Non-commutative linear logic with sub-context-free complexity

Yusaku Nishimiya

RIKEN Center for Advanced Intelligence Project (AIP)
&
B4 at the University of Illinois

joint work with Masaya Taniguchi

9 September @ TbiLLC 2025

Summary

Result:

Linear logic (Lambek calculus) analogue of the Greibach Normal Form of context-free grammar.

Merit:

Direct translation between logical inference & string generation rules.

Structure of the talk

1. Motivations
   - Complexity of linear logic
   - Structure of formal language
2. Sketch of proof
3. Ideas for extension/applications
   - Non-Chomsky hierarchy languages
   - Exponentials
   - Geometric group theory

Linear logic

Linear logic [Girard and Lafont, 1987]

Linear logic is a logic with *restricted contraction and weakening*.

$$\frac{X, X \to A}{X \to A}\ \text{Contraction} \qquad \frac{X \to A}{X, Y \to A}\ \text{Weakening}$$

Intuitively, linear logic prohibits *freely*

- 'throwing away' an existing formula
- 'introducing' a new formula

thus more resource conscious.

Linear logic (intuitionistic multiplicative exponential)

Axiom

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

Structural rules

$$\frac{\Gamma \to A \qquad \Delta, A, \Theta \to B}{\Delta, \Gamma, \Theta \to B}\ \mathrm{Cut} \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ \mathrm{Exchange}$$

$$\frac{\Gamma, !A, !A, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \mathrm{Ctr} \qquad \frac{\Gamma, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \mathrm{Weak}$$

$$\frac{\Gamma, A, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \mathrm{Der} \qquad \frac{!\Gamma \to A}{!\Gamma \to !A}\ \mathrm{Prom}$$

Inference rules

$$\frac{\Gamma \to A \qquad \Delta, B, \Theta \to C}{\Delta, \Gamma, A \multimap B, \Delta \to C}\ \mathrm{L}\multimap \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \otimes B, \Delta \to C}\ \mathrm{L}\otimes$$

$$\frac{A, \Gamma \to B}{\Gamma \to A \multimap B}\ \mathrm{R}\multimap \qquad \frac{\Gamma \to A \qquad \Delta \to B}{\Gamma, \Delta \to A \otimes B}\ \mathrm{R}\otimes$$

Computational power of linear logic

Known decidability and complexity results

- Full (propositional) linear logic is undecidable. [Lincoln et al., 1992]
- Multiplicative-additive linear logic is PSPACE-complete. [Lincoln et al., 1992]
- Multiplicative linear logic (MLL) is NP-complete. [Kanovich, 1991]

No complexity difference between full and intuitionistic fragments for the above. [Lincoln, 1995][1]

Open problem:
Decidability/complexity of multiplicative exponential linear logic (MELL) is unknown.

[1] First-Order Linear logic is also undecidable. [Girard and Lafont, 1987].
First-Order MALL is NEXPTIME-hard. [Lincoln and Scedrov, 1994]

Research motivation #1

Given that...

- (Propositional) Linear logic is undecidable.[Lincoln et al., 1992]
- Multiplicative-Additive Linear logic is PSPACE-complete. [Lincoln et al., 1992]
- Multiplicative Linear logic (MLL) is NP-complete. [Kanovich, 1991]

What linear logic fragments correspond to lower complexity classes?

Weaker logics for 'simpler' computation?

The Chomsky Hierarchy

Formal Language	Automaton
Recursively enumerable	Turing machine
Context-sensitive	Linear bounded automaton (PSPACE-complete)
Context-free	Nondeterministic pushdown automaton
Linear	One-turn pushdown automaton
Regular (semi-linear)	Finite automaton (deterministic/nondeterministic)

$\rightarrow$ non-commutative fragments

in which $\dfrac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}$ Exchange does not hold.

Rules in the Lambek calculus: intuitionistic non-commutative linear logic

The Lambek calculus L [Lambek, 1958] consists of

$$\frac{}{A \to A}\ \text{Axiom} \qquad \frac{\Gamma, X, \Theta \to B; \qquad \Delta \to X}{\Gamma, \Delta, \Theta \to B}\ \text{Cut}$$

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

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

for any $A, B, C \in Tp$, $\Gamma, \Delta, \Theta \in Tp^*$ (finite sequences of types).

Lambek, 1958 *The mathematics of sentence structure*.

