Transformation of DPO Grammars into Hypergraph Lambek Grammars With The Conjunctive Kleene Star

Tikhon Pshenitsyn
ptihon@yandex.ru

Department of Mathematical Logic and Theory of Algorithms
Faculty of Mathematics and Mechanics
Lomonosov Moscow State University
Russia

TERMGRAPH 2022

Categorial Grammars

- Rule-based approaches: productions of the form $\alpha \to \beta$ where α, β are strings.

Categorial Grammars

- Rule-based approaches: productions of the form $\alpha \to \beta$ where α, β are strings.
- Another approach is categorial grammars.

Categorial Grammars

- Rule-based approaches: productions of the form $\alpha \to \beta$ where α, β are strings.
- Another approach is categorial grammars.
- One assigns categories (=types) to symbols of an alphabet.

Categorial Grammars

- Rule-based approaches: productions of the form $\alpha \to \beta$ where α, β are strings.
- Another approach is categorial grammars.
- One assigns categories (=types) to symbols of an alphabet.
- A general mechanism deals with sequences of categories.

Categorial Grammars

- Rule-based approaches: productions of the form $\alpha \to \beta$ where α, β are strings.
- Another approach is categorial grammars.
- One assigns categories (=types) to symbols of an alphabet.
- A general mechanism deals with sequences of categories.
- A string is correct if one can replace each its symbol by a corresponding category and obtain a sequence of categories, which is accepted by a general mechanism.

Lambek Calculus

- Lambek 1958: a logical formalism designed to model natural languages = *a general mechanism*.

Lambek Calculus

- Lambek 1958: a logical formalism designed to model natural languages = *a general mechanism*.
- Types in the Lambek calculus:

Lambek Calculus

- Lambek 1958: a logical formalism designed to model natural languages = *a general mechanism*.
- Types in the Lambek calculus:
  - Primitive (atomic) types Pr (denoted as p, q, r, np etc.)

Lambek Calculus

- Lambek 1958: a logical formalism designed to model natural languages = *a general mechanism*.
- Types in the Lambek calculus:
  - Primitive (atomic) types Pr (denoted as p, q, r, np etc.)
  - Complex types are built from primitive ones using binary connectives $\backslash$, $/$ and $\cdot$.

Lambek Calculus

- Lambek 1958: a logical formalism designed to model natural languages = *a general mechanism*.
- Types in the Lambek calculus:
  - Primitive (atomic) types Pr (denoted as p, q, r, np etc.)
  - Complex types are built from primitive ones using binary connectives $\backslash$, $/$ and $\cdot$.
- Example: $p/(q\backslash p)$, $(p \cdot q)/(q \cdot p)$ are types.

Lambek Calculus

- Lambek 1958: a logical formalism designed to model natural languages = *a general mechanism*.
- Types in the Lambek calculus:
  - Primitive (atomic) types Pr (denoted as p, q, r, np etc.)
  - Complex types are built from primitive ones using binary connectives $\backslash$, $/$ and $\cdot$.
- Example: $p/(q\backslash p)$, $(p \cdot q)/(q \cdot p)$ are types.
- A sequent is a structure of the form $A_1, \ldots, A_n \to B$ where A_i, B are types.

Lambek calculus: Axiom and Rules

$$\frac{}{A \to A}\ (\mathrm{Ax})$$

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

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

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

$\Gamma, \Delta, \Pi, \Psi$ are sequences of types ($\Pi, \Psi \neq \Lambda$), A, B, C are types.

Definition

A sequent $\Pi \to A$ is derivable if it can be obtained from axiom sequents using rules. This is denoted as $\mathrm{L} \vdash \Pi \to A$.

