Untyped polarized calculi

Xavier MONTILLET

July 20, 2023

0. Introduction

The goal of this thesis is to provide an introduction to polarized L calculi and to demonstrate their usefulness in studying untyped λ-calculi.

0.1. Motivation

The study of programming languages aims at making reasoning about the behavior of programs easier, and at identifying desirable properties for future programming languages. When studied formally, programming languages are equipped with a semantics, i.e. a map that assigns to each program a mathematical object that represents its behavior. The semantics then induces an equivalence relation: two programs are considered equivalent when they have the same semantics. Some aspects of the behavior of programs can be either useful or superfluous depending on the context. For example, the time a program takes to compute its result is irrelevant when reasoning about its adherence to a specification, but crucial when trying to optimize the program. This leads to some programming languages having several semantics, ranging from loose ones that account for very few aspects of the behavior and are easy to reason about, to more precise ones that account for more aspects of the behavior but are more complex.

One very desirable property of a semantics is compositionality: program fragments should also have a semantics, and the semantics of the whole program should be expressible in terms of the semantics of its fragments. For example, to get the smallest element of a list, we can write a program that sorts the list and returns the first element of the sorted list, and this works independently of the how exactly the list is sorted. The existence of a compositional semantics is a fundamental property for programming languages because it allows for large collaborative programs without requiring each individual contributor to understand every part of the program in details. The execution of a program by a computer is an inherently non-compositional process because any operation can a priori observe any part of the state of the computer. This leads to some low-level programming languages suffering from a lack of compositionality, e.g. assembly languages or those that use the `goto` statement [Dij68]. This lead to the introduction of high-level programming languages that encourage writing programs in a compositional way by disallowing the natural non-compositional ways of writing programs and providing compositional abstractions as an alternative.

One of the most popular and widely spread of those abstractions is the concept of function that allows writing program fragments that takes some inputs, and uses them to compute some output. Functions can be through of as a sort of restricted `goto` statements that eventually returns to where it started[1]. This restriction makes reasoning on what happens after calling a function much easier than on what happens after a `goto` statement: we know that whatever instruction is placed after a function call will eventually be executed[2]. The λ-calculus [Bar84] is a bare-bones programming language used to study the expressiveness of functions. Its bare-bones nature makes studying it mathematically easier, but unsuitable to write complex programs, which is why real-world programming languages based on the λ-calculus extend it with some datatypes (e.g. numbers) and operations (e.g. addition). While those additional operations can be encoded into the λ-calculus (just like functions can be encoded with `goto` statements), the encodings can be used in more ways than intended, which

[1] Modulo termination.
[2] Again modulo termination.

[Dij68] "Letters to the Editor: Go to Statement Considered Harmful", Dijkstra, 1968
[Bar84] *The lambda calculus: its syntax and semantics*, Barendregt, 1984

makes them harder to reason about. In the words of Robert Harper[3]:

> The expressive power of a programming language is derived from its strictures, not its affordances.

When trying to study programming languages with additional datatypes, a new difficulty appears: scalability. Indeed, some proofs scale quadratically in the number of datatypes, and hence become unmanageable as soon as a handful of datatypes are added. In a typed setting, it is well-known that many proofs are easier in sequent calculi than in natural deduction. In this thesis, we look at the untyped counterpart of this statement, i.e. we compare two untyped calculi: the sequent-calculus-inspired $\overline{\lambda}\mu\widetilde{\mu}$-calculus [CurHer00], and the natural-deduction-like λ-calculus. It turns out that, while the $\overline{\lambda}\mu\widetilde{\mu}$-calculus has a higher initial cost of entry, it scales much better when adding datatypes[4], elucidates the connections between several well-known variants of the λ-calculus[5], and suggests new better-behaved variants[6].

