Complexity of the Lambek Calculus and Its Extensions

Stepan L. Kuznetsov

LACompLing 2021 · MALIN 2021

Steklov Mathematical Institute, RAS

About This Talk

- This talk will be probably more technical than most of the talks at the conference.

About This Talk

- This talk will be probably more technical than most of the talks at the conference.
- We shall discuss some of the *mathematical tools* under categorial grammar formalisms.

About This Talk

- This talk will be probably more technical than most of the talks at the conference.
- We shall discuss some of the *mathematical tools* under categorial grammar formalisms.
- Once a formalism gets introduced, and it is validated that it captures the desired language phenomena, mathematical questions concerning this formalism arise — most importantly algorithmic ones.

About This Talk

- This talk will be probably more technical than most of the talks at the conference.
- We shall discuss some of the *mathematical tools* under categorial grammar formalisms.
- Once a formalism gets introduced, and it is validated that it captures the desired language phenomena, mathematical questions concerning this formalism arise — most importantly algorithmic ones.
- And this is where mathematicians enter the battle.

Lambek Calculus

- We start with the Lambek calculus, which was introduced by Joachim Lambek (1958) for mathematical description of natural language syntax.

Lambek Calculus

- We start with the Lambek calculus, which was introduced by Joachim Lambek (1958) for mathematical description of natural language syntax.
- We formulate it as a sequent calculus.

Lambek Calculus

- We start with the Lambek calculus, which was introduced by Joachim Lambek (1958) for mathematical description of natural language syntax.
- We formulate it as a sequent calculus.
- Sequents are expressions of the form $A_1, \ldots, A_n \to B$, where A_i and B are formulae built using $\backslash$, $/$, $\cdot$.

Lambek Calculus

- We start with the Lambek calculus, which was introduced by Joachim Lambek (1958) for mathematical description of natural language syntax.
- We formulate it as a sequent calculus.
- Sequents are expressions of the form $A_1, \ldots, A_n \to B$, where A_i and B are formulae built using $\backslash$, $/$, $\cdot$.
- There is a subtle question whether one should allow empty antecedents ($n = 0$). The constraint which disallows them is called *Lambek's non-emptiness restriction.*

Lambek Calculus

- Axioms and inference rules of the Lambek calculus are as follows:

$$\frac{}{A \to A}\ Id \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \cdot B, \Delta \to C}\ \cdot L \qquad \frac{\Pi \to A \quad \Delta \to B}{\Pi, \Delta \to A \cdot B}\ \cdot R$$

$$\frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, \Pi, A \setminus B, \Delta \to C}\ \setminus L \qquad \frac{A, \Pi \to B}{\Pi \to A \setminus B}\ \setminus R$$

$$\frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, B / A, \Pi, \Delta \to C}\ / L \qquad \frac{\Pi, A \to B}{\Pi \to B / A}\ \setminus R$$

Lambek Calculus

- Axioms and inference rules of the Lambek calculus are as follows:

$$\frac{}{A \to A}\ Id \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \cdot B, \Delta \to C}\ \cdot L \qquad \frac{\Pi \to A \quad \Delta \to B}{\Pi, \Delta \to A \cdot B}\ \cdot R$$

$$\frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, \Pi, A \setminus B, \Delta \to C}\ \setminus L \qquad \frac{A, \Pi \to B}{\Pi \to A \setminus B}\ \setminus R$$

$$\frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, B / A, \Pi, \Delta \to C}\ / L \qquad \frac{\Pi, A \to B}{\Pi \to B / A}\ \setminus R$$

- Lambek's restriction here means adding "Π is non-empty" to $\setminus R$ and $/ R$.

Lambek Calculus

- The Lambek calculus is used for modelling natural language:

Pete	met	the	girl	whom	John	likes
N,	$(N \setminus S) / N$,	N / CN,	CN,	$(CN \setminus CN) / (S / N)$,	N,	$(N \setminus S) / N$
						$\rightarrow S$

Lambek Calculus

- The Lambek calculus is used for modelling natural language:

Pete	met	the	girl	whom	John	likes	
N,	$(N \setminus S) / N$,	N / CN,	CN,	$(CN \setminus CN) / (S / N)$,	N,	$(N \setminus S) / N$	$\to S$

- As one can see from this example, the Lambek calculus is capable of handling quite involved syntax.

Lambek Calculus

- The Lambek calculus is used for modelling natural language:

Pete	met	the	girl	whom	John	likes	
N,	$(N \setminus S) / N$,	N / CN,	CN,	$(CN \setminus CN) / (S / N)$,	N,	$(N \setminus S) / N$	$\to S$

- As one can see from this example, the Lambek calculus is capable of handling quite involved syntax.
- Moreover, it can be considered as the preferable system among others, since it is exactly the atomic theory of the language algebra (Pentus 1995).

