bullet \\ \boxed{\times(\widehat{L})} \\ \bullet \\ 2 \quad 3 \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \rightarrow \times \left(\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \right) \quad t^\bullet \rightarrow t \quad f^\bullet \rightarrow f}{\begin{array}{c} 1 \\ \bullet \\ \boxed{f} \\ \bullet \\ 1 \quad t \quad 3 \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \quad \boxed{\text{DPO}(\rho)} \rightarrow \times \left(\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \right)} (\div \rightarrow)
 \end{array}$$

Here the sequent $Y \rightarrow A$ equals $\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \rightarrow \times \left(\begin{array}{c} l \quad r \\ \bullet \quad \bullet \\ \diagdown \quad \diagup \\ \bullet \quad \bullet \\ p \end{array} \right)$, and the rule r equals ρ from Example 1.

It remains to prove Lemma 3:

Proof (of Lemma 3). The ‘‘only if’’ direction straightforwardly follows from Lemma 2: if we take $Y = X^\bullet$, $A = X$, then $Y \rightarrow A$ is an axiom $X^\bullet \rightarrow X$, hence $Y' + \sum_{r \in P_N} k_r \cdot \text{DPO}(r)^\bullet \rightarrow A$ is derivable for some k_r .

Finally, one can derive the sequent $Y' + \sum_{r \in P_N} (!\text{DPO}(r))^\bullet \rightarrow X$ from it using the rules $(! \rightarrow)$, (w), and (c).

The ‘‘if’’ direction is proved by induction on the length of a derivation of $Y' + \sum_{r \in P_N} b_r \cdot (!\text{DPO}(r))^\bullet + \sum_{r \in P_N} k_r \cdot (\text{DPO}(r))^\bullet \rightarrow X$ in HMEL_0 . For the sake of brevity, we denote the sum $\sum_{r \in P_N} b_r \cdot (!\text{DPO}(r))^\bullet + \sum_{r \in P_N} k_r \cdot (\text{DPO}(r))^\bullet$ as $\sigma\{b_r; k_r\}$.

The base case is trivial. Let us prove the induction step. The proof is done by considering the last rule applied in a derivation of a sequent.

Case 1. The last rule applied is $(\div \rightarrow)$. In this case, its application must be of the form

$$\frac{Y^0 \left[e_0 / (\times(\widehat{L}))^\bullet \right] + \sigma\{b_r^0, k_r^0\} \rightarrow X \quad Y^1 + \sigma\{b_r^1, k_r^1\} \rightarrow \text{lab}_{\widehat{R}}(e_1) \quad \dots \quad Y^l + \sigma\{b_r^l, k_r^l\} \rightarrow \text{lab}_{\widehat{R}}(e_l)}{Y' + \sigma\{b_r', k_r'\} + (\text{DPO}(r_0))^\bullet \rightarrow X}$$

Here $\text{DPO}(r_0) = \times(\widehat{L}) \div (\widehat{R} + \mathbb{S}_0^\bullet)$; $E_{\widehat{R}} = \{e_1, \dots, e_l\}$; $b_r' = b_r$ for all $r \in P_N$, $k_r' = k_r$ for $r \neq r_0$ and $k_{r_0}' = k_{r_0} - 1$; for all $r \in P_N$ it holds that $b_r' = b_r^0 + \dots + b_r^l$, $k_r' = k_r^0 + \dots + k_r^l$; and

$$Y' = Y^0 \left[e_0 / \widehat{R} \left[e_1 / Y^1, \dots, e_l / Y^l \right] \right].$$

This is just an explicit general form of the rule application of $(\div \rightarrow)$ possible for the given sequent. Remark 1 implies that the sequent $Y^0 \left[e_0/\widehat{L} \right] + \sigma\{b_r^0, k_r^0\} \rightarrow X$ is also derivable. Applying the induction hypothesis, we obtain that $X^\bullet \Rightarrow^* Y^0 \left[e_0/\widehat{L} \right]$. Then, $Y^0 \left[e_0/\widehat{L} \right] \Rightarrow_{r_0} Y^0 \left[e_0/\widehat{R} \right]$. Finally, let us apply the induction hypothesis to the sequent $Y^i + \sigma\{b_r^i, k_r^i\} \rightarrow lab_{\widehat{R}}(e_i)$ for each $i = 1, \dots, l$, therefore concluding that $lab_{\widehat{R}}(e_i)^\bullet \Rightarrow^* Y^i$. Now it remains to start with the hypergraph $Y^0 \left[e_0/\widehat{R} \right]$ and to successively remake each derivation $lab_{\widehat{R}}(e_i)^\bullet \Rightarrow^* Y^i$ within it (for $i = 1, \dots, l$), thus replacing each hyperedge e_i of \widehat{R} by Y^i . Therefore, we obtain that