Examples

- $\mathrm{L} \vdash np, (np\backslash s)/np, np \to s$:

$$\dfrac{\dfrac{s \to s \quad np \to np}{np, np\backslash s \to s}\ (\backslash \to) \qquad np \to np}{np, (np\backslash s)/np, np \to s}\ (/ \to)$$

Examples

- $\mathrm{L} \vdash np, (np\backslash s)/np, np \to s$:

$$\dfrac{\dfrac{s \to s \quad np \to np}{np, np\backslash s \to s}\,(\backslash \to) \qquad np \to np}{np, (np\backslash s)/np, np \to s}\,(/ \to)$$

- $\mathrm{L} \vdash p \cdot (p\backslash q) \to q$:

$$\dfrac{\dfrac{q \to q \quad p \to p}{p, (p\backslash q) \to q}\,(\backslash \to)}{p \cdot (p\backslash q) \to q}\,(\cdot \to)$$

Lambek Categorial Grammars

Definition

A Lambek categorial grammar consists of

- An alphabet Σ;
- A finite correspondence $\triangleright$, which assigns one or several types to each symbol in Σ.
- A distinguished type S.

Lambek Categorial Grammars

Definition

A Lambek categorial grammar consists of

- An alphabet Σ;
- A finite correspondence $\triangleright$, which assigns one or several types to each symbol in Σ.
- A distinguished type S.

It generates the language of the strings $a_1 \dots a_n$ over Σ such that:

$$\begin{array}{cccc} & a_1 \quad \dots \quad a_n & \\ & \triangleright \qquad\quad\; \triangleright & \\ \mathrm{L} \vdash & T_1, \dots, T_n & \to S \end{array}$$

Example

- s, np — primitive types.

Example

- s, np — primitive types.
- Alphabet Σ: *Olaf*, *Astrid*, *loves*, *sleeps*.

Example

- s, np — primitive types.
- Alphabet Σ: *Olaf*, *Astrid*, *loves*, *sleeps*.
- Correspondence:

$$\begin{array}{ccc} Astrid & \triangleright & np \\ Olaf & \triangleright & np \\ loves & \triangleright & (np\backslash s)/np \\ sleeps & \triangleright & np\backslash s \end{array}$$

Example

- s, np — primitive types.
- Alphabet Σ: *Olaf*, *Astrid*, *loves*, *sleeps*.
- Correspondence:

$$\begin{array}{ccc} \mathit{Astrid} & \triangleright & np \\ \mathit{Olaf} & \triangleright & np \\ \mathit{loves} & \triangleright & (np\backslash s)/np \\ \mathit{sleeps} & \triangleright & np\backslash s \end{array}$$

- Distinguished type: s (sentence).

Example

- s, np — primitive types.
- Alphabet Σ: *Olaf*, *Astrid*, *loves*, *sleeps*.
- Correspondence:

$$\begin{array}{ccc} Astrid & \triangleright & np \\ Olaf & \triangleright & np \\ loves & \triangleright & (np\backslash s)/np \\ sleeps & \triangleright & np\backslash s \end{array}$$

- Distinguished type: s (sentence).
- This grammar accepts strings like *Olaf sleeps*, *Olaf loves Astrid* etc.

$$\begin{array}{ccccc} & Olaf & loves & Astrid & \\ & \triangleright & \triangleright & \triangleright & \\ \mathrm{L} \vdash & np, & (np\backslash s)/np, & np & \to s \end{array}$$

Hypergraph Lambek Grammars

ICGT 2021:

- Hypergraph Lambek calculus HL and hypergraph Lambek grammars.

Hypergraph Lambek Grammars

ICGT 2021:

- Hypergraph Lambek calculus HL and hypergraph Lambek grammars.
- Generalized types, sequents, axioms and rules.

Hypergraph Lambek Grammars

ICGT 2021:

- Hypergraph Lambek calculus HL and hypergraph Lambek grammars.
- Generalized types, sequents, axioms and rules.
- The membership problem for hypergraph Lambek grammars is NP-complete.