Lambek Calculus

- The Lambek calculus is used for modelling natural language:

Pete	met	the	girl	whom	John	likes	
N,	$(N \setminus S) / N$,	N / CN,	CN,	$(CN \setminus CN) / (S / N)$,	N,	$(N \setminus S) / N$	$\to S$

- As one can see from this example, the Lambek calculus is capable of handling quite involved syntax.
- Moreover, it can be considered as the preferable system among others, since it is exactly the atomic theory of the language algebra (Pentus 1995).
- However, it has severe theoretical limitations.

Limitations of the Lambek Calculus

- The first issue is **expressivity.**

Limitations of the Lambek Calculus

- The first issue is **expressivity.**
- Lambek grammars can generate only context-free languages (Pentus 1993 — the proof of Chomsky conjecture).

Limitations of the Lambek Calculus

- The first issue is **expressivity.**
- Lambek grammars can generate only context-free languages (Pentus 1993 — the proof of Chomsky conjecture).
- On the other hand, natural language syntax includes non-context-free phenomena (Shieber 1985).

Limitations of the Lambek Calculus

- The first issue is **expressivity.**
- Lambek grammars can generate only context-free languages (Pentus 1993 — the proof of Chomsky conjecture).
- On the other hand, natural language syntax includes non-context-free phenomena (Shieber 1985).
- The second issue is **complexity.**

Limitations of the Lambek Calculus

- The first issue is **expressivity.**
- Lambek grammars can generate only context-free languages (Pentus 1993 — the proof of Chomsky conjecture).
- On the other hand, natural language syntax includes non-context-free phenomena (Shieber 1985).
- The second issue is **complexity.**
- The Lambek calculus is NP-complete (Pentus 2006), as well as its $(\backslash, /)$- and $(\backslash, \cdot)$-fragments (Savateev 2008).

Limitations of the Lambek Calculus

- The first issue is **expressivity.**
- Lambek grammars can generate only context-free languages (Pentus 1993 — the proof of Chomsky conjecture).
- On the other hand, natural language syntax includes non-context-free phenomena (Shieber 1985).
- The second issue is **complexity.**
- The Lambek calculus is NP-complete (Pentus 2006), as well as its $(\backslash, /)$- and $(\backslash, \cdot)$-fragments (Savateev 2008).
- The only polynomially decidable fragment is the one with only $\backslash$ (Savateev 2007).

Limitations of the Lambek Calculus

- The first issue is **expressivity.**
- Lambek grammars can generate only context-free languages (Pentus 1993 — the proof of Chomsky conjecture).
- On the other hand, natural language syntax includes non-context-free phenomena (Shieber 1985).
- The second issue is **complexity.**
- The Lambek calculus is NP-complete (Pentus 2006), as well as its $(\backslash, /)$- and $(\backslash, \cdot)$-fragments (Savateev 2008).
- The only polynomially decidable fragment is the one with only $\backslash$ (Savateev 2007).
- Thus, we should *simultaneously* try to increase expressivity and harness complexity.

This Talk

- A survey of complexity results for the Lambek calculus itself and its fragment was given at the talk of Mati Pentus at *Advances in Modal Logic 2010.*

This Talk

- A survey of complexity results for the Lambek calculus itself and its fragment was given at the talk of Mati Pentus at *Advances in Modal Logic 2010.*
- In this talk, we shall discuss algorithmic complexity for extensions and variants of the Lambek calculus, trying to see how expressive power correlates with complexity growth.

This Talk

- A survey of complexity results for the Lambek calculus itself and its fragment was given at the talk of Mati Pentus at *Advances in Modal Logic 2010.*
- In this talk, we shall discuss algorithmic complexity for extensions and variants of the Lambek calculus, trying to see how expressive power correlates with complexity growth.
- The presentation is based on joint work and discussions with Mati Pentus, Max Kanovich, Andre Scedrov, Glyn Morrill, and Stanislav Speranski.

Categorial Grammar Parsers

- The importance of algorithmic decidability and reasonable complexity bounds is connected to the usage of Lambek-style formalisms is natural language parsers.

Categorial Grammar Parsers

- The importance of algorithmic decidability and reasonable complexity bounds is connected to the usage of Lambek-style formalisms is natural language parsers.
- The (probably) most well-known Lambek-based parsers are *Grail* (Moot et al.) and *CatLog* (Morrill et al.).

Categorial Grammar Parsers

- The importance of algorithmic decidability and reasonable complexity bounds is connected to the usage of Lambek-style formalisms is natural language parsers.
- The (probably) most well-known Lambek-based parsers are *Grail* (Moot et al.) and *CatLog* (Morrill et al.).
- Grail uses the *non-associative* Lambek calculus as the basic system, which is out of the scope of this talk.

Categorial Grammar Parsers

- The importance of algorithmic decidability and reasonable complexity bounds is connected to the usage of Lambek-style formalisms is natural language parsers.
- The (probably) most well-known Lambek-based parsers are *Grail* (Moot et al.) and *CatLog* (Morrill et al.).
- Grail uses the *non-associative* Lambek calculus as the basic system, which is out of the scope of this talk.
- However, the non-associative Lambek calculus, unlike the associative one, is polynomially decidable.