$$X^\bullet \Rightarrow^* Y^0 \left[e_0/\widehat{L} \right] \Rightarrow Y^0 \left[e_0/\widehat{R} \right] \Rightarrow^* Y^0 \left[e_0/\widehat{R} \left[e_1/Y^1, \dots, e_l/Y^l \right] \right] = Y'.$$

Case 2. The last rule applied is either (i) the rule $(! \rightarrow)$, or (ii) the rule (w) , or (iii) the rule (c) . Assume that the main type of this rule application is $!DPO(r_0)$ for $r_0 \in P_N$. Then it must be of the form

$$\frac{Y' + \sigma\{b'_r, k'_r\} \rightarrow X}{Y' + \sigma\{b_r, k_r\} \rightarrow X}$$

Here $b'_r = b_r, k'_r = k_r$ for $r \neq r_0$; in case (i), $b'_{r_0} = b_{r_0} - 1, k'_{r_0} = k_{r_0} + 1$; in case (ii), $b'_{r_0} = b_{r_0} - 1, k'_{r_0} = k_{r_0}$; in case (iii), $b'_{r_0} = b_{r_0} + 1, k'_{r_0} = k_{r_0}$. The induction hypothesis completes the proof in all the three subcases. \square

Now we introduce an $HMELO_0$ -grammar corresponding to a given DPO grammar.

Construction 3. Let $HGr = \langle N, \Sigma, P, S^\bullet \rangle$ be a normalized DPO grammar. Then $LG(HGr) = \langle \Sigma, S', \triangleright \rangle$ where \triangleright consists of pairs $a \triangleright T_a$, and $S' = S \div \left(\sum_{r \in P_N} (!DPO(r))^\bullet + \$\mathbf{0}^\bullet \right)$.

Here we apply the exponential to each type $DPO(r)$ (for $r \in P_N$) and store the result in S' .

This construction turns out to be a straightforward generalization of the one from [4]. Using it we can prove the following lemma:

Lemma 4. *If HGr is a normalized DPO grammar, then $L(LG(HGr)) = L(HGr)$.*

Proof. A hypergraph $H \in \mathcal{H}(\Sigma)$ belongs to $L(LG(HGr))$ if and only if $HMELO_0 \vdash t(H) \rightarrow S'$ where $t(H)$ is the relabeling of H such that $t(e) = T_{lab_H(e)}$. This sequent is equiderivable with the sequent $t(H) + \sum_{r \in P_N} (!DPO(r))^\bullet \rightarrow S$ according to Remark 1 for $HMELO_0$. According to Lemmas 2 and 3, this happens if and only if $S^\bullet \Rightarrow^* t(H)$ by means of nonterminal rules of HGr . It is straightforward to show that this is equivalent to the fact that $H \in L(HGr)$. \square

Finally we come up with the following result:

Theorem 2. *$HMELO_0$ -grammars are at least as expressive as DPO grammars.*

4 DPO Grammars With Linear Restriction and HL-Grammars

The above construction strongly relies on the exponential modality, which enables one to compress several copies of the same type in the antecedent into a single one. In the case of HL where we do not have the $!$ modality, Construction 3 does not work anymore. Moreover, it is clear that it must fail: languages generated by HL-grammars are decidable and even are in NP while DPO grammars generate

at least all recursively enumerable string languages, so the two classes of grammars do not generate the same languages. However, Construction 2 still works since it uses only \times and \div . It facilitates to link HL-grammars with DPO grammars with a certain restriction formally defined below.

Definition 2. Given a DPO hypergraph grammar $HGr = \langle N, \Sigma, P, S^\bullet \rangle$, let $L_c(HGr)$ consist of all hypergraphs $H \in L(HGr)$ such that there exists a derivation $S^\bullet \Rightarrow^* H$ with no more than $c \cdot |E_H|$ steps.