Type-logical grammar

Type-logical grammar

A type-logical grammar $\mathcal{G}$ consists of

1. Tp: set of types, recursively generated from primitive types and connectives
2. **L**: a sequent calculus and *Axioms*
3. $f : \Sigma \to \mathcal{P}(Tp)$: an assignment function, extendable to strings
   For $w = a_1 \cdots a_n$, $f^*(w) = \{\Gamma = T_1, ..., T_n | T_k \in f(a_k) \text{ for all } k \in [1, n]\} \subseteq Tp^*$
4. $S_\mathcal{G} \in Tp$: a *distinguished type*

To 'recognise' language $\mathcal{L} \subseteq \Sigma^*$;

$$w \in \mathcal{L} \text{ iff } \exists \Gamma \in f^*(w) \text{ s.t. } \Gamma \to S_\mathcal{G} \textit{ is derivable in } \mathbf{L}.$$

where w is some string and $f^*(w)$ is the set of sequences of type assigned to w which includes Γ.

Lambek calculus / grammar complexity

- 1958: Lambek calculus [Lambek, 1958]
- 63: Chomsky conjectures: Lambek *grammar* = CFG [Chomsky, 1963]
- 87: Linear logic [Girard and Lafont, 1987]
- 90: Lambek calculus as fragment of LL [Abrusci, 1990]
- 93/97: Lambek *grammar* = CFG, proved [Pentus, 1993, Pentus, 1997]
- 2006: Lambek *calculus* is NP-complete [Pentus, 2006]

Formal language

Review: formal languages

Let
Σ be a finite set of symbols
Σ^* the set of all finite-length strings of symbols from Σ
(i.e. the free monoid generated by Σ).

A *formal language* on Σ is any subset $\mathcal{L} \subseteq \Sigma^*$.

A *formal grammar* is an effective procedure for the formal language membership decision.

Examples of formal languages

Let $\Sigma = \{a, b, c\}$

- $\{ab^*c\}$: strings in which an 'a' is followed by a finite number of 'b's, then by a 'c'.
- $\{a^n b^n : n \geq 1\}$: strings in which n 'a's are followed by n 'b's.

Let $\Sigma = \{(,)\}$

- Dyck language: strings of 'balanced' parentheses. e.g. (()(()))

Formal grammar

A formal grammar $\mathbf{G}$ consists of

1. N: non-terminal symbols
2. $S_{\mathbf{G}} \in N$: the start symbol
3. Σ: terminal symbols ($N \cap \Sigma = \emptyset$)
4. P: production rules, to rewrite a non-terminal symbol to some string of terminal and non-terminal symbols i.e. $P \subseteq (\Sigma \cup N)^+ \to (\Sigma \cup N)^*$

Language $\mathcal{L} \subseteq \Sigma^*$ is defined by;

$$w \in \mathcal{L} \text{ iff } w \text{ is } \textit{producible from } S_{\mathbf{G}}$$

where w is some string.

Example (regular grammar & nondeterministic finite automaton)

$\mathcal{L} = (ab \cup b)^* ba \subseteq \{a, b\}^*$

Grammar

$\mathbf{G} = (\Sigma, N, P, S_\mathbf{G})$

$\Sigma = \{a, b\}$

$N = \{S_\mathbf{G}, A, B, C\}$

P contains;

$S_\mathbf{G} \to aA|bB$

$A \to bS_\mathbf{G}$

$B \to bB|aC|a$

$C \to bS_\mathbf{G}$

Automaton

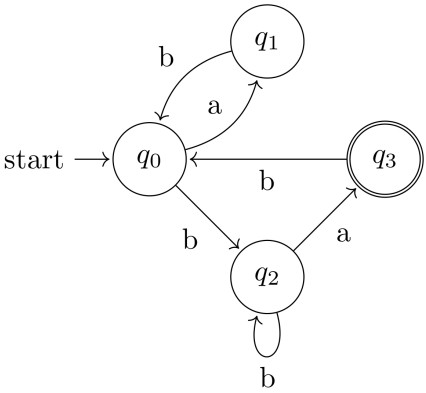

Figure modified and adapted from [Berry and Sethi, 1986].

Examples of formal languages *revisited*

Let $\Sigma = \{a, b, c\}$

- $\{ab^*c\}$ is **regular**.
- $\{a^n b^n : n \geq 1\}$ is **linear but non-regular**.

Let $\Sigma = \{(,)\}$

- Dyck language: strings of 'balanced' parentheses. e.g. (()(())) is **context-free but non-linear.**

'Varied sophisticatedness' in *the ability to count* required to parse the language.