[3]This is a quote I remember hearing at OPLSS 2019. A similar sentence can be found in an email by Robert Harper on the TYPES mailing list:

> The power of a type system arises from its strictures, which can be selectively relaxed, not its affordances, which sacrifice the ability to draw sharp distinctions.

[4]Many definitions and proofs scale quadratically in the number of datatype constructors in the λ-calculus, and only linearly in the $\overline{\lambda}\mu\widetilde{\mu}$-calculus.

[5]For example, in $\overline{\lambda}\mu\widetilde{\mu}$, the distinction between evaluating with the head reduction or with the weak head reductions in call-by-name can be understood as being dual to the distinction between evaluating open expressions or closed expressions in call-by-value.

[6]This includes our calculus $\boldsymbol{\lambda}_{\mathrm{P}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$ which can be seen as a version of Call-by-push-value [Lev04; Lev06] with what Levy calls "complex values", and our dynamically typed calculus $\boldsymbol{\lambda}_{\mathrm{N}}^{\mathcal{PN}}$ that avoids clashes while remaining untyped.

[CurHer00] "The duality of computation", Curien and Herbelin, 2000

0.2. Background

0.2.1. Calculi

λ-calculi and Call-by-push-value The λ-calculus [Bar84] is a well-known abstraction used to study programming languages. It has two main evaluation strategies: *call-by-name* (CBN) evaluates arguments only when they are used, while *call-by-value* (CBV) evaluates arguments immediately. Each strategy has its own advantage: call-by-name ensures that no unnecessary computations are done, while call-by-value ensures that no computations are done more than once. We write $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ and $\boldsymbol{\lambda}^{\to}_{\mathrm{V}}$ for the call-by-name and call-by-value λ-calculi respectively. Each strategy induces two reductions: the strong reduction $-\!\!\triangleright$ that can reduce anywhere in the expression, an the operational reduction $\triangleright$ (often called the weak head reduction) that never reduces under λ-abstractions and is deterministic. While the strong reduction is the most common in the literature, the operational reduction is more closely related to real-world programming languages [Ong88; Abr90].

The call-by-name λ-calculus has be thoroughly studied [Bar84] and is well-understood. By contrast, the current understanding of the call-by-value lags behind. This is due to its study being more involved than that of call-by-name, for example requiring computation monads [Mog89; Mog91] to build models, and σ-reductions / commuting conversions to get a well-behaved reduction on open expressions [AccGue16; AccPao12; PaoRon99; GarNog16]. *Call-by-push-value* (CBPV) [Lev04; Lev06] decomposes Moggi's computation monad as an adjunction, subsumes both call-by-name and call-by-value, and sheds some light on the interactions and differences of both strategies. CBPV also adds some datatypes (sums and pairs), and its pure fragment has been studied under the name Bang calculus [EhrGue16; BucKesRíoVis20].

The $\bar{\lambda}\mu\tilde{\mu}$-calculus Another direction the λ-calculus has evolved in is the computational interpretation of classical logic, with continuation-passing style translations and the $\lambda\mu$-calculus [Par92]. This eventually led to the $\bar{\lambda}\mu\tilde{\mu}$-calculus [CurHer00], which can be understood as denoting proofs in the sequent calculus, just like λ-terms denote proofs in natural deduction. An interesting property of the $\bar{\lambda}\mu\tilde{\mu}$-calculus is that it resembles both the

