Standard classical logic as protocol for process communication

Dragiša Žunić

Carnegie Mellon University in Qatar

Keywords:

Classical logic, process calculi, asterix calculus, pi calculus, session types

Introduction

We propose a process calculus for classical logic with explicit structural rules. Rather than going though linear logic we use a standard classical logic formalized in the sequent calculus. In this ongoing research, we illustrate that classical logic has potential to naturally serve as protocol for concurrent process communication.

Being in the era of concurrent, distributed and parallel computing, we are concerned with searching for a calculus for concurrency - which fits naturally the concurrent setting and which is rooted in logic, in the manner λ-calculus is a foundation for functional programming. This paradigm is thus the following: proofs are processes; propositions are session types; proof-transformation (here cut-elimination) is communication.

As in the works of Wadler [1, 2] and Caires and Pfenning [3], we let logic guide the design of the "right" process calculus (let us mention a more recent work dealing with multiparty sessions [4]). The logic we consider here is classical logic with explicit structural rules formalized in two-sided sequent calculus (close to Gentzen's LK). This is a promising logical setting for concurrency because it features symmetry, non-confluence and control over erasing and duplicating terms. Moreover, this research itself (process communication being of spatial nature) provides new insight into the essence of logic. Building upon the work by Bierman and Urban [5], we have previously explored classical computation with explicit structural rules at different levels of abstraction [6, 7], and similarly in the intuitionistic setting [8].

$$P, Q \;::=\; -x- \;\mid\; (\nu x)(P \mid Q) \;\mid\; x[y,z].(P \mid Q) \;\mid\; x(y,z).P$$
$$\mid\; x^e[\,].P \;\mid\; x^e(\,).P \;\mid\; x^d[y,z].P \;\mid\; x^d(y,z).P$$

Figure 1: Syntax

$$\frac{}{-x- \;:\cdot\; x : A \vdash x : A}\,(\mathrm{Ax}) \qquad \frac{P \;:\cdot\; \Gamma \vdash x : A, \Delta \qquad Q \;:\cdot\; \Gamma', x : A \vdash \Delta'}{(\nu x)(P \mid Q) \;:\cdot\; \Gamma, \Gamma' \vdash \Delta, \Delta'}\,(\mathrm{Cut})$$

$$\frac{P \;:\cdot\; \Gamma \vdash y : A, \Delta \qquad Q \;:\cdot\; \Gamma' \vdash z : B, \Delta'}{x[y,z].(P \mid Q) \;:\cdot\; \Gamma, \Gamma' \vdash x : A \wedge B, \Delta, \Delta'}\,(\wedge_R) \qquad \frac{R \;:\cdot\; \Gamma, y : A, z : B \vdash \Delta}{x(y,z).R \;:\cdot\; \Gamma, x : A \wedge B \vdash \Delta}\,(\wedge_L)$$

$$\frac{P \;:\cdot\; \Gamma \vdash \Delta}{x^e[\,].P \;:\cdot\; \Gamma \vdash x : A, \Delta}\,(\mathrm{W}_R) \qquad \frac{Q \;:\cdot\; \Gamma \vdash \Delta}{x^e(\,).Q \;:\cdot\; \Gamma, x : A \vdash \Delta}\,(\mathrm{W}_L)$$

$$\frac{P \;:\cdot\; \Gamma \vdash y : A, z : A, \Delta}{x^d[y,z].P \;:\cdot\; \Gamma \vdash x : A, \Delta}\,(\mathrm{C}_R) \qquad \frac{Q \;:\cdot\; \Gamma, y : A, z : A \vdash \Delta}{x^d(y,z).Q \;:\cdot\; \Gamma, x : A \vdash \Delta}\,(\mathrm{C}_L)$$

Figure 2: The type system

Classical logic as process calculus

We will see that this setting has a particular flavor comparing to what is seen before. We have explicit weakening which introduces a tool (call it channel-eraser) that a process can, under certain assumptions, use to erase other processes. However, this will happen only when another process tries to establish a communication over that channel-eraser.

Also we have explicit contraction which implements a tool (channel-duplicator) that a process can use (provided that it has one) for duplication of processes that try to communicate over that channel.

We present a blueprint on how all this works through providing syntax, type assignment rules and several reduction rules.

Elements of the syntax are given in Fig. 1. They are assigned types as presented in Fig 2, thus setting up to have classical propositions as session types, i.e., a protocol for process communication.

We have the reduction rules of type β (logical reduction rules) and of type σ (structural reduction rules). Few of the reduction rules are given in Fig. 3. We assume that free names in Q are u and v, and that free names of P are r, s. Computation preserves free names and types.

$$
\begin{array}{lcll}
(\sigma_{W_R}) & : & (\nu x)(x^e[\,].P \mid x(w).Q) & \rightarrow \quad u^e[\,].v^e[\,].P \\
(\sigma_{W_L}) & : & (\nu x)(x[w].P \mid x^e(\,).Q) & \rightarrow \quad r^e(\,).s^e(\,).Q \\
(\sigma_{C_R}) & : & (\nu x)(x^d[y,z].P \mid x(w).Q) & \rightarrow \quad u^d[u_1,u_2].v^d[v_1,v_2].(\nu z)((\nu y)(P \mid Q\{y/w\}) \mid Q\{z/w\}) \\
(\sigma_{C_L}) & : & (\nu x)(x[w].P \mid x^d(y,z).Q) & \rightarrow \quad r^d(r_1,r_2).s^d(s_1,s_2).(\nu z)((\nu y)(Q \mid P\{y/w\}) \mid P\{z/w\})
\end{array}
$$

Figure 3: Partial reduction rules for process calculus

Acknowledgements

This paper was made possible by NPRP 7-988-1-178 from the Qatar National Research Fund (a member of Qatar Foundation). The statements made herein are solely the responsibility of the authors.

References

[1] P. Wadler, *Propositions as sessions*, ACM SIGPLAN International Conference on Functional Programming, ICFP 2012, pp. 273–286, 2012.

[2] P. Wadler, *Propositions as sessions*, Journal of Functional Programming, Vol. 24, No. 2-3, pp. 384–418, 2014.

[3] L. Caires, F. Pfenning, *Session Types as Intuitionistic Linear Propositions*, CONCUR 2010, pp. 222–236, 2010.

[4] M. Carbone, S. Lindley, F. Montesi, C, Schürmann, P. Wadler, *Coherence Generalises Duality: A Logical Explanation of Multiparty Session Types*, CONCUR 2016, pp. 33:1-33:15, 2016.

[5] C. Urban, *Classical Logic and Computation*, University of Cambridge (PhD thesis), 2000.

[6] D. Zunic, *Computing With Sequent and Diagrams in Classical Logic - Calculi $^*\mathcal{X}$, $©\mathcal{X}$ and $d\mathcal{X}$*, ENS Lyon (PhD thesis), 2007. (`https://tel.archives-ouvertes.fr/tel-00265549`)

[7] D. Zunic, P. Lescanne, *A congruence relation for restructuring classical terms*, ICTCS 2017, accepted.

[8] S. Ghilezan, J. Ivetic, P. Lescanne and D. Zunic, *Intuitionistic Sequent-Style Calculus with Explicit Structural Rules*, TbiLLC 2009, pp. 101–124, 2009.