Research motivation #2

To study structural properties of formal languages
via translation to logic.

$\rightarrow$ a step towards exploiting rich literature on semantics of linear logic for applications in formal language & complexity.

Main Result

Let $Tp_n(\heartsuit)$ be the set of types with at most n connectives in $\heartsuit$.

Theorem

Lambek grammar with Cut and ...	assigning...	is equivalent to...
$(/ \to)$	$Tp_1(/)$	Regular
$(/ \to)$ and $(\backslash \to)$	$Tp_1(/, \backslash)$	Linear
$(/ \to)$	$Tp_n(/)$	Context-free
		... languages without the empty string.

$\therefore$ A single introduction rule sufficient to parse any CFLs.

Lambek grammar with Cut and ...	assigning...	is equivalent to...
$(/ \to)$	$Tp_1(/)$	Regular
$(/ \to)$ and $(\backslash \to)$	$Tp_1(/, \backslash)$	Linear
$(/ \to)$	$Tp_n(/)$	Context-free ... languages.

Steps

1. Cut-elimination
2. Structural induction on the length of Cut-free proof

The $(/ \to)$ rule. $$\frac{\Gamma \to A; \qquad \Delta, B, \Theta \to C}{\Delta, (B/A), \Gamma, \Theta \to C} (/ \to)$$

Cut-elimination [Lambek, 1958]

$$\frac{\Gamma, \alpha, \Theta \to \beta; \qquad \Delta \to \alpha}{\Gamma, \Delta, \Theta \to \beta}\ \mathsf{Cut}$$

$$\Downarrow$$

$$\frac{\dfrac{\Gamma, \alpha, \Theta \to \beta; \qquad \Delta' \to \alpha}{\Gamma, \Delta', \Theta \to \beta;}\ \mathsf{Cut'} \qquad \Xi \to \alpha'}{\Gamma, \Delta, \Theta \to \beta}\ (/ \to)$$

Δ' contains one less '/' than Δ.

Recalling formal grammar

G is ...	if every $p \in P$ is in the form(s)
Regular (semi-linear)	$A \to aB$
Linear	$A \to aB$ or $A \to Ba$
Context-free	$A \to \beta \quad \beta \in (N \cup \Sigma)^*$

Theorem ([Greibach, 1965])

Every ϵ-free CFL can be generated by a CFG in Greibach normal form:

$$A \to aB_1B_2 \cdots B_n$$

For ref:
Context-sensitive if $\alpha A \beta \to \alpha \gamma \beta$ with $A \in N$, $\alpha, \beta \in (N \cup \Sigma \setminus \{S\})^*$ and $\gamma \in (N \cup \Sigma \setminus \{S\})^+$

Structural lemma to show language equivalence (for CFG)

Lemma

Let Γ be a non-empty sequence of types in $Tp(/)$.

$$(/ \to) \vdash \Gamma \to T$$
iff
$$\Gamma = (\cdots((T/\beta_n)/\beta_{n-1})/\cdots)/\beta_1, \Delta_1, ..., \Delta_n$$
and
$$\mathbf{L}(/ \to) \vdash \Delta_k \to \beta_k \text{ for all } 1 \leq k \leq n$$

For all $T \in Tp$.

Illustration

$$\frac{\Delta_1 \to \alpha_1; \qquad (\cdots((S_{\mathbf{G}}/\alpha_n)/\alpha_{n-1})/\cdots)/\alpha_2, \Delta_2, ..., \Delta_n \to S_{\mathcal{G}}}{(\cdots((S_{\mathbf{G}}/\alpha_n)/\alpha_{n-1})/\cdots)/\alpha_1, \Delta_1, ..., \Delta_n \to S_{\mathcal{G}}}\ (/ \to)$$

Sketch of proof (main results, CFG)

Finite length Cut-free proofs (only $(/ \rightarrow)$) exist

$$\Downarrow$$

Construct corresponding production rule;

$$\alpha \rightarrow a\beta_1\beta_2 \cdots \beta_n \in P \text{ if } (\cdots((\alpha/\beta_n)/\beta_{n-1})/\cdots)/\beta_1 \in f(a)$$

$$\alpha \rightarrow a \in P \text{ if } \alpha \in f(a)$$

... recursively and vice versa following the structural lemma.