Hypergraph Lambek Grammars

ICGT 2021:

- Hypergraph Lambek calculus HL and hypergraph Lambek grammars.
- Generalized types, sequents, axioms and rules.
- The membership problem for hypergraph Lambek grammars is NP-complete.
- Hypergraph Lambek grammars are able to generate more languages than hyperedge replacement grammars (the language of all graphs, languages with non-linear growth of the number of edges etc.).

Hypergraph Lambek Grammars

ICGT 2021:

- Hypergraph Lambek calculus HL and hypergraph Lambek grammars.
- Generalized types, sequents, axioms and rules.
- The membership problem for hypergraph Lambek grammars is NP-complete.
- Hypergraph Lambek grammars are able to generate more languages than hyperedge replacement grammars (the language of all graphs, languages with non-linear growth of the number of edges etc.).
- What class of languages do hypergraph Lambek grammars generate?

Hypergraphs

- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.

Hypergraphs

- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.
- One can replace a hyperedge e_0 in G by a hypergraph H as follows:

Hypergraphs

- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.
- One can replace a hyperedge e_0 in G by a hypergraph H as follows:
  1. Remove e_0 from G.

Hypergraphs

- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.
- One can replace a hyperedge e_0 in G by a hypergraph H as follows:
  1. Remove e_0 from G.
  2. Insert a copy of H.

Hypergraphs

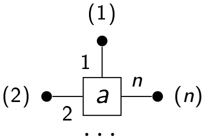
- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.
- One can replace a hyperedge e_0 in G by a hypergraph H as follows:
  1. Remove e_0 from G.
  2. Insert a copy of H.
  3. For each i, fuse the i-th external node of H with the i-th attachment node of e_0.

Hypergraphs

- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.
- One can replace a hyperedge e_0 in G by a hypergraph H as follows:
  1. Remove e_0 from G.
  2. Insert a copy of H.
  3. For each i, fuse the i-th external node of H with the i-th attachment node of e_0.
  4. The result is denoted as $G[e_0/H]$.

Hypergraphs

- A hypergraph consists of nodes V and of hyperedges E. A function $att : E \to V^*$ attaches hyperedges to nodes; a function $lab : E \to C$ labels hyperedges; $ext \in V^*$ are external nodes.
- One can replace a hyperedge e_0 in G by a hypergraph H as follows:
  1. Remove e_0 from G.
  2. Insert a copy of H.
  3. For each i, fuse the i-th external node of H with the i-th attachment node of e_0.
  4. The result is denoted as $G[e_0/H]$.
- $a^\bullet$ is a hypergraph with one a-labeled hyperedge attached to n distinct nodes, which are all external in the same order as attachment nodes:



Hypergraph DPO Grammars

- We consider the category of hypergraphs without external nodes.

Hypergraph DPO Grammars

- We consider the category of hypergraphs without external nodes.
- Rules are of the form $(L \overset{\varphi_L}{\leftarrow} I \overset{\varphi_R}{\rightarrow} R)$ with a *discrete* hypergraph I.

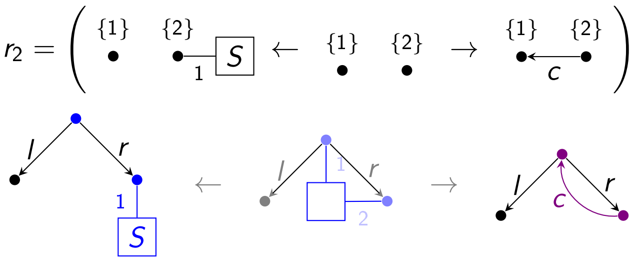
Hypergraph DPO Grammars

- We consider the category of hypergraphs without external nodes.
- Rules are of the form $(L \stackrel{\varphi_L}{\leftarrow} I \stackrel{\varphi_R}{\rightarrow} R)$ with a *discrete* hypergraph I.

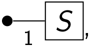
Hypergraph DPO Grammars

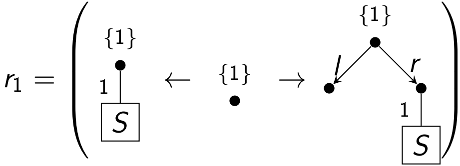
- We consider the category of hypergraphs without external nodes.
- Rules are of the form $(L \xleftarrow{\varphi_L} I \xrightarrow{\varphi_R} R)$ with a *discrete* hypergraph I.