Construction 4. Let $HGr = \langle N, \Sigma, P, S^\bullet \rangle$ be a normalized grammar; let $c \in \mathbb{N}$. Then we construct an HL-grammar $LG_c(HGr) = \langle \Sigma, S, \triangleright \rangle$ where \triangleright contains exactly the following pairs:

$$a \triangleright \times \left(T_a^\bullet + \sum_{r \in P_N} k_r \cdot \text{DPO}(r)^\bullet \right) \text{ for } k_r \geq 0, \sum_{r \in P_N} k_r \leq c.$$

Note that \triangleright is a finite relation since there are finitely many $k_r \in \mathbb{N}$ satisfying the above requirements.

Example 7. Consider a DPO grammar $HGr = \langle N, \Sigma, P, S^\bullet \rangle$ where $N = \{S\}$, $\Sigma = \{a\}$, and $P = \{r_1, r_2, r_3\}$:

1. $r_1 = \left(\boxed{S} \leftarrow D[0] \rightarrow D[0] \right)$;
2. $r_2 = \left(\boxed{S} \leftarrow D[0] \rightarrow \boxed{S} \bullet \right)$;
3. $r_3 = \left(\{1\} \bullet \bullet \{2\} \leftarrow D[2] \rightarrow \{1\} \bullet \xrightarrow{a} \bullet \{2\} \right)$.

It is not hard to see that it generates all graphs (with edges having two attachment nodes): the rule r_2 produces nodes while r_3 produces edges. Consider e.g. the following derivation:

$$\boxed{S} \xRightarrow{r_2} \boxed{S} \bullet \xRightarrow{r_2} \boxed{S} \bullet \bullet \xRightarrow{r_1} \bullet \bullet \xRightarrow{r_3} \bullet \bullet \xrightarrow{a} \bullet \bullet \xRightarrow{r_3} \bullet \bullet \xrightarrow{a} \bullet \bullet \xrightarrow{a} \bullet \bullet \xrightarrow{r_3} \bullet \bullet \xrightarrow{a} \bullet \bullet \xrightarrow{r_3} \bullet \bullet \xrightarrow{a} \bullet \bullet \quad (4)$$

Note that HGr is not normalized; using Construction 1 we replace r_3 by the following two rules:

1. $r'_3 = \left(\{1\} \bullet \bullet \{2\} \leftarrow D[2] \rightarrow \{1\} \bullet \xrightarrow{T_a} \bullet \{2\} \right)$;
2. $r''_3 = \left(\{1\} \bullet \xrightarrow{T_a} \bullet \{2\} \leftarrow D[2] \rightarrow \{1\} \bullet \xrightarrow{a} \bullet \{2\} \right)$.

Let us denote this new normalized grammar HGr' . Then we convert its nonterminal rules into types using Construction 2:

1. $X_1 = \text{DPO}(r_1) = \times \left(\boxed{S} \right) \div \left(\boxed{\$}_0 \right)$;
2. $X_2 = \text{DPO}(r_2) = \times \left(\boxed{S} \right) \div \left(\boxed{S} \bullet \boxed{\$}_0 \right)$;
3. $X_3 = \text{DPO}(r'_3) = \times \left((1) \bullet \bullet (2) \right) \div \left((1) \bullet \xrightarrow{T_a} \bullet (2) \boxed{\$}_0 \right)$;

Finally, we introduce an HL-grammar $LG_2(HGr') = \langle \Sigma, S, \triangleright \rangle$ according to Construction 4. The binary relation \triangleright consists of the following 13 pairs (in fact, of the 10 distinct pairs) where $i, j \in \{1, 2, 3\}$:

$$\begin{aligned} \bullet a \triangleright T &= \times \left((1) \bullet \xrightarrow{T_a} \bullet (2) \right); & \bullet a \triangleright T_{ij} &= \times \left((1) \bullet \xrightarrow{T_a} \bullet (2) \boxed{X_i} \boxed{X_j} \right). \\ \bullet a \triangleright T_i &= \times \left((1) \bullet \xrightarrow{T_a} \bullet (2) \boxed{X_i} \right); \end{aligned}$$

