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

remove todo notes

Abstract

We present the syntax, operational semantics, and typing rules of a π-calculus with linear and shared types. We use leftover typing [1] to encode our typing rules in a way that propagates linearity constraints into process continuations. We generalize the algebras on multiplicities using indexed sets of *partial commutative monoids*, allowing the user to choose a mix of linear, affine, gradual and shared typing. 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

The π-calculusmodels communication.

why resource-aware typing

extensional typing rules for a given syntax and operational semantics

leftover typing

1.1 Contribution

Machine verified formalisation of the linear pi calculus
Typing with leftovers applied to the pi calculus
Abstraction over multiplicities
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

$$\frac{n : \mathbb{N}}{\mathrm{VAR}_n : Set} \qquad \frac{n : \mathbb{N}}{0 : \mathrm{VAR}_n} \qquad \frac{x : \mathrm{VAR}_n}{{}_{1+}x : \mathrm{VAR}_{1+n}}$$

Figure 1 Types of size n

$$\frac{n : \mathbb{N}}{\mathrm{PROCESS}_n : Set}$$

$$\begin{aligned}\mathrm{PROCESS}_n ::= {} & \mathbf{0}_n \\ | {} & \boldsymbol{\nu}\, \mathrm{PROCESS}_{1+n} \\ | {} & \mathrm{PROCESS}_n \parallel \mathrm{PROCESS}_n \\ | {} & \mathrm{VAR}_n\, ()\, \mathrm{PROCESS}_{1+n} \\ | {} & \mathrm{VAR}_n\, \langle \mathrm{VAR}_n \rangle\, \mathrm{PROCESS}_n\end{aligned}$$

Figure 2 Well-scoped grammar using de Bruijn indices

4 Semantics

4.1 Structural congruence

$$\frac{P\ Q : \mathrm{PROCESS}_n}{P \equiv_n Q : Set} \qquad \frac{P\ Q\ R : \mathrm{PROCESS}_n}{comp - assoc : P \parallel Q \parallel R \equiv_n P \parallel Q \parallel R}$$

$$\frac{P\ Q : \mathrm{PROCESS}_n}{comp - sym : P \parallel Q \equiv_n Q \parallel P} \qquad \frac{P : \mathrm{PROCESS}_n}{comp - end : P \parallel \mathbf{0}_n \equiv_n P}$$

$$\frac{}{scope - end : \boldsymbol{\nu}\, \mathbf{0}_{1+n} \equiv_n \mathbf{0}_n} \qquad \frac{P\,Q : \mathrm{PROCESS}_{1+n} \qquad uQ : \mathrm{UNUSED}_0\, Q}{scope - ext : \boldsymbol{\nu}\, (P \parallel Q) \equiv_n (\boldsymbol{\nu}\, P) \parallel lower\ 0\ \ Q\ uQ}$$

$$\frac{P : \mathrm{PROCESS}_{1+1+n}}{scope - comm : \boldsymbol{\nu}\, \boldsymbol{\nu}\, P \equiv_n \boldsymbol{\nu}\, \boldsymbol{\nu}\, swap\ 0\ P}$$

Figure 3 Structural rewriting rules

congruence relationship indexed by recursive tree

4.2 Reduction

keeping track of the variable on which communication occurs

$$\frac{}{\text{REC} : Set} \qquad \frac{}{zero : \text{REC}} \qquad \frac{r : \text{REC}}{one\ r : \text{REC}} \qquad \frac{r\ s : \text{REC}}{two\ r\ s : \text{REC}}$$

$$\frac{P\ Q : \text{PROCESS}_n \qquad r : \text{REC}}{P =_r Q : Set} \qquad \frac{P\ Q : \text{PROCESS}_n \qquad P \equiv_n Q}{struct : P =_{zero} Q}$$

$$\frac{P\ P' : \text{PROCESS}_{1+n} \qquad P =_r P'}{cong - scope : \boldsymbol{\nu}\ P =_{one\ r} \boldsymbol{\nu}\ P'} \qquad \frac{P\ P' : \text{PROCESS}_n \qquad P =_r P'}{cong - comp : P \parallel Q =_{one\ r} P' \parallel Q}$$

