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

Yusaku Nishimiya[1,2] Masaya Taniguchi[2]

[1]University of Illinois Springfield, IL, USA
[2]RIKEN Center for Advanced Intelligence Project (AIP), Tokyo, Japan

Keywords: Formal language, context-free language, substructural logic, linear logic, Lambek calculus, intuitionistic logic, descriptive complexity, type-logical grammar

Summary

The present work constructs a bridge between logic (sequent calculus) and computation (automata) by showing their equivalent formal language parsing power for lower classes of the Chomsky Hierarchy. Though analogous correspondences had long been known, our proof employs a significantly more straightforward and direct translation between sequent rewriting rules (logic) and string production rules (computation). We believe the result presented herein constitutes a first step toward a more extensive and richer characterisation of the interaction between computation and logic as well as a finer-grained complexity separation of various sequent calculi.

Introduction

$$\frac{\alpha, \alpha \to \beta}{\alpha \to \beta}\ \text{Contraction} \qquad \frac{\alpha \to \gamma}{\alpha, \beta \to \gamma}\ \text{Weakening} \qquad \text{restricted in } \mathbf{LL}.$$

$$\frac{\Gamma, \alpha, \beta, \Delta \to \gamma}{\Gamma, \beta, \alpha, \Delta \to \gamma}\ \text{Exchange} \qquad \frac{\Gamma, \alpha, \Theta \to \beta; \qquad \Delta \to \alpha}{\Gamma, \Delta, \Theta \to \beta}\ \text{Cut} \qquad \text{allowed in } \mathbf{LL}.$$

Proof theorists study substructural logics to understand the effect of admitting or eliminating *structural rules* on the properties of proof systems, usually presented as a sequent calculus. Of particular interest for computation is linear logic (**LL**) [Gir87] as it restricts Contraction and Weakening rules, making proofs more *resource-conscious* (and thus computation-relevant) than its classical counterpart.

The multiplicative-additive fragment of linear logic (**MALL**)[1] [LMSS92] in which each formula is used exactly once was shown to be PSPACE-complete [LMSS92] and further restriction by removal of additives, to multiplicative linear logic (**MLL**) makes the calculus NP-complete [Kan91].

Little is known about the computational complexity of proof systems based on even 'weaker' fragments of **LL**, except for the NP-completeness [Pen06] of the Lambek calculus (**L**), the intuitionistic, non-commutative, multiplicative fragment of **LL** with direction-sensitive implications. **L** was originally introduced as a proof system for formalising natural language syntax in Lambek's seminal paper [Lam58] but was later shown by Abrusci [Abr90] to be a fragment of **LL** without Exchange, leaving Cut as the sole allowed structural rule. Chomsky conjectured in 1963 [Cho63] the equivalence between type-logical grammars based on **L** and context-free grammars (CFG) and thus began the research into the expressivity of Lambek grammar. Pentus confirmed Chomsky's conjecture [Pen93] and further proved [Pen97] that removal of multiplicative connective does not change the expressivity such that the *product-free* Lambek grammar is also context-free. However, no fragments of **L** corresponding in expressivity to lower classes of formal grammars (equivalently automata) in the Chomsky hierarchy have been identified.

Here, we show that, intuitively, the restriction on the size and directionality of the logical formulae permitted in the proof system, more so than on inference rules, yields the variation in expressivity that corresponds exactly to the difference between context-free, linear context-free and regular grammar.

[1]Here, we focus exclusively on propositional logic without the *exponential* connectives. For a survey of computability and complexity for more general linear logic and its first-order extension, see [Lin95].

Preliminaries

The expressive power of a class of automata is defined by the formal languages it can parse (i.e. decide the set membership). Formal languages in turn are characterised by the nature of grammatical *production rules* required to generate all strings therein. Here, we consider the class of context-free grammars.

