Research summary by Lê Thành Dũng (Tito) Nguyễn, September 2020

Broadly speaking, most of my research involves or is inspired by linear logic (LL) in some way. That said, it is never *only* about LL: I try to draw *connections with various other fields*. Of course, as an offspring of the celebrated Curry–Howard–Lambek correspondence, LL was connected to programming languages and to category theory from its birth. But I am referring to topics such as combinatorics or automata theory, whose relationship to proof theory hasn't been explored as much.

Past: proof nets through the lens of graph theory. Linear logic was first defined both by sequent calculus and by a new kind of proof system, called *proof nets*. The latter marks a radical departure from tradition: proofs are now *graph-like*, that are *not defined inductively* unlike usual tree-like proofs; correctness is checked via a *global* combinatorial criterion, instead of a mere local verification of the validity of the inference rules. The *sequentialization theorem* says that these proof nets are equivalent to the sequent calculus for LL; it is arguably the hardest result in Girard's seminal paper on LL (1987).

It seems like a natural idea to use *graph theory* in the study of proof nets, but this has been pursued mostly by a single person, namely Christian Retoré. He discovered in the 1990s that proof net correctness can be recast using the notion of *perfect matching* in a graph, and that the sequentialization theorem is equivalent to Kotzig's theorem on perfect matchings (1959; see also Szeider 2004). My first research paper (FSCD 2018 / LMCS 2020) seems to be the first attempt at exploiting the full potential of Retoré's idea: I show that it allows for "unreasonably effective" applications of the vast literature on graph algorithms and structural graph theory to proof nets for Multiplicative Linear Logic with Mix (MLL+Mix).

More recently, **Lutz Straßburger** and I have applied (in unpublished work) these graph-theoretic tools to Retoré's *pomset logic*, an extension of MLL+Mix with a self-dual non-commutative connective. What's interesting about this logic is that it is defined by a system of proof nets for which no well-behaved sequent calculus counterpart is known. The challenge of designing an inductively defined proof system accomodating self-dual non-commutativity led Alessio Guglielmi to introduce a logic called *system BV* in the late 1990s. It was the initial use case for the *deep inference* methodology which has become since then an important approach in proof theory (see `http://alessio.guglielmi.name/res/cos/`).

Thanks to an obscure paper on edge-colored directed graphs, Lutz and I discovered that provability in pomset logic is a strictly harder decision problem than in BV, assuming $\mathsf{NP} \neq \mathsf{coNP}$. This led us to find a concrete formula that is provable in pomset logic but not in BV, refuting the longstanding conjecture that the two are equivalent. (But please wait for a preprint to be uploaded before spreading the news!)

Present: implicit automata in typed λ-calculi. The field of *implicit computational complexity* seeks to characterize complexity classes by using constrained programming languages; it is practiced by several people at LIPN (Damiano, Paulin, Thomas, Virgile). The research programme that I am carrying out with **Pierre Pradic** explores a counterpart for automata and transducers.

Remarkably, automata naturally appear in the study of the *simply typed λ-calculus* (STLC). Consider the types $\texttt{Bool} = o \to o \to o$ of Church booleans and $\texttt{Str}[A] = (A \to A) \to (A \to A) \to A \to A$ of *Church-encoded binary strings* – A may be an arbitrary type, while o is a base type ($A, B ::= o \mid A \to B$). Hillebrand and Kanellakis (LICS 1996) showed that a language $L \subseteq \{0,1\}^*$ is definable in STLC by some $t : \texttt{Str}[A] \to \texttt{Bool}$ (A may be chosen depending on L) if and only if it is *regular*.

Pierre and I published at ICALP 2020 a characterization of *star-free languages* in a λ-calculus with *non-commutative types* – morally, such a type system forces functions to use their arguments in the order that they are given in. After this, we turned to string-to-string functions instead of languages, working with various *transducer* models whose theory is currently an active research topic; this led to several discussions around transducers with automata theorists (Mikołaj Bojańczyk and Amina Doumane). We have also come to realize that equivalence results about definability of languages or functions are mere epiphenomena: they come from deeper structural connections that can be expressed in the language of *category theory* (some relevant keywords are "Dialectica categories" and "Geometry of Interaction").

Future. Well, rumor has it that I'm supposed to write a PhD manuscript this year, so the rest can wait… But a lot of questions on implicit automata are still open (e.g. $\texttt{Str}[A] \to \texttt{Str}[o]$ in STLC). I am also interested in semantics of polymorphism (e.g. hypercoherences + normal functors) which play a role in my solution to an open problem in implicit complexity posed by Patrick Baillot (post-proceedings of DICE-FOPARA 2019) – in fact this result involves regular languages and led me to implicit automata.