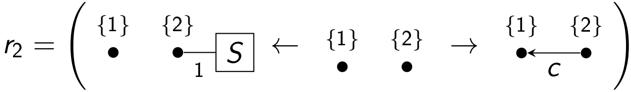
Example



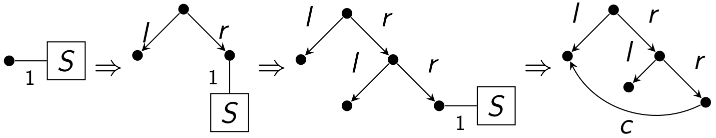
DPO grammar — Example

Let the start hypergraph be , and let the productions be as follows:





Example



Results of this work

Definition

Given a DPO hypergraph grammar with a start hypergraph S and a set of productions P, let $L_c(HGr)$ consist of all hypergraphs $H \in L(HGr)$ such that there exists a derivation $S \overset{*}{\Rightarrow}_P H$ with no more than $c \cdot |E_H|$ steps.

Results of this work

Definition

Given a DPO hypergraph grammar with a start hypergraph S and a set of productions P, let $L_c(HGr)$ consist of all hypergraphs $H \in L(HGr)$ such that there exists a derivation $S \overset{*}{\Rightarrow}_P H$ with no more than $c \cdot |E_H|$ steps.

Theorem

If HGr is a DPO grammar and $1 \leq c \in \mathbb{N}$, then the language $L_c(HGr)$ can be generated by a hypergraph Lambek grammar.

Results of this work

Definition

Given a DPO hypergraph grammar with a start hypergraph S and a set of productions P, let $L_c(HGr)$ consist of all hypergraphs $H \in L(HGr)$ such that there exists a derivation $S \overset{*}{\Rightarrow}_P H$ with no more than $c \cdot |E_H|$ steps.

Theorem

If HGr is a DPO grammar and $1 \leq c \in \mathbb{N}$, then the language $L_c(HGr)$ can be generated by a hypergraph Lambek grammar.

Claim: each language generated by a hypergraph Lambek grammar is of the form $L_c(HGr)$ for some DPO grammar HGr.

Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.

Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.
- They are defined inductively.

Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.
- They are defined inductively.
- There is the set of primitive types Pr and two constructors of types: $\div$ (division) and $\times$ (product).

Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.
- They are defined inductively.
- There is the set of primitive types Pr and two constructors of types: $\div$ (division) and $\times$ (product).
- Let N be a type and let D be a hypergraph such that all its hyperedges except for one are labeled by types, and the remaining one is labeled by a special $\$_n$ label. Then $N \div D$ is a type.

Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.
- They are defined inductively.
- There is the set of primitive types Pr and two constructors of types: $\div$ (division) and $\times$ (product).
- Let N be a type and let D be a hypergraph such that all its hyperedges except for one are labeled by types, and the remaining one is labeled by a special $\$_n$ label. Then $N \div D$ is a type.
- Let M be a hypergraph labeled by types. Then $\times(M)$ is a type.

Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.
- They are defined inductively.
- There is the set of primitive types Pr and two constructors of types: $\div$ (division) and $\times$ (product).
- Let N be a type and let D be a hypergraph such that all its hyperedges except for one are labeled by types, and the remaining one is labeled by a special $\$_n$ label. Then $N \div D$ is a type.
- Let M be a hypergraph labeled by types. Then $\times(M)$ is a type.