Likewise for linear & regular languages;

1. $A/B \in f(a)$ iff $A \rightarrow aB \in P$
2. $B\backslash A \in f(a)$ iff $A \rightarrow Ba \in P$, and
3. $A \in f(a)$ iff $A \rightarrow a \in P$.

Innovation/contribution

A (more) direct translation: formal grammar $\leftrightarrow$ inference rule

- Analog of the Greibach normal form of CFG in Lambek calculus/linear logic
- Characterisation of propositional formula-size's effect on complexity

Future

Topic 1

Lambek grammar of non-Chomsky hierarchy languages
&
Generalisation of $(/ \to)$ rules

LL-parser

Definition

Let $k \in \mathbb{N}$.
A language generated by a CFG $\mathbf{G} = (\Sigma, N, P, S_{\mathbf{G}})$ is $LL(k)$ iff for any $A \in N$; $w, x, y \in \Sigma^*$; $\beta, \beta', \gamma \in (\Sigma \cup N)^*$; and any two derivations;

$$S \Rightarrow^* wA\gamma \Rightarrow w\beta\gamma \Rightarrow^* wx$$

$$S \Rightarrow^* wA\gamma \Rightarrow w\beta'\gamma \Rightarrow^* wy$$

if x and y share the first k symbols, we necessarily have $\beta = \beta'$.

Intuition: rewriting of A 'becomes' deterministic if given k symbols *look-ahead* (from after w).

‘n-skip’ ($/ \to$) rule [Taniguchi, 2024]

Recall the left /-introduction ‘($/ \to$)’ rule;

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

Generalised ‘n-skip ($/ \to$)’ rule

$$\frac{\Gamma \to A; \qquad \Delta, B, D_1, ..., D_n, \Theta \to C}{\Delta, B/A, \underline{D_1, ..., D_n, \mathbf{\Gamma}}, \Theta \to C} (/ \to)_n \qquad n \in \mathbb{N}$$

Idea: long-range-dependency inspired by natural language syntax.

Conjecture: CFG $\subsetneq$ $(/ \to)_n$-grammar $\subsetneq$ CSG^2

More generally: Which linear logic fragment corerspond to...

- star-free language
- indexed language

... ?

Topic 2

Formal language theoretic analysis of (sub)exponentials & proof differentiation

Subexponentials

Recall structural rules in LL

$$\frac{\Gamma, !A, !A, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \text{Ctr} \qquad \frac{\Gamma, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \text{Weak}$$

Subexponential: separating *linear* into affine and *relevant*

- relevant: at least once (no cont)
- affine: at most once (no weak)

Complexity results [Kanovich et al., 2019]:

- *Relevant*-subexponential Lambek calculus with...
  - multiplicative & additive: PSPACE
  - multiplicative: NP
- *Affine*-subexponential Lambek calculus: undecidable.

→ Relevant Lambek grammar's expressivity?

‘Differential’ linear logic: an extension of linear logic with

costructural rules

Applications

- Logic for (functional) analysis
- Program synthesis

Rules of (intuitionistic) differential linear logic (DILL)

$$\frac{\Gamma, A, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \text{der} \qquad \frac{\Gamma, !A, \Delta \to B}{\Gamma, A, \Delta \to B}\ \text{coder}$$

$$\frac{\Gamma, !A, !A, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \text{ctr} \qquad \frac{\Gamma, !A, \Delta \to B}{\Gamma, !A, !A, \Delta \to B}\ \textit{coctr}$$

$$\frac{\Gamma, \Delta \to B}{\Gamma, !A, \Delta \to B}\ \text{weak} \qquad \frac{\Gamma, !A, \Delta \to B}{\Gamma, \Delta \to B}\ \text{coweak}$$

$$\frac{\Gamma, A, B, \Delta \to C}{\Gamma, B, A, \Delta \to C}\ \text{Exchange}$$

$$\frac{}{A \to A}\ \text{Ax} \qquad \frac{\Gamma \to A \qquad \Delta, A, \Theta \to B}{\Delta, \Gamma, \Theta \to B}\ \text{cut} \qquad \frac{!\Gamma \to A}{!\Gamma \to !A}\ \text{prom}$$

$$\frac{\Gamma \to A \qquad \Delta, B, \Theta \to C}{\Delta, \Gamma, A \multimap B, \Delta \to C}\ \text{L}\multimap \qquad \frac{\Gamma, A, B, \Delta \to C}{\Gamma, A \otimes B, \Delta \to C}\ \text{L}\otimes$$

$$\frac{A, \Gamma \to B}{\Gamma \to A \multimap B}\ \text{R}\multimap \qquad \frac{\Gamma \to A \qquad \Delta \to B}{\Gamma, \Delta \to A \otimes B}\ \text{R}\otimes$$

$$\text{DILL} = \text{IMELL} + \text{coder} + \text{coctr} + \text{coweak}$$

‘Differentiation’ of proof

A procedure to convert a non-linear proof to linear proof. [Clift, 2017]

