University of BRISTOL

DEPARTMENT OF COMPUTER SCIENCE

Modal types for distributed programming

Joseph Eastoe

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Wednesday 31st May, 2023

Abstract

As distributed programming gains prominence with the shift to cloud-based architectures, addressing challenges such as race conditions, deadlocks, livelocks, and consistency loss is crucial. In this work, we define a dual-context calculus similar to Moody's and describe a distributed abstract machine inspired by Moody's, however adopting the style of the π-calculus. Using a bi-simulation method, we establish the computational equivalence of the dual-context calculus and the distributed abstract machine, ensuring that problems associated with distributed systems, such as deadlock and livelock, do not occur within the machine; This serves as the initial step toward improving non-serial programming by developing a programming language that can handle the challenges associated with it, in a similar way to how type safety of language guarantees progress and preservation. We conclude with a literature review that compares our dual calculus and the machine to other relevant works in the field.

Declaration

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Taught Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, this work is my own work. Work done in collaboration with, or with the assistance of others, is indicated as such. I have identified all material in this dissertation which is not my own work through appropriate referencing and acknowledgement. Where I have quoted or otherwise incorporated material which is the work of others, I have included the source in the references. Any views expressed in the dissertation, other than referenced material, are those of the author.

Joseph Eastoe, Wednesday 31st May, 2023

Contents

List of Figures

Ethics Statement

This project did not require ethical review, as determined by my supervisor, Alex Kavvos

Chapter 1

Introduction

Distributed programming is gaining prominence as many applications move towards cloud-based architectures [Shi22]. However, programming distributed systems can be complex due to challenges such as race conditions, deadlocks, livelocks, and consistency loss.

In Moody's paper, modal types were proposed as a means to represent spatial properties, mobility and locality[Moo05]. Moody introduced a dual-context calculus based on modal logic that leveraged these modal types and defined a distributed abstract machine representing the concurrent execution of the calculus across multiple processes at different locations. However, Moody's paper lacked proof of the computational equivalence between the abstract machine and the dual-context calculus.

The aims of this project are:

- Discuss the origins behind Moody's language.
- Define a dual-context similar to Moody's calculus.
- Describe a distributed abstract machine following Moody's approach, presented in the style of π-calculus.
- Prove the computational equivalence of the dual-context and the distributed abstract machine using a bi-simulation method.
- Conclude with a literature review comparing the dual calculus and the defined machine to other relevant works.

Chapter 2

Background

2.1 Motivation

Distributed programming is becoming increasingly necessary as more and more applications move towards cloud-based architectures [Shi22]. Switching to the cloud offers a range of benefits, such as improved performance and scalability. However, programming distributed systems can be challenging, as it requires dealing with race conditions, deadlocks, livelocks, and loss of consistency. Similarly, multithreaded applications offer equal benefits but have the same complexities and potential pitfalls. However, despite these hurdles, the benefits that this type of computing brings are too substantial to ignore.

As the demand for distributed programming and parallel computing continues to grow, it becomes essential to develop tools to assist with the challenges that come with this type of programming. By supporting developers and organizations working with distributed systems or multithreaded applications, we can minimize bugs, enhance reliability and strengthen security. We propose the first steps towards improving non-serial programming by further developing a programming language that theoretically could not have deadlocks or livelocks by design, similar to how the type safety of a programming language guarantees progress and preservation.

2.2 Origins

Modal logic is a formal system that adds concepts of necessity and possibility. In modal logic, these modal concepts are represented by modal operators, such as "necessarily" ("$\Box$") and "possibly" ("$\Diamond$"). The feature of modal logic is its ability to reason about truth from various viewpoints, referred to as "possible worlds" or just "worlds." These worlds can have different sets of relative truths. For example, in some worlds, it might be sunny, while in others, it might not be.
Kripke semantics provides a formal framework for understanding modal logic. In this framework, necessarily A ($\Box A$) is true at a world w if and only if A is true at all accessible worlds from w; for example, necessarily $1 + 1 = 2$ is true. Possibly A ($\Diamond A$) is true at a world w if and only if A is true in some possible world which is accessible from w. Which worlds are accessible from w is defined via the accessibility relation in the Kripke model. Allowing the accessibility relation to have certain properties such as reflexivity, symmetry, and transitivity leads to different types of modal logic. S4 modal logic is where the relation has reflexive and transitive, and S5 is where the relation also has symmetry.
The Curry-Howard correspondence is the relationship between proofs in a formal system and programs such as ones from lambda calculus. A proof of a logical proposition can be viewed as a program which when "executed" produces evidence for the proposition. Programs can be considered as proofs where the inputs are the premises and outputs are the conclusions of the proposition. For example, using the correspondence type checking can be seen as proof checking, where a program with type A corresponds to a valid proof of the logical proposition encoded with type A.

Davies and Pfenning [DP01] proposed a modal logic approach to staged computation, staged computation refers to a technique in which a program's computation is split into multiple stages. They extended the Curry-Howard correspondence to include modal logic(S4), using it to produce their dual-context calculus (global context and local context), which in turn had modal type ($\Box A$, $\Diamond A$). Each world in the Kripke semantic of modal logic corresponds to a stage in computation, and terms(code) that have type

$\Box A$ (in the context of modal logic, meaning necessarily A) can be executed in a future stage of computation.

Moody presented almost identical calculus (derived from Pfenning and Davies [PD01]) based on S4 modal logic that also utilizes the Curry-Howard correspondence, interpreting propositions as types and programs as proofs, and incorporating modal types [Moo05]. In this case, each world in the Kripke semantics is interpreted as a site of computation; however, the specific locations remain abstract. Moody also defined a distributed abstract machine that represents multiple processes running in parallel, each performing computation within the calculus with the worlds corresponding to the different processes.

Moody utilized modal types from the correspondence to encapsulate spatial properties (mobility and locality) on terms where mobility refers to the ability of terms to move within the distributed abstract machine. One example of modal types is the type $\Box A$, which represents a term of type A that possesses location independence. This means it can be evaluated at any arbitrary location and thus has mobility. Another type, $\Diamond A$, gives the calculus locality.
Below we present a language based on S4 modal logic that uses the type $\Box A$ giving it mobility, however not locality.

2.3 Dual-context calculus

We present a typing judgment of the below form.

$$\Delta; \Gamma \vdash M : A$$

where Δ, Γ are contexts, M is a term and A is the type of the term M. The contexts are defined by $\Gamma ::= \cdot \mid \Gamma',\ x\ :\ B$ where x is a variable and B is the type of N.

We assume that no variables are duplicated within the contexts; therefore, if x appears in Γ, it cannot also be present in Δ. When introducing a new variable to these contexts, we assume it to be distinct from all existing variables. If this condition is not met, we can always alpha rename to guarantee its uniqueness. We make no distinction between contexts that are only different in the order of their assumptions.

We will now define this formally following Davies and Pfenning [DP01].

2.3.1 Statics

types	A, B	$::=$	Num	numbers
			$A \to B$	function type
			$\Box A$	modal type
contexts	Γ, Δ	$::=$		empty context
			$\Gamma, x : A$	context extension
terms	M, N	$::=$	x	variables
			$\lambda x : A.\, M$	function
			$M(N)$	function application
			$\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N$	let box
values	V, W	$::=$	$\overline{n}$	number
			$\lambda x : A.\, M$	function
			$\mathsf{box}\ M$	box

Let M, N_1, N_2 be terms. Then the set of free variables is defined as follows:

$$\mathrm{fv}(x) = \{x\} \qquad \mathrm{fv}(N_1(N_2)) = \mathrm{fv}(N_1) \cup \mathrm{fv}(N_2)$$

$$\mathrm{fv}(\lambda x.\, N_1) = \mathrm{fv}(N_1) \setminus \{x\} \qquad \mathrm{fv}(\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2) = \mathrm{fv}(N_1) \cup (\mathrm{fv}(N_2) \setminus \{u\})$$

$$\textsc{Num}\ \frac{n \in \mathbb{N}}{\Gamma \vdash \overline{n} : \mathsf{Num}}$$

(a) Determines that a natural number is a term of type Num.

$$\textsc{Var}\ \frac{}{\Delta; \Gamma, x : A \vdash x : A}$$

(b) A variable can derive its type from local context.

$$\textsc{MVar}\ \frac{}{\Delta, u : A; \Gamma \vdash u : A}$$

(c) A variable can derive its type from global context.

$$\textsc{Lam}\ \frac{\Delta; \Gamma, x : A \vdash M : B}{\Delta; \Gamma \vdash \lambda x : A.\, M : A \to B}$$

(d) A lambda abstraction taking in x with type A surrounding a term of type B given x added to its local context, has type A to B

$$\textsc{App}\ \frac{\Delta; \Gamma \vdash M : A \to B \qquad \Delta; \Gamma \vdash N : A}{\Delta; \Gamma \vdash M(N) : B}$$

(e) Application of a function term M to an argument term N results in a term of type B.

$$\textsc{Box}\ \frac{\Delta; \cdot \vdash M : A}{\Delta; \Gamma \vdash \mathsf{box}\ M : \Box A}$$

(f) A term M of type A which only uses the global context can be boxed, resulting in a term of type □A.

$$\textsc{LetBox}\ \frac{\Delta; \Gamma \vdash M : \Box A \qquad \Delta, u : A; \Gamma \vdash N : C}{\Delta; \Gamma \vdash \mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N : C}$$

(g) Unboxing a term M of type □A and using it in term N by adding it to the global context results in a term of type C.

Figure 2.1: Typing rules

$$\frac{\dfrac{\dfrac{}{u : \mathsf{Num}; \cdot \vdash u : \mathsf{Num}}}{u : \mathsf{Num}; \cdot \vdash \mathsf{box}\ u : \Box\mathsf{Num}} \qquad \dfrac{\dfrac{}{u : \mathsf{Num}, v : \mathsf{Num}; \cdot, x : \mathsf{Num} \vdash x : \mathsf{Num}}}{u : \mathsf{Num}, v : \mathsf{Num}; \cdot \vdash \lambda x : \mathsf{Num}.\, x : \mathsf{Num} \to \mathsf{Num}}}{u : \mathsf{Num}; \cdot \vdash \mathsf{let\ box}\ v \Leftarrow \mathsf{box}\ u\ \mathsf{in}\ \lambda x : \mathsf{Num}.\, x : \mathsf{Num} \to \mathsf{Num}}$$

Figure 2.2: Typing derivation tree of $u : \mathsf{Num}; \cdot \vdash \mathsf{let\ box}\ v \Leftarrow \mathsf{box}\ u\ \mathsf{in}\ \lambda x : \mathsf{Num}.\, x : \mathsf{Num} \to \mathsf{Num}$.

More examples can be found here A.2

The idea of different worlds expands elegantly to dual-context calculi. Γ, Δ represents the local and global context respectively, meaning that the context Γ is local to the machine (in the context of modal logic, this would be a world) and the context Δ is shared and accessed by all terms no matter of location (in modal logic this would be a shared truth across all the worlds). This allows $\Box A$ typed terms to be evaluated anywhere (mobile) as that type is only given to terms, which only depend on the global context Δ to give it the type A. Allowing the term to have type A in all worlds as the global context is accessible from anywhere.

Lemma 2.3.1 (Weakening)**.**

1. If $\Delta; \Gamma \vdash M : A$ then $\Delta; \Gamma, x : A \vdash M : A$.
2. If $\Delta; \Gamma \vdash M : A$ then $\Delta, u : A; \Gamma \vdash M : A$.

Theorem 2.3.2 (Substitution)**.**

1. If $\Delta; \Gamma \vdash M : A$ and $\Delta; \Gamma, x : A \vdash N : C$ then $\Delta; \Gamma \vdash N[M/x] : C$.
2. If $\Delta; \cdot \vdash M : A$ and $\Delta, u : A; \Gamma \vdash N : C$ then $\Delta; \Gamma \vdash N[M/u] : C$.

2.3.2 Dynamics

We augment the dual-context calculus with call-by-value dynamics (with the exception of the D-BOXBETA, which is slightly CBN). To prevent an excessive number of reduction rules such as D-App-1, D-App-2, and D-LetBox-1, we employ *evaluation contexts* in the manner of Felleisen and Hieb [FH92]. These contexts precisely indicate the locations where reductions may occur within a term.
Using Backus–Naur Form, we define.

$$
\begin{array}{rcl}
\mathcal{E} & ::= & [\,] \\
 & & \mathcal{E}(N) \\
 & & V\big(\mathcal{E}\big) \\
 & & \mathsf{let\ box}\ u \Leftarrow \mathcal{E}\ \mathsf{in}\ N
\end{array}
$$

We write $\mathcal{E}[M]$ for the term that results from replacing $[\,]$ with M.
For example, $\mathcal{E} = V(\mathcal{E}') = V([\,](N))$ then, $\mathcal{E}[M] = V(M(N))$

$$\textsc{Val-Num}\quad \frac{n \in \mathbb{N}}{\overline{n}\ \mathsf{val}}$$

(a) A natural number is a value.

$$\textsc{Val-Lam}\quad \frac{}{\lambda x : A.\, M\ \mathsf{val}}$$

(b) A lambda abstraction is a value.

$$\textsc{Val-Box}\quad \frac{}{\mathsf{box}\ M\ \mathsf{val}}$$

(c) A boxed term is a value.

$$\textsc{D-Beta}\quad \frac{}{(\lambda x : A.\, M)(V) \longmapsto M[V/x]}$$

(d) Application of a lambda abstraction and value results in a substitution.

$$\textsc{D-BoxBeta}\quad \frac{}{\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N \longmapsto N[M/u]}$$

(e) A letbox term results in unboxing a term M and using it in another term N via a substitution.

$$\textsc{D-Eval}\quad \frac{M \longmapsto M'}{\mathcal{E}[M] \longmapsto \mathcal{E}[M']}$$

(f) If a term M steps to M', then the term within the evaluation context $\mathcal{E}$ also steps accordingly.

Figure 2.3: Dynamics

$$
\begin{aligned}
(\lambda x : \mathsf{Num}.\, (\lambda y : \mathsf{Num}.\, (\lambda z : \mathsf{Num}.\, \mathsf{plus}(x; z))(y))(x))(\overline{1}) &\longmapsto (\lambda y : \mathsf{Num}.\, (\lambda z : \mathsf{Num}.\, \mathsf{plus}(\overline{1}; z))(y))(\overline{1}) \\
&\longmapsto (\lambda z : \mathsf{Num}.\, \mathsf{plus}(\overline{1}; z))(\overline{1}) \\
&\longmapsto \mathsf{plus}(\overline{1}; \overline{1}) \\
&\longmapsto \overline{2}
\end{aligned}
$$

Figure 2.4: Example of dynamics rules

We define a “basic reduction” being one that uses D-Beta or D-BoxBeta rules.

Lemma 2.3.3. Suppose $\cdot;\cdot \vdash M : A$ then either M val or there exist unique $\mathcal{E}$ and N such that $M = \mathcal{E}[N]$ and $N \longmapsto N'$ by a “basic reduction” (i.e. either D-Beta or D-BoxBeta)

Proof. By induction on $\cdot;\cdot \vdash M : A$

Case(NUM). Suppose $\cdot;\cdot \vdash M : A$ of the form

$$\frac{n \in \mathbb{N}}{\cdot;\cdot \vdash \overline{n} : \mathsf{Num}}$$

By VAL-NUM M val holds

Case(LAM). Similar to NUM

Case(BOX). Similar to NUM

Case(VAR). This case is not possible due to M being closed

Case(MVAR). Similar to VAR

Case(APP). Suppose $\cdot;\cdot \vdash M : A$ of the form

$$\frac{\cdot;\cdot \vdash M_1 : A \to B \qquad \cdot;\cdot \vdash M_2 : A}{\cdot;\cdot \vdash M_1(M_2) : B}$$

There are three cases

- M_1 val and M_2 val then for M to be well typed, M_1 must be of the form $\lambda x : A.\, P$ writing M_1 in this form and M_2 has V_2. for $M = \mathcal{E}[N]$ there are three cases.
  - $\mathcal{E}[(\lambda x : A.\, P)(V_2)] = (\lambda x : A.\, P)(V_2)$ where $\mathcal{E} = [\,]$. Thus $N \longmapsto N'$ exists by D-Beta rule.
  - $\mathcal{E}[V_2] = (\lambda x : A.\, P)(V_2)$ where $\mathcal{E} = (\lambda x : A.\, P)\Big(\mathcal{E}'\Big) = \lambda x : A.\, P([\,])$. As V_2 is a value, then $\mathcal{E}'$ has to be a hole and this shape of $\mathcal{E}'$ does not yield a reduction as V_2 is a value.
  - $\mathcal{E}[\lambda x : A.\, P] = (\lambda x : A.\, P)(V_2)$ where $\mathcal{E} = \mathcal{E}'(V_2) = \mathcal{E}'([\,])$. This similar to the above case.