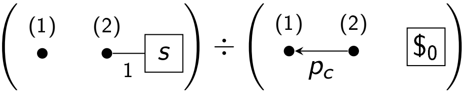
Hypergraph Lambek Calculus — Types

- Types are labels of hyperedges.
- They are defined inductively.
- There is the set of primitive types Pr and two constructors of types: $\div$ (division) and $\times$ (product).
- Let N be a type and let D be a hypergraph such that all its hyperedges except for one are labeled by types, and the remaining one is labeled by a special $\$_n$ label. Then $N \div D$ is a type.
- Let M be a hypergraph labeled by types. Then $\times(M)$ is a type.

Example

Let s, p_c be primitive types.

$\mathrm{DPO}(r_2) = \times$



Hypergraph Lambek Calculus — Sequents

Definition

A sequent is a structure of the form $H \to A$ where H is a hypergraph labeled by types and A is a type.

Hypergraph Lambek Calculus — Axiom and Rules

Axiom: $A^\bullet \to A$ (A is a type).

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

$$\frac{D[e^\$_D/F] \to N}{F \to N \div D}\ (\to \div)$$

$$\frac{H_1 \to lab_M(m_1) \quad \dots \quad H_l \to lab_M(m_l)}{M[m_1/H_1]\dots[m_l/H_l] \to \times(M)}\ (\to \times)$$

$$\frac{H[e/M] \to A}{H[e/(\times(M))^\bullet] \to A}\ (\times \to)$$

Here $e \in E_H$; $E_D = \{e^\$_D, d_1, \dots, d_k\}$; $E_M = \{m_1, \dots, m_l\}$.

Hypergraph Lambek Grammars

Definition

A hypergraph Lambek grammar consists of

- A terminal alphabet of labels Σ;
- A finite correspondence $\triangleright$, which assigns one or several types of the hypergraph Lambek calculus to each symbol in Σ.
- A distinguished type S.

Hypergraph Lambek Grammars

Definition

A hypergraph Lambek grammar consists of

- A terminal alphabet of labels Σ;
- A finite correspondence $\triangleright$, which assigns one or several types of the hypergraph Lambek calculus to each symbol in Σ.
- A distinguished type S.

This grammar generates the language of hypergraphs G over Σ, for which a function $f_G : E_G \to Tp$ exists such that

1. $lab_G(e) \triangleright f_G(e)$ for all hyperedges $e \in E_G$;
2. If $f_G(G)$ is the result of relabeling of G using f_G, then

$$\mathrm{HL} \vdash f_G(G) \to S.$$

From DPO to Hypergraph Lambek Grammars

- Hypergraph Lambek calculus HL works with types and sequents = with nested structures made of hypergraphs.

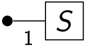
From DPO to Hypergraph Lambek Grammars

- Hypergraph Lambek calculus HL works with types and sequents = with nested structures made of hypergraphs.
- Each rule application leaves a "trace", namely, a new type appears after any rule application.

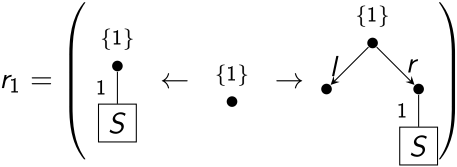
From DPO to Hypergraph Lambek Grammars

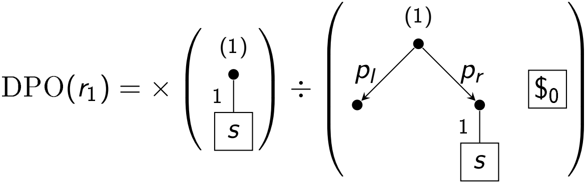
- Hypergraph Lambek calculus HL works with types and sequents = with nested structures made of hypergraphs.
- Each rule application leaves a "trace", namely, a new type appears after any rule application.
- To prove the theorem we encode DPO rules by types of the hypergraph Lambek calculus.

Example of a convertion

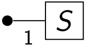
Consider the DPO grammar with the start hypergraph 

and with the productions r_1, r_2. We introduce new primitive types p_l, p_r, p_c, s.