Categorial Grammar Parsers

- The importance of algorithmic decidability and reasonable complexity bounds is connected to the usage of Lambek-style formalisms is natural language parsers.
- The (probably) most well-known Lambek-based parsers are *Grail* (Moot et al.) and *CatLog* (Morrill et al.).
- Grail uses the *non-associative* Lambek calculus as the basic system, which is out of the scope of this talk.
- However, the non-associative Lambek calculus, unlike the associative one, is polynomially decidable.
- CatLog's calculus is associative, but includes a mechanism of *brackets* for controlled non-associativity.

Is NP Too Hard?

- Pentus' result on NP-hardness of the Lambek calculus was considered as a negative one, limiting the use of the Lambek calculus in practices (in comparison with other categorial grammar formalisms, which are polynomially tractable).

Is NP Too Hard?

- Pentus' result on NP-hardness of the Lambek calculus was considered as a negative one, limiting the use of the Lambek calculus in practices (in comparison with other categorial grammar formalisms, which are polynomially tractable).
- However, in fact the input parameter is the length of a *sentence*, not the whole *text*, so it is roughly $\lesssim 20$.

Is NP Too Hard?

- Pentus' result on NP-hardness of the Lambek calculus was considered as a negative one, limiting the use of the Lambek calculus in practices (in comparison with other categorial grammar formalisms, which are polynomially tractable).
- However, in fact the input parameter is the length of a *sentence*, not the whole *text*, so it is roughly $\lesssim 20$.
- This makes NP brute-force-search algorithms, with good optimisations, practically usable.

Is NP Too Hard?

- Pentus' result on NP-hardness of the Lambek calculus was considered as a negative one, limiting the use of the Lambek calculus in practices (in comparison with other categorial grammar formalisms, which are polynomially tractable).
- However, in fact the input parameter is the length of a *sentence*, not the whole *text*, so it is roughly $\lesssim 20$.
- This makes NP brute-force-search algorithms, with good optimisations, practically usable.
- Moreover, those can be easily expanded to some "algorithmically harmless" extensions of the system.

Extensions of the Lambek Calculus

- In real-life applications, the Lambek calculus is extended by a huge amount of extra operations and syntactic mechanisms.

Extensions of the Lambek Calculus

- In real-life applications, the Lambek calculus is extended by a huge amount of extra operations and syntactic mechanisms.
- For example, the calculus of Morrill's CatLog uses about 45 operations (and this number could grow in the future).

Extensions of the Lambek Calculus

- In real-life applications, the Lambek calculus is extended by a huge amount of extra operations and syntactic mechanisms.
- For example, the calculus of Morrill's CatLog uses about 45 operations (and this number could grow in the future).
- While it is impossible to discuss all these operations in detail during this talk, we shall try to classify them from the algorithmic point of view.

Extensions of the Lambek Calculus

- In real-life applications, the Lambek calculus is extended by a huge amount of extra operations and syntactic mechanisms.
- For example, the calculus of Morrill's CatLog uses about 45 operations (and this number could grow in the future).
- While it is impossible to discuss all these operations in detail during this talk, we shall try to classify them from the algorithmic point of view.
- Most of these operations are *"harmless,"* in the sense that adding these operations does not increase algorithmic complexity (if we are already in a non-deterministic class like NP or PSPACE).

Extensions of the Lambek Calculus

- In real-life applications, the Lambek calculus is extended by a huge amount of extra operations and syntactic mechanisms.
- For example, the calculus of Morrill's CatLog uses about 45 operations (and this number could grow in the future).
- While it is impossible to discuss all these operations in detail during this talk, we shall try to classify them from the algorithmic point of view.
- Most of these operations are *"harmless,"* in the sense that adding these operations does not increase algorithmic complexity (if we are already in a non-deterministic class like NP or PSPACE).
- Others are *"dangerous,"* as they lead to complexity growth and even to undecidability.

“Harmless” Extensions

- Fortunately, most of the extensions are “harmless.”

“Harmless” Extensions

- Fortunately, most of the extensions are “harmless.”
- One of such extensions introduces the *bracketing* mechanism with modalities and bracket structure which restrict associativity.

“Harmless” Extensions

- Fortunately, most of the extensions are “harmless.”
- One of such extensions introduces the *bracketing* mechanism with modalities and bracket structure which restrict associativity.
- This disallows incorrect sentences like “the girl whom John likes Mary and Pete likes” (which are validated by the original Lambek calculus), by making the dependent clause a strong island: “[[John likes Mary and Pete likes …]].”

“Harmless” Extensions

- Fortunately, most of the extensions are “harmless.”
- One of such extensions introduces the *bracketing* mechanism with modalities and bracket structure which restrict associativity.
- This disallows incorrect sentences like “the girl whom John likes Mary and Pete likes” (which are validated by the original Lambek calculus), by making the dependent clause a strong island: “[[John likes Mary and Pete likes ...]].”
- Another example is *medial extraction:* in order to parse “the girl whom John met yesterday,” we have to move N to the middle of the dependent clause.