  Only one of these cases works, thus unique.
- M_1 val and M_2 is not a value, writing M_1 to be V_1, for $M = \mathcal{E}[N]$ there are three cases.
  - $\mathcal{E}[V_1(M_2)] = V_1(M_2)$ where $\mathcal{E} = [\,]$. Thus $N = M$ there is no possible basic reduction from $V_1(M_2)$ as M_2 is not a value.
  - $\mathcal{E}[V_1] = V_1(M_2)$ where $\mathcal{E} = \mathcal{E}'(M_2) = [\,](M_2)$. This similar to the case 1.1.
  - $\mathcal{E}[M_2] = V_1(M_2)$ where $\mathcal{E} = V_1\Big(\mathcal{E}'\Big) = V_1([\,])$. Applying **IH** and M_2 is not a value then there must be a unique $\mathcal{F}$ and N_1 such that $M_2 = \mathcal{F}[N_1]$ and $N_1 \longmapsto N_1'$ by a basic reduction. $M_2 = \mathcal{F}[N_1]$ and $\mathcal{F}' = V_1\Big(\mathcal{F}\Big)$ thus $V_1\Big(\mathcal{F}[N_1]\Big) = \mathcal{F}'[N_1] = V_1(M_2)$ and by $N_1 \longmapsto N_1'$ this is complete.

  Only one of these cases works, thus unique.
- M_1 is not a value then for $M = \mathcal{E}[N]$ there are two cases.
  - $\mathcal{E}[M_1(M_2)] = M_1(M_2)$ where $\mathcal{E} = [\,]$. Thus $N = M$ there is no possible basic reduction from $M_1(M_2)$ has M_1 is not a value.
  - $\mathcal{E}[M_1] = M_1(M_2)$ where $\mathcal{E} = \mathcal{E}(M_2) = [\,](M_2)$. This similar to the case 2.3

  Only one of these cases works, thus unique.

Case(LETBOX). Suppose $\cdot;\cdot \vdash M : A$ of the form

$$\frac{\cdot;\cdot \vdash M_1 : \Box A \qquad \cdot; u : A \vdash M_2 : C}{\cdot;\cdot \vdash \mathsf{let\ box}\ u \Leftarrow M_1\ \mathsf{in}\ M_2 : C}$$

One thing to note is that the form of M only allows for $\mathcal{E} = [\,]$ or $\mathcal{E} = \mathsf{let\ box}\ u \Leftarrow \mathcal{E}'\ \mathsf{in}\ N$.

- if M_1 val then by canonical forms M_1 val is of the form $\mathsf{box}\ M_1'$. If this was the case then there is only one possibility to get a reduction.
  Due to the fact that $\mathcal{E} = \mathsf{let\ box}\ u \Leftarrow \mathcal{E}'\ \mathsf{in}\ N$ does not work has M_1 val thus no further reduction. So $\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M_1'\ \mathsf{in}\ M_2 = \mathcal{E}[\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M_1'\ \mathsf{in}\ M_2]$ where $\mathcal{E} = [\,]$ and $\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M_1'\ \mathsf{in}\ M_2 \longmapsto M_2[M_1'/u]$ by D-BOXBETA.
- if M_1 val does not hold then by applying **IH** on M_1 then there must be a unique $\mathcal{E}$ and N_1 such that $M_1 = \mathcal{E}[N_1]$ and $N_1 \longmapsto N_1'$ by a basic reduction. $\mathcal{E} = [\,]$ does not work in this case has M_1 is not of the form $\mathsf{box}\ M_1'$ thus unable to apply D-BOXBETA.
  Using $M_1 = \mathcal{E}[N_1]$ and $\mathcal{E} = \mathsf{let\ box}\ u \Leftarrow \mathcal{E}\ \mathsf{in}\ N$ then $(\mathsf{let\ box}\ u \Leftarrow \mathcal{E}[N_1]\ \mathsf{in}\ N) = (\mathsf{let\ box}\ u \Leftarrow M_1\ \mathsf{in}\ N) = \mathcal{E}[N_1] = M$.

Thus only one $\mathcal{E}$ works in each case so unique.

□

Theorem 2.3.4 (Type safety)**.**

1. (Preservation) If $\Delta; \Gamma \vdash M : A$ and $M \longmapsto N$ then $\Delta; \Gamma \vdash N : A$.
2. (Progress) If $\cdot; \cdot \vdash M : A$ either M val or $M \longmapsto N$ for some N.

2.4 A distributed abstract machine

We adopt Moody's [Moo05] approach in defining the distributed abstract machine but present it in the style of the π-calculus [Mil92]. As a result, our abstract machine does not employ a store, which is in contrast to Moody's. We assume an infinite set of channels names $a, b, \dots \in \mathcal{N}$ meaning that we never run out and can always create a new channel.

$$
\begin{array}{llcll}
\text{runtime terms} & M & ::= & \dots & \text{terms} \\
 & & & ?a & \text{listen for a value from } a \\
\text{thread} & T & ::= & \langle M : a \rangle & \text{runtime term } M \text{ outputs value on channel } a \\
\text{configuration} & C & ::= & \mathbf{0} & \text{stopped} \\
 & & & T & \text{thread} \\
 & & & C_1 \mid C_2 & \text{parallel composition} \\
 & & & \nu a.\, C & \text{channel scope}
\end{array}
$$

The difference between a term and a runtime term is that runtime terms and their subterms may take the form of

$$?a$$

which means listen to channel a for a value. If a value is passed down the channel, then $?a$ would become the value passed down (there are some conditions to this, however).

The machine is structured in terms of *configurations*.

The most basic configuration is one of the form, this is commonly refer to as a thread, and a being the output channel for the thread.

$$\langle M : a \rangle$$

This part of the machine will evaluate the term M, until it becomes a value which then can be sent along channel a. Any two configurations C_1 and C_2 may be combined into

$$C_1 \mid C_2$$

The two configurations will then run in parallel.
The configuration

$$\nu a.\, C$$

declares a new channel a, with scope C. Within the configuration C, a may be used to send and receive values only to other configurations which are also in the scope of a. This property is not enforced; it comes naturally due to how $?a$ appears within the configurations (this is further explained in the caption text of the M-BoxBeta rule). Note that $\nu a.\, C$ binds a in C, so we consider it up to α-conversion. Meaning that all occurrences of a can be renamed, that is, channels of the form a and questions marks of the form $?a$. An example of this renaming is below.

By α-conversion on the channel c

$$\nu c.\, \Big(\langle \lambda u : A.\, (u)(?c) : b\rangle \;\Big|\; \langle P : c\rangle\Big) \equiv \nu e.\, \Big(\langle \lambda u : A.\, (u)(?e) : b\rangle \;\Big|\; \langle P : e\rangle\Big)$$

One thing to note is that during substitution $?a$ will get ignored thus $?b[M/b] \equiv ?b$. Apart from this, substitution is performed as normal.

Let C_1, C_2 be configuration and M, N_1, N_2 be runtime terms. Then the set of free names is defined as follows:

$$\begin{aligned} fn(?a) &= \{a\} \\ fn(\lambda x.\, N_1) &= fn(N_1) \\ fn(N_1(N_2)) &= fn(N_1) \cup fn(N_2) \\ fn(\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2) &= fn(N_1) \cup fn(N_2) \end{aligned}$$

$$\begin{aligned} fn\Big(\langle M : a\rangle\Big) &= \{a\} \cup fn(M) \\ fn(\nu c.\, C_1) &= fn(C_1) \setminus \{c\} \\ fn(C_1 \mid C_2) &= fn(C_1) \cup fn(C_2) \end{aligned}$$

2.4.1 Structural congruence

We present structural congruence, which holds some similarity to π-calculus structural congruence [MPW92a; MPW92b]. Structural congruence enables configurations that are different syntactically (however should be indistinguishable) to be equivalent. For example, it allows you to reorder threads, reassociate parallel compositions, and 'garbage-collect' threads that will no longer communicate a value.

$$\frac{}{C_1 \mid C_2 \equiv C_2 \mid C_1}$$

(a) Renders the order of configurations irrelevant by considering two configurations with differing orders as equivalent.

$$\frac{}{C \mid \mathbf{0} \equiv C}$$

(b) Simplifies a parallel composition where one of the components is stopped; this is equivalent to just the running component.

$$\frac{}{(C_1 \mid C_2) \mid C_3 \equiv C_1 \mid (C_2 \mid C_3)}$$

(c) Gives the configurations association

$$\frac{a \notin fn(C_1)}{\nu a.\, (C_1 \mid C_2) \equiv C_1 \mid \nu a.\, C_2}$$

(d) This rule bears a resemblance to scope extrusion in the π-calculus; if C_1 does not utilize a, there is no need for it to be within its scope.

$$\frac{}{\nu a.\, \langle M : a\rangle \equiv \mathbf{0}}$$

(e) This enables garbage collection. Due to a only being able to send and receive values to other configurations that are also in the scope of a (this property is further explained in the caption text of the M-BoxBeta rule) and rule d), the above can only happen if the other configurations running do not use the channel a. Thus the output on that channel does not matter as no other configuration are listening, allowing it to be equivalent to the stopped configuration.

$$\frac{}{\nu a.\, \mathbf{0} \equiv \mathbf{0}}$$

(f) A stopped configuration has no need for a channel, thus just equal to the stopped configuration.

Figure 2.5: Structural congruence rules

Examples using these rules can be found here A.4

2.4.2 Dynamics of the abstract machine

The following rules show how the abstract machine steps from configuration to configuration. Note that “abstract machine” can be used synonymously with the word configuration.

Again we will use *evaluation contexts* however, the Ns below are runtime terms and not normal terms allowing for $?a$.

$$\begin{array}{rcl} \mathcal{E} & ::= & [\,] \\ & & \mathcal{E}(N) \\ & & V\big(\mathcal{E}\big) \\ & & \mathsf{let\ box}\ u \Leftarrow \mathcal{E}\ \mathsf{in}\ N \end{array}$$

We write $\mathcal{E}[M]$ for the term that results from replacing $[\,]$ with M.
For example $\mathcal{E} = (\mathsf{let\ box}\ u \Leftarrow \mathcal{E}\ \mathsf{in}\ ?b) = (\mathsf{let\ box}\ u \Leftarrow (\lambda x : A.\, N)([\,])\ \mathsf{in}\ ?b)$
thus $\mathcal{E}[M] = (\mathsf{let\ box}\ u \Leftarrow (\lambda x : A.\, N)(M)\ \mathsf{in}\ ?b)$.

M-Beta

$$\frac{}{\Big\langle \mathcal{E}[(\lambda x : A.\, M)(V)] : a \Big\rangle \longrightarrow \Big\langle \mathcal{E}\Big[M[V/x]\Big] : a \Big\rangle}$$

(a) This rule corresponds to D-Beta

M-BoxBeta

$$\frac{}{\Big\langle \mathcal{E}[\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ M\ \mathsf{in}\ N] : a \Big\rangle \longrightarrow \nu c.\, \Bigg(\Big\langle \mathcal{E}\Big[N[?c/u]\Big] : a \Big\rangle \;\Big|\; \langle M : c \rangle\Bigg)}$$

(b) This rule corresponds to D-BoxBeta. Due to $\mathsf{box}\ M$ having a box type, it is allowed to be computed in a different location. This rule creates a new channel c and spawns a new thread with M inside that output on c. This new thread does computation on M until it becomes a value, by the computation which is done on it. Then using M-Recv rule, this value is output on channel c. Due to runtime terms only having $?a$ because of M-BoxBeta rule and it creating a new channel each time, it's impossible to have two threads that depend on each other, causing deadlock.
This also means that if a runtime term of the form $?a$ appears, the configuration it appears in must be in the scope of a and the thread that outputs on channel a. One caveat is this does not apply to the thread at the start of the computation, as this thread was not spawned by M-BoxBeta.

M-Par-1

$$\frac{C_1 \longrightarrow C_1'}{C_1 \mid C_2 \longrightarrow C_1' \mid C_2}$$

(c) Steps on configurations can be done in parallel

M-Nu-1

$$\frac{C \longrightarrow C'}{\nu a.\, C \longrightarrow \nu a.\, C'}$$

(d) Computation can step into channels scopes

M-Recv

$$\frac{}{\Big\langle \mathcal{E}[?a] : c \Big\rangle \;\Big|\; \langle V : a \rangle \longrightarrow \Big\langle \mathcal{E}[V] : c \Big\rangle \;\Big|\; \langle V : a \rangle}$$

(e) This rule is responsible in for values being sent back along their respective channels.

Figure 2.6: Dynamics of abstract machine

Notice how the evaluation context does not have $\mathsf{box}\ \mathcal{E}$ included in its definition. This is due to the fact that, for well typed terms, $\mathsf{box}\ P$ should only appear surrounded by a letbox, thus the moment there is $\mathsf{box}\ P$ then we can use M-BoxBeta and before this point $\mathcal{E}$ allows for computation to be done inside the let box (via $\mathsf{let\ box}\ u \Leftarrow \mathcal{E}\ \mathsf{in}\ N$). There is also no definition $\mathcal{E} = \mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ \mathcal{E}'$ which is distinct from Davies and Pfenning [DP01] cong_letbox2. However, this is not a concern as if a reduction by cong_letbox2 is possible, say on term $\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2$ which is inside of a thread; then it must also be possible for a reduction by cong_letbox2 on the term $\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2'$ where N_2' is N_2 with u being replaced with a term. Thus if this reduction is possible, we can use M-BoxBeta, spawning a new thread with the term $N_2[?c/u]$ inside of it and do the reduction on this thread as normal.

As mentioned above, Moody's configurations employ a store, while ours do not; however, there are some similarities between ours and Moody's. Moody's configurations take the form $\langle l : M\rangle \triangleleft C$ where l represents a unique label for the configuration, and M is a term. This bears a resemblance to a thread in our calculus when considering the label as the output channel of the thread. Both labels and output channels are unique; thus, this is a fair comparison.
Moody's configuration has some similar rules to ours also. The "letbox" rule behaves the same as ours, the "syncr" rule which has the same function as M-Recv, and the "app" would be M-Beta in our calculus. None of the above Moody's rules uses the store; thus, they are almost identical to ours. However, there are some rules our configuration to not have that are in Moody's; this is due to the $\diamond$ in ours being omitted. Another difference is that Moody's configuration can not move around; they are fixed in place, distinct from our configurations which are able to via structural congruence.

2.5 How to evaluate a term

Suppose you would like to evaluate a closed term $\cdot;\cdot \vdash M : \mathsf{Num}$. Pick a channel name as a distinguished output channel, say a. Then start the machine in the configuration.

$$\langle M : a\rangle$$

Hopefully by using the dynamics rules the eventually end state is

$$\langle V : a\rangle$$

With V having type Num. A example is below.

$$\mathcal{E} = ([\,])(\mathsf{let\ box}\ u \Leftarrow \overline{2}\ \mathsf{in}\ u) \qquad A \equiv \mathsf{Num}$$

$$\langle((\lambda x : A.\, \lambda y : A.\, \mathsf{plus}(x;y))(\overline{6}))(\mathsf{let\ box}\ u \Leftarrow \overline{2}\ \mathsf{in}\ u) : a\rangle \equiv \left\langle \mathcal{E}[(\lambda x : A.\, \lambda y : A.\, \mathsf{plus}(x;y))(\overline{6})] : a \right\rangle$$

by M-Beta

$$\longrightarrow \left\langle \mathcal{E}\left[\lambda y : A.\, \mathsf{plus}(x;y)[\overline{6}/x]\right] : a \right\rangle$$

$$\equiv \left\langle \mathcal{E}[\lambda y : A.\, \mathsf{plus}(\overline{6};y)] : a \right\rangle$$

$$\equiv \langle(\lambda y : A.\, \mathsf{plus}(\overline{6};y))(\mathsf{let\ box}\ u \Leftarrow \overline{2}\ \mathsf{in}\ u) : a\rangle$$

$$\mathcal{E} = (\lambda y : A.\, \mathsf{plus}(\overline{6};y))([\,])$$

by M-BoxBeta

$$\longrightarrow \nu b.\left(\left\langle \mathcal{E}\left[u[?b/u]\right] : a \right\rangle \,\middle|\, \langle \overline{2} : b\rangle\right)$$

$$\equiv \nu b.\left(\left\langle \mathcal{E}[?b] : a \right\rangle \,\middle|\, \langle \overline{2} : b\rangle\right)$$

by M-Recv

$$\longrightarrow \nu b.\left(\left\langle \mathcal{E}[\overline{2}] : a \right\rangle \,\middle|\, \langle \overline{2} : b\rangle\right)$$

$$\equiv \left\langle \mathcal{E}[\overline{2}] : a \right\rangle \,\middle|\, \nu b.\left(\langle \overline{2} : b\rangle\right)$$

$$\equiv \left\langle \mathcal{E}[\overline{2}] : a \right\rangle \,\middle|\, \mathbf{0}$$

$$\equiv \left\langle \mathcal{E}[\overline{2}] : a \right\rangle$$

$$\equiv \langle(\lambda y : A.\, \mathsf{plus}(\overline{6};y))(\overline{2}) : a\rangle$$

$$\mathcal{E} = [\,]$$

by M-Beta

$$\longrightarrow \left\langle \mathsf{plus}(x;y)[\overline{2}/y] : a \right\rangle$$

$$\equiv \langle \mathsf{plus}(\overline{6};\overline{2}) : a\rangle$$

by Plus

$$\longrightarrow \langle \overline{8} : a\rangle$$

Figure 2.7: Example of dynamics of the abstract machine.

