Typing a linear π-calculus

Uma Zalakain
University of Glasgow, Scotland
u.zalakain.1@research.gla.ac.uk

Ornela Dardha
University of Glasgow, Scotland
ornela.dardha@glasgow.ac.uk

Abstract

We present the syntax, operational semantics, and typing rules of a π-calculus with linear and shared types. We abstract over types (using *shapes* to untangle usage contexts from typing contexts) and generalize the algebras on multiplicities (using indexed sets of *partial commutative monoids*). We use leftover typing [1] to encode our typing rules in a way that propagates linearity constraints into process continuations. We provide framing, weakening and strengthening proofs that we then use to prove subject congruence. We show that the type system is stable under substitution and prove subject reduction.

This formalization has been fully mechanized with Agda and is available at `https://github.com/umazalakain/typing-linear-pi`.

2012 ACM Subject Classification Theory of computation → Process calculi

Keywords and phrases pi calculus, linear, types, concurrency

Digital Object Identifier 10.4230/LIPIcs...

Supplement Material `https://github/umazalakain/typing-linear-pi`

Acknowledgements I want to thank . . .

1 Introduction

extensional typing rules for a given syntax and operational semantics

1.1 Contribution

machine verified formalisation of the pi calculus

typing with leftovers applied to the pi calculus

multiple multiplicities per variable

multiple multiplicity types

full formalisation available in Agda

2 Related work

[?] polymorphic tokens, HOAS

[?]

3 Syntax

variable references (strings, locally named, de Bruijn)

allows to ignore alpha conversion, or proofs of inequality between strings

strings to maybe de Bruijn, names can be kept in context as well, just not doing it

4 Semantics

4.1 Structural congruence

congruence relationship indexed by recursive tree

4.2 Reduction

keeping track of the variable on which communication occurs

5 Linear typing rules

5.1 Variable references

Polarities

Multiple variables

Vectors Most general solution, n possible multiplicities

5.2 Multiplicities

Generalisation over indexed sets of *partial commutative monoids*

properties of the underlaying monoid

given two multiplicities, the third is uniquely determined. this is better modelled through a function rather than an inductive type.

5.3 Variable types and multiplicities

two-layered approach: types on one hand, capabilities on the other removing from context vs keeping in context but marking it used

5.4 Typing with leftovers

5.4.1 Typing relation

Variable references as proofs of capability

Context splits at each variable reference

6 Subject reduction

6.1 Framing

Definition Let $\Gamma \mathsf{P} \boxtimes \Delta$ and $\Gamma - Delta \equiv \Xi - \phi$. Then $\Xi \mathsf{P} \boxtimes \phi$.

By defining of $\uplus$ as a total function, $\Gamma - \Delta$ is no longer functional: $\omega - \omega$ results in any multiplicity, including 0. Therefore $\Xi \mathsf{P} \boxtimes \phi$ would imply that such a variable cannot appear in P.

6.2 Weakening

6.3 Strengthening

6.4 Swapping

6.5 Substitution

7 Future work

Work that will be done time permiting:

Soundness and completeness with respect to an alternative formalization.
Proof of progress
Product types
Sum types
Decidable typechecking
Encoding of session types

References

1 Guillaume Allais. Typing with Leftovers - A mechanization of Intuitionistic Multiplicative-Additive Linear Logic. page 22 pages, 2018. http://drops.dagstuhl.de/opus/volltexte/2018/10049/. doi:10.4230/lipics.types.2017.1.