$$\frac{P\ P' : \text{PROCESS}_{1+n} \qquad P =_r P'}{cong - recv : x\,()\ P =_{one\ r} x\,()\ P'} \qquad \frac{P\ P' : \text{PROCESS}_n \qquad P =_r P'}{cong - send : x\,\langle y\rangle\ P =_{one\ r} x\,\langle y\rangle\ P'}$$

$$\frac{P : \text{PROCESS}_n}{refl : P =_{zero} P} \qquad \frac{P\ Q : \text{PROCESS}_n \qquad P =_r Q}{sym : Q =_{one\ r} P}$$

$$\frac{P\ Q\ R : \text{PROCESS}_n \qquad P =_r Q \qquad Q =_s R}{trans : P =_{two\ r\ s} R}$$

Figure 4 Structural rewriting rules lifted to a congruent equivalence relation

5 Linear typing rules

5.1 Multiplicities

A type system with both linear and shared resources has multiplicities 0, 1 and ω.

5.2 Variable references

5.3 Contexts

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

6.2 Weakening

Suffices to model as a series of insertions.

$$\frac{n : \mathbb{N}}{\mathrm{CHANNEL}_n : Set} \qquad \frac{}{nothing : \mathrm{CHANNEL}_n} \qquad \frac{i : \mathrm{VAR}_n}{just\ i : \mathrm{CHANNEL}_n}$$

$$\frac{i : \mathrm{CHANNEL}_n \qquad P\ Q : \mathrm{PROCESS}_n}{P \longrightarrow_i Q : Set}$$

$$\frac{i\ j : \mathrm{VAR}_n \qquad P : \mathrm{PROCESS}_{1+n} \qquad Q : \mathrm{PROCESS}_n \qquad uP : \mathrm{UNUSED}_0\ P}{comm : i\,()\,P \parallel i\,\langle j \rangle\,Q \longrightarrow_{just\ i} lower\ 0\ P[j/0]\ uP \parallel Q}$$

$$\frac{P \longrightarrow_i P'}{par : P \parallel Q \longrightarrow_i P' \parallel Q} \qquad \frac{P \longrightarrow_i Q}{res : \boldsymbol{\nu}\,P \longrightarrow_{dec\ i} \boldsymbol{\nu}\,Q} \qquad \frac{P = P' \qquad P' \longrightarrow_i Q}{struct : P \longrightarrow_i Q}$$

Figure 5 Operational semantics indexed by reducing channel

$$\frac{n : \mathbb{N}}{\mathrm{PRECTX}_n : Set} \qquad \frac{}{[] : \mathrm{PRECTX}_0} \qquad \frac{\gamma : \mathrm{PRECTX}_n \qquad t : \mathrm{TYPE}}{\gamma, t : \mathrm{PRECTX}_{1+n}}$$

6.3 Strengthening

6.4 Swapping

6.5 Substitution

7 Future work

Work that will be done time permiting:

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

$$\frac{n : \mathbb{N}}{\mathrm{IDXS}_n : Set} \qquad \frac{}{[] : \mathrm{IDXS}_0} \qquad \frac{\imath : \mathrm{IDXS}_n \qquad x : \mathrm{IDX}}{\imath, x : \mathrm{IDXS}_{1+n}}$$

$$\frac{\imath : \mathrm{IDXS} n}{\mathrm{CTX}_\imath : Set} \qquad \frac{}{[] : Ctx_{[]}} \qquad \frac{\Gamma : \mathrm{CTX}_\imath \qquad c : \mathrm{CARRIER}_x}{\Gamma, c : \mathrm{CTX}_{\imath,x}}$$

$$\frac{\gamma : \mathrm{PRECTX}_n \qquad \begin{array}{c}\imath : \mathrm{IDXS}_n \\ \Gamma : \mathrm{CTX}_\imath\end{array} \qquad P : \mathrm{PROCESS}_n \qquad \Delta : \mathrm{CTX}_\imath}{\gamma \propto \Gamma \vdash P \boxtimes \Delta : Set}$$