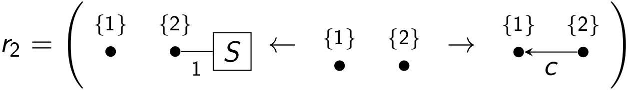


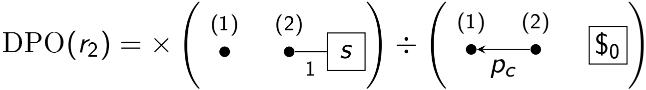


Example of a convertion

Consider the DPO grammar with the start hypergraph 

and with the productions r_1, r_2. We introduce new primitive types p_l, p_r, p_c, s.





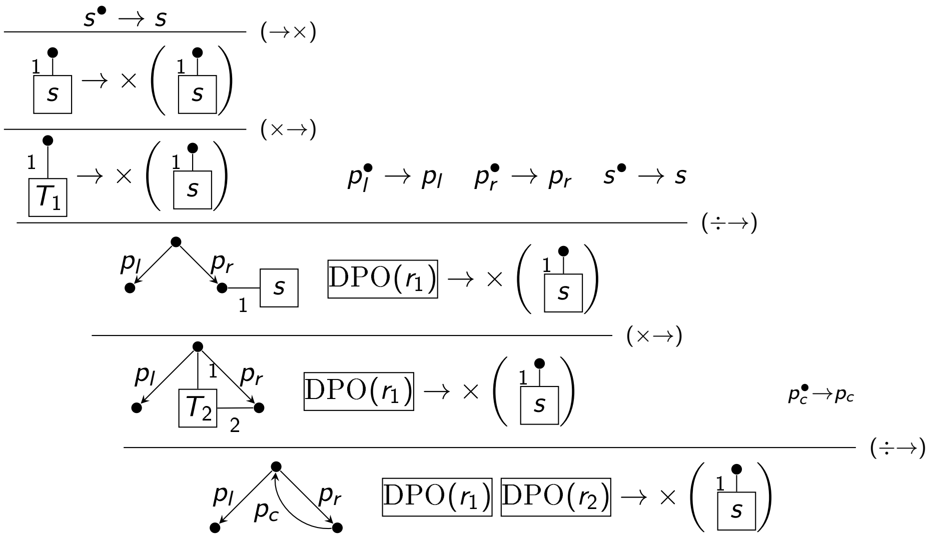
Example of a convertion

If we want to generate $L_2(HGr)$, then we construct the following grammar:

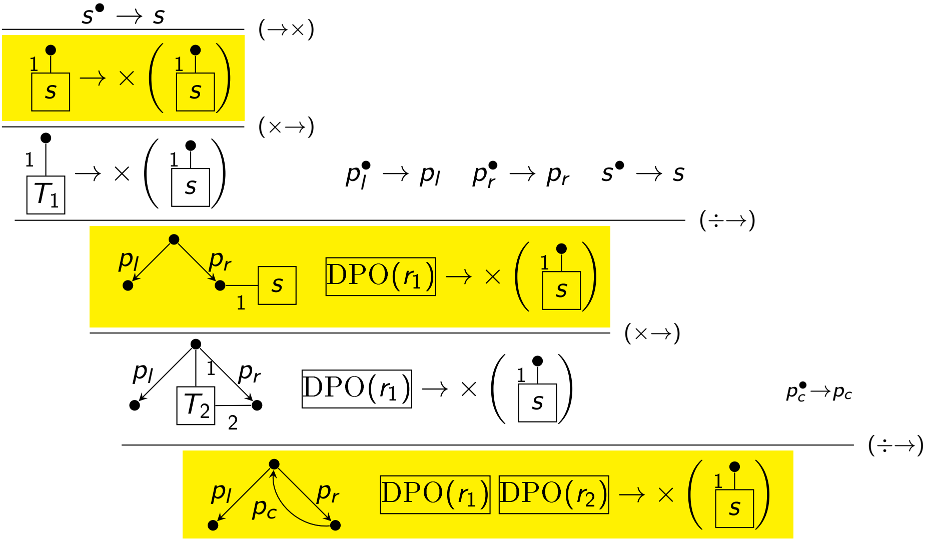
1. Alphabet: l, r, c.
2. A correspondence:
   - $l \triangleright p_l$, $r \triangleright p_r$, $c \triangleright p_c$;
   - $l \triangleright \times\left((1)\ \bullet \xrightarrow{p_l} \bullet\ (2) \quad \boxed{\mathrm{DPO}(r_i)} \right)$, $r \triangleright \times\left((1)\ \bullet \xrightarrow{p_r} \bullet\ (2) \quad \boxed{\mathrm{DPO}(r_i)} \right)$,
     $c \triangleright \times\left((1)\ \bullet \xrightarrow{p_c} \bullet\ (2) \quad \boxed{\mathrm{DPO}(r_i)} \right)$;
   - $l \triangleright \times\left((1)\ \bullet \xrightarrow{p_l} \bullet\ (2) \quad \boxed{\mathrm{DPO}(r_i)} \ \boxed{\mathrm{DPO}(r_j)} \right)$,
     $r \triangleright \times\left((1)\ \bullet \xrightarrow{p_r} \bullet\ (2) \quad \boxed{\mathrm{DPO}(r_i)} \ \boxed{\mathrm{DPO}(r_j)} \right)$,
     $c \triangleright \times\left((1)\ \bullet \xrightarrow{p_c} \bullet\ (2) \quad \boxed{\mathrm{DPO}(r_i)} \ \boxed{\mathrm{DPO}(r_j)} \right)$.