Recall that in order to check that a hypergraph belongs to $L(\text{LG}_2(\text{HGr}'))$ we need 1) to replace labels of its hyperedges by types corresponding to them via \triangleright ; 2) to construct a sequent with the resulting hypergraph in the antecedent and with S in the succedent; 3) to derive this sequent. Let us check, for example, that $H = a \circlearrowleft \xrightarrow{a} \circlearrowright a$ belongs to $L(\text{LG}_2(\text{HGr}'))$. We replace each label a by one of the

types T , T_i , or T_{ij} as follows: $T_{22} \circlearrowleft \xrightarrow{T_{13}} \circlearrowright T_{33}$ (compare the indices of types with the numbers of rules

applied in (4)). Then, it remains to check that $T_{22} \circlearrowleft \xrightarrow{T_{13}} \circlearrowright T_{33} \rightarrow S$ is derivable:

$$\frac{\frac{\frac{T_a \circlearrowleft \xrightarrow{T_a} \circlearrowright T_a \quad \boxed{X_2} \quad \boxed{X_2} \quad \boxed{X_1} \quad \boxed{X_3} \quad \boxed{X_3} \quad \boxed{X_3} \rightarrow S}{T_a \circlearrowleft \xrightarrow{T_a} \circlearrowright T_{33} \quad \boxed{X_2} \quad \boxed{X_2} \quad \boxed{X_1} \quad \boxed{X_3} \rightarrow S} (\times \rightarrow)}{T_a \circlearrowleft \xrightarrow{T_{13}} \circlearrowright T_{33} \quad \boxed{X_2} \quad \boxed{X_2} \rightarrow S} (\times \rightarrow)}{T_{22} \circlearrowleft \xrightarrow{T_{13}} \circlearrowright T_{33} \rightarrow S} (\times \rightarrow)$$

The uppermost sequent in the above derivation is derivable, which follows from the proof of Lemma 2. This justifies that $H \in L(\text{LG}_2(\text{HGr}'))$.

The above example illustrates the following theorem, the main one in this work:

Theorem 3. *For each DPO grammar HGr and each c there is an HL-grammar generating $L_c(\text{HGr})$.*

Lemma 5. *If HGr is a normalized DPO grammar and $1 \leq c \in \mathbb{N}$, then $L(\text{LG}_{c-1}(\text{HGr})) = L_c(\text{HGr})$.*

The inclusion $L(\text{LG}_{c-1}(\text{HGr})) \supseteq L_c(\text{HGr})$ is proved by using Lemma 2 in the same way as in Example 7. The other inclusion is proved by using Lemma 3 and by noticing that the number of $\text{DPO}(r)$ -labeled hyperedges in an antecedent here strictly corresponds to the number of rule applications in HGr (since we do not have types with the exponential).

The idea of Construction 4 and of Theorem 5 is that we store $\text{DPO}(r)$ -labeled hyperedges in each type corresponding to a terminal symbol (since we cannot store them in the succedent as in Construction 3).

Then, for $G \in L(\text{LG}_c(\text{HGr}))$, after we replace each symbol a in G by a type $\times \left(T_a^\bullet + \sum_{r \in P_N} k_r \cdot \text{DPO}(r)^\bullet \right)$ for some k_r , and start deriving a corresponding sequent, these hyperedges eventually appear in the antecedent where they play their role shown in Lemma 2. The total number of these hyperedges, however, is limited by the number of hyperedges in G , hence the language $L_c(\text{HGr})$ is generated instead of $L(\text{HGr})$.

Theorem 5 says that HL-grammars are powerful enough to generate hypergraphs of a language generated by a DPO grammar such that the number of steps in their derivation is bounded by a linear function of the number of hyperedges. It might be the case for a DPO grammar HGr that $L(\text{HGr}) = L_c(\text{HGr})$ for some $c \in \mathbb{N}$; in fact, we claim that for each HL-grammar $\text{HLGr} = \langle \Sigma, S, \triangleright \rangle$ there is a DPO grammar HGr and $c \in \mathbb{N}$ such that $L(\text{HLGr}) = L_c(\text{HGr}) = L(\text{HGr})$, although we do not prove this here (this should be a matter of another paper). In general, however, $L(\text{HGr}) \neq L_c(\text{HGr})$ (e.g. in Example 7 $L_{k+1}(\text{HGr})$ contains only graphs G such that $|V_G| < k \cdot |E_G|$).