Definition 1. A context-free grammar (CFG) $\mathbf{G} = (N, \Sigma, S_{\mathbf{G}}, P)$ consists of sets N of *non-terminal symbols*, Σ of *terminal symbols*, a *start symbol* $S_{\mathbf{G}} \in N$ and $P \subset N \times (\Sigma \cup N)^+$, the set of production rules, where $(\Sigma \cup N)^+$ consists of finite non-empty sequences of terminal and non-terminal symbols.

Any rule $p \in P$ of the form $A \to aB$ (resp. $A \to Ba$) is said to be *right linear* (resp. *left linear*).

A CFG is a linear context-free grammar (LCFG) if all production rules are either right or left-linear.

A (right) regular grammar (REG) is a context-free grammar all of whose production rules are right linear. Likewise and equivalently for the left regular grammar.

We identify the fragments of Lambek calculus $\mathbf{L}$ with equivalent expressivity to context-free grammar subclasses by constructing a suitable type-logical grammar, 'Lambek grammar'. To illustrate, we present the Axiom, Cut-rule and two inference rules: $(/ \to), (\backslash \to)$ considered herein.[2] Let $\mathbf{L}(/ \to)$ be the fragment with Axiom, Cut and $(/ \to)$. We likewise define $\mathbf{L}(/ \to, \backslash \to)$. We let $\Gamma, \Delta, \Theta...$ be sequences of propositions or *types*, which we denote by $\alpha, \beta, \gamma...$ and are elements of Tp the set of types.

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

$$\frac{\Gamma \to \alpha; \qquad \Delta, \beta, \Theta \to \gamma}{\Delta, (\beta/\alpha), \Gamma, \Theta \to \gamma}\ (/ \to) \qquad \frac{\Gamma \to \alpha; \qquad \Delta, \beta, \Theta \to \gamma}{\Delta, \Gamma, (\alpha\backslash\beta), \Theta \to \gamma}\ (\backslash \to)$$

Given a finite set $Pr = \{A, B, C, ...\}$ (of *primitive types*), we define $Tp(/, \backslash)$ as the smallest set such that i. $Pr \subset Tp$ and ii. for any $\alpha, \beta \in Tp$, $\alpha/\beta, \alpha\backslash\beta \in Tp$. We let Tp_n for any $n \in \mathbb{N}$ be the set of types in which the number of distinct occurrences of connective symbols $/, \backslash$, or the type's 'degree', is at most n. We now define the Lambek grammar.

Definition 2. A Lambek grammar $\mathcal{G}$ is a quadruple $(Pr, V, S_{\mathcal{G}}, f)$, with the set of primitive types Pr, the finite set of symbols or *alphabet* V, the *distinguished type* $S_{\mathcal{G}} \in Pr$ and the type assignment function $f : V \to \Omega^{Tp}$, where Ω^{Tp} is the powerset of Tp. f is naturally extended to strings; $f^+ : V^+ \to \Omega^{Tp^+}$ defined by $\forall w \in V^+$ s.t. $w = a_1...a_n$, $\quad f^+(w) = \{\Gamma \in Tp^+ | \Gamma = \alpha_1...\alpha_n \text{ s.t. } \forall k, \alpha_k \in f(a_k)\}$ where V^+ is the set of all finite non-empty strings of symbols in V and Tp^+ is the set of all finite non-empty sequences of types. The *language* $\mathcal{L}$ *recognised by* $\mathcal{G}$ is a subset of V^+, such that for any $w \in V^+$, $w \in \mathcal{L}$ iff $\exists \Gamma \in f^+(w)$ s.t. $\mathbf{L} \vdash \Gamma \to S_{\mathcal{G}}$ (*i.e.* any given string is in the language iff there is a sequence of types assigned to it which is *reducible* to $S_{\mathcal{G}}$ in $\mathbf{L}$). The grammar and language for fragments of $\mathbf{L}$ are analogously defined.

Main results

The construction of corresponding grammars relies on structural inductions on the sequent, which in turn requires the existence of a Cut-free proof for any provable sequents.