   Here $i, j = 1, 2, 3$.
3. A distinguished type: $\times\left(\bullet \underset{1}{\text{—}} \boxed{s} \right)$.

Example of a derivation



Example of a derivation



Results

Theorem

If HGr is a DPO grammar and $1 \leq c \in \mathbb{N}$*, then the language* $L_c(HGr)$ *can be generated by a hypergraph Lambek grammar.*

- The limitation is essential: DPO grammars are universal while HL-grammars are NP-complete.

Results

Theorem

If HGr is a DPO grammar and $1 \leq c \in \mathbb{N}$, then the language $L_c(HGr)$ can be generated by a hypergraph Lambek grammar.

- The limitation is essential: DPO grammars are universal while HL-grammars are NP-complete.
- Q: How to generalize the hypergraph Lambek calculus in order to overcome this limitation?

Results

Theorem

If HGr is a DPO grammar and $1 \leq c \in \mathbb{N}$, then the language $L_c(HGr)$ can be generated by a hypergraph Lambek grammar.

- The limitation is essential: DPO grammars are universal while HL-grammars are NP-complete.
- Q: How to generalize the hypergraph Lambek calculus in order to overcome this limitation?
- A: To allow one unlimitedly copying types of the form $\mathrm{DPO}(r)$ in left-hand sides of sequents.

Hypergraph Conjunctive Kleene Star

- Palka 2007, Kuznetsov 2021, . . . : the Lambek calculus enriched with Kleene star (aka action logic).
- It allows one to unlimitedly copy types in right-hand sides of sequents.

Hypergraph Conjunctive Kleene Star

- Palka 2007, Kuznetsov 2021, . . . : the Lambek calculus enriched with Kleene star (aka action logic).
- It allows one to unlimitedly copy types in right-hand sides of sequents.

Definition

The hypergraph Lambek calculus with conjunctive Kleene star ${}^{*}\mathrm{HL}_{\omega}$:

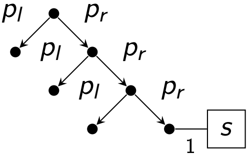
- Types are built using an additional constructor: if A is a type of rank 0, then $\overset{*}{\circ}A$ is also a type.
- We add the following rules:

$$\frac{(H \to \times(n \cdot A^{\bullet}))_{n=0}^{\infty}}{H \to \overset{*}{\circ}A}\ (\to^{*})_{\omega} \qquad\qquad \frac{G + n \cdot A^{\bullet} \to B}{G + \overset{*}{\circ}A^{\bullet} \to B}\ ({}^{*}\!\to),\ n \geq 0$$

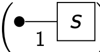
Here $G + H$ is a disjoint sum of hypergraphs, and $n \cdot H$ is H copied n times.