5 HL with the Conjunctive Kleene Star

In Section 3, we added the exponential modality to HL, which helped us to prove Theorem 4. Another operation, which behaves similarly, is Kleene star. The Lambek calculus with Kleene star is studied by several researchers, in particular, in [7]. An extension of L by intersection, union, and Kleene star is known as infinitary action logic [12], or, in the algebraic setting, as the logic of action algebras. Note that Kleene star can be described in terms of actions within a transition system: if A is a class of actions, then A^* means actions from A repeated several times [7]. This understanding is very close to what we use in Construction 3 since our goal is to be able to apply encoded DPO rules arbitrarily many times within HL. Hence, let us now look at the Lambek calculus with the unit and Kleene star $L_{\mathbf{1}\omega}^*$ and, more specifically, at the rules for Kleene star [7]:

$$\frac{\Pi \rightarrow A^n}{\Pi \rightarrow A^*} (\rightarrow^*), n \geq 0 \qquad \frac{(\Gamma, A^n, \Delta \rightarrow B)_{n=0}^\infty}{\Gamma, A^*, \Delta \rightarrow B} (*\rightarrow)_\omega$$

Here $A^0 := \mathbf{1}$, $A^{n+1} := A^n \cdot A$; $\mathbf{1}$ is the unit of the product satisfying the axiom $A \cdot \mathbf{1} \leftrightarrow A$. Note that the rule (\rightarrow^*) is in fact a countable set of rules for each $n \geq 0$; contrarily, $(*\rightarrow)_\omega$ is a single rule with countably many premises. Let us clarify the notion of being derivable in this calculus: the set of derivable sequents in $L_{\mathbf{1}\omega}^*$ is the least set S containing all axioms of $L_{\mathbf{1}\omega}^*$ (i.e., all sequents of the form $A \rightarrow A$ and the sequent $\rightarrow \mathbf{1}$) such that it is closed under applications of all inference rules (i.e., if, for some rule, all sequents above the line belong to S , then the sequent below the line must also belong to S). In other words, a derivation in $L_{\mathbf{1}\omega}^*$ is again a sequence of rule applications, which now can be countable in size but which does not have branches of infinite length.

Unfortunately, the rules for Kleene star work in an undesirable way: they allow unlimited copying types in succedents of sequents (namely, if we have n copies of A in a succedent, then we can wrap them into a single type A^*) but not in antecedents. This motivates us to consider an operation behaving dually:

$$\frac{(\Pi \rightarrow A^n)_{n=0}^\infty}{\Pi \rightarrow *A} (\rightarrow^*)_\omega \qquad \frac{\Gamma, A^n, \Delta \rightarrow B}{\Gamma, *A, \Delta \rightarrow B} (*\rightarrow), n \geq 0$$

We call the operation $*A$ *the conjunctive Kleene star*. Algebraically, it can be defined in complete residuated lattices using infinitary conjunction as $*a = \bigwedge_{n=0}^\infty a^n = \inf\{a^n \mid n \in \mathbb{N}\}$ (this is why we call it conjunctive). Note that the language semantics of this operation is poor: if L is a language ($L \subseteq \Sigma^*$), $\mathbf{1}$ equals $\{\Lambda\}$, and multiplication of languages means pairwise concatenation of their words while conjunction means intersection, then $*L = \{\Lambda\}$ if $\Lambda \in L$ and $*L = \emptyset$ otherwise.

Unfortunately, we have found little about this operation in the literature. Nevertheless, it is mentioned in [10] in the context of storage operators. A storage operator I in an MTL-algebra works as follows: $I(a)$ is the greatest idempotent among those not greater than a (it must exist in an MTL-algebra with storage). It can be shown that, if $\inf\{a^n \mid n \in \mathbb{N}\}$ exists, then it equals $I(a)$, so these operators are very close. In [10], it is mentioned that the storage operator has many analogies with Girard's exponential !.

Let us show how to generalize the conjunctive Kleene star and inference rules for it to hypergraphs. A question arises: how should one understand an iteration of a type, namely, A^n ? In the string case, this means repeating a type n times and writing copies in line connecting them by \cdot . We need to extend this iteration procedure to hypergraphs. We suggest the following general definitions:

Definition 3. A *template* T of rank k is a hypergraph $T = \langle V_T, [2], att_T, lab_T, ext_T \rangle$ such that $rk_T(1) = rk_T(2) = rk(T) = k$. In other words, T has two hyperedges of the same rank, which coincides with the rank of T . Hereinafter $T(H_1, H_2)$ is a shorthand notation for $T[1/H_1, 2/H_2]$.