“Harmless” Extensions

- Fortunately, most of the extensions are “harmless.”
- One of such extensions introduces the *bracketing* mechanism with modalities and bracket structure which restrict associativity.
- This disallows incorrect sentences like “the girl whom John likes Mary and Pete likes” (which are validated by the original Lambek calculus), by making the dependent clause a strong island: “[[John likes Mary and Pete likes ...]].”
- Another example is *medial extraction:* in order to parse “the girl whom John met yesterday,” we have to move N to the middle of the dependent clause.
- This is achieved by assigning type $(CN \setminus CN) / (S / !N)$ to “whom,” where the ! modality allows permutation.

“Harmless” Extensions

- There are also toy examples of “harmless” extensions, like adding the word reversal operation ^{R} (K. 2012).

“Harmless” Extensions

- There are also toy examples of “harmless” extensions, like adding the word reversal operation ^{R} (K. 2012).
- These extensions, and many others, still keep the derivability problem in the NP class.

“Harmless” Extensions

- There are also toy examples of “harmless” extensions, like adding the word reversal operation ^{R} (K. 2012).
- These extensions, and many others, still keep the derivability problem in the NP class.
- For brackets, there is also a specific question of *bracket guessing*, or *bracket induction* (Morrill et al. 2018), since the bracketing (island) structure is not given as the input.

“Harmless” Extensions

- There are also toy examples of “harmless” extensions, like adding the word reversal operation ^{R} (K. 2012).
- These extensions, and many others, still keep the derivability problem in the NP class.
- For brackets, there is also a specific question of *bracket guessing*, or *bracket induction* (Morrill et al. 2018), since the bracketing (island) structure is not given as the input.
- In the implementations, several techniques are used for optimisation, one of which is *focusing*. Focusing means reorganising a cut-free sequent proof by exchanging rules in a systematic way, which reduces proof-search space.

Pseudopolynomial Algorithms

- For the original Lambek calculus, it is even possible to perform polynomial-time parsing, in the following sense.

Pseudopolynomial Algorithms

- For the original Lambek calculus, it is even possible to perform polynomial-time parsing, in the following sense.
- Let us consider *two* complexity parameters: n for the size of the sequent and d for its *depth* (roughly speaking, nesting of divisions and multiplications).

Pseudopolynomial Algorithms

- For the original Lambek calculus, it is even possible to perform polynomial-time parsing, in the following sense.
- Let us consider *two* complexity parameters: n for the size of the sequent and d for its *depth* (roughly speaking, nesting of divisions and multiplications).
- Pentus (2010) proposed an algorithm whose running time is *polynomial* in n and exponential in d: $\mathrm{poly}(n, 2^d)$.

Pseudopolynomial Algorithms

- For the original Lambek calculus, it is even possible to perform polynomial-time parsing, in the following sense.
- Let us consider *two* complexity parameters: n for the size of the sequent and d for its *depth* (roughly speaking, nesting of divisions and multiplications).
- Pentus (2010) proposed an algorithm whose running time is *polynomial* in n and exponential in d: $\mathrm{poly}(n, 2^d)$.
- Kanovich et al. (2017) extended this to the Lambek calculus with brackets, but the complexity now is only $\mathrm{poly}(n, 2^d, n^b)$, where b is bracket nesting depth.

Pseudopolynomial Algorithms

- For the original Lambek calculus, it is even possible to perform polynomial-time parsing, in the following sense.
- Let us consider *two* complexity parameters: n for the size of the sequent and d for its *depth* (roughly speaking, nesting of divisions and multiplications).
- Pentus (2010) proposed an algorithm whose running time is *polynomial* in n and exponential in d: $\mathrm{poly}(n, 2^d)$.
- Kanovich et al. (2017) extended this to the Lambek calculus with brackets, but the complexity now is only $\mathrm{poly}(n, 2^d, n^b)$, where b is bracket nesting depth.
- Unfortunately, while d is usually small, b could be large (in sentences like “the house that Jack built”).

Pseudopolynomial Algorithms

- For the original Lambek calculus, it is even possible to perform polynomial-time parsing, in the following sense.
- Let us consider *two* complexity parameters: n for the size of the sequent and d for its *depth* (roughly speaking, nesting of divisions and multiplications).
- Pentus (2010) proposed an algorithm whose running time is *polynomial* in n and exponential in d: $\mathrm{poly}(n, 2^d)$.
- Kanovich et al. (2017) extended this to the Lambek calculus with brackets, but the complexity now is only $\mathrm{poly}(n, 2^d, n^b)$, where b is bracket nesting depth.
- Unfortunately, while d is usually small, b could be large (in sentences like “the house that Jack built”).
- Thus, (pseudo)polynomiality is not robust even for “harmless” extensions.

“Dangerous” Extensions

- Unlike “harmless” extensions discussed before, there are certain operation which significantly increase algorithmic complexity.

“Dangerous” Extensions

- Unlike “harmless” extensions discussed before, there are certain operation which significantly increase algorithmic complexity.
- We shall consider the following ones:
  - additive conjunction and disjunction ($\wedge$, $\vee$);
  - subexponentials (!) which allow non-local contraction

$$\frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, !A, \Phi, \Delta \to C}\ !C$$

    or a variant of this rule;
  - the Kleene star, *.