Hypergraph Conjunctive Kleene Star

- Palka 2007, Kuznetsov 2021, . . . : the Lambek calculus enriched with Kleene star (aka action logic).
- It allows one to unlimitedly copy types in right-hand sides of sequents.

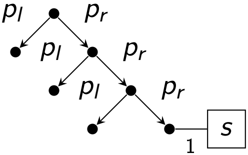


$\boxed{\mathrm{DPO}(r_1)}\ \boxed{\mathrm{DPO}(r_1)}\ \boxed{\mathrm{DPO}(r_1)} \to \times\Big($

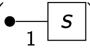


$\Big)$

$({}^{*}\!\to)$



$\boxed{\overset{*}{O}(\mathrm{DPO}(r_1))} \to \times\Big($



$\Big)$

Results

Theorem (cut elimination)

If ${}^*\mathrm{HL}_\omega \vdash H \to A$ *and* ${}^*\mathrm{HL}_\omega \vdash G[e/A^\bullet] \to B$, *then* ${}^*\mathrm{HL}_\omega \vdash G[e/H] \to B$.

Results

Theorem (cut elimination)

If ${}^*\mathrm{HL}_\omega \vdash H \to A$ *and* ${}^*\mathrm{HL}_\omega \vdash G[e/A^\bullet] \to B$*, then* ${}^*\mathrm{HL}_\omega \vdash G[e/H] \to B$*.*

Theorem

Each language generated by a DPO grammar can be generated by a categorial grammar over the calculus ${}^*\mathrm{HL}_\omega$*.*

Results

Theorem (cut elimination)

If $^*\mathrm{HL}_\omega \vdash H \to A$ *and* $^*\mathrm{HL}_\omega \vdash G[e/A^\bullet] \to B$*, then* $^*\mathrm{HL}_\omega \vdash G[e/H] \to B$*.*

Theorem

Each language generated by a DPO grammar can be generated by a categorial grammar over the calculus $^*\mathrm{HL}_\omega$*.*

Theorem

The problem of whether a sequent is derivable in $^*\mathrm{HL}_\omega$ *is undecidable.*

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.
- The former class itself is interesting.

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.
- Future work:

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ ($=$) languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.
- Future work:
    1. To complete the description of languages generated by HL-grammars.

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.
- Future work:
  1. To complete the description of languages generated by HL-grammars.
  2. To develop a more general definition of the hypergraph Lambek calculus using the category theory.

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.
- Future work:
  1. To complete the description of languages generated by HL-grammars.
  2. To develop a more general definition of the hypergraph Lambek calculus using the category theory.
  3. How powerful are categorial grammars based on ${}^{*}\mathrm{HL}_{\omega}$?

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ ($=$) languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.
- Future work:
  1. To complete the description of languages generated by HL-grammars.
  2. To develop a more general definition of the hypergraph Lambek calculus using the category theory.
  3. How powerful are categorial grammars based on ${}^{*}\mathrm{HL}_{\omega}$?

Conclusion

- DPO rules can be modeled by types of the hypergraph Lambek calculus.
- Languages generated by DPO grammars with the length of a derivation linearly bounded by the number of hyperedges $\subseteq$ $(=)$ languages generated by HL-grammars.
- The former class itself is interesting.
- Hypergraph Lambek grammars enriched with the conjunctive Kleene star subsume DPO grammars and thus are universal.
- Future work:
  1. To complete the description of languages generated by HL-grammars.
  2. To develop a more general definition of the hypergraph Lambek calculus using the category theory.
  3. How powerful are categorial grammars based on ${}^*\mathrm{HL}_\omega$?

Thank you for attention!