Definition 4. A template T of rank k is *monoidal* if for all hypergraphs A, B, C of rank k it holds that 1. $T(A, T(B, C)) \cong T(T(A, B), C)$, 2. a hypergraph U_T of rank k exists such that $T(U_T, A) \cong T(A, U_T) \cong A$.

Definition 5. The T -iteration $T^n(A)$ of a type A (where T is a monoidal template) such that $rk(A) = rk(T)$ is defined as follows: $T^0(A) := U_T$; $T^{n+1}(A) := T(T^n(A), A^\bullet)$ (for $n \geq 0$).

Example 8. Two examples of monoidal templates are

- $O = \boxed{X} \quad \boxed{X}$ (i.e., $V_O = \emptyset$, $E_O = [2]$, $att_O(1) = att_O(2) = ext_O = \Lambda$). Note that $U_O = D[0]$.
- $Str = (1) \bullet \xrightarrow{Y} \bullet \xrightarrow{Y} \bullet (2)$.

Here X, Y are arbitrary labels, they do not matter. Note that $O(H, G) = H + G$ for zero-rank H, G . Consequently, $O^m(A) = m \cdot A^\bullet$ (where $rk(A) = 0$).

Using monoidal templates we can define the hypergraph conjunctive Kleene star. Types of the hypergraph Lambek calculus with the conjunctive Kleene star ${}^*HL_\omega$ are built as described in Section 2.2 but we add one more item to the definition: if A is a type such that $rk(A) = n$ and if T is a monoidal template of rank n , then ${}^*_T A$ is also a type of rank n . We also add two inference rules for the new operation:

$$\frac{(H \rightarrow \times (T^n(A)))_{n=0}^\infty}{H \rightarrow {}^*_T A} (\rightarrow^*)_\omega \qquad \frac{G[e/T^n(A)] \rightarrow B}{G[e/({}^*_T A)^\bullet] \rightarrow B} (*\rightarrow), n \geq 0$$

Usual logical questions concerning ${}^*HL_\omega$ arise. In particular, the cut elimination theorem can be proved:

Theorem 4. *If ${}^*HL_\omega \vdash H \rightarrow A$ and ${}^*HL_\omega \vdash G[e/A^\bullet] \rightarrow B$, then ${}^*HL_\omega \vdash G[e/H] \rightarrow B$.*

The theorem is proved by a transfinite induction in a similar way to that from [12].

Note that we can define *the hypergraph Kleene star* generalizing A^* studied in [7, 12] in the same way as ${}^*_T A$. Even if the conjunctive Kleene star is something weird and useless, we think that the definitions of template, of T -iteration and so on are useful in the respect that using them we could define and study the hypergraph Kleene star.

Returning to ${}^*HL_\omega$, we can now define ${}^*HL_\omega$ -grammars and repeat Construction 3:

Construction 5. Let $HGr = \langle N, \Sigma, P, S^\bullet \rangle$ be a normalized grammar. Then $LG_\omega(HGr) = \langle \Sigma, S', \triangleright \rangle$ where \triangleright consists of pairs $a \triangleright T_a$, and $S' = S \div \left(\sum_{r \in P_N} ({}^*_O DPO(r))^\bullet + \$\bullet \right)$.

Here we apply the hypergraph conjunctive Kleene star to each type $DPO(r)$ (for $r \in P_N$) and store the result in S' . This trick enables us to prove the following

Theorem 5. *If HGr is a normalized DPO grammar, then $L(LG_\omega(HGr)) = L(HGr)$.*

6 Conclusion

In this work, we have made several steps in relating DPO grammars, which represent the rule-based approach in the field of graph grammars, and HL-grammars, which are an ambassador of the type-logical approach. We have proved that any DPO grammar can be transformed into an equivalent HMEL₀-grammar using a generalization of the method from [3]. Then we restricted derivations in DPO grammars and proved that grammars with this restriction can be converted into equivalent HL-grammars. This restriction is a promising tool in our opinion: we claim that linearly-restricted DPO grammars are equivalent to HL-grammars. Recently we have successfully applied the same idea of imposing a linear restriction to solve an open problem concerning the Lambek calculus with permutation [14]. In the future, we are going to establish a precise connection between HL-grammars and DPO grammars.