Let π be a proof of $!A \to B$. The *derivative* of π is defined as the following proof.

$$\dfrac{\dfrac{\dfrac{\begin{array}{c}\pi\\ \vdots \\ !A \to B\end{array}}{!A, !A \to B}\,\mathrm{coctr}}{!A, A \to B}\,\mathrm{coder}}{A \to B}\,\mathrm{coweak}$$

Question:

- Is there a formal language-theoretic phenomenon/construction analogous to proof differentiation (in non-commutative DILL)?

* 'provability' not interesting for DILL as

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{A \to A}\,\mathrm{Ax}}{!\Gamma, A \to A}\,\mathrm{weak}^n}{\Gamma, A \to A}\,\mathrm{coder}^n}{\Gamma, !A \to A}\,\mathrm{der}}{\Gamma \to A}\,\mathrm{coweak}$$

... any sequent is provable (n: length of Γ).

Topic 3

Interaction with geometric group theory

Group presentation & word problem

Let $X = \{x_1, x_2, ...x_n\}$ be a set.
Define the formal inverses as $X^{-1} = \{x_1^{-1}, x_2^{-1}, ..., x_n^{-1}\}$, $X \cap X^{-1} = \emptyset$.
Take $\Sigma = X \cup X^{-1}$ and its free monoid Σ^* (i.e. free group of X).
Consider a finite $R \subset \Sigma^*$ called *relators*[3], which induces an equivalence relation;

$$u \sim_R v$$

if v can be obtained from u by a finite sequence of insertions or deletions of $r \in R$.
Let G be the group formed by the set of such equivalence classes.

Definition

The **word problem** $W(G)$, a formal language (over $\Sigma = X \cup X^{-1}$), of a finitely presented group $G = \langle X|R\rangle$ is the set;

$$W(G) = \{w \in \Sigma^* | w \sim_R 1\} \subseteq \Sigma^*$$

where 1 is the group identity (empty string).

[3] $\forall x \in X, \{xx^{-1}, x^{-1}x\} \subset R$, 'trivial relators'

EDT0L $\subsetneq$ ET0L languages

Extended alphabet (Deterministic) Table 0-interaction Lindenmayer

Definition

[Rozenberg, 1973, Bishop et al., 2025]
Let V (*variables*) and Σ (*terminals*) be finite sets s.t. $\Sigma \subseteq V$.
Fix $S \in V$, an *start symbol*.
The grammar $G = (V, P, S, \Sigma)$ is

an *EDT0L*-system	if $p \in \mathrm{End}(V^*)$ s.t. $\sigma \cdot p = \sigma$ for all $\sigma \in \Sigma$
an *ET0L*-system	if $p \in \mathcal{P}(V \times V^*)$
	... for all $p \in P$

Relation with the Chomsky Hierarchy

'$\to$' known set containement

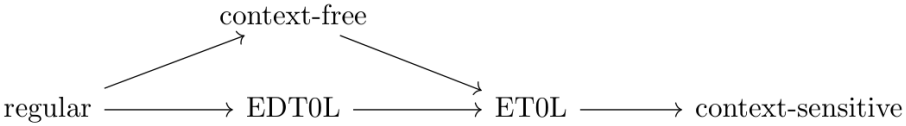

Figure adapted from [Bishop et al., 2025].

Open problems [Ciobanu et al., 2018] (and related result)

Let $\pi_{WP}(X)$: class of groups with word problems in the family of language X.

$$\pi_{WP}(\text{EDT0L}) \stackrel{?}{=} \pi_{WP}(\text{regular})$$

($\pi_{WP}(\text{regular})$= finite groups [Anisimov, 1971])

$$\pi_{WP}(\text{ET0L}) \stackrel{?}{=} \pi_{WP}(\text{context-free})$$

($\pi_{WP}(\text{context-free})$ = virtually free groups [Muller and Schupp, 1983])

Structural similarity between sequential & parallel rewriting?