More examples found here A.4

Chapter 3

Simulation

In Moody's paper, he introduces a dual context calculus and distributed abstract machine; however, he does not provide proof demonstrating that they are equivalent in the context of computation. In this work, we present a proof of computational equivalence by employing a bi-simulation method to our dual context and our distributed abstract machine. This is achieved by defining a relationship between them and showing that a step on one can be simulated on the other with the relationship holding.
Prior to defining the relationship, we will now formally describe some preliminary definitions that are used within the relationship.

3.1 Definitions

Sequential Substitution:

σ is of the form : $\sigma \ ::= \ . \mid M/x, \ \sigma'$ where $x \notin dom(\sigma')$
Given a term N and a σ we define a term

$$N|\sigma\rangle$$

by

$$N|\cdot\rangle \ := \ N$$
$$N|M/x, \ \sigma\rangle \ := \ (N[M/x])|\sigma\rangle$$

The $x \notin dom(\sigma)$ ensures that each variable corresponds to subbing only one term; thus, $\sigma = P/x, \ Q/y, \ N/x$ is not valid. This is beneficial later in defining configuration substitution, where each M/x represents an individual thread, meaning channels with identical names cannot produce different values. Additionally, if $N|\sigma\rangle$ is closed then $\forall \ M/x \ \in \ \sigma. \ x \ \notin \ \text{fv}(M)$, if this were the case, then $N|\sigma\rangle$ would not be closed as by the above no other substitution could substitute for x.[1]

An example is below:

$$\begin{aligned} \sigma &= y/z, s/x, t/y \\ N|\sigma\rangle &= z(xy)|y/z, s/x, t/y\rangle \\ &= y(xy)|s/x, t/y\rangle \\ &= y(sy)|t/y\rangle \\ &= t(st) \end{aligned}$$

[1] $\forall \ M/x \ \in \ \sigma. \ x \ \notin \ \text{fv}(M)$ reads as, for all M/x in σ (recall σ is of the form $P/y, \ Q/z, \ \ldots$), x is not a free value of M

Question Substitution:

We make it so that there is a distinguished channel a_x for each variable x

Given σ, we define a term

$$N^{?\sigma}$$

by

$$x^{?\sigma} \;:=\; \begin{cases} ?a_x & \text{if } x \in \mathit{dom}(\sigma) \\ x & \text{otherwise} \end{cases}$$

$$(N_1(N_2))^{?\sigma} \;:=\; {N_1}^{?\sigma}({N_2}^{?\sigma})$$

Rename u, if $u \in \mathit{dom}(\sigma)$ for the below rules

$$(\lambda u : A.\, N_1)^{?\sigma} \;:=\; \lambda u : A.\, {N_1}^{?\sigma}$$

When renaming the below, only rename in N_2 as this is where u is bound

$$(\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2)^{?\sigma} \;:=\; \mathsf{let\ box}\ u \Leftarrow {N_1}^{?\sigma}\ \mathsf{in}\ {N_2}^{?\sigma}$$

An example is below:

$$\begin{aligned}
\sigma :&\quad w/z, s/x, t/y \\
N^{?\sigma} &= (\mathsf{let\ box}\ x \Leftarrow \lambda y : \mathsf{Num}.\, x\ \mathsf{in}\ z(x))^{?\sigma} \\
&= \mathsf{let\ box}\ s \Leftarrow \lambda y : \mathsf{Num}.\, x^{?\sigma}\ \mathsf{in}\ (z(s))^{?\sigma} \quad \text{(rename } x \text{ to } s \text{ in } N_2) \\
&= \mathsf{let\ box}\ s \Leftarrow \lambda y : \mathsf{Num}.\, x^{?\sigma}\ \mathsf{in}\ z^{?\sigma}(s^{?\sigma}) \\
&= \mathsf{let\ box}\ s \Leftarrow \lambda y : \mathsf{Num}.\, ?a_x\ \mathsf{in}\ ?a_z(s)
\end{aligned}$$

Configuration Substitution

Given σ, we define a process

$$[\![\sigma]\!]$$

by

$$[\![\cdot]\!] \;:=\; \mathbf{0}$$

$$[\![M/x,\ \sigma]\!] \;:=\; \langle M^{?\sigma} : a_x \rangle \;\Big|\; [\![\sigma]\!]$$

The definitions below allows us to manipulating the configuration substitutions

$$[\![\cdot]\!]^{?\beta} \;:=\; \mathbf{0}$$

$$[\![M/x,\ \alpha]\!]^{?\beta} \;:=\; \langle M^{?(\alpha,\ \beta)} : a_x \rangle \;\Big|\; [\![\alpha]\!]^{?\beta}$$

$$[\![\alpha,\ \beta]\!] \;:=\; [\![\alpha]\!]^{?\beta} \mid [\![\beta]\!]$$

One may regard configuration substitution as the configuration based adaptation of sequential substitution. In the definition of configuration substitution, there is $M^{?\sigma}$; this is because terms that are subbed in by σ can have free variables in them, which due to sequences of substitution will be resolved, given that $N|\sigma\rangle$ is closed, by the free variables appearing later down the sequence. This also means that each term that gets subbed in via $N|\sigma\rangle$ can not have free variables that only appear earlier in the sequence. Meaning that $\langle M^{?\sigma} : a_x \rangle$ in the configuration $\langle M^{?\sigma} : a_x \rangle \;\Big|\; [\![\gamma]\!]$ can only question variables in γ.

Examples are below:

$$\begin{aligned}
\sigma &= M/z, P/y, Q/w \\
\sigma' &= P/y, Q/w \\
[\![\sigma]\!] &= \langle M^{?\sigma'} : a_z \rangle \;\Big|\; [\![P/y, Q/w]\!] \\
&= \langle M^{?\sigma'} : a_z \rangle \;\Big|\; \langle P^{?(Q/w)} : a_y \rangle \;\Big|\; [\![Q/w]\!]
\end{aligned}$$

$$\begin{aligned}
&\text{Using } Q/w \;=\; Q/w \;\cdot \\
&= \langle M^{?\sigma'} : a_z\rangle \,\Big|\, \langle P^{?(Q/w)} : a_y\rangle \,\Big|\, \langle Q^{?(\cdot)} : a_w\rangle \,\Big|\, [\![\cdot]\!] \\
&= \langle M^{?\sigma'} : a_z\rangle \,\Big|\, \langle P^{?(Q/w)} : a_y\rangle \,\Big|\, \langle Q^{?(\cdot)} : a_w\rangle \,\Big|\, \mathbf{0} \\
&\equiv \langle M^{?\sigma'} : a_z\rangle \,\Big|\, \langle P^{?(Q/w)} : a_y\rangle \,\Big|\, \langle Q^{?(\cdot)} : a_w\rangle \\
&= \langle M^{?\sigma'} : a_z\rangle \,\Big|\, \langle P^{?(Q/w)} : a_y\rangle \,\Big|\, \langle Q : a_w\rangle
\end{aligned}$$

$$\begin{aligned}
\sigma :&\quad N/z, M/x, Q/w, P/s \\
\sigma' :&\quad Q/w, P/s \\
[\![\sigma]\!] &= [\![M/x]\!]^{?\sigma'} \mid [\![Q/w,\ P/s]\!] \\
&= \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, [\![M/x]\!]^{?\sigma'} \,\Big|\, [\![Q/w,\ P/s]\!] \\
&\text{Using } M/x \;=\; M/x \;\cdot \\
&= \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, \langle M^{?\sigma',\cdot} : a_x\rangle \,\Big|\, [\![\cdot]\!]^{?\sigma'} \,\Big|\, [\![Q/w,\ P/s]\!] \\
&= \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, \langle M^{?\sigma',\cdot} : a_x\rangle \,\Big|\, \mathbf{0} \,\Big|\, [\![Q/w,\ P/s]\!] \\
&\equiv \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, \langle M^{?\sigma',\cdot} : a_x\rangle \,\Big|\, [\![Q/w,\ P/s]\!] \\
&\equiv \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, \langle M^{?\sigma',\cdot} : a_x\rangle \,\Big|\, \langle Q^{?(P/s)} : a_w\rangle \,\Big|\, [\![P/s]\!] \\
&\text{Using } P/s \;=\; P/s,\ \cdot \\
&= \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, \langle M^{?\sigma',\cdot} : a_x\rangle \,\Big|\, \langle Q^{?(P/s)} : a_w\rangle \,\Big|\, \langle P^{?\cdot} : a_s\rangle \,\Big|\, [\![\cdot]\!] \\
&= \langle N^{?(M/x,\ \sigma')} : a_z\rangle \,\Big|\, \langle M^{?\sigma',\cdot} : a_x\rangle \,\Big|\, \langle Q^{?(P/s)} : a_w\rangle \,\Big|\, \langle P : a_s\rangle
\end{aligned}$$

We define a relationship R. First, there is a term N (that could have free variables in), which gets applied a sequential substitution. On the other side of the relationship, there is a thread with N inside of it, and each substitution in σ is in its own thread. The term $N|\sigma\rangle$ and the configuration described above would be related under the relationship R. We now define this formally.

Relationship (R):

We define $\nu\sigma\ ::=\ .\mid \nu x.\nu\sigma'$ where $\sigma\ ::=\ .\mid M/x,\ \sigma'$

Given $\mathrm{fv}(N|\sigma\rangle) = \emptyset$
We say that M and C are related ($M\ R\ C$), if they are of the form

$$\begin{aligned}
M &\equiv\ N|\sigma\rangle \\
C &\equiv\ \nu\sigma.\Big(\langle N^{?\sigma} : a\rangle \,\Big|\, [\![\sigma]\!]\Big)
\end{aligned}$$

Thus $N|\sigma\rangle\ R\ \nu\sigma.\Big(\langle N^{?\sigma} : a\rangle \,\Big|\, [\![\sigma]\!]\Big)$

3.2 Proof

The rationale for the below proof is to demonstrate that the distributed abstract machine is computationally equivalent to the dual-context calculus. The dual-context calculus cannot livelock or deadlock as its computation is serial; thus, proofing the equivalence would consequently mean that the abstract machine also cannot do this, implying that if a language was to be built on this theory, then no complications that come with non-serial programming can occur. As they would be handled by the compiler when checking if the program is "valid".

The proof is broken down by each type of reduction on the dual-context calculus, ending with the D-Eval rule. We will now outline the cases.

- For D-Beta, we first apply M-Recv repeatedly until M-Beta reduction becomes possible. Subsequently, we perform the M-Beta reduction, demonstrating that the resulting term is closed, allowing for the relationship to be applied to it.
- D-BoxBeta, we apply the M-BoxBeta reduction. Then define a new sigma allowing the resulting term to be rewritten; we show this is closed and the relationship holds.
- D-Eval, we apply the induction hypothesis. Then on the resulting configuration, we use lemma 3.2.5, ending up with a configuration with the relationship still holding

The main challenges in the proof stem from handling cases where N is just a variable before the sequence substitution is applied and the M-Recv rule altering the abstract machine when used.

We now define the lemmata used in the proof.

Lemma 3.2.1. $(\mathcal{E}[N])^{?\sigma} = \mathcal{E}^{?\sigma}[N^{?\sigma}]$

Proof. By induction on $\mathcal{E}$

Case(HOLE). Suppose $\mathcal{E}$ of the form $[\,]$

$$N'^{?\sigma} = N'^{?\sigma}$$

Case(APP1). Suppose $\mathcal{E}$ of the form $\mathcal{E}'(P)$

$$\begin{aligned}(\mathcal{E}[N])^{?\sigma} &= ((\mathcal{E}'(P))[N])^{?\sigma}\\ &= ((\mathcal{E}'[N])(P))^{?\sigma}\\ &= (\mathcal{E}'[N])^{?\sigma}(P^{?\sigma})\\ &\text{By induction hypothesis}\\ &= \left(\mathcal{E}'^{?\sigma}[N^{?\sigma}]\right)(P^{?\sigma})\\ &= (\mathcal{E}'^{?\sigma}(P^{?\sigma}))[N^{?\sigma}]\\ &= \mathcal{E}^{?\sigma}[N^{?\sigma}]\end{aligned}$$

Case(APP2). Similar to APP1

Case(LETBOX). Suppose $\mathcal{E}$ of the form $\mathsf{let\ box}\ u \Leftarrow \mathcal{E}'\ \mathsf{in}\ P$

$$\begin{aligned}
\mathcal{E}[N]^{?\sigma} &= ((\mathsf{let\ box}\ u \Leftarrow \mathcal{E}'\ \mathsf{in}\ P)[N])^{?\sigma} \\
&= (\mathsf{let\ box}\ u \Leftarrow (\mathcal{E}'[N])\ \mathsf{in}\ P)^{?\sigma} \\
&\text{Rename } u\text{, if } u \in \mathit{dom}(\sigma) \\
&= \mathsf{let\ box}\ u \Leftarrow (\mathcal{E}'[N])^{?\sigma}\ \mathsf{in}\ P^{?\sigma} \\
&\text{By induction hypothesis} \\
&= \mathsf{let\ box}\ u \Leftarrow \mathcal{E}'^{?\sigma}[N^{?\sigma}]\ \mathsf{in}\ P^{?\sigma} \\
&= (\mathsf{let\ box}\ u \Leftarrow \mathcal{E}'^{?\sigma}\ \mathsf{in}\ \mathcal{P}^{?\sigma})[N^{?\sigma}] \\
&= \mathcal{E}^{?\sigma}[N^{?\sigma}]
\end{aligned}$$

□

The below lemma is very similar to the above lemma

Lemma 3.2.2. $\mathcal{E}[N]|\sigma\rangle = \mathcal{E}|\sigma\rangle[N|\sigma\rangle]$

Proof. By induction on $\mathcal{E}$

CASE(HOLE). Suppose $\mathcal{E}$ of the form $[\,]$

$$N'|\sigma\rangle = N'|\sigma\rangle$$

CASE(APP1). Suppose $\mathcal{E}$ of the form $\mathcal{E}'(P)$

$$\begin{aligned}
\mathcal{E}[N]|\sigma\rangle &= ((\mathcal{E}'(P))[N])|\sigma\rangle \\
&= ((\mathcal{E}'[N])(P))|\sigma\rangle \\
&= (\mathcal{E}'[N])|\sigma\rangle(P|\sigma\rangle) \\
&\text{By induction hypothesis} \\
&= \Big(\mathcal{E}'|\sigma\rangle[N|\sigma\rangle]\Big)(P|\sigma\rangle) \\
&= (\mathcal{E}'|\sigma\rangle(P|\sigma\rangle))[N|\sigma\rangle] \\
&= \mathcal{E}|\sigma\rangle[N|\sigma\rangle]
\end{aligned}$$

CASE(APP2). Similar to APP1

CASE(LETBOX). Suppose $\mathcal{E}$ of the form $\mathsf{let\ box}\ u \Leftarrow \mathcal{E}'\ \mathsf{in}\ P$

$$\begin{aligned}
\mathcal{E}[N]|\sigma\rangle &= ((\mathsf{let\ box}\ u \Leftarrow \mathcal{E}'\ \mathsf{in}\ P)[N])|\sigma\rangle \\
&= (\mathsf{let\ box}\ u \Leftarrow (\mathcal{E}'[N])\ \mathsf{in}\ P)|\sigma\rangle \\
&\text{Renaming } u\text{, if } u \in \mathit{dom}(\sigma) \\
&= \mathsf{let\ box}\ u \Leftarrow (\mathcal{E}'[N])|\sigma\rangle\ \mathsf{in}\ P|\sigma\rangle \\
&\text{By induction hypothesis} \\
&= \mathsf{let\ box}\ u \Leftarrow \mathcal{E}'|\sigma\rangle[N|\sigma\rangle]\ \mathsf{in}\ P|\sigma\rangle \\
&= (\mathsf{let\ box}\ u \Leftarrow \mathcal{E}'|\sigma\rangle\ \mathsf{in}\ \mathcal{P}|\sigma\rangle)[N|\sigma\rangle] \\
&= \mathcal{E}|\sigma\rangle[N|\sigma\rangle]
\end{aligned}$$

□

Lemma 3.2.3. If $u \notin \mathit{dom}(\sigma)$ and $\forall\ M/x\ \in\ \sigma.\ u\ \notin\ \mathrm{fv}(M)$. Then $(P[Q/u])|\sigma\rangle \equiv P|\sigma\rangle[Q|\sigma\rangle/u]$

Proof. By induction on σ

CASE($\sigma = \cdot$). Suppose σ of the form $\cdot$

$$(P[Q/u])|\cdot\rangle = P[Q/u] = P|\cdot\rangle[Q|\cdot\rangle/u]$$

CASE($\sigma = M/x,\ \sigma'$). Suppose σ of the form $M/x, \sigma'$

$$\begin{aligned}(P[Q/u])|M/x,\sigma'\rangle &= ((P[Q/u])[M/x])|\sigma'\rangle\\ &\text{As } x \neq u \text{ and } u \notin \text{fv}(M)\\ &= P[M/x]\Big[Q[M/x]/u\Big]|\sigma'\rangle\\ &\text{By induction hypothesis}\\ &= P[M/x]|\sigma'\rangle\Big[Q[M/x]|\sigma'\rangle/u\Big]\\ &= P|\sigma\rangle[Q|\sigma\rangle/u]\end{aligned}$$

□

Lemma 3.2.4. If $x|\sigma\rangle\ =\ M$. Then σ is of the form $\alpha,\ M'/z_0,\ \beta$ where z_0 is a variable

$$\begin{aligned}M &\equiv N_1(N_2) & M' &\equiv N_1'(N_2')\\ &\equiv \mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2 & &\equiv \mathsf{let\ box}\ u \Leftarrow N_1'\ \mathsf{in}\ N_2'\\ &\equiv \lambda x : A.\, N_1 & &\equiv \lambda x : A.\, N_1'\\ &\equiv \mathsf{box}\ N_1 & &\equiv \mathsf{box}\ N_1'\end{aligned}$$

With each row representing a possible M and M' pair.
$M'|\beta\rangle = M$ and $x|\alpha\rangle\ =\ z_0$ (where z_0 is the last variable which this is true for).