“Harmless” Extensions as “Linear” Ones

- The “harmless” extensions have the following common property: the new rules of inference maintain *linearity*, in the sense that each formula occurrences in the conclusion of a rule maps to *not more than one* occurrence in the premise(s).

“Harmless” Extensions as “Linear” Ones

- The “harmless” extensions have the following common property: the new rules of inference maintain *linearity*, in the sense that each formula occurrences in the conclusion of a rule maps to *not more than one* occurrence in the premise(s).
- This makes the size of cut-free proof polynomial w.r.t. the size of the goal sequent, thus an NP upper bound.

“Harmless” Extensions as “Linear” Ones

- The “harmless” extensions have the following common property: the new rules of inference maintain *linearity*, in the sense that each formula occurrences in the conclusion of a rule maps to *not more than one* occurrence in the premise(s).
- This makes the size of cut-free proof polynomial w.r.t. the size of the goal sequent, thus an NP upper bound.
- In contrast, the rules for “dangerous” operations have more complex premises, which leads to blowup of proof search.

“Harmless” Extensions as “Linear” Ones

- Thus, rules for “harmless” operations, from a bird-eye view, just perform some reorganisation of the same material (formulae).

“Harmless” Extensions as “Linear” Ones

- Thus, rules for “harmless” operations, from a bird-eye view, just perform some reorganisation of the same material (formulae).
- The differences are only in the structure of meta-formulae which form the sequent.

“Harmless” Extensions as “Linear” Ones

- Thus, rules for “harmless” operations, from a bird-eye view, just perform some reorganisation of the same material (formulae).
- The differences are only in the structure of meta-formulae which form the sequent.
- Recently, Pshenitsyn (2021) proposed a general framework for such situations, called the *hypergraph Lambek calculus* HL.

“Harmless” Extensions as “Linear” Ones

- Thus, rules for “harmless” operations, from a bird-eye view, just perform some reorganisation of the same material (formulae).
- The differences are only in the structure of meta-formulae which form the sequent.
- Recently, Pshenitsyn (2021) proposed a general framework for such situations, called the *hypergraph Lambek calculus* HL.
- The syntax of HL is quite involved (using hypergraphs instead of formulae), but it is still in the NP class.

“Harmless” Extensions as “Linear” Ones

- Thus, rules for “harmless” operations, from a bird-eye view, just perform some reorganisation of the same material (formulae).
- The differences are only in the structure of meta-formulae which form the sequent.
- Recently, Pshenitsyn (2021) proposed a general framework for such situations, called the *hypergraph Lambek calculus* HL.
- The syntax of HL is quite involved (using hypergraphs instead of formulae), but it is still in the NP class.
- Moreover, it absorbs many of the known “harmless” extensions of the Lambek calculus, so it could probably become the “umbrella logic” for them.

Additive Operations

- Let us return to more interesting, “dangerous” stuff.

Additive Operations

- Let us return to more interesting, "dangerous" stuff.
- Additive operations are governed by the following inference rules:

$$\frac{\Gamma, A_1, \Delta \to C \quad \Gamma, A_2, \Delta \to C}{\Gamma, A_1 \vee A_2, \Delta \to C} \vee L \qquad \frac{\Pi \to A_i}{\Pi \to A_1 \vee A_2} \vee R$$

$$\frac{\Gamma, A_i, \Delta \to C}{\Gamma, A_1 \wedge A_2, \Delta \to C} \wedge L \qquad \frac{\Pi \to A_1 \quad \Pi \to A_2}{\Pi \to A_1 \wedge A_2} \wedge R$$

Additive Operations

- Let us return to more interesting, “dangerous” stuff.
- Additive operations are governed by the following inference rules:

$$\frac{\Gamma, A_1, \Delta \to C \quad \Gamma, A_2, \Delta \to C}{\Gamma, A_1 \vee A_2, \Delta \to C} \vee L \qquad \frac{\Pi \to A_i}{\Pi \to A_1 \vee A_2} \vee R$$

$$\frac{\Gamma, A_i, \Delta \to C}{\Gamma, A_1 \wedge A_2, \Delta \to C} \wedge L \qquad \frac{\Pi \to A_1 \quad \Pi \to A_2}{\Pi \to A_1 \wedge A_2} \wedge R$$

- Additive operations allow imposing several syntactic conditions on a word at the same time.

Additive Operations

- This allows describing finite intersections of context-free languages (Kanazawa 1992) and, moreover, languages generated by conjunctive grammars (K., Okhotin 2017) and closures of such under symbol-to-symbol homomorphisms.

Additive Operations

- This allows describing finite intersections of context-free languages (Kanazawa 1992) and, moreover, languages generated by conjunctive grammars (K., Okhotin 2017) and closures of such under symbol-to-symbol homomorphisms.
- The latter class includes an NP-hard language, thus we have no hope for a pseudopolynomial algorithm here.