Theorem 1. The elimination of Cut from $\mathbf{L}(/ \to)$ does not change the set of provable formulae and likewise holds for $\mathbf{L}(/ \to, \backslash \to)$.

Sketch of proof. (*Gentzen's Theorem* in [Lam58]) To illustrate, we present the sequent replacement procedure for $\mathbf{L}(/ \to)$.

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

Assume that the premises of the Cut on the left are provable without Cut. Then, the last step in the derivation of $\Delta \to \alpha$ is the $(/ \to)$ as shown below.

$$\frac{\Delta' \to \alpha; \qquad \Xi \to \alpha'}{\Delta \to \alpha}\ (/ \to)$$

[2] Readers can consult [Bus10] for a clear presentation of the rest of the rules in Lambek calculus in Gentzen-style.

Readers can verify that the replacement of Cut by a 'smaller' Cut (Cut' on the right) is possible due to the assumed Cut-free provability of relevant premises, noting in particular that Δ' contains one less / connective than Δ. The procedure is analogous for $\mathbf{L}(/\,\rightarrow, \backslash\,\rightarrow)$. We now state our main theorem.

Theorem 2. The following three pairs of Lambek-fragment grammar and formal grammar (without the empty string) are of equivalent expressive power.

$$\begin{aligned} \mathbf{L}(/\,\rightarrow)\text{-grammar with } Tp(/) &\Leftrightarrow \text{CFG} \\ \mathbf{L}(/\,\rightarrow, \backslash\,\rightarrow)\text{-grammar with } Tp_1(/, \backslash) &\Leftrightarrow \text{LCFG} \\ \mathbf{L}(/\,\rightarrow)\text{-grammar with } Tp_1(/) &\Leftrightarrow \text{REG} \end{aligned}$$

Sketch of proof. The construction of a Lambek grammar given a CFG begins by identifying:

$$Pr = N,\ V = \Sigma \text{ and } S_{\mathcal{G}} = S_{\mathbf{G}} \text{ (hereafter } S).$$

The type assignment $f : V \rightarrow \Omega^{Tp(/)}$ is defined as follows, assuming Greibach normal form [Gre65]. For any $a \in V$, $f(a) \subseteq Tp(/)$ is the smallest set such that $A \in f(a)$ if $A \rightarrow a \in P$ and $(\cdots((A/B_n)/B_{n-1})/\cdots)/B_1 \in f(a)$ if $A \rightarrow aB_1...B_{n-1}B_n \in P$.

The converse direction involves the same identities of terminals/alphabet and start symbol⇔distinguished type but the treatment of assigned types is more subtle: we let $N = \overline{f}(V)$, the set of all sub-types[3] of all types assigned to symbols in V by $f : V \rightarrow \Omega^{Tp(/)}$. The corresponding production rules are; $\alpha \rightarrow a\beta_1\beta_2\cdots\beta_n \in P$ if $(\cdots((\alpha/\beta_n)/\beta_{n-1})/\cdots)/\beta_1 \in f(a)$ and $\alpha \rightarrow a \in P$ if $\alpha \in f(a)$.

To show the language equivalence, we use the following structural lemma that follows the form of sequents constructible by applying the $(/\,\rightarrow)$ rule.

Lemma 3. ('Reducibility condition') Let Γ be a non-empty sequence of types in $Tp(/)$. $\mathbf{L}(/\,\rightarrow) \vdash \Gamma \rightarrow S$ iff $\Gamma = \alpha, \Delta_1, ..., \Delta_n$ where

1. α is of the form $(\cdots((S/\beta_n)/\beta_{n-1})/\cdots)/\beta_1$ where $\beta_1, .., \beta_n \in Tp(/)$ and
2. for all $1 \leq k \leq n$, $\mathbf{L}(/\,\rightarrow) \vdash \Delta_k \rightarrow \beta_k$.

Moreover, likewise holds for reducibility to any other types besides S.