Proof. By contradiction

CASE(APP). Suppose M is of the form $N_1(N_2)$, thus $M'\ =\ N_1'(N_2')$

It is always true that $\sigma\ =\ \alpha, ...$ as α can be empty ($\alpha = \cdot$).
$\therefore\ x|\sigma\rangle\ =\ z_0$ holds
There must be a substitution on z_0, which is not a variable, as M is not a variable.
Consider the next substitution on z_0; it could be another variable substitution (for example z_1/z_0) however this would been include into α.
Thus the term substituted after α is not a variable.
Say this term is not of the form $(N_1'(N_2'))$. The next substitution will only apply to the subterms of the above term (N_n'), thus leaving the top-level shape of the term intact.
Thus M is not of the form $N_1(N_2)$ which is a contradiction, thus the next sub after α is $(N_1(N_2))/z_0$
Recall $M'|\beta\rangle = M$

Thus $\sigma = \alpha, (N_1(N_2))/z_0, \beta$

CASE(LAM). Similar to APP

CASE(LETBOX). Similar to APP

CASE(BOX). Similar to APP

□

The above lemma demonstrates that, given the outermost rule of M
(for example $(\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2)(\lambda x : A.\, N_3)$ would be the APP rule) this rule must be subbed into x first, ignoring pointless substitution, which maps variables to other variables.

Lemma 3.2.5. If $\nu\sigma.\left(\langle N : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma'.\left(\langle N' : a\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$
Then $\nu\sigma.\left(\left\langle \mathcal{E}[N] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma'.\left(\left\langle \mathcal{E}[N] : a\right\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$

Proof. By exhaustion on $\nu\sigma.\left(\langle N : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma'.\left(\langle N' : a\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$

Case(σ). Suppose the reduction happens in $\llbracket\sigma\rrbracket$. This case is trivially true.

Case(M-BETA). Suppose $\nu\sigma.\left(\langle N : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma'.\left(\langle N' : a\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$ of the form

$$\frac{}{\nu\sigma.\left(\left\langle \mathcal{E}'[(\lambda x : A.\, M)(V)] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma.\left(\left\langle \mathcal{E}'\left[M[V/x]\right] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)}$$

let $\mathcal{E}'' = \mathcal{E}\left[\mathcal{E}'[\,]\right]$
$\therefore \nu\sigma.\left(\left\langle \mathcal{E}''[(\lambda x : A.\, M)(V)] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma.\left(\left\langle \mathcal{E}''\left[M[V/x]\right] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$
$\therefore \nu\sigma.\left(\left\langle \mathcal{E}\left[\mathcal{E}'[(\lambda x : A.\, M)(V)]\right] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma.\left(\left\langle \mathcal{E}\left[\mathcal{E}'\left[M[V/x]\right]\right] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$
$\therefore \nu\sigma.\left(\left\langle \mathcal{E}[N] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow \nu\sigma.\left(\left\langle \mathcal{E}[N'] : a\right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$

Case(M-BETABOX). Similar to M-BETA case

Case(M-RECV). Similar to M-BETA case

□

Below is the proof.

Conjecture 3.2.6. If $M \longmapsto M'$ and $M\ R\ C$ Then $C \longrightarrow^* C'$ with $M'\ R\ C'$.

Proof. By induction on $M \longmapsto M'$

Case(D-BETA). Suppose $M \longmapsto M'$ of the form and $M\ R\ C$

$$\frac{\textsc{D-Beta}}{(\lambda x : A.\, N_1)(V_1) \longmapsto N_1[V_1/x]}$$

By inversion on relationship (R)
$\therefore M = N|\sigma\rangle$ and M is closed
$\therefore V_1$ is closed and N_1 is closed apart from x
We will assume $x \ \notin\ dom(\sigma)$ and $\forall\ M/y\ \in\ \sigma.\ x\ \notin\ \mathrm{fv}(M)$ as if this was not the case we can rename x so it is.

N can take multiple forms: $x,\ N_1'(N_2'),\ (\lambda x : A.\, N_1')(N_2')$, where N_1' could equal N_1 same with N_2'.

- $N = (\lambda x : A.\, N_1')(N_2')$

  $M = ((\lambda x : A.\, N_1')(N_2'))|\sigma\rangle$
  by $x\ \notin\ dom(\sigma)$
  $\therefore N_1'|\sigma\rangle \ = \ N_1$, $N_2'|\sigma\rangle \ = \ V_1$
  $\therefore N_2'|\sigma\rangle$ is closed and $N_1'|\sigma\rangle$ is closed apart from x

  $C = \nu\sigma.\left(\langle((\lambda x : A.\, N_1')(N_2'))^{?\sigma} : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$
  by $x\ \notin\ dom(\sigma)$
  $= \nu\sigma.\left(\langle(\lambda x : A.\, N_1'^{?\sigma})(N_2'^{?\sigma}) : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$

  N_2' could not be a value, however due to $N_2'|\sigma\rangle \ = \ V_1$ along the way applying the sequential substitution on N_2', a value must be formed. By Val-Num, Val-Lam and Val-Box this value must be of the form $\overline{n}$ or $\lambda x : A.\, M$ or $\mathsf{box}\ M$. By lemma 3.2.4, N_2' is either of the above form or a variable, where some sequence of variable substitution chained together could occur eventually subbing in a value V_1'. For example z_0/N_2', z_1/z_0, z_2/z_1, $\mathsf{box}\ P/z_2$, where z_n are variables and V_1' in this case being $\mathsf{box}\ P$.

We say the variables in this chain are z_0 to z_n.

$\therefore \sigma$ has two parts $\alpha,\ \beta$, where α is the part of σ that contains the variable substitution chain for N_2' making it into V_1', with the last substitution in α being of the form V_1'/z_n and β being the rest of σ. α will mostly likely contain other substitutions as well as ones that sub into N_2'.

If N_2' is already a value then alpha would be $\alpha = \cdot$ and $\beta = \sigma$

$\therefore \sigma = \alpha,\ \beta$ where $N_2'|\alpha\rangle = V_1'$ and $V_1'|\beta\rangle = V_1$

$= \nu\sigma.\, \Big(\langle(\lambda x : A.\, N_1'^{?\sigma})(N_2'^{?\sigma}) : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\beta} \;\Big|\; \llbracket\beta\rrbracket\Big)$

We now apply M-Recv, zero to n times depending on the size of the variable substitution chain, passing the value $V_1'^{?\beta}$ back.[2] When passing it through the substitution variable chain, each $?a_{z_n}$ gets revolved with $V_1'^{?\beta}$

$\longrightarrow^* \nu\sigma.\, \Big(\langle(\lambda x : A.\, N_1'^{?\sigma})(V_1'^{?\beta}) : a\rangle \;\Big|\; \llbracket\alpha'\rrbracket^{?\beta} \;\Big|\; \llbracket\beta\rrbracket\Big)$

let $\sigma' = \alpha',\ \beta$

$dom(\sigma) = dom(\sigma')$ due to the fact that the only difference is that the variables in the substitution chain now sub in a different term (V_1').

Thus for all P, $P^{?\sigma} = P^{?\sigma'}$ and $\nu\sigma.\, C = \nu\sigma'.\, C$.

As the sequential substitution can at most sub one term for each variable and β subs in the variables in V_1' making it closed, then $V_1'|\beta\rangle = V_1'|\omega\rangle$ where ω contains β.

Thus $V_1'|\beta\rangle = V_1'|\sigma'\rangle$ and $V_1'^{?\beta} = V_1'^{?\sigma'}$.

$= \nu\sigma'.\, \Big(\langle(\lambda x : A.\, N_1'^{?\sigma'})(V_1'^{?\sigma'}) : a\rangle \;\Big|\; \llbracket\alpha'\rrbracket^{?\beta} \;\Big|\; \llbracket\beta\rrbracket\Big)$

$= \nu\sigma'.\, \Big(\langle(\lambda x : A.\, N_1'^{?\sigma'})(V_1'^{?\sigma'}) : a\rangle \;\Big|\; \llbracket\sigma'\rrbracket\Big)$

by the M-Beta rule

$\longrightarrow \nu\sigma'.\, \Big(\Big\langle N_1'^{?\sigma'}[V_1'^{?\sigma'}/x] : a\Big\rangle \;\Big|\; \llbracket\sigma'\rrbracket\Big)$

by $x \notin dom(\sigma)$ and $dom(\sigma) = dom(\sigma')$

$= \nu\sigma'.\, \Big(\Big\langle (N_1'[V_1'/x])^{?\sigma'} : a\Big\rangle \;\Big|\; \llbracket\sigma'\rrbracket\Big) \;=\; C'$

Due to V_1 being closed and $V_1 = V_1'|\beta\rangle$, $V_1'|\beta\rangle = V_1'|\sigma'\rangle$.

Thus $V_1'|\sigma'\rangle$ is closed.

Recall $N_1'|\sigma\rangle$ is closed apart from x where $x \notin dom(\sigma)$ and thus $x \notin dom(\sigma')$.

Observe that, when applying $N_1'|\sigma\rangle$, if a variable appears which is in the variable substitution chain (z_0 to z_n), then for $N_1'|\sigma\rangle$ to be closed, it must follow the rest of the chain. Now considering $N_1'|\sigma'\rangle$ instead, because of α' rather than going along the chain, it immediately goes to the end of the chain (subbing in V_1'), and from the above $V_1'|\beta\rangle = V_1'|\omega\rangle$ (where ω contains β), then subbing in V_1' early does not affect the final term.

Thus $N_1'|\sigma\rangle = N_1'|\sigma'\rangle$

By the above, therefore $N_1'|\sigma'\rangle[V_1'|\sigma'\rangle/x]$ is closed.

As $x \notin dom(\sigma)$, $\forall\ M/y \in \sigma.\ x \notin \mathrm{fv}(M)$ and $dom(\sigma) = dom(\sigma')$.

We can apply the lemma 3.2.3

$N_1'|\sigma'\rangle[V_1'|\sigma'\rangle/x] \;=\; (N_1'[V_1'/x])|\sigma'\rangle$

$\therefore\ (N_1'[V_1'/x])|\sigma'\rangle$ is closed.

Thus matching the condition for the relationship R

$\therefore\ (N_1'[V_1'/x])|\sigma'\rangle \;R\; \nu\sigma.\, \Big(\Big\langle (N_1'[V_1'/x])^{?\sigma'} : a\Big\rangle \;\Big|\; \llbracket\sigma'\rrbracket\Big)$

$$\begin{aligned}(N_1'[V_1'/x])|\sigma'\rangle &= N_1'|\sigma'\rangle[V_1'|\sigma'\rangle/x] \\ &= N_1'|\sigma\rangle[V_1'|\beta\rangle/x]\end{aligned}$$

[2] $?\beta$ comes from the definition of the configuration substitution and the last entry of α being of the form V_1'/z_n

$$\begin{aligned} &= N_1[V_1/x] \\ &= M' \end{aligned}$$

$\therefore\ M'\ R\ C'$

- $N = x$

  $M = x|\sigma\rangle$
  by lemma 3.2.4
  $\therefore \sigma\ =\ \alpha,\ (N_1'(N_2'))/z_0,\ \beta$
  N_1' is either of the form $\lambda x : A.\, N_1''$ or is a variable.
  We now assume that N_1' is a variable, as it is clear this carries over to the case where it is of the form $\lambda x : A.\, N_1''$. We rename N_1' to the variable z_1 to reflect the above.
  $\therefore \sigma\ =\ \alpha,\ (z_1(N_2'))/z_0,\ \beta$ where $x|\alpha\rangle = z_0,\ z_1|\beta\rangle = \lambda x : A.\, N_1$ and $N_2'|\beta\rangle = N_2$
  $\therefore x|\alpha,\ (z_1(N_2'))/z_0,\ \beta\rangle = z_0|(z_1(N_2'))/z_0,\ \beta\rangle = (z_1(N_2'))|\beta\rangle = M$

  We now can apply lemma 3.2.4 on z_1 as $z_1|\beta\rangle = \lambda x : A.\, N_1$.
  $\therefore \beta = \alpha',\ \lambda x : A.\, N_4'/z_3, \gamma$ where $z_1|\alpha'\rangle = z_3$, and $(\lambda x : A.\, N_4')|\gamma\rangle = \lambda x : A.\, N_1$
  $\therefore z_1|\alpha',\ \lambda x : A.\, N_4'/z_3,\ \gamma\rangle = z_3|\lambda x : A.\, N_4'/z_3,\ \gamma\rangle = (\lambda x : A.\, N_4')|\gamma\rangle = \lambda x : A.\, N_1$
  Combining the above, we get
  $\therefore \sigma\ =\ \alpha,\ (z_1(N_2'))/z_0,\ \beta,\ (\lambda x : A.\, N_4')/z_3,\ \gamma$

  $$\begin{aligned} x|\alpha\rangle\ &=\ z_0,\ z_1|\beta\rangle\ =\ z_3 \\ N_2'|\beta\rangle\ &=\ N_5',\ N_5'|\lambda x : A.\, N_4'/z_3\rangle\ =\ N_6' \\ N_4'|\gamma\rangle\ &=\ N_1,\ N_6'|\gamma\rangle\ =\ V_1 \end{aligned}$$

  let $\sigma_0 = (z_1(N_2'))/z_0,\ \beta,\ (\lambda x : A.\, N_4')/z_3,\ \gamma$
  let $\sigma_1 = \beta,\ (\lambda x : A.\, N_4')/z_3,\ \gamma$
  let $\sigma_2 = (\lambda x : A.\, N_4')/z_3,\ \gamma$

  $x|\sigma\rangle\ = z_0|z_1(N_2')/z_0,\ \sigma_1\rangle = (z_1(N_2'))|\sigma_1\rangle =$
  $(z_3(N_5'))|\lambda x : A.\, N_4'/z_3,\ \gamma\rangle = (\lambda x : A.\, N_4'(N_6'))|\gamma\rangle = M$

  Recall M is closed, $N_2'|\sigma_1\rangle = V_1$ and $N_4'|\gamma\rangle$
  $\therefore N_2'|\sigma_1\rangle$ is closed and $N_4'|\gamma\rangle$ is closed apart from x

  $C = \nu\sigma.\left(\langle x^{?\sigma} : a\rangle \;\middle|\; \llbracket\sigma\rrbracket\right)$
  $= \nu\sigma.\left(\langle x^{?\sigma} : a\rangle \;\middle|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\middle|\; \langle(z_1(N_2'))^{?\sigma_1} : a_{z_0}\rangle \;\middle|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\middle|\; \langle(\lambda x : A.\, N_4')^{?\gamma} : a_{z_3}\rangle \;\middle|\; \llbracket\gamma\rrbracket\right)$
  by $x\ \notin\ dom(\sigma)$
  $= \nu\sigma.\left(\langle x^{?\sigma} : a\rangle \;\middle|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\middle|\; \langle(z_1{}^{?\sigma_1}(N_2'{}^{?\sigma_1})) : a_{z_0}\rangle \;\middle|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\middle|\; \langle(\lambda x : A.\, N_4'{}^{?\gamma}) : a_{z_3}\rangle \;\middle|\; \llbracket\gamma\rrbracket\right)$