Additive Operations

- This allows describing finite intersections of context-free languages (Kanazawa 1992) and, moreover, languages generated by conjunctive grammars (K., Okhotin 2017) and closures of such under symbol-to-symbol homomorphisms.
- The latter class includes an NP-hard language, thus we have no hope for a pseudopolynomial algorithm here.
- The Lambek calculus with additives (MALC) itself is PSPACE-complete (Kanovich 1994), and this holds even for its $(\backslash, \wedge)$-fragment (Kanovich et al. 2019).

Additive Operations

- This allows describing finite intersections of context-free languages (Kanazawa 1992) and, moreover, languages generated by conjunctive grammars (K., Okhotin 2017) and closures of such under symbol-to-symbol homomorphisms.
- The latter class includes an NP-hard language, thus we have no hope for a pseudopolynomial algorithm here.
- The Lambek calculus with additives (MALC) itself is PSPACE-complete (Kanovich 1994), and this holds even for its $(\backslash, \wedge)$-fragment (Kanovich et al. 2019).
- Additive operations raise complexity, but keep decidability, and they are still under the scope of exponential-free linear logic.

Subexponentials with Contraction

- Real nonlinearity comes with the contraction rule for (sub)exponential modalities.

Subexponentials with Contraction

- Real nonlinearity comes with the contraction rule for (sub)exponential modalities.
- The natural language phenomenon here is the situation of *multiple extraction:* “the paper that the author of signed without reading.”

Subexponentials with Contraction

- Real nonlinearity comes with the contraction rule for (sub)exponential modalities.
- The natural language phenomenon here is the situation of *multiple extraction:* "the paper that the author of signed without reading."
- Here the dependent clause ("the author of ... signed ... without reading ...") includes several *gaps* which should be filled by copies of *the same* N ("the paper").

Subexponentials with Contraction

- Real nonlinearity comes with the contraction rule for (sub)exponential modalities.
- The natural language phenomenon here is the situation of *multiple extraction:* "the paper that the author of signed without reading."
- Here the dependent clause ("the author of ... signed ... without reading ...") includes several *gaps* which should be filled by copies of *the same* N ("the paper").
- This requires some form of contraction rule for $!N$:

$$\frac{\ldots !N \ldots !N \ldots \to C}{\ldots !N \ldots \ \ldots \to C}$$

Subexponentials with Contraction

- The *non-local contraction rule* is formulated as follows:

$$\frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, !A, \Phi, \Delta \to C}\ !NC_1 \qquad \frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, \Phi, !A, \Delta \to C}\ !NC_2$$

Subexponentials with Contraction

- The *non-local contraction rule* is formulated as follows:

$$\frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, !A, \Phi, \Delta \to C}\ !NC_1 \qquad \frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, \Phi, !A, \Delta \to C}\ !NC_2$$

- Non-locality is necessary in the absence of permutation (which is usually not the case); otherwise, logical properties like cut-elimination get broken (Kanovich et al. 2018).

Subexponentials with Contraction

- The *non-local contraction rule* is formulated as follows:

$$\frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, !A, \Phi, \Delta \to C}\ !NC_1 \qquad \frac{\Gamma, !A, \Phi, !A, \Delta \to C}{\Gamma, \Phi, !A, \Delta \to C}\ !NC_2$$

- Non-locality is necessary in the absence of permutation (which is usually not the case); otherwise, logical properties like cut-elimination get broken (Kanovich et al. 2018).
- Extensions of the Lambek calculus with a ! which allows $NC_{1,2}$ are undecidable (Kanovich et al. 2018), more precisely, Σ^0_1-complete.

Subexponentials with Contraction: Undecidability

- The undecidability argument is a refinement of the one used by Lincoln et al. (1992) for proving undecidability for propositional linear logic with the exponential modality (which allows all structural rules: contraction, weakening, and permutation).

Subexponentials with Contraction: Undecidability

- The undecidability argument is a refinement of the one used by Lincoln et al. (1992) for proving undecidability for propositional linear logic with the exponential modality (which allows all structural rules: contraction, weakening, and permutation).
    - Notice that in the non-commutative case the undecidability proof does not require additives.

Subexponentials with Contraction: Undecidability

- The undecidability argument is a refinement of the one used by Lincoln et al. (1992) for proving undecidability for propositional linear logic with the exponential modality (which allows all structural rules: contraction, weakening, and permutation).
  - Notice that in the non-commutative case the undecidability proof does not require additives.
- The argument is based on encoding derivability from finite sets of non-logical axioms.

Subexponentials with Contraction: Undecidability

- The undecidability argument is a refinement of the one used by Lincoln et al. (1992) for proving undecidability for propositional linear logic with the exponential modality (which allows all structural rules: contraction, weakening, and permutation).
  - Notice that in the non-commutative case the undecidability proof does not require additives.
- The argument is based on encoding derivability from finite sets of non-logical axioms.
- The latter problem is also undecidable (Buszkowski 1982, 2005), which is quite unfortunate, since sometimes we want to express extra properties of syntactic types (like subtyping: $N^{\text{inanimate}} \to N$).

Subexponentials with Contraction: Decidability

- However, it turns out that for practical applications subexponentials which allow contraction can be considered harmless!