Proof theoretic methods for group classifications?

Sequents/formulae/proof in logic $\overset{?}{\leftrightarrow}$ group structure/actions

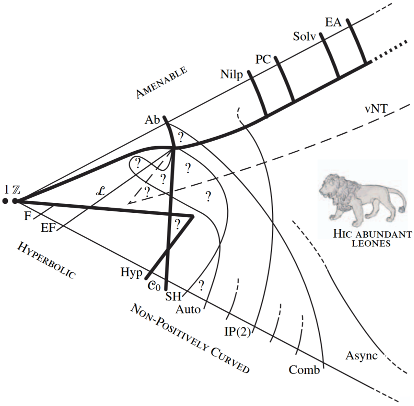

Figure: Bridson's universe of finitely presented groups, figure adapted from [Bridson, 2006]

Conclusion

A direct & straightforward translation between logic and formal language established.

Potential applications:

- More fine-grained logical characterisations of various formal languages
- Better understanding of exponential connective's '!' computational behaviour
- Proof theoretic & logical methods to study groups

References I

Abrusci, V. M. (1990).
A comparison between lambek syntactic calculus and intuitionistic linear propositional logic.
Mathematical Logic Quarterly, 36(1).

Anisimov, A. V. (1971).
Group languages.
Cybernetics, 7(4):594–601.

Berry, G. and Sethi, R. (1986).
From regular expressions to deterministic automata.
Theoretical computer science, 48:117–126.

Bishop, A., Elder, M., Evetts, A., Gallot, P., and Levine, A. (2025).
On groups with edt0l word problem.
arXiv preprint arXiv:2505.20057.

References II


Bridson, M. R. (2006).
Non-positive curvature and complexity for finitely presented groups.
In *International Congress of Mathematicians*, volume 2, pages 961–987.
European Math. Soc.


Chomsky, N. (1963).
Formal properties of grammars.
Handbook of Math. Psychology, 2:328–418.


Ciobanu, L., Elder, M., and Ferov, M. (2018).
Applications of l systems to group theory.
International Journal of Algebra and Computation, 28(02):309–329.


Clift, J. (2017).
Turing machines and differential linear logic.

References III

Girard, J.-Y. and Lafont, Y. (1987).
Linear logic and lazy computation.
In *International Joint Conference on Theory and Practice of Software Development*, pages 52–66. Springer.

Greibach, S. A. (1965).
A new normal-form theorem for context-free phrase structure grammars.
Journal of the ACM, 12(1):42–52.

Kanovich, M., Kuznetsov, S., Nigam, V., and Scedrov, A. (2019).
Subexponentials in non-commutative linear logic.
Mathematical Structures in Computer Science, 29(8):1217–1249.

Kanovich, M. I. (1991).
The multiplicative fragment of linear logic is np-complete.

References IV


Lambek, J. (1958).
The mathematics of sentence structure.
The American Mathematical Monthly, 65(3):154–170.


Lincoln, P. and Scedrov, A. (1994).
First-order linear logic without modalities is nexptime-hard.
Theoretical Computer Science, 135(1):139–153.


Lincoln, P. D. (1995).
Deciding provability of linear logic formulas.
London Mathematical Society Lecture Note Series, pages 109–122.


Lincoln, P. D., Mitchell, J., Scedrov, A., and Shankar, N. (1992).
Decision problems for propositional linear logic.
Annals of pure and applied logic, 56(1-3):239–311.

References V


Muller, D. E. and Schupp, P. E. (1983).
Groups, the theory of ends, and context-free languages.
Journal of Computer and system sciences, 26(3):295–310.


Pentus, M. (1993).
Lambek grammars are context free.
In *[1993] Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 429–433. IEEE.


Pentus, M. (1997).
Product-free lambek calculus and context-free grammars.
The Journal of Symbolic Logic, 62(2):648–660.


Pentus, M. (2006).
Lambek calculus is np-complete.
Theoretical Computer Science, 357(1-3):186–201.

References VI


Rozenberg, G. (1973).
Extension of tabled 0 l-systems and languages.
International Journal of Computer & Information Sciences, 2(4):311–336.


Taniguchi, M. (2024).
Substructural logics weaker than commutative lambek calculus.
In *Book of abstracts of TACL 2024: Topol- ogy, Algebra, and Categories in Logic*, pages 250–251.

Thank you :)

* I'm looking for an internship: Jan-Aug 2026 *