  We now repeat similar steps to the above case $(N = (\lambda x : A.\, N_1')(N_2'))$.
  Note that $z_1|\beta\rangle = z_3$, where each sub that is able to be applied is a sequence of variable substitution chained together (by lemma 3.2.4) and $\lambda x : A.\, N_4'{}^{?\gamma}$ val this allows for the M-Recv rule to be applied. β turns into β', as when passing down $\lambda x : A.\, N_4'{}^{?\gamma}$ the $?a$'s of the variables in the chain will now get resolved to $\lambda x : A.\, N_4'{}^{?\gamma}$
  $\longrightarrow^*\ \nu\sigma.\left(\langle x^{?\sigma} : a\rangle \;\middle|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\middle|\; \langle\lambda x : A.\, N_4'{}^{?\gamma}(N_2'{}^{?\sigma_1}) : a_{z_0}\rangle \;\middle|\; \llbracket\beta'\rrbracket^{?\sigma_2} \;\middle|\; \langle(\lambda x : A.\, N_4'{}^{?\gamma}) : a_{z_3}\rangle \;\middle|\; \llbracket\gamma\rrbracket\right)$

  let $\sigma_1' = \beta',\ (\lambda x : A.\, N_4')/z_3,\ \gamma$
  $dom(\sigma_1') = dom(\sigma_1)$, as the only thing that has changed is that variables in the chain, we now sub in a different term $(\lambda x : A.\, N_4')$ instead of the variable next in the chain.
  Thus for all P, $P^{?\sigma_1} = P^{?\sigma_1'}$
  As sequential substitution can at most sub one term for each variable and γ subs in the variables in N_4', making it closed apart from x where $x \notin dom(\sigma)$ thus $x \notin dom(\sigma_1')$. Then $N_4'|\gamma\rangle = N_4'|\omega\rangle$ where ω contains γ and $x \notin dom(\omega)$
  Thus $N_4'|\gamma\rangle = N_4'|\sigma_1'\rangle$ and $N_4'{}^{?\gamma} = N_4'{}^{?\sigma_1'}$
  $\equiv \nu\sigma.\left(\langle x^{?\sigma} : a\rangle \;\middle|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\middle|\; \langle\lambda x : A.\, N_4'{}^{?\sigma_1'}(N_2'{}^{?\sigma_1'}) : a_{z_0}\rangle \;\middle|\; \llbracket\sigma_1'\rrbracket\right)$

Recall $N_2'|\sigma_1\rangle$ is closed
Observe that when doing $N_2'|\sigma_1\rangle$, if a variable appears which is in the variable substitution chain, then for $N_2'|\sigma_1\rangle$ to be closed, it must follow the rest of the chain. Now consider what would happen with $N_2'|\sigma_1'\rangle$ instead because of β' the sequence substitution immediately goes to the end of the chain (subbing in $\lambda x : A.\, N_4'$).
Recall $N_4'|\gamma\rangle = N_4'|\omega\rangle$ where ω contains γ and $x \notin dom(\omega)$ this implies $(\lambda x : A.\, N_4')|\gamma\rangle = (\lambda x : A.\, N_4')|\omega\rangle$
The above can be applied to σ_1' and everything in σ_1' which also contains γ (for example $(\lambda x : A.\, N_4')/z_3,\ \gamma)$ due to $x \notin dom(\sigma_1')$; thus subbing in $(\lambda x : A.\, N_4')$ early does not affect the final term.
Thus $N_2'|\sigma_1'\rangle = N_2'|\sigma_1\rangle$
Recall $N_2'|\sigma_1\rangle = V_1$
Thus $N_2'|\sigma_1'\rangle = N_2'|\sigma_1\rangle = V_1$
The above also implies that possible terms σ_1' could sub in is a subset of the possible terms σ_1 could sub in. With σ_1' not subbing in the terms (in this case variables) that are subbed in for the variables in the substitution chain.

As N_2' could not be a value and $N_2'|\sigma_1'\rangle = V_1$, we repeat the same steps in the above case $(N = (\lambda x : A.\, N_1')(N_2'))$ but with σ_1' instead of σ.
Thus σ_1' can also be express as $\sigma_1' = \eta,\ \theta$, where η is the part of σ_1' which contains the variable substitution chain for N_2' and θ is the rest of σ_1'. Thus $N_2'|\eta\rangle = V_2'$ and $V_2'|\theta\rangle = V_1$
$= \nu\sigma.\, \Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle \lambda x : A.\, N_4'^{?\sigma_1'}(N_2'^{?\sigma_1'}) : a_{z_0}\rangle \;\Big|\; \llbracket\eta\rrbracket^{?\theta} \;\Big|\; \llbracket\theta\rrbracket\Big)$

We now apply M-Recv, zero to n times depending on the size of the variable substitution chain, passing the value $V_2'^{?\theta}$ back.
$\longrightarrow^* \nu\sigma.\, \Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle \lambda x : A.\, N_4'^{?\sigma_1'}(V_2'^{?\theta}) : a_{z_0}\rangle \;\Big|\; \llbracket\eta'\rrbracket^{?\theta} \;\Big|\; \llbracket\theta\rrbracket\Big)$

let $\sigma_1'' = \eta',\ \theta$
Repeating similar steps to the case where $(N = (\lambda x : A.\, N_1')(N_2'))$.
We get $dom(\sigma_1'') = dom(\sigma_1')$, thus for all P, $P^{?\sigma'} = P^{?\sigma''}$ and $V_2'|\theta\rangle = V_2'|\sigma_1''\rangle$ thus $V_2'^{?\theta} = V_2'^{?\sigma_1''}$
Applying the same logic above that we did with $N_2'|\sigma_1\rangle = N_2'|\sigma_1'\rangle$.
Thus $N_4'|\sigma_1'\rangle = N_4'|\sigma_1''\rangle$ and the terms σ_1'' could sub in being a subset of terms σ_1' could sub in which is also subset of the terms σ_1 could sub in, from above.
$= \nu\sigma.\, \Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle \lambda x : A.\, N_4'^{?\sigma_1''}(V_2'^{?\sigma_1''}) : a_{z_0}\rangle \;\Big|\; \llbracket\sigma_1''\rrbracket\Big)$
We now apply the M-Beta rule.
$\longrightarrow \nu\sigma.\, \Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \Big\langle N_4'^{?\sigma_1''}[V_2'^{?\sigma_1''}/x] : a_{z_0}\Big\rangle \;\Big|\; \llbracket\sigma_1''\rrbracket\Big)$
$\equiv \nu\sigma.\, \Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \Big\langle (N_4'[V_2'/x])^{?\sigma_1''} : a_{z_0}\Big\rangle \;\Big|\; \llbracket\sigma_1''\rrbracket\Big)$

Recall $dom(\sigma_1'') = dom(\sigma_1') = dom(\sigma_1)$
Recall $\sigma = \alpha,\ (z_1(N_2'))/z_0,\ \sigma_1$
let $\sigma' = \alpha,\ (N_4'[V_2'/x])/z_0,\ \sigma_1''$
$\therefore dom(\sigma) = dom(\sigma')$

Recall $\sigma_0 = (z_1(N_2'))/z_0,\ \sigma_1$
let $\sigma_0'' = (N_4'[V_2'/x]))/z_0,\ \sigma_1''$
$\therefore dom(\sigma_0) = dom(\sigma_0'')$
$\therefore\ \llbracket\alpha\rrbracket^{?\sigma_0} = \llbracket\alpha\rrbracket^{?\sigma_0''}$

$$\begin{aligned}\therefore\ \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \Big\langle (N_4'[V_2'/x])^{?\sigma_1''} : a_{z_0}\Big\rangle \;\Big|\; \llbracket\sigma_1''\rrbracket &= \llbracket\alpha\rrbracket^{?\sigma_0''} \;\Big|\; \Big\langle (N_4'[V_2'/x])^{?\sigma_1''} : a_{z_0}\Big\rangle \;\Big|\; \llbracket\sigma_1''\rrbracket \\ &= \llbracket\alpha,\ (N_4'[V_2'/x])/z_0,\ \sigma_1''\rrbracket \\ &= \llbracket\sigma'\rrbracket\end{aligned}$$

$= \nu\sigma.\, \Big(\langle x^{?\sigma'} : a\rangle \;\Big|\; \llbracket\sigma'\rrbracket\Big)$
Recall $dom(\sigma) = dom(\sigma')$
$= \nu\sigma'.\, \Big(\langle x^{?\sigma'} : a\rangle \;\Big|\; \llbracket\sigma'\rrbracket\Big) =\ C'$

$x|\sigma'\rangle = (N_4'[V_2'/x])|\sigma_1''\rangle$

As $x \notin dom(\sigma)$, $\forall\ M/y \in \sigma.\ x \notin \mathrm{fv}(M)$, $\sigma_1 \in \sigma$ and the terms σ_1'' could sub in being a subset of the terms σ_1 could sub in, thus $\forall\ M/x \in \sigma_1''.\ u \notin \mathrm{fv}(M)$.

We can apply the lemma 3.2.3

$= N_4'|\sigma_1''\rangle[V_2'|\sigma_1''\rangle/x]$

recall $N_4'|\sigma_1''\rangle = N_4'|\sigma_1'\rangle = N_4'|\gamma\rangle = N_1$

$= N_1[V_2'|\sigma_1''\rangle/x]$

recall $V_2'|\sigma_1''\rangle = V_2'|\theta\rangle = V_1$

$= N_1[V_1/x]$

recall V_1 is closed and N_1 is closed apart from x.

$\therefore N_1[V_1/x]$ is closed.

$\therefore x|\sigma'\rangle$ is closed.

Thus matching the condition for the relationship R

$\therefore\ x|\sigma'\rangle\ R\ \nu\sigma'.\left(\langle x^{?\sigma'} : a\rangle \,\middle|\, [\![\sigma']\!]\right)$

recall $x|\sigma'\rangle\ =\ N_1[V_1/u]$

$\therefore\ x|\sigma'\rangle\ =\ M'$

$\therefore\ M'\ R\ C'$

- $N = N_1'(N_2')$
  Similar to $N = x$

CASE(D-BOXBETA). Suppose $M \longmapsto M'$ of the form and $M\ R\ C$

$$\frac{}{\text{let box } u \Leftarrow \text{box } N_1 \text{ in } N_2 \longmapsto N_2[N_1/u]}$$

By inversion on relationship (R)

$\therefore\ M = N|\sigma\rangle$ and M is closed

$\therefore\ N_1$ is closed and N_2 is closed apart from u

We will assume $u \notin dom(\sigma)$ and $\forall\ M/x \in \sigma.\ u \notin \mathrm{fv}(M)$, as if this was not the case we can rename u so it is.

N can take multiple forms: x, $\text{let box } u \Leftarrow N_1' \text{ in } N_2'$, $\text{let box } u \Leftarrow \text{box } N_1' \text{ in } N_2'$ where N_1' could equal N_1 same with N_2'.

- $N = \text{let box } u \Leftarrow \text{box } N_1' \text{ in } N_2'$

  $M = (\text{let box } u \Leftarrow \text{box } N_1' \text{ in } N_2')|\sigma\rangle$

  by $u \notin dom(\sigma)$

  $= \text{let box } u \Leftarrow \text{box } N_1'|\sigma\rangle \text{ in } N_2'|\sigma\rangle$

  $\therefore\ N_1'|\sigma\rangle = N_1$
  $N_2'|\sigma\rangle = N_2$

  $C = \nu\sigma.\,\langle(\text{let box } u \Leftarrow \text{box } N_1' \text{ in } N_2')^{?\sigma} : a\rangle \,\Big|\, [\![\sigma]\!]$

  by $u \notin dom(\sigma)$

  $= \nu\sigma.\,\langle\text{let box } u \Leftarrow \text{box } N_1'^{?\sigma} \text{ in } N_2'^{?\sigma} : a\rangle \,\Big|\, [\![\sigma]\!]$

  $\longrightarrow \nu\sigma.\left(\nu a_u.\left(\left\langle N_2'^{?\sigma}[?a_u/u] : a\right\rangle \,\middle|\, \langle N_1'^{?\sigma} : a_u\rangle\right) \,\middle|\, [\![\sigma]\!]\right)$

  As a_u is a new channel $?a_u \notin \mathrm{fv}([\![\sigma]\!])$, if it was rename it

$$\equiv \nu\sigma.\nu a_u.\left(\left\langle {N_2'}^{?\sigma}[?a_u/u] : a\right\rangle \,\middle|\, \langle {N_1'}^{?\sigma} : a_u\rangle \,\middle|\, [\![\sigma]\!]\right)$$

Let $\sigma' \;=\; N_1'/u, \sigma$

Recall $u \notin dom(\sigma)$

Thus, adding it to σ does not affect the sequential substitution property where each variable subs in at most one term.

$\therefore {N_1'}^{?\sigma} = {N_1'}^{?\sigma'}$

$$\begin{aligned}\langle {N_1'}^{?\sigma} : a_u\rangle \,\big|\, [\![\sigma]\!] &= \langle {N_1'}^{?\sigma'} : a_u\rangle \,\big|\, [\![\sigma]\!] \\ &= [\![\sigma']\!]\end{aligned}$$

$$= \nu\sigma'.\left\langle {N_2'}^{?\sigma}[?a_u/u] : a\right\rangle \,\big|\, [\![\sigma']\!]$$

by definition $?\sigma$

$= \nu\sigma'.\langle {N_2'}^{?\sigma'} : a\rangle \,\big|\, [\![\sigma']\!] \equiv C'$

Recall N_1 is closed and N_2 is closed apart from u

Recall $N_1'|\sigma\rangle = N_1$ and $N_2'|\sigma\rangle = N_2$

$\therefore$ $N_1'|\sigma\rangle$ and $N_2'|\sigma\rangle$ is closed apart from u

$\therefore$ $N_2'|\sigma\rangle[N_1'|\sigma\rangle/u]$ is closed

By lemma 3.2.3

$$\begin{aligned}N_2'|\sigma\rangle[N_1'|\sigma\rangle/u] &= (N_2'[N_1'/u])|\sigma\rangle \\ &= N_2'|N_1'/u, \sigma\rangle \\ &= N_2'|\sigma'\rangle\end{aligned}$$

$\therefore N_2'|\sigma'\rangle$ is closed

$\therefore$ $N_2'|\sigma'\rangle \;R\; \nu\sigma'.\langle {N_2'}^{?\sigma'} : a\rangle \,\big|\, [\![\sigma']\!]$

$N_2'|\sigma\rangle[N_1'|\sigma\rangle/u] = N_2[N_1/u] = M'$

Recall $N_2'|\sigma\rangle[N_1'|\sigma\rangle/u] = N_2'|\sigma'\rangle$

$\therefore$ $N_2'|\sigma'\rangle = M'$

$\therefore$ $M' \;R\; C'$

- $N = x$

  We now repeat a similar step to the start of case D-BETA ($N = x$).

  $M = x|\sigma\rangle$

  by applying lemma 3.2.4

  σ is of the form $\sigma \;=\; \alpha,\; (\mathsf{let\ box}\ u \Leftarrow N_1\ \mathsf{in}\ N_2')/z_0,\; \beta$

  N_1' is either of the form $\mathsf{box}\ N_1''$ or is a variable

  We now assume that N_1' is a variable as it is clear this carries over to the case where its of the form $\mathsf{box}\ N_1''$. We rename N_1' to the variable z_1 to reflect the above.

  by applying lemma 3.2.4 again on z_1 and repeating similar steps in D-BETA ($N = x$).