Subexponentials with Contraction: Decidability

- However, it turns out that for practical applications subexponentials which allow contraction can be considered harmless!
- Let us consider the so-called “relevant modality,” which allows contraction and permutation, but not weakening.

Subexponentials with Contraction: Decidability

- However, it turns out that for practical applications subexponentials which allow contraction can be considered harmless!
- Let us consider the so-called “relevant modality,” which allows contraction and permutation, but not weakening.
- This ! gets applied to types of syntactic objects which get abstracted.

Subexponentials with Contraction: Decidability

- However, it turns out that for practical applications subexponentials which allow contraction can be considered harmless!
- Let us consider the so-called "relevant modality," which allows contraction and permutation, but not weakening.
- This ! gets applied to types of syntactic objects which get abstracted.
- Usually, this is a noun phrase: $!N$. One could potentially think of abstracting a verb, $!(N \setminus S)$, but definitely not more complicated objects.

Subexponentials with Contraction: Decidability

- Thus, formulae which *really* appear under ! are $(\backslash, /)$-formulae of depth not greater than 1: $(q_1 \dots q_n) \setminus p \,/ (r_1 \dots r_m)$.

Subexponentials with Contraction: Decidability

- Thus, formulae which *really* appear under ! are $(\backslash, /)$-formulae of depth not greater than 1: $(q_1 \dots q_n) \setminus p \,/ (r_1 \dots r_m)$.
- And for such formulae under !, the derivability problem is decidable and belongs to the NP class (Dudakov et al. 2021).

Subexponentials with Contraction: Decidability

- Thus, formulae which *really* appear under ! are $(\backslash, /)$-formulae of depth not greater than 1: $(q_1 \dots q_n) \backslash p /(r_1 \dots r_m)$.
- And for such formulae under !, the derivability problem is decidable and belongs to the NP class (Dudakov et al. 2021).
- The algorithm uses a dyadic syntax which propagates all applications of contraction up to the axioms.

Subexponentials with Contraction: Decidability

- Thus, formulae which *really* appear under ! are $(\backslash, /)$-formulae of depth not greater than 1: $(q_1 \dots q_n) \backslash p /(r_1 \dots r_m)$.
- And for such formulae under !, the derivability problem is decidable and belongs to the NP class (Dudakov et al. 2021).
- The algorithm uses a dyadic syntax which propagates all applications of contraction up to the axioms.
- Axiom checking is quite involved, and requires so-called total derivability in context-free grammars.

Subexponentials with Contraction: Brackets

- In Morrill's systems for CatLog, contraction has a subtle interaction with the bracketing structure.
- This is due to the fact that in multiple extraction there is one principal gap, and others are *parasitic* ones, appearing in islands: "[the author of ...] signed ... [without reading ...]."
- Morrill uses different rules in different versions of his system, we cite one of the most recent ones (Morrill 2018):

$$\frac{\Xi(A) \to C}{\Xi(!A) \to C}\ !L \qquad \frac{!A \to B}{!A \to !B}\ !R \qquad \frac{\Xi(\Gamma_1, !A, [!A, \Gamma_2], \Gamma_3) \to C}{\Xi(\Gamma_1, !A, [[\Gamma_2]], \Gamma_3) \to C}\ !C$$

$$\frac{\Xi(\Gamma, !A) \to C}{\Xi(!A, \Gamma) \to C}\ !P_1 \qquad \frac{\Xi(!A, \Gamma) \to C}{\Xi(\Gamma, !A) \to C}\ !P_2$$

Subexponentials with Contraction: Brackets

- In our JoLLI paper (Kanovich, K., Scedrov 2021) we give proof-theoretic analysis of Morrill's systems and prove their undecidability.

Subexponentials with Contraction: Brackets

- In our JoLLI paper (Kanovich, K., Scedrov 2021) we give proof-theoretic analysis of Morrill's systems and prove their undecidability.
- However, the argument involves using bracket modalities under ! to break the bracket discipline.

Subexponentials with Contraction: Brackets

- In our JoLLI paper (Kanovich, K., Scedrov 2021) we give proof-theoretic analysis of Morrill's systems and prove their undecidability.
- However, the argument involves using bracket modalities under ! to break the bracket discipline.
- In real applications this does not happen, and standard proof search algorithms solve the derivability problem.

Subexponentials with Contraction: Brackets

- In our JoLLI paper (Kanovich, K., Scedrov 2021) we give proof-theoretic analysis of Morrill's systems and prove their undecidability.
- However, the argument involves using bracket modalities under ! to break the bracket discipline.
- In real applications this does not happen, and standard proof search algorithms solve the derivability problem.
- Theoretically, this is supported by decidability results for certain conditions on bracket modalities under ! (Kanovich, Stepan G. Kuznetsov, K., Scedrov 2021), which make ! "harmless" (NP for Lambek, PSPACE for MALC).