Consider a production rule of the form $S \rightarrow a\beta_1\beta_2\cdots\beta_n$ with S on the left-hand side. The application of Lemma 3 to the corresponding type assignment $(\cdots((S/\beta_n)/\beta_{n-1})/\cdots)/\beta_1 \in f(a)$ and recursively to the type assignments that correspond to production rules with $\beta_1, \beta_2, ...$ or β_n on the left-hand side and so on, implies the language equivalence by induction on the length of Cut-free derivation.

The results for LCFG and REG follow the analogous construction of corresponding grammars with appropriate restrictions on production rules and types.

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 $A \rightarrow a \in P$.

Discussion & future

The result presented here shows the Lambek grammar's sensitivity to the restrictions on the type degree and directionality. The unidirectional $\mathbf{L}(/\,\rightarrow)$-grammar is the simplest Lambek grammar with context-free complexity and may be considered as the proof-theoretic analogue of the Greibach normal form. We further note that the type degree restriction to one naturally corresponds to the (bi)linearity of production rules.

Though the language equivalences themselves are not particularly surprising, we believe the directness of the production rule⇔inference rule translation equips us with an intuition to extend the result to related and more general classes of interesting problems. For example, an intimate understanding of the computational behaviour of sequent calculus at inference rule resolution would, we believe, encourage more interaction between proof complexity and formal languages. For more concrete developments, promising directions include i. the formal language-theoretic characterisation of slightly modified Lambek calculus and linear logic inference rules, ii. further work in fine-grained descriptive complexity of other linear logic fragments and iii. identification of Lambek grammars that correspond in complexity to star-free languages, mildly context-sensitive languages, Lindenmayer-systems[LR72], etc.

[3] Considering a type α as a string, its substring α' is a subtype of α if it is a type.

References

[Abr90] V Michele Abrusci. A comparison between lambek syntactic calculus and intuitionistic linear propositional logic. *Mathematical Logic Quarterly*, 36(1):11–15, 1990.

[Bus10] Wojciech Buszkowski. Lambeck calculus and substructural logics. *Linguistic Analysis*, 36(1):15–48, 2010.

[Cho63] Noam Chomsky. Formal properties of grammars. *Handbook of Math. Psychology*, 2:328–418, 1963.

[Gir87] Jean-Yves Girard. Linear logic. *Theoretical Computer Science*, 50(1):1–102, 1987.

[Gre65] Sheila A Greibach. A new normal-form theorem for context-free phrase structure grammars. *Journal of the ACM*, 12(1):42–52, 1965.

[Kan91] Max I Kanovich. The multiplicative fragment of linear logic is np-complete. 1991. (No full text available.).

[Lam58] Joachim Lambek. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170, 1958.

[Lin95] Patrick D Lincoln. Deciding provability of linear logic formulas. *London Mathematical Society Lecture Note Series*, pages 109–122, 1995.

[LMSS92] Patrick D Lincoln, John Mitchell, Andre Scedrov, and Natarajan Shankar. Decision problems for propositional linear logic. *Annals of Pure and Applied Logic*, 56(1-3):239–311, 1992.

[LR72] Aristid Lindenmayer and Grzegorz Rozenberg. Developmental systems and languages. In *Proceedings of the fourth annual ACM symposium on theory of computing*, pages 214–221, 1972.

[Pen93] Mati Pentus. Lambek grammars are context free. In *Proceedings Eighth Annual IEEE Symposium on Logic in Computer Science*, pages 429–433. IEEE, 1993.

[Pen97] Mati Pentus. Product-free lambek calculus and context-free grammars. *The Journal of Symbolic Logic*, 62(2):648–660, 1997.

[Pen06] Mati Pentus. Lambek calculus is np-complete. *Theoretical Computer Science*, 357(1-3):186–201, 2006.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 24K16077. YN thanks the Neural Circuits and Computations Unit, RIKEN Center for Brain Science, for providing a friendly working space and colleagues from various centres of RIKEN for stimulating questions. The authors further acknowledge Naoki Negishi for participating in our weekly discussions and thank the anonymous reviewers for fruitful recommendations that led to the present form of this paper.