  $\therefore$ $\sigma \;=\; \alpha,\; (\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')/z_0,\; \beta,\; \mathsf{box}\ N_4'/z_3,\; \gamma$, where z_i are variables and N_4', N_2' being terms.

  let $\sigma_0 = (\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')/z_0,\; \beta,\; \mathsf{box}\ N_4'/z_3,\; \gamma$

  let $\sigma_1 = \beta, \mathsf{box}\ N_4'/z_3,\; \gamma$

  let $\sigma_2 = \mathsf{box}\ N_4'/z_3,\; \gamma$

  $x|\sigma\rangle \;= z_0|\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2'/z_0,\; \sigma_1\rangle = (\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')|\sigma_1\rangle =$
  $(\mathsf{let\ box}\ u \Leftarrow z_3\ \mathsf{in}\ N_5')|\mathsf{box}\ N_4'/z_3,\; \gamma\rangle = (\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ N_4'\ \mathsf{in}\ N_6')|\gamma\rangle = M$

  $\therefore$ $x|\alpha\rangle \;=\; z_0,\; z_1|\beta\rangle \;=\; z_3$

$$N_2'|\beta\rangle \;=\; N_5',\; N_5'|\mathsf{box}\ N_4'/z_3\rangle \;=\; N_6'$$
$$N_4'|\gamma\rangle \;=\; N_1,\; N_6'|\gamma\rangle \;=\; N_2$$

Recall N_2 is closed apart from u and N_1 is closed
$N_2'|\sigma_1\rangle = N_2$ and $N_4'|\gamma\rangle = N_1$
$\therefore N_2'|\sigma_1\rangle$ is closed and $N_4'|\gamma\rangle$ is closed

$$C = \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\sigma\rrbracket\Big)$$
$$= \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle(\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')^{?\sigma_1} : a_{z_0}\rangle \;\Big|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\Big|\; \langle(\mathsf{box}\ N_4')^{?\gamma} : a_{z_3}\rangle \;\Big|\; \llbracket\gamma\rrbracket\Big)$$

by $u \notin dom(\sigma)$

$$= \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle\mathsf{let\ box}\ u \Leftarrow {z_1}^{?\sigma_1}\ \mathsf{in}\ {N_2'}^{?\sigma_1} : a_{z_0}\rangle \;\Big|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\Big|\; \langle(\mathsf{box}\ N_4')^{?\gamma} : a_{z_3}\rangle \;\Big|\; \llbracket\gamma\rrbracket\Big)$$

The below steps are similar to the case D-BETA ($N = x$).
Note that $z_1|\beta\rangle = z_3$, where each sub that is able to be applied is a sequence of variable substitution chained together (by lemma 3.2.4) and $(\mathsf{box}\ N_4')^{?\gamma}$ val this allows for the M-Recv rule to be applied. β turns into β', as when passing down $(\mathsf{box}\ N_4')^{?\gamma}$ the $?a$ of the variables chained together will now get resolved to $(\mathsf{box}\ N_4')^{?\gamma}$

$$\longrightarrow^{*}\ \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle\mathsf{let\ box}\ u \Leftarrow (\mathsf{box}\ N_4')^{?\gamma}\ \mathsf{in}\ {N_2'}^{?\sigma_1} : a_{z_0}\rangle \;\Big|\; \llbracket\beta'\rrbracket^{?\sigma_2} \;\Big|\; \langle(\mathsf{box}\ N_4')^{?\gamma} : a_{z_3}\rangle \;\Big|\; \llbracket\gamma\rrbracket\Big)$$

let $\sigma_1' = \beta',\ \mathsf{box}\ N_4'/z_3,\ \gamma$
$dom(\sigma_1') = dom(\sigma_1)$, as the only thing that has changed is that variables in the chain we now sub in a different term ($\mathsf{box}\ N_4'$) instead of the variable next in the chain.
Thus for all P, $P^{?\sigma_1} = P^{?\sigma_1'}$.
As sequential substitution can at most sub one term for each variable and γ subs in the variables in N_4' making it closed, then $N_4'|\gamma\rangle = N_4'|\omega\rangle$ where ω contains γ.
Thus $N_4'|\gamma\rangle = N_4'|\sigma_1'\rangle$ and ${N_4'}^{?\gamma} = {N_4'}^{?\sigma_1'}$

$$= \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \langle\mathsf{let\ box}\ u \Leftarrow (\mathsf{box}\ N_4')^{?\sigma_1'}\ \mathsf{in}\ {N_2'}^{?\sigma_1'} : a_{z_0}\rangle \;\Big|\; \llbracket\sigma_1'\rrbracket\Big)$$

Recall $\sigma_0 = (\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')/z_0,\ \sigma_1$
let $\sigma_0' = (\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')/z_0,\ \sigma_1'$
Recall $dom(\sigma_1) = dom(\sigma_1')$
$\therefore dom(\sigma_0) = dom(\sigma_0')$
Thus $\llbracket\alpha\rrbracket^{?\sigma_0} = \llbracket\alpha\rrbracket^{?\sigma_0'}$

$$= \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0'} \;\Big|\; \langle\mathsf{let\ box}\ u \Leftarrow (\mathsf{box}\ N_4')^{?\sigma_1'}\ \mathsf{in}\ {N_2'}^{?\sigma_1'} : a_{z_0}\rangle \;\Big|\; \llbracket\sigma_1'\rrbracket\Big)$$

Recall $N_2'|\sigma_1\rangle$ is closed
Observe that when doing $N_2'|\sigma_1\rangle$, if a variable appears which is in the variable substitution chain, then for $N_2'|\sigma_1\rangle$ to be closed, it must follow the rest of the chain. However, with $N_2'|\sigma_1'\rangle$ because of β' the sequence substitution immediately goes to the end of the chain (subbing in $\mathsf{box}\ N_4'$), and due to the fact that $\mathsf{box}\ N_4'|\gamma\rangle = \mathsf{box}\ N_4'|\omega\rangle$ (coming from $N_4'|\gamma\rangle = N_4'|\omega\rangle$) where ω contains γ; subbing in $\mathsf{box}\ N_4'$ early does not affect the final term.
Thus $N_2'|\sigma_1'\rangle = N_2'|\sigma_1\rangle = N_2$
The above also implies that possible terms σ_1' could sub in is a subset of the possible terms σ_1 could sub in.

We now apply the M-BoxBeta rule.

$$\longrightarrow \nu\sigma.\Big(\langle x^{?\sigma} : a\rangle\Big) \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0'} \;\Big|\; \nu a_u.\Big(\Big\langle {N_2'}^{?\sigma_1'}[?a_u/u] : a_{z_0}\Big\rangle\Big) \;\Big|\; \langle {N_4'}^{?\sigma_1'} : a_u\rangle \;\Big|\; \llbracket\sigma_1'\rrbracket$$

As a_u is a new channel, it does not appear anyway in the other configurations; if it does rename it

$$\equiv \nu\sigma.\Big(\nu a_u.\Big(\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0'} \;\Big|\; \Big\langle {N_2'}^{?\sigma_1'}[?a_u/u] : a_{z_0}\Big\rangle \;\Big|\; \langle {N_4'}^{?\sigma_1'} : a_u\rangle \;\Big|\; \llbracket\sigma_1'\rrbracket\Big)\Big)$$

The below steps are similar to case where $N = \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ N_1'\ \mathsf{in}\ N_2'$

Let $\sigma_1'' \;=\; N_4'/u, \sigma_1'$
Recall $u \notin dom(\sigma)$, $\forall\ M/x\ \in\ \sigma.\ u\ \notin\ \mathrm{fv}(M)$, $dom(\sigma_1) \in dom(\sigma)$ and $dom(\sigma_1') = dom(\sigma_1)$. Thus $u \notin dom(\sigma_1')$
$\therefore$ Adding it to σ_1' does not affect the sequential substitution

property where each variable subs in at most one term.

$\therefore N_4'^{?\sigma_1'} = N_4'^{?\sigma_1''}$

$$\begin{aligned} \langle N_4'^{?\sigma_1'} : a_u \rangle \;\Big|\; \llbracket \sigma_1' \rrbracket &= \langle N_4'^{?\sigma_1''} : a_u \rangle \;\Big|\; \llbracket \sigma_1' \rrbracket \\ &= \llbracket \sigma_1'' \rrbracket \end{aligned}$$

$= \nu\sigma.\left(\nu a_u.\left(\langle x^{?\sigma} : a \rangle \;\Big|\; \llbracket \alpha \rrbracket^{?\sigma_0'} \;\Big|\; \left\langle N_2'^{?\sigma_1'}[?a_u/u] : a_{z_0} \right\rangle \;\Big|\; \llbracket \sigma_1'' \rrbracket\right)\right)$

by definition $?\sigma$

$= \nu\sigma.\left(\nu a_u.\left(\langle x^{?\sigma} : a \rangle \;\Big|\; \llbracket \alpha \rrbracket^{?\sigma_0'} \;\Big|\; \langle N_2'^{?\sigma_1''} : a_{z_0} \rangle \;\Big|\; \llbracket \sigma_1'' \rrbracket\right)\right)$

Recall $\sigma_0' = (\mathsf{let\ box}\ u \Leftarrow z_1\ \mathsf{in}\ N_2')/z_0,\ \sigma_1'$

let $\sigma_0'' = N_2'/z_0\ , N_4'/u, \sigma_1'$

$\therefore \mathit{dom}(\sigma_0') = \mathit{dom}(\sigma_0'') + u$

Recall $u\ \notin\ \mathit{dom}(\sigma)$

$\therefore u\ \notin\ \mathit{dom}(\alpha)$

$\therefore \llbracket \alpha \rrbracket^{?\sigma_0'} = \llbracket \alpha \rrbracket^{?\sigma_0''}$

$= \nu\sigma.\left(\nu a_u.\left(\langle x^{?\sigma} : a \rangle \;\Big|\; \llbracket \alpha \rrbracket^{?\sigma_0''} \;\Big|\; \langle N_2'^{?\sigma_1''} : a_{z_0} \rangle \;\Big|\; \llbracket \sigma_1'' \rrbracket\right)\right)$

let $\sigma' = \alpha,\ \sigma_0''$

The only difference between σ and σ' is the added u and β being β'

Recall $\mathit{dom}(\sigma_0') = \mathit{dom}(\sigma_0)$ with the difference between σ_0 and σ_0' being β and β'

$\therefore \mathit{dom}(\beta) = \mathit{dom}(\beta')$

$\therefore \mathit{dom}(\sigma') = \mathit{dom}(\sigma) + u$

$= \nu\sigma'.\langle x^{?\sigma} : a \rangle \;\Big|\; \llbracket \alpha \rrbracket^{?\sigma_0''} \;\Big|\; \langle N_2'^{?\sigma_1''} : a_{z_0} \rangle \;\Big|\; \llbracket \sigma_1'' \rrbracket$

M' is not equal to x thus $x \in \mathit{dom}(\sigma)$

by $u \notin \mathit{dom}(\sigma)$ and the above, $u \neq x$

$= \nu\sigma'.\langle x^{?\sigma'} : a \rangle \;\Big|\; \llbracket \sigma' \rrbracket \equiv C'$

$$\begin{aligned} x|\sigma'\rangle &= x|\alpha,\ \sigma_0''\rangle \\ &= z_0|\sigma_0''\rangle \\ &= z_0|N_2'/z_0\ , \sigma_1''\rangle \\ &= N_2'|\sigma_1''\rangle \\ &= N_2'|N_4'/u, \sigma_1'\rangle \end{aligned}$$

By definition of sequential substitution

$= (N_2'[N_4'/u])|\sigma_1'\rangle$

Recall $u\ \notin\ \mathit{dom}(\sigma_1')$, $\sigma_1 \in \sigma$, $\forall\ M/x\ \in\ \sigma.\ u\ \notin\ \mathrm{fv}(M)$ and σ_1' could sub in is a subset of the possible terms σ_1 could sub in, thus $\forall\ M/x\ \in\ \sigma_1'.\ u\ \notin\ \mathrm{fv}(M)$

By lemma 3.2.3

$= N_2'|\sigma_1'\rangle[N_4'|\sigma_1'\rangle/u]$

Recall $N_4'|\gamma\rangle = N_4'|\sigma_1'\rangle$ and $N_2'|\sigma_1'\rangle = N_2'|\sigma_1\rangle$

$= N_2'|\sigma_1\rangle[N_4'|\gamma\rangle/u]$

Recall $N_2'|\sigma_1\rangle = N_2$ and $N_4'|\gamma\rangle = N_1$

$= N_2[N_1/u]$

Recall N_2 is closed apart from u and N_1 is closed

$\therefore N_2[N_1/u]$ is closed

$\therefore x|\sigma'\rangle$ is closed

$x|\sigma'\rangle\ R\ \nu\sigma'.\langle x^{?\sigma'} : a \rangle \;\Big|\; \llbracket \sigma' \rrbracket$

$x|\sigma'\rangle = N_2[N_1/u] = M'$

$M'\ R\ C'$

- $N = \mathsf{let\ box}\ u \Leftarrow N_1'\ \mathsf{in}\ N_2'$
  Similar to $N = x$ case

CASE(D-EVAL). Suppose $M \longmapsto M'$ of the form and $M\ R\ C$

$$\frac{P \longmapsto P'}{\mathcal{E}[P] \longmapsto \mathcal{E}[P']}$$

By inversion on relationship (R)
$\therefore\ M = N|\sigma\rangle$ and M is closed
$\therefore\ \mathcal{E}$ is closed excluding the hole and P is closed

N can take two forms: x, $\mathcal{E}'[N_1']$ where N_1' could equal P and $\mathcal{E}'$ could equal $\mathcal{E}$.

- $N = \mathcal{E}'[N_1']$

  $M = (\mathcal{E}'[N_1'])|\sigma\rangle$
  by lemma 3.2.2
  $= \mathcal{E}'|\sigma\rangle[N_1'|\sigma\rangle]$
  $\therefore \mathcal{E} = \mathcal{E}'|\sigma\rangle$ and $P = N_1'|\sigma\rangle$
  $\therefore \mathcal{E}'|\sigma\rangle$ and $N_1'|\sigma\rangle$ are closed

  $C = \nu\sigma.\left(\left\langle (\mathcal{E}'[N_1'])^{?\sigma} : a \right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$
  by lemma 3.2.1
  $= \nu\sigma.\left(\left\langle \mathcal{E}'^{?\sigma}[N_1'^{?\sigma}] : a \right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$

  Recall $N_1'|\sigma\rangle$ is closed
  $\therefore N_1'|\sigma\rangle\ R\ \nu\sigma.\left(\langle N_1'^{?\sigma} : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right)$
  Recall $N_1'|\sigma\rangle \longmapsto P'$

  We can now apply the induction hypothesis on $N_1'|\sigma\rangle$
  $\nu\sigma.\left(\langle N_1'^{?\sigma} : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow^* \nu\sigma'.\left(\langle N_1''^{?\sigma'} : a\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$ with $P'\ R\ \nu\sigma'.\left(\langle N_1''^{?\sigma'} : a\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$
  P' is closed by inversion on the relationship R and $P' = N_1''|\sigma'\rangle$ thus $N_1''|\sigma'\rangle$ is closed

  As $\nu\sigma.\left(\langle N_1'^{?\sigma} : a\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow^* \nu\sigma'.\left(\langle N_1''^{?\sigma'} : a\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$
  by applying lemma 3.2.5 multiple times
  $\therefore \nu\sigma.\left(\left\langle \mathcal{E}'^{?\sigma}[N_1'^{?\sigma}] : a \right\rangle \,\middle|\, \llbracket\sigma\rrbracket\right) \longrightarrow^* \nu\sigma'.\left(\left\langle \mathcal{E}'^{?\sigma}[N_1''^{?\sigma'}] : a \right\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$
  The difference between $dom(\sigma)$ and $dom(\sigma')$ is at most u which we can renamed so $u \notin \mathcal{E}'$
  Thus $\mathcal{E}'^{?\sigma} = \mathcal{E}'^{?\sigma'}$
  The difference between σ and σ' is at most a new thread outputting on channel a_u (which does not alter $\mathcal{E}'$ as $u \notin \mathcal{E}'$) and the variables which are in the variable substitution chains of the subterms in N_1', are now resolved by M-Recv. However, from the above cases D-BETA and D-BOXBETA, this resolving never affects the other terms (which are inside the threads in the configuration), final term when the substitution is applied. For example in the case D-BETA $N = x$ where $N_2'|\sigma_1'\rangle = N_2'|\sigma_1\rangle = V_1$ (found in the first paragraph of page 21)
  Thus $\mathcal{E}'|\sigma\rangle = \mathcal{E}'|\sigma'\rangle$

  $= \nu\sigma'.\left(\left\langle \mathcal{E}'^{?\sigma'}[N_1''^{?\sigma'}] : a \right\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$
  By lemma 3.2.1
  $= \nu\sigma'.\left(\left\langle (\mathcal{E}'[N_1''])^{?\sigma'} : a \right\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right) \equiv C'$

  Recall $\mathcal{E}'|\sigma\rangle$, $N_1''|\sigma'\rangle$ are closed and $\mathcal{E}'|\sigma\rangle = \mathcal{E}'|\sigma'\rangle$
  Thus $\mathcal{E}'|\sigma'\rangle[N_1''|\sigma'\rangle]$ is closed.
  By lemma 3.2.2
  $\therefore (\mathcal{E}'[N_1''])|\sigma'\rangle$ is closed.
  $\therefore (\mathcal{E}'[N_1''])|\sigma'\rangle\ R\ \nu\sigma'.\left(\left\langle (\mathcal{E}'[N_1''])^{?\sigma'} : a \right\rangle \,\middle|\, \llbracket\sigma'\rrbracket\right)$