Subexponentials with Contraction: Brackets

- In our JoLLI paper (Kanovich, K., Scedrov 2021) we give proof-theoretic analysis of Morrill's systems and prove their undecidability.
- However, the argument involves using bracket modalities under ! to break the bracket discipline.
- In real applications this does not happen, and standard proof search algorithms solve the derivability problem.
- Theoretically, this is supported by decidability results for certain conditions on bracket modalities under ! (Kanovich, Stepan G. Kuznetsov, K., Scedrov 2021), which make ! "harmless" (NP for Lambek, PSPACE for MALC).
    - Notice that now no restriction is imposed on the depth of formulae under !.

Kleene Star

- We finish our survey by discussion one of the most interesting operations, namely *iteration* or *Kleene star*, A^*.

Kleene Star

- We finish our survey by discussion one of the most interesting operations, namely *iteration* or *Kleene star,* A^*.
- In Morrill's systems, it is denoted by $?A$ and called "existential exponential."

Kleene Star

- We finish our survey by discussion one of the most interesting operations, namely *iteration* or *Kleene star,* A^*.
- In Morrill's systems, it is denoted by $?A$ and called "existential exponential."
- However, it obeys standard rules for Kleene star, one of which is the omega-rule:

$$\frac{\left(\Gamma, A^n, \Delta \to C\right)_{n=0}^{\infty}}{\Gamma, A^*, \Delta \to C}\ *L_\omega \qquad \frac{}{\to A^*}\ *R_0 \qquad \frac{\Pi \to A \quad \Delta \to A^*}{\Pi, \Delta \to A^*}\ *R$$

Kleene Star

- With $*L_\omega$, the Lambek calculus with Kleene star are undecidable and Π^0_1-complete (Buszkowski, Palka 2007; K. 2020).

Kleene Star

- With $*L_\omega$, the Lambek calculus with Kleene star are undecidable and Π^0_1-complete (Buszkowski, Palka 2007; K. 2020).
- This means that *disproving* a sequent is an enumerable task (due to the finite model property), but proving requires infinitary mechanisms.

Kleene Star

- With $*L_\omega$, the Lambek calculus with Kleene star are undecidable and Π^0_1-complete (Buszkowski, Palka 2007; K. 2020).
- This means that *disproving* a sequent is an enumerable task (due to the finite model property), but proving requires infinitary mechanisms.
- Fortunately, Morrill uses his "existential exponential" only in positive positions, which do not involve $*L_\omega$.

Kleene Star

- With $*L_\omega$, the Lambek calculus with Kleene star are undecidable and Π^0_1-complete (Buszkowski, Palka 2007; K. 2020).
- This means that *disproving* a sequent is an enumerable task (due to the finite model property), but proving requires infinitary mechanisms.
- Fortunately, Morrill uses his "existential exponential" only in positive positions, which do not involve $*L_\omega$.
- Namely, "and" in iterated coordination situations like "John, Bill, Mary and Suzy" receives the type $(?N \setminus N) \,/\, N$

Kleene Star

- For logical completeness, however, we need some sort of a left rule for Kleene star.

Kleene Star

- For logical completeness, however, we need some sort of a left rule for Kleene star.
- One option could be an inductive-style axiomatisation (like in action logic, Pratt 1991):

$$\frac{\to B \quad A, B \to B}{A^* \to B}\ {*L} \qquad \frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ Cut$$

Kleene Star

- For logical completeness, however, we need some sort of a left rule for Kleene star.
- One option could be an inductive-style axiomatisation (like in action logic, Pratt 1991):

$$\frac{\to B \quad A, B \to B}{A^* \to B}\ {*L} \qquad \frac{\Pi \to A \quad \Gamma, B, \Delta \to C}{\Gamma, \Pi, \Delta \to C}\ Cut$$

- This again leads an undecidable system, now it is Σ^0_1-complete (K. 2019 — solving a problem raised by Kozen in 1994).

Kleene Star and Subexponential

- We conclude by considering an extension of the Lambek calculus with both * and !.

Kleene Star and Subexponential

- We conclude by considering an extension of the Lambek calculus with both * and !.
- Formally speaking, Morrill's system is such a system.

Kleene Star and Subexponential

- We conclude by considering an extension of the Lambek calculus with both * and !.
- Formally speaking, Morrill's system is such a system.
- In the presence of $*L_\omega$ and $!NC_{1,2}$, this system's complexity is as high as Π^1_1 (K., Speranski 2021).

Kleene Star and Subexponential

- We conclude by considering an extension of the Lambek calculus with both * and !.
- Formally speaking, Morrill's system is such a system.
- In the presence of $*L_\omega$ and $!NC_{1,2}$, this system's complexity is as high as Π^1_1 (K., Speranski 2021).
- Namely, they are capable of encoding well-foundedness of recursively defined infinite graphs (Kozen 2002).

Kleene Star and Subexponential

- We conclude by considering an extension of the Lambek calculus with both * and !.
- Formally speaking, Morrill's system is such a system.
- In the presence of $*L_\omega$ and $!NC_{1,2}$, this system's complexity is as high as Π^1_1 (K., Speranski 2021).
- Namely, they are capable of encoding well-foundedness of recursively defined infinite graphs (Kozen 2002).
- There are intermediate fragments which fall into the hyperarithmetical hierarchy, but these questions are definitely beyond any reasonable linguistic applications.

Thank you!