Theoretical Computer Science 411 (2010) 2223–2238

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

An exact correspondence between a typed pi-calculus and polarised proof-nets

Kohei Honda [a], Olivier Laurent [b,*]

[a] *Department of Computer Science, Queen Mary, University of London, United Kingdom*
[b] *Preuves Programmes Systèmes, CNRS, Universtité Paris 7, France*

ARTICLE INFO

Article history:
Received 27 November 2008
Received in revised form 29 September 2009
Accepted 24 January 2010
Communicated by P.-L. Curien

Keywords:
Pi-calculus
Proof-nets
Processes
Logics
Proofs
Types
Interaction
Concurrency
Linear Logic
Polarity
Embedding
Determinism
Non-determinism

ABSTRACT

This paper presents an exact correspondence in typing and dynamics between polarised linear logic and a typed π-calculus based on IO-typing. The respective incremental constraints, one on geometric structures of proof-nets and one based on types, precisely correspond to each other, leading to the exact correspondence of the respective formalisms as they appear in Olivier Laurent (2003) [27] (for proof-nets) and Kohei Honda et al. (2004) [24] (for the π-calculus).

1. Introduction

This paper presents exact mutual embeddings between the proof-nets and the π-calculus. More precisely, we show that a specific form of *polarised linear logic* [27] and a typed version of the asynchronous π-calculus [24] are essentially different ways of presenting the same structure. On the one hand, the π-calculus is a formalism which can represent a wide variety of computational phenomena starting from sequential programming languages to distributed computation through a simple operation, channel passing, originating in CCS; on the other hand, proof-nets represent dynamics of cut elimination in Linear Logic, which is a proof-theoretic reformulation of intuitionistic logic. These two formalisms come from quite different backgrounds: this paper pinpoints an exact way in which we can relate these two different formalisms.

As is well known, the so-called Curry–Howard correspondence allows us to relate proofs of intuitionistic logic to the typed λ-calculus, so that a proof corresponds to a typed λ-term, the assumptions and conclusion of a proof to the types of that term, and cut elimination (a mechanical procedure to simplify proofs preserving the same assumptions and conclusions) to

* Corresponding author. Tel.: +33 4 72 72 84 34.
E-mail address: olivier.laurent@ens-lyon.fr (O. Laurent).

0304-3975/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2010.01.028

reduction in the λ-term. Linear Logic [15] can fully embed this isomorphism: further the logic enjoys duality of the classical logic, and the notion of cut elimination in proof-nets is more fine-grained than the cut elimination in the intuitionistic logic. The connectives of Linear Logic arise as decomposition of the semantic universe underlying intuitionistic logic. The study of Linear Logic and proof-nets is usually motivated by logical or logically oriented semantic concerns, though some of the ideas in Linear Logic (such as linearity) are widely used in computational formalisms other than logically motivated ones.

In contrast, the π-calculus [35] is a relatively recent development in process algebras, syntactic formalisms for understanding concurrent computation which include, among others, CCS [32], CSP [19] and ACP [11]. In particular, all algebraic operators in the π-calculus are those of CCS: the π-calculus merely adds a syntactic treatment of name passing to CCS. The motivation to add name passing in the π-calculus is practical: the authors of [35] wish to model so-called *mobile systems*, i.e., communicating systems in which connectivity among processes dynamically changes. Such systems are commonplace nowadays, be they mobile phone networks or email communication where email addresses themselves are exchanged. The study of the π-calculus in particular and process algebras in general is motivated by modelling practical systems rather than logical concern, even though it is also used as a tool to study semantics and logics of computation.

The present work shows that, in spite of these two very different origins and orientations, there is an expressive common part shared by these two formalisms. This "common part" is obtained through a natural restriction of each formalism: proof-nets are restricted so that they are "polarised" [27], i.e. we have distinction between positive and negative formulae. For the π-calculus we use, other than asynchrony in communication, a restriction by types used for studying control operators [24]. As we shall show, the correspondence between two fragments is as exact as can be: two translations are mutually inverse (up to the structural equality of the π-calculus); the well-formed proof-nets are precisely well-typed processes via translation and vice versa; and one-step reduction in one formalism is precisely mirrored by one-step reduction in another formalism.

The result poses many questions: why one of the (arguably) most advanced tools to study dynamics of logics and one of the (arguably) most advanced tools to study dynamics of concurrent computation coincide so closely? What merits does this correspondence give us? For the former the authors do not have an answer (another related mathematical tool for computation is *game semantics* [4,18], which again has a different origin and motivation but has a close tie with both formalisms). For the second question, it may be worth recording a couple of observations we have gained through the present inquiry. First, our correspondence results treat the proof-nets which are semantically non-deterministic (in the π-calculus notation we allow typed processes of the form $!x.0|!x.0$ which introduce racing at x). While non-deterministic proof-nets have been studied before ([30] for example), the presented variant would be of interest because of its precise correspondence with the π-calculus: the whole laws and theories of non-deterministic processes from process algebras can now be applied (for example we can now discuss the notion of interactive behaviour of proof-nets directly using transition relations of the π-calculus). Second, the syntactic constructs in the π-calculus are given an exact geometric presentation through the corresponding proof-nets. While many graphical formalisms for the π-calculi have been studied [8,25,29,31], our work differs in that the target graphical formalism is directly based on proof-nets. This allows us to compare incremental geometric constraints in proof-nets on the one hand and incremental type-based constraints in the π-calculus. For emphasising this point, our presentation starts from the correspondence result for the respective non-deterministic extensions and adds incremental constraints: for each constraint we again establish one-to-one correspondence between them, finally reaching the deterministic, terminating formalisms studied in [27,24]. Thus already the both-way interactions enrich these two theories. We hope that the present inquiry serves as a starting point of further dialogues between these two different fields of study.

The connection between process algebras and Linear Logic was first observed by Samson Abramsky in his computational interpretation of Linear Logic [1,2]. The process formalism he used is not a standard process algebra but a syntactic construction which directly represents proofs of Linear Logic as terms. Bellin and Scott [10] have shown an embedding of a version of proof-nets with a notion of polarities to a "synchronous" π-calculus, a non-standard version of the π-calculus with commutativity of prefixes. Laneve, Parrow and Victor [29] showed that some parts of dynamics of the original proof-nets can be captured by the fusion calculus, which is a variant of the π-calculus with (what corresponds to) axiom links. Beffara [5] has shown a realisability semantics of Linear Logic which uses a synchronous version of the π-calculus with a dynamic link. The present work differs from these works (and also from [8,14,12]) in that we focus on significant, and independently conceived, fragments of the original formalisms, and that there is an exact correspondence in term/graph constructions and dynamics.

In the remainder, we first outline a fragment of proof-nets we shall use (which is a non-deterministic extension of polarised proof-nets in [27]). Then we outline a fragment of the π-calculus we shall use (which is a relaxing of type constraints in the typed π-calculus in [24]). We then relate these two formalisms. In the next section we discuss how incremental restrictions in these two formalisms lead to more restricted classes of processes/proofs. This is the core of our correspondence where we finally reach a precise coincidence between polarised proof-nets [27] and the control π-calculus [24]. The final section is dedicated to extensions of the correspondence between the two formalisms.

2. Proof-nets

The notion of proof-nets we consider here comes from polarised proof-nets [27,28] with two important extensions: n-ary !-nodes (responsible for non-determinism) and named cuts. In addition, it is presented in the so-called "focalised" form [17] (using n-ary multiplicative nodes, $\otimes$ and $\parr$); we omit axiom nodes; and we only consider exponential cuts.