By lemma 3.2.2
$\therefore \mathcal{E}'|\sigma'\rangle[N_1''|\sigma'\rangle] \; R \; \nu\sigma'.\left(\left\langle(\mathcal{E}'[N_1''])^{?\sigma'} : a\right\rangle \middle| \llbracket\sigma'\rrbracket\right)$
Recall $\mathcal{E} = \mathcal{E}'|\sigma\rangle$, $P' = N_1''|\sigma'\rangle$ and $\mathcal{E}'|\sigma\rangle = \mathcal{E}'|\sigma'\rangle$

$\therefore M' \; R \; C'$

- $N = x$

We now repeat a similar step to the start of case D-BETA ($N = x$).
$M = x|\sigma\rangle$
by applying lemma 3.2.4
σ is of the form $\sigma \;=\; \alpha,\ \mathcal{E}_1[N_1']/z_0,\ \beta$
N_1' is either of the form $\mathsf{box}\ N_1''$ or $\lambda x : A.\, N_1''$ (depending which reduction on P is used) or is a variable.
We now assume that N_1' is a variable as its clear, this combine with the above case $N = \mathcal{E}'[N_1']$ carries over to where N_1' is not a variable. We rename N_1' to the variable z_1 to reflect this.
By applying lemma 3.2.4 again on z_1
$\sigma = \alpha,\ \mathcal{E}_1[z_1]/z_0,\ \beta,\ N'/z_2,\ \gamma$ where S_i are variables and N_1' being terms.

let $\sigma_0 = \mathcal{E}_1[z_1]/z_0,\ \beta,\ N'/z_2,\ \gamma$
let $\sigma_1 = \beta,\ N'/z_2,\ \gamma$
let $\sigma_2 = N'/z_2,\ \gamma$

$x|\sigma\rangle \;= z_0|\mathcal{E}_1[z_1]/z_0,\ \sigma_1\rangle = (\mathcal{E}_1[z_1])|\sigma_1\rangle =$
$(\mathcal{E}_2[z_2])|N'/z_2,\ \gamma\rangle = (\mathcal{E}_3[N'])|\gamma\rangle = M$

$$\begin{aligned}\therefore\ x|\alpha\rangle &= z_0,\ z_1|\beta\rangle \;=\; z_2\\ \mathcal{E}_1|\beta\rangle &= \mathcal{E}_2,\ \mathcal{E}_2|N'/z_2\rangle \;=\; \mathcal{E}_3\\ \mathcal{E}_3|\gamma\rangle &= \mathcal{E},\ N'|\gamma\rangle \;=\; P\end{aligned}$$

$$C = \nu\sigma.\,\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \left\langle(\mathcal{E}_1[z_1])^{?\sigma_1} : ?a_{z_0}\right\rangle \;\Big|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\Big|\; \langle N'^{?\gamma} : a_{z_2}\rangle \;\Big|\; \llbracket\gamma\rrbracket$$

Recall P is closed and $\mathcal{E}$ is closed.
$\therefore N'|\gamma\rangle$ and $\mathcal{E}_3|\gamma\rangle$ are closed.
$\therefore N'|\gamma\rangle \; R \; \nu\gamma.\,\langle N'^{?\gamma} : a_{z_2}\rangle \;\Big|\; \llbracket\gamma\rrbracket$
Recall $N'|\gamma\rangle \longmapsto P'$

We can now apply the induction hypothesis on $N'|\gamma\rangle$
$\nu\gamma.\left(\langle N'^{?\gamma} : a_{z_2}\rangle \middle| \llbracket\gamma\rrbracket\right) \longrightarrow^* \nu\gamma'.\left(\langle N''^{?\gamma'} : a_{z_2}\rangle \middle| \llbracket\gamma'\rrbracket\right)$ with $P' \; R \; \nu\gamma'.\left(\langle N''^{?\gamma'} : a_{z_2}\rangle \middle| \llbracket\gamma'\rrbracket\right)$
P' is closed by inversion on the relationship R and $P' = N_1''|\gamma'\rangle$ thus $N_1''|\gamma'\rangle$ is closed.

As $\nu\gamma.\left(\langle N'^{?\gamma} : a_{z_2}\rangle \middle| \llbracket\gamma\rrbracket\right) \longrightarrow^* \nu\gamma'.\left(\langle N''^{?\gamma'} : a_{z_2}\rangle \middle| \llbracket\gamma'\rrbracket\right)$
Observe that all the reduction rules (M-Beta, M-BoxBeta and M-Recv) do not involve channels apart from M-BoxBeta, which adds a new channel a_u. This new channel can be renamed; thus, using structural congruence we can bring this channel to the front, making σ'. Thus, the above combined with M-PAR-1.

$$\begin{aligned}&\therefore \nu\sigma.\,\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \left\langle(\mathcal{E}_1[z_1])^{?\sigma_1} : ?a_{z_0}\right\rangle \;\Big|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\Big|\; \langle N'^{?\gamma} : a_{z_2}\rangle \;\Big|\; \llbracket\gamma\rrbracket\\ &\qquad\longrightarrow^* \nu\sigma'.\,\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\alpha\rrbracket^{?\sigma_0} \;\Big|\; \left\langle\mathcal{E}_1[z_1]^{?\sigma_1} : ?a_{z_0}\right\rangle \;\Big|\; \llbracket\beta\rrbracket^{?\sigma_2} \;\Big|\; \langle N''^{?\gamma'} : a_{z_2}\rangle \;\Big|\; \llbracket\gamma'\rrbracket \equiv C'\end{aligned}$$

where $\sigma' = \alpha,\ \mathcal{E}_1[z_1]/z_0,\ \beta,\ N''/z_2,\ \gamma'$
As the difference between σ and σ' is at most u which we can rename so $u \notin dom(\sigma)$, then adding u to σ does not affect $\llbracket\sigma\rrbracket$ up to where the recursive definition reaches u which is in γ'.
$= \nu\sigma'.\,\langle x^{?\sigma} : a\rangle \;\Big|\; \llbracket\sigma'\rrbracket$

We will assume $u \notin dom(\sigma)$ and $\forall\ M/x \in \sigma.\ u \notin \text{fv}(M)$ as if this was not the case we can rename u so it is.

Recall $\mathcal{E}_3|\gamma\rangle$, $N_1''|\gamma'\rangle$ are closed.
$\therefore \mathcal{E}_3|\gamma\rangle[N''|\gamma'\rangle]$ is closed.
The difference between γ and γ' is at most u. By $u \in dom(\sigma)$, $\forall\ M/x \in \sigma.\ u \notin \text{fv}(M)$ and $\gamma \in \sigma$ thus $u \in dom(\gamma)$ and $\forall\ M/x \in \gamma.\ u \notin \text{fv}(M)$. As $\mathcal{E}_3|\gamma\rangle$ is closed.
$\therefore \mathcal{E}_3|\gamma\rangle = \mathcal{E}_3|\gamma'\rangle$
$\therefore \mathcal{E}_3|\gamma'\rangle[N''|\gamma'\rangle]$ is closed.

$$\begin{aligned} x|\sigma'\rangle &= (\mathcal{E}_1[z_1])|\beta,\ N''/z_2,\ ,\gamma'\rangle \\ &= (\mathcal{E}_2[z_2])|N''/z_2,\ \gamma'\rangle \\ &= (\mathcal{E}_3[N''])|\gamma'\rangle \\ &\text{by lemma 3.2.2} \\ &= \mathcal{E}_3|\gamma'\rangle[N''|\gamma'\rangle] \end{aligned}$$

$\therefore x|\sigma'\rangle$ is closed.
$\therefore x|\sigma'\rangle\ R\ \nu\sigma'.\langle x^{?\sigma} : a\rangle \bigm| [\![\sigma']\!]$

$x|\sigma'\rangle = \mathcal{E}_3|\gamma'\rangle[N''|\gamma'\rangle]$
Recall $P = N_1''|\gamma\rangle$, $\mathcal{E}|\gamma'\rangle = \mathcal{E}|\gamma\rangle$ and $\mathcal{E}|\gamma\rangle = \mathcal{E}$
$= \mathcal{E}[P']$
$= M'$

$M'\ R\ C'$

□

Chapter 4

Related Work

There are multiple calculi that have distributed operations; the most prominent one is the π-calculus [MPW92a; MPW92b]. The π-calculus is a process based calculus that allows for processes or threads to be executed concurrently while passing information (names) via named channels; these names being passed can themselves become channels. The concept of mobility can to be defined here due to the fact that terms are not bounded to a location or fixed to a point. This is due to structural congruence in the calculus, which enables processes to be rearranged and named channels to change scope (via scope extrusion). However, due to this there is an inability to define locations; for example, scopes can not represent defined locations as there are able to be changed due to scope extrusion. Thus locality can not be defined in the calculus. Some extensions to π-calculus rectify this by adding locations, DPI calculus by Hennessy and Riely [HR02] and lsdλ by Ravara, Matos, Vasconcelos, and Lopes [Rav+03]. These calculi enable certain channel names to be designated to a specific location while others follow scope extrusion. Other extensions bring confidentiality [PV21], make positive adjustments from a security respective [ABF17] and adapt the calculus for modeling Mobile Ad Hoc Wireless Networks (MANETs) [SRS10]. A couple of calculi are build on the core of π-calculus, such as Seal calculus [CVZ05] and Kell calculus [SS04].
Many different type systems have been proposed for π-calculus [Gay93; KPT99; PS96]. However, there is yet to be a clear option on which type system is most canonical.

One calculus that is similar to ours is Lambda 5, a dual context calculus which is based on modal logic IS5 via the Curry-Howard correspondence [Mur+04; Sim94]. Where each possible world represents nodes on a network, in this network it is assumed that all nodes can communicate with each other; due to this Murphy VII, Crary, Harper, and Pfenning chose an accessibility relation that is reflexive (allowing a node(world) to be accessible itself) symmetric (if a node A can access node B then node B can assess node A), and transitive (if a node A can access node B, and node B can access node C then node A can access node C) hence using modal S5 logic. This gives two more axioms compared with S4, $\Diamond A \to \Box\Diamond A$ and $\Diamond\Box A \to \Box A$. Similar to ours, a term of type $\Box A$ represents a term (mobile code) of type A at any world, allowing it to be evaluated anywhere and a term of type $\Diamond A$ represents the address of a remote term that as type A. They explicitly define location rather than leaving it abstracted away like in our case; this is done by having judgements of the form $\Omega; \Gamma \vdash M : A@\omega$ where M has type A at world ω.
For the dynamics of calculus they define an abstract machine which is a network of nodes, and the steps of computation are distributed along the nodes. This machine is sequential and deterministic, unlike our configurations, which are concurrent and non-deterministic.
The network contains a fixed number of nodes w_i, where each node is a table in which the programmer specifies what terms this table contains for each node. Term with type $\Diamond A$ contains the world number and label, thus acting as an address to look up a term with type A.
They present two RPC (Remote Procedure Call) calls fetch$[\omega']M$ and get$< \omega' >M$. Fetch executes code M in the node of ω' then retrieves the result value of type $\Box$, and get$< \omega' >M$, which behaves similarly but returns a term of type $\Diamond$. These RPC are made possible due to the extra axioms bought with S5. This is similar to our calculus as via M-BoxBeta rule terms can be evaluated at a different world which in our case is a different thread with the value being ”returned” via M-Recv rule.

Other calculus that bear a resemblance is λrpc[JW04]. Its again is dual context, based on S5 logic having types $\Box$, $\Diamond$ and worlds representing processes at *places* in a network. However its based on hybrid logic meaning that worlds are inside propositions thus the judgments in this calculus contain the following,

"A at ω" meaning type A resides in world ω. They define additions types $\tau@z$ and $n[\tau]$ called placements. $\tau@z$ at ω would entail reasoning about world z from world ω and $n[\tau]$ at ω reasons about a world which is from traversing edge n from ω.
The operational semantics also incorporate RPC calls that return values. However, their semantics involve synchronization and adopt a process calculus approach, which sets them apart from Lambda 5 and makes them more akin to our configurations.

The ambient calculus is a process-based calculus where ambients exist defined as a bounded place where computation can occur, for example a web page that is bounded by a file (possibly a .html file). Mobility in this calculus is the act of ambients crossing boundaries which can be restricted providing security. This notation of mobility is distinct from π-calculus as it does not involve communication over channels. An ambient can contain other ambients, and so on. Unlike our calculus, this particular calculus is not inherently logic-based, as it did not utilize the Curry-Howard correspondence. Instead, it begins with a process calculus foundation and builds logic on top of it in order to further analyze the language's behaviors. The computation is the movement of ambients. Untyped ambients have no fixed scope as they are allowed to move around; thus locations can not be defined, meaning no locality.
Some other research papers that present languages that have mobility of code are De Nicola, Ferrari, and Pugliese [DFP98], Borghuis and Feijs [BF00] and Bonelli and Feller [BF12].

In general, what sets our calculus apart from the others is that it is based on logic which adds to its robustness, while its notation being relatively light-weight compared to the others.

In a paper by Milner, it was shown that λ-calculus is able to be encoded into the π-calculus [Mil92]; he defined a relation and showed that each reduction in λ-calculus can be mimicked by sequences of reductions in π-calculus with the relation holding though out. However, during Milner's proof the strong bisimilarity on processes was used to show that relation still held after the sequence of π-calculus reductions. The paper by Accattoli [Acc13] makes steps to rectify this by enabling the reductions of the λ-calculus to be more closely mirrored in the π-calculus. Sangiorgi [San99] further analyses the relationship between λ-calculus and π-calculus.
Our proof used a similar method to one shown in Milner's paper and certain aspects of our proof were also inspired by it. For example, the decision to use a sequence substitution in the definition of the relationship. This approach enables sigma to substitute a term containing free variables. This is required for the D-BoxBeta case to hold as if this was not the case; then one is unable to define a new sigma which maps u to a term that could contain free variables N_1'.

Proofs that also use a similar approach to our are Vasconcelos [Vas05], Sangiorgi and Xu [SX14], Cimini, Coen, and Sangiorgi [CCS10] and Boudol [Bou97]

Chapter 5

Conclusion

In conclusion, distributed programming is becoming increasingly significant as more applications adopt a cloud-based architecture. However, this type of programming is complex, as it must grapple with issues such as race conditions, deadlocks, livelocks, and consistency loss. Moody's paper introduced modal types as a way to represent spatial properties and a dual-context calculus based on using these modal types. He proposed an distributed abstract machine enabling concurrent execution of the dual-context calculus across multiple processors at different locations. However, Moody's work did not include proof of computational equivalence between the dual-context calculus and distributed abstract machine.

This project explains the origins behind Moody's language. Defines a dual-context calculus akin to Moody's proposal, presents a distributed abstract machine in the style of π-calculus and demonstrates their computational equivalence using a bi-simulation method, ensuring that deadlock and livelock do not occur within the machine; this serves as the initial first steps toward improving non-serial programming, by further developing a programming language that by design is not able to deadlock or livelock. We then conclude with a literature review that compares the dual calculus and the machine we define to other relevant works. Due to the above this project achieves all of its aims.

Future research for the project could be adding more features to dual-context calculus and the abstract machine, then proofing a similar conjecture. Some possible features are: adding a store, which can be used to derive the $\diamond$ type, similar to Moody's approach. Expand the calculus to make it based on S5 defining new rules based on the new axioms gained.

Bibliography

[ABF17] Martın Abadi, Bruno Blanchet, and Cédric Fournet. "The applied pi calculus: Mobile values, new names, and secure communication". In: *Journal of the ACM (JACM)* 65.1 (2017), pp. 1–41 (cit. on p. 29).

[Acc13] Beniamino Accattoli. "Evaluating functions as processes". In: *Electronic Proceedings in Theoretical Computer Science* 110 (Feb. 2013), pp. 41–55. DOI: 10.4204/eptcs.110.6. URL: https://doi.org/10.4204%5C%2Feptcs.110.6 (cit. on p. 30).

[BF12] Eduardo Bonelli and Federico Feller. "Justification Logic as a foundation for certifying mobile computation". In: *Annals of Pure and Applied Logic* 163.7 (2012). The Symposium on Logical Foundations of Computer Science 2009, pp. 935–950. ISSN: 0168-0072. DOI: https://doi.org/10.1016/j.apal.2011.09.007. URL: https://www.sciencedirect.com/science/article/pii/S0168007211001291 (cit. on p. 30).

[BF00] Tijn Borghuis and Loe Feijs. "A Constructive Logic for Services and Information Flow in Computer Networks". In: *The Computer Journal* 43.4 (Jan. 2000), pp. 274–289. ISSN: 0010-4620. DOI: 10.1093/comjnl/43.4.274. eprint: https://academic.oup.com/comjnl/article-pdf/43/4/274/1112015/430274.pdf. URL: https://doi.org/10.1093/comjnl/43.4.274 (cit. on p. 30).

