Session Types, Solos, and the Computational Contents of Sequent Calculus Proofs

Nicolas Guenot

IT University of Copenhagen
ngue@itu.dk

The original form of the Curry-Howard correspondence established a connection between intuitionistic natural deduction and the simply-typed λ-calculus, but this has been extended to various other logical systems and other calculi. One problem with such extensions arise when considering proofs in the sequent calculus: the correspondence in such a setting is difficult to design, so that the proof systems considered are often quite constrained — for example using a stoup in the logic to restrict the shape of proofs — and term calculi are not necessarily well-behaved — for example when one interprets the left rule for implication as a floating `let` construct, which is problematic in the usual theory of λ-calculi. We discuss in this work the status of this problem in the light of some recent developments relating linear sequent proofs to session-typed processes.

1 Cut Elimination and Permutations

A quite striking difference between normalisation in the natural deduction NJ and any cut elimination procedure for the sequent calculus is the fact that eliminating cuts from a proof is performed in small steps, such that important steps, the so-called *principal cases*, require some form of *synchronisation* of the shape of the two subproofs above the cut. The other cases are mere *trivial permutations* that reflect the lax structure of proofs in the sequent calculus: the order between the rule instances in a given proof are often irrelevant, leading to the notion of proof-nets in linear logic [Gir87].

While the view of principal cases in cut elimination as synchronisation has lead to the *concurrent* interpretation of sequent calculi, as done in linear logic [CP10], this emphasis on trivial permutations leads to the idea that instead of using proof-nets to represent parallel processes, one can adopt a syntactic approach where permutations correspond to the equations of a congruence on processes — as usually considered in the process calculi community. The correspondence we are looking for is thus:

propositions	session types
proofs	processes
cut elimination	communication
congruence	permutation

2 Linear Logic and Session Solos

The session types system introduced by Caires and Pfenning [CP10], based on the intuitionistic variant of linear logic and adapted to linear logic by Wadler [Wad12] establishes a correspondence that provides strong guarantees on typed π-terms, but the connection it shows between proofs

and processes is not as tight as one might expect, in the sense that some equations on processes cannot be reflected in the structure of proofs. To improve this, we propose to use the *solos* calculus [LV03], a restricted setting where inputs and outputs are free of explicit sequentialisation — it drops the prefix operation just as the asynchronous π-calculus drops it for outputs, leaving only implicit, *causal* dependencies. Consider for example the typing of some process starting with an output in Wadler's typed π-calculus:

$$\dfrac{\overset{\Pi_1}{\triangledown}\quad P \vdash \Gamma, y : A \qquad \overset{\Pi_2}{\triangledown}\quad Q \vdash \Delta, x : B}{\overline{x}\langle y\rangle.(P \,|\, Q) \vdash \Gamma, \Delta, x : A \otimes B}$$

and observe that typing involves the decomposition of several operators, where in particular the prefix operation supposedly prevents processes inside P or Q to *move out* even if they are not using x or y. As a consequence, typeability is not preserved by the structural congruence usually applied on π-terms, as illustrated by the two terms below:

$$\overline{x}\langle y\rangle.(P \,|\, (\nu z)\,(Q \,|\, R)) \;\equiv\; \overline{x}\langle y\rangle.(\nu z)\,((P \,|\, Q) \,|\, R)$$

where the left one is typeable and the right one is not. Moving to the more parallel setting of solos allows to write this process as $(y)\,(\overline{x}\langle y\rangle|(z)\,(P|Q))$ or equivalently as $(y)\,(z)\,(\overline{x}\langle y\rangle|(P|Q))$, which is much closer to the structure of the corresponding proof in the sequent calculus, where some rule instances in Π_1 and Π_2 might permute downwards.

The adoption of solos as the underlying computational model yields a session type system where the correspondence between proofs and processes is tighter, and all this can also be adapted to the intuitionistic variant of linear logic. This might place solos in the position of being an interesting implementation language for typesafe concurrent programming, but it does not solve all problems of previous systems nor create a perfect correspondence. Because of the shape of proofs in the sequent calculus, an exact matching of π-terms or solos processes with such proofs is impossible. This offers two orthogonal possibilities: either making concessions on the side of process calculi and constrain further the structure of processes, or use a different logical formalism where the structure of proofs is even more lax than in the sequent calculus.

References

[CP10] L. Caires and F. Pfenning. Session types as intuitionistic linear propositions. In P. Gastin and F. Laroussinie, editors, *CONCUR'10*, volume 6269 of *LNCS*, pages 222–236, 2010.

[Gir87] J-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.

[LV03] C. Laneve and B. Victor. Solos in concert. *Mathematical Structures in Computer Science*, 13(5):657–683, 2003.

[Wad12] P. Wadler. Propositions as sessions. In P. Thiemann and R. B. Findler, editors, *ICFP'12*, pages 273–286. ACM, 2012.