Acknowledgments

I thank prof. Mati Pentus and Stepan L. Kuznetsov for fruitful discussions, and anonymous reviewers for valuable remarks at all the rounds of reviewing (in particular, for suggesting studying connections with process algebras, which resulted in starting a research of another subject).

References

- [1] Frank Drewes, Hans-Jörg Kreowski & Annegret Habel (1997): *Hyperedge Replacement Graph Grammars*. In Grzegorz Rozenberg, editor: *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, World Scientific, pp. 95–162, doi:10.1142/9789812384720_0002.
- [2] Hartmut Ehrig, Michael Pfender & Hans Jürgen Schneider (1973): *Graph-Grammars: An Algebraic Approach*. In: *14th Annual Symposium on Switching and Automata Theory, Iowa City, Iowa, USA, October 15-17, 1973*, IEEE Computer Society, pp. 167–180, doi:10.1109/SWAT.1973.11.
- [3] Makoto Kanazawa (1992): *The Lambek calculus enriched with additional connectives*. *J. Log. Lang. Inf.* 1(2), pp. 141–171, doi:10.1007/BF00171695.
- [4] Makoto Kanazawa (1999): *Lambek Calculus: Recognizing Power and Complexity*. In Jelle Gerbrandy, Maarten Marx, Maarten de Rijke & Yde Venema, editors: *Essays Dedicated to Johan van Benthem on the Occasion of His 50th Birthday*, Amsterdam University Press.
- [5] Max I. Kanovich, Stepan L. Kuznetsov, Vivek Nigam & Andre Scedrov (2020): *Soft Subexponentials and Multiplexing*. In: *IJCAR (1), Lecture Notes in Computer Science 12166*, Springer, pp. 500–517, doi:10.1007/978-3-030-51074-9_29.
- [6] Barbara König, Dennis Nolte, Julia Padberg & Arend Rensink (2018): *A Tutorial on Graph Transformation*. In Reiko Heckel & Gabriele Taentzer, editors: *Graph Transformation, Specifications, and Nets - In Memory of Hartmut Ehrig, Lecture Notes in Computer Science 10800*, Springer, pp. 83–104, doi:10.1007/978-3-319-75396-6_5.
- [7] Stepan L. Kuznetsov (2021): *Complexity of the Infinitary Lambek Calculus with Kleene Star*. *Rev. Symb. Log.* 14(4), pp. 946–972, doi:10.1017/S1755020320000209.
- [8] Joachim Lambek (1958): *The Mathematics of Sentence Structure*. *The American Mathematical Monthly* 65(3), pp. 154–170, doi:10.1080/00029890.1958.11989160.
- [9] Patrick Lincoln, John C. Mitchell, Andre Scedrov & Natarajan Shankar (1992): *Decision Problems for Propositional Linear Logic*. *Ann. Pure Appl. Log.* 56(1-3), pp. 239–311, doi:10.1016/0168-0072(92)90075-B.
- [10] Franco Montagna (2004): *Storage Operators and Multiplicative Quantifiers in Many-valued Logics*. *J. Log. Comput.* 14(2), pp. 299–322, doi:10.1093/logcom/14.2.299.
- [11] Richard Moot & Mario Piazza (2001): *Linguistic Applications of First Order Intuitionistic Linear Logic*. *J. Log. Lang. Inf.* 10(2), pp. 211–232, doi:10.1023/A:1008399708659.
- [12] Ewa Palka (2007): *An Infinitary Sequent System for the Equational Theory of *-continuous Action Lattices*. *Fundam. Informaticae* 78(2), pp. 295–309.
- [13] Tikhon Pshenitsyn (2021): *Grammars Based on a Logic of Hypergraph Languages*. In Berthold Hoffmann & Mark Minas, editors: *Proceedings Twelfth International Workshop on Graph Computational Models, GCM@STAF 2021, Online, 22nd June 2021, EPTCS 350*, pp. 1–18, doi:10.4204/EPTCS.350.1.
- [14] Tikhon Pshenitsyn (2022): *Grammars over the Lambek Calculus with Permutation: Recognizing Power and Connection to Branching Vector Addition Systems with States*, doi:10.48550/ARXIV.2209.08415. Available at <https://arxiv.org/abs/2209.08415>.
- [15] Tikhon Pshenitsyn (2022): *Hypergraph Lambek grammars*. *Journal of Logical and Algebraic Methods in Programming* 129, p. 100798, doi:10.1016/j.jlamp.2022.100798.