[Bou97] Gérard Boudol. "The π-calculus in direct style". In: *Proceedings of the 24th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*. 1997, pp. 228–242 (cit. on p. 30).

[CVZ05] G. Castagna, J. Vitek, and F. Zappa Nardelli. "The Seal Calculus". In: *Information and Computation* 201.1 (2005), pp. 1–54. ISSN: 0890-5401. DOI: https://doi.org/10.1016/j.ic.2004.11.005. URL: https://www.sciencedirect.com/science/article/pii/S0890540105000635 (cit. on p. 29).

[CCS10] Matteo Cimini, Claudio Sacerdoti Coen, and Davide Sangiorgi. "Functions as Processes: Termination and the $\lambda\mu\mu$-Calculus". In: *Trustworthly Global Computing: 5th International Symposium, TGC 2010, Munich, Germany, February 24-26, 2010, Revised Selected Papers 5*. Springer. 2010, pp. 73–86 (cit. on p. 30).

[DP01] Rowan Davies and Frank Pfenning. "A modal analysis of staged computation". In: *Journal of the ACM* 48.3 (2001), pp. 555–604. DOI: 10.1145/382780.382785 (cit. on pp. 2, 3, 10).

[DFP98] R. De Nicola, G.L. Ferrari, and R. Pugliese. "KLAIM: a kernel language for agents interaction and mobility". In: *IEEE Transactions on Software Engineering* 24.5 (1998), pp. 315–330. DOI: 10.1109/32.685256 (cit. on p. 30).

[FH92] Matthias Felleisen and Robert Hieb. "The revised report on the syntactic theories of sequential control and state". In: *Theoretical Computer Science* 103.2 (1992), pp. 235–271. DOI: 10.1016/0304-3975(92)90014-7 (cit. on p. 5).

[Gay93] Simon J. Gay. "A Sort Inference Algorithm for the Polyadic π-Calculus". In: *Proceedings of the 20th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*. POPL '93. Charleston, South Carolina, USA: Association for Computing Machinery, 1993, pp. 429–438. ISBN: 0897915607. DOI: 10.1145/158511.158701. URL: https://doi.org/10.1145/158511.158701 (cit. on p. 29).

[HR02] Matthew Hennessy and James Riely. "Resource Access Control in Systems of Mobile Agents". In: *Information and Computation* 173.1 (2002), pp. 82–120. ISSN: 0890-5401. DOI: https://doi.org/10.1006/inco.2001.3089. URL: https://www.sciencedirect.com/science/article/pii/S0890540101930895 (cit. on p. 29).

[JW04] Limin Jia and David Walker. “Modal Proofs as Distributed Programs”. In: *Programming Languages and Systems.* Ed. by David Schmidt. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 219–233. ISBN: 978-3-540-24725-8 (cit. on p. 29).

[KPT99] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. “Linearity and the Pi-Calculus”. In: *ACM Trans. Program. Lang. Syst.* 21.5 (Sept. 1999), pp. 914–947. ISSN: 0164-0925. DOI: 10.1145/330249.330251. URL: https://doi.org/10.1145/330249.330251 (cit. on p. 29).

[Mil92] Robin Milner. “Functions as processes”. In: *Mathematical Structures in Computer Science* 2.2 (1992), pp. 119–141. DOI: 10.1017/S0960129500001407 (cit. on pp. 7, 30).

[MPW92a] Robin Milner, Joachim Parrow, and David Walker. “A calculus of mobile processes, I”. In: *Information and Computation* 100.1 (1992), pp. 1–40. ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(92)90008-4. URL: https://www.sciencedirect.com/science/article/pii/0890540192900084 (cit. on pp. 8, 29).

[MPW92b] Robin Milner, Joachim Parrow, and David Walker. “A calculus of mobile processes, II”. In: *Information and Computation* 100.1 (1992), pp. 41–77. ISSN: 0890-5401. DOI: https://doi.org/10.1016/0890-5401(92)90009-5. URL: https://www.sciencedirect.com/science/article/pii/0890540192900095 (cit. on pp. 8, 29).

[Moo05] Jonathan Moody. “Logical Mobility and Locality Types”. In: *Logic Based Program Synthesis and Transformation.* Ed. by Sandro Etalle. Vol. 3573. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 69–84. DOI: 10.1007/11506676_5 (cit. on pp. 1, 3, 7).

[Mur+04] Tom Murphy VII, Karl Crary, Robert Harper, and Frank Pfenning. “A Symmetric Modal Lambda Calculus for Distributed Computing”. In: *Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science.* LICS ’04. USA: IEEE Computer Society, 2004, pp. 286–295. ISBN: 0769521924 (cit. on p. 29).

[PD01] Frank Pfenning and Rowan Davies. “A judgmental reconstruction of modal logic”. In: *Mathematical structures in computer science* 11.4 (2001), pp. 511–540 (cit. on p. 3).

[PS96] Benjamin Pierce and Davide Sangiorgi. “Typing and subtyping for mobile processes”. In: *Mathematical Structures in Computer Science* 6.5 (1996), pp. 409–453. DOI: 10.1017/S096012950007002X (cit. on p. 29).

[PV21] Ivan Prokić and Hugo Torres Vieira. “The Cπ-calculus: A model for confidential name passing”. In: *Journal of Logical and Algebraic Methods in Programming* 119 (2021), p. 100622. ISSN: 2352-2208. DOI: https://doi.org/10.1016/j.jlamp.2020.100622. URL: https://www.sciencedirect.com/science/article/pii/S2352220820301073 (cit. on p. 29).

[Rav+03] António Ravara, Ana G. Matos, Vasco T. Vasconcelos, and Luís. Lopes. “Lexically scoped distribution: what you see is what you get”. In: *Electronic Notes in Theoretical Computer Science* 85.1 (2003). FGC, Foundations of Global Computing, 2nd EATCS Workshop (Satellite Event of ICALP 2003), pp. 61–79. ISSN: 1571-0661. DOI: https://doi.org/10.1016/S1571-0661(05)80088-X. URL: https://www.sciencedirect.com/science/article/pii/S157106610580088X (cit. on p. 29).

[San99] DAVIDE Sangiorgi. “From to ; or, Rediscovering continuations”. In: *Mathematical Structures in Computer Science* 9.4 (1999), pp. 367–401. DOI: 10.1017/S0960129599002881 (cit. on p. 30).

[SX14] Davide Sangiorgi and Xian Xu. “Trees from functions as processes”. In: *CONCUR 2014–Concurrency Theory: 25th International Conference, CONCUR 2014, Rome, Italy, September 2-5, 2014. Proceedings 25.* Springer. 2014, pp. 78–92 (cit. on p. 30).

[SS04] Alan Schmitt and Jean-Bernard Stefani. “The Kell Calculus: A Family of Higher-Order Distributed Process Calculi”. In: *International Conferences on Graph Computing.* 2004 (cit. on p. 29).

[Shi22] Michael Shirer. *Worldwide Public Cloud Services revenues grew 29.0% to $408.6 billion in 2021, according to IDC.* June 2022. URL: https://www.idc.com/getdoc.jsp?containerId=prUS49420022 (cit. on pp. 1, 2).

[Sim94] Alex K. Simpson. “The proof theory and semantics of intuitionistic modal logic”. In: 1994 (cit. on p. 29).

[SRS10] Anu Singh, C.R. Ramakrishnan, and Scott A. Smolka. “A process calculus for Mobile Ad Hoc Networks”. In: *Science of Computer Programming* 75.6 (2010), pp. 440–469. ISSN: 0167-6423. DOI: `https://doi.org/10.1016/j.scico.2009.07.008`. URL: `https://www.sciencedirect.com/science/article/pii/S0167642309001142` (cit. on p. 29).

[Vas05] Vasco Thudichum Vasconcelos. “Lambda and pi calculi, CAM and SECD machines”. In: *Journal of Functional Programming* 15.1 (2005), pp. 101–127. DOI: `10.1017/S0956796804005386` (cit. on p. 30).

Appendix A

A.1 Rules made redundant due to reduction contexts

$$\textsc{D-App-1}\;\dfrac{M \longmapsto M'}{M(N) \longmapsto M(N)} \qquad \textsc{D-App-2}\;\dfrac{N \longmapsto N'}{V(N) \longmapsto V(N')} \qquad \textsc{D-LetBox-1}\;\dfrac{M \longmapsto M'}{\mathsf{let\ box}\ u \Leftarrow M\ \mathsf{in}\ N \longmapsto \mathsf{let\ box}\ u \Leftarrow M'\ \mathsf{in}\ N}$$

A.2 Example of the typing rules

$$\dfrac{\dfrac{\dfrac{\dfrac{}{\cdot;\cdot, f : \mathsf{Num} \to \mathsf{Num} \vdash f : \mathsf{Num} \to \mathsf{Num}} \qquad \dfrac{2 \in \mathbb{N}}{\cdot;\cdot \vdash \overline{2} : \mathsf{Num}}}{\cdot;\cdot, f : \mathsf{Num} \to \mathsf{Num} \vdash f(\overline{2}) : \mathsf{Num}}}{\cdot;\cdot \vdash \lambda f : \mathsf{Num} \to \mathsf{Num}.\, f(\overline{2}) : (\mathsf{Num} \to \mathsf{Num}) \to \mathsf{Num}} \qquad \dfrac{}{\cdot, g : \mathsf{Num} \to \mathsf{Num}; \cdot \vdash g : \mathsf{Num} \to \mathsf{Num}}}{\cdot; g : \mathsf{Num} \to \mathsf{Num} \vdash (\lambda f : \mathsf{Num} \to \mathsf{Num}.\, (f)(\overline{2}))(g) : \mathsf{Num}}$$

(a) Typing derivation tree of the term $(\lambda f : \mathsf{Num} \to \mathsf{Num}.\, f(\overline{2}))(g)$ of type Num with local context having $g : \mathsf{Num} \to \mathsf{Num}$.

$$\dfrac{\dfrac{\dfrac{3 \in \mathbb{N}}{\cdot;\cdot \vdash \overline{3} : \mathsf{Num}}}{\cdot;\cdot \vdash \mathsf{box}\ \overline{3} : \Box\mathsf{Num}} \qquad \dfrac{\dfrac{}{u : \mathsf{Num}; x : \mathsf{Num} \vdash x : \mathsf{Num}}}{u : \mathsf{Num}; \cdot \vdash \lambda x : \mathsf{Num}.\, x : \mathsf{Num} \to \mathsf{Num}}}{\cdot;\cdot \vdash \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ \overline{3}\ \mathsf{in}\ \lambda x : \mathsf{Num}.\, x : \mathsf{Num} \to \mathsf{Num}}$$

(b) Typing derivation tree for the term $\mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ \overline{3}\ \mathsf{in}\ \lambda x : \mathsf{Num}.\, x$ of type $\mathsf{Num} \to \mathsf{Num}$.

Figure A.2.1: Examples of typing derivation trees.

A.3 Addition dynamic rules used in examples

$$\textsc{D-Plus}\;\dfrac{n_1 + n_2 = n}{\mathrm{plus}\,(\overline{n_1}; \overline{n_2}) \longmapsto \overline{n}} \qquad \textsc{D-Succ}\;\dfrac{n_1 + 1 = n}{\mathrm{succ}\,(\overline{n_1}) \longmapsto \overline{n}}$$

A.4 Examples of dynamics of the abstract machine

$$\mathcal{E} = [\,]$$

$$\nu b.\left(\nu a.\left(\langle \overline{3} : a\rangle \;\middle|\; \langle ?a : b\rangle\right)\right) \equiv \nu b.\left(\nu a.\left(\langle ?a : b\rangle \;\middle|\; \langle \overline{3} : a\rangle\right)\right)$$

by M-Recv

$$\longrightarrow \nu b.\left(\nu a.\left(\langle \overline{3} : b\rangle\right)\right)$$

$$\equiv \nu a.\left(\nu b.\left(\langle \overline{3} : b\rangle\right)\right)$$

$$\equiv \nu a.\,(\mathbf{0})$$

$$\equiv \mathbf{0}$$

$$\mathcal{E} = [\,]$$

$$\langle \mathsf{let\ box}\ u \Leftarrow \mathsf{box\ succ}(\overline{3})\ \mathsf{in}\ u : a\rangle$$by M-BoxBeta

$$\longrightarrow \nu b.\left(\left\langle u[?b/u] : a\right\rangle \;\middle|\; \langle \mathsf{succ}(\overline{3}) : b\rangle\right)$$

by Succ

$$\longrightarrow \nu b.\left(\left\langle u[?b/u] : a\right\rangle \;\middle|\; \langle \overline{4} : b\rangle\right)$$

$$\equiv \nu b.\left(\langle ?b : a\rangle \;\middle|\; \langle \overline{4} : b\rangle\right)$$

by M-Recv

$$\longrightarrow \nu b.\left(\langle \overline{4} : a\rangle \;\middle|\; \langle \overline{4} : b\rangle\right)$$

$$\equiv \langle \overline{4} : a\rangle \;\middle|\; \nu b.\left(\langle \overline{4} : b\rangle\right)$$

$$\equiv \langle \overline{4} : a\rangle \;\middle|\; \mathbf{0}$$

$$\equiv \langle \overline{4} : a\rangle$$

$$\mathcal{E} = (\lambda x : A.\, x)([\,]) \qquad A \equiv \mathsf{Num}$$

$$\langle (\lambda x : A.\, x)((\lambda x : A.\, x)(\overline{9})) : a\rangle \equiv \left\langle \mathcal{E}[(\lambda x : A.\, x)(\overline{9})] : a\right\rangle$$

by M-Beta

$$\longrightarrow \left\langle \mathcal{E}\left[x[\overline{9}/x]\right] : a\right\rangle$$

$$\equiv \left\langle \mathcal{E}[\overline{9}] : a\right\rangle$$

$$\equiv \langle (\lambda x : A.\, x)(\overline{9}) : a\rangle$$

$$\mathcal{E} = [\,]$$

by M-Beta

$$\longrightarrow \left\langle x[\overline{9}/x] : a\right\rangle$$

$$\equiv \langle \overline{9} : a\rangle$$

$\mathcal{E} = \mathsf{let\ box}\ u \Leftarrow [\,]\ \mathsf{in}\ ((\lambda x : \mathsf{Num}.\, \mathsf{succ}(x))(u)) \qquad A \equiv \mathsf{Num}$

$$\langle \mathsf{let\ box}\ u \Leftarrow ((\lambda x : A.\, \mathsf{box}\ x)(\overline{5}))\ \mathsf{in}\ ((\lambda x : A.\, \mathsf{succ}(x))(u)) : a\rangle \equiv \Big\langle \mathcal{E}[(\lambda x : A.\, \mathsf{box}\ x)(\overline{5})] : a \Big\rangle$$

by M-Beta

$$\longrightarrow \Big\langle (\mathcal{E}[\mathsf{box}\ x])[\overline{5}/x] : a \Big\rangle$$

$$\equiv \Big\langle \mathcal{E}[\mathsf{box}\ \overline{5}] : a \Big\rangle$$

$$\equiv \langle \mathsf{let\ box}\ u \Leftarrow \mathsf{box}\ \overline{5}\ \mathsf{in}\ ((\lambda x : A.\, \mathsf{succ}(x))(u)) : a\rangle$$

$\mathcal{E} = [\,]$

by M-BoxBeta

$$\longrightarrow \nu b.\, \Big(\langle (\lambda x : A.\, \mathsf{succ}(x))(?b) : a\rangle \,\Big|\, \langle \overline{5} : b\rangle \Big)$$

by M-Recv

$$\longrightarrow \nu b.\, \Big(\langle (\lambda x : A.\, \mathsf{succ}(x))(\overline{5}) : a\rangle \,\Big|\, \langle \overline{5} : b\rangle \Big)$$

by M-Beta

$$\longrightarrow \nu b.\, \Big(\Big\langle \mathsf{succ}(x)[\overline{5}/x] : a \Big\rangle \,\Big|\, \langle \overline{5} : b\rangle \Big)$$

$$\equiv \nu b.\, \Big(\langle \mathsf{succ}(\overline{5}) : a\rangle \,\Big|\, \langle \overline{5} : b\rangle \Big)$$

by Succ

$$\longrightarrow \nu b.\, \Big(\langle \overline{6} : a\rangle \,\Big|\, \langle \overline{5} : b\rangle \Big)$$

$$\equiv \langle \overline{6} : a\rangle \,\Big|\, \nu b.\, \Big(\langle \overline{5} : b\rangle \Big)$$

$$\equiv \langle \overline{6} : a\rangle \,\Big|\, \nu b.\, (\mathbf{0})$$

$$\equiv \langle \overline{6} : a\rangle \,\Big|\, \mathbf{0}$$

$$\equiv \langle \overline{6} : a\rangle$$

Figure A.4.2: Examples of dynamics of the abstract machine.