[Bar84] *The lambda calculus: its syntax and semantics*, Barendregt, 1984
[Ong88] "Fully Abstract Models of the Lazy Lambda Calculus", Ong, 1988
[Abr90] "The lazy lambda calculus", Abramsky, 1990
[Mog89] "Computational Lambda-Calculus and Monads", Moggi, 1989
[Mog91] "Notions of Computation and Monads", Moggi, 1991
[AccGue16] "Open Call-by-Value", Accattoli and Guerrieri, 2016
[AccPao12] "Call-by-Value Solvability, Revisited", Accattoli and Paolini, 2012
[PaoRon99] "Call-by-value Solvability", Paolini and Ronchi Della Rocca, 1999
[GarNog16] "No solvable lambda-value term left behind", García-Pérez and Nogueira, 2016
[Lev04] *Call-By-Push-Value: A Functional/Imperative Synthesis*, Levy, 2004
[Lev06] "Call-by-push-value: Decomposing call-by-value and call-by-name", Levy, 2006
[EhrGue16] "The Bang Calculus: An Untyped Lambda-Calculus Generalizing Call-by-Name and Call-by-Value", Ehrhard and Guerrieri, 2016
[BucKesRíoVis20] "The Bang Calculus Revisited", Bucciarelli *et al.*, 2020
[Par92] "$\lambda\mu$-Calculus: An algorithmic interpretation of classical natural deduction", Parigot, 1992
[CurHer00] "The duality of computation", Curien and Herbelin, 2000

λ-calculus and the Krivine abstract machine [Kri07; CurMun10; MunSch15], which makes it suitable to study both the equational theory and the operational semantics. The full $\overline{\lambda}\mu\widetilde{\mu}$-calculus is not confluent, but two natural fragments, the call-by-name and call-by-value fragments, are. Further restricting those to their intuitionistic fragments yields calculi that correspond to the call-by-name and call-by-value λ-calculi. Since call-by-value is syntactically dual to call-by-name in the full $\overline{\lambda}\mu\widetilde{\mu}$-calculus [CurHer00; DowAri18], the additional difficulty in the study of call-by-value can be understood as stemming from the restriction to the intuitionistic fragment which breaks this symmetry.

Polarized sequent calculi Those two lines of work (CBPV and $\overline{\lambda}\mu\widetilde{\mu}$) can be combined into a polarized sequent calculus LJ_p^η [CurFioMun16] or $\mathrm{L}_{\mathrm{int}}$ [MunSch15]. It inherits all the advantages of CBPV (subsumes CBV and CBN without loss of confluence, allows both strategies to interact, has nice models, has nice η-rules for functions, pairs and sums, ...) and of $\overline{\lambda}\mu\widetilde{\mu}$ (CBV and CBN are dual, has a simple top-level reduction that generalizes both movements of the focus inside expressions of abstract machines and commuting conversions, has classical logic built-in but can easily be restricted to intuitionistic logic, ...).

0.2.2. Solvability in arbitrary programming languages

Observational equivalence and preorder The compilation of programs often involves many optimizations where some parts of the programs are replaced by faster ones. The soundness of those transformations is studied in a compositional way by using an *observational equivalence*: two expressions, i.e. program fragments, are said to be observationally equivalent when replacing one by the other never changes the observable behavior of the encompassing program. The observational equivalence is often refined to an *observational preorder* that takes into account that some replacements are sound in one direction but not in the other, i.e. that some expressions are strictly better than others.

Operational relevance and solvability The study of the observational equivalence often relies on two notions that it preserves:

O*perationally relevant* expressions are those that can be used to form a program that returns a result on at least one input, i.e. those that are not completely useless. Expressions that are not operationally relevant are called *operationally irrelevant* and are often exactly the least elements of the observational preorder.

Solvable expressions are those that can be used to form programs of any chosen behavior, and expressions that are not solvable are called *unsolvable*. The intuition behind solvability is that it is an indirect way of stating that the expression computes some intermediate result

[Kri07] "A call-by-name lambda-calculus machine", Krivine, 2007
[CurMun10] "The duality of computation under focus", Curien and Munch-Maccagnoni, 2010
[MunSch15] "Polarised Intermediate Representation of Lambda Calculus with Sums", Munch-Maccagnoni and Scherer, 2015
[CurHer00] "The duality of computation", Curien and Herbelin, 2000
[DowAri18] "A tutorial on computational classical logic and the sequent calculus", Downen and Ariola, 2018
[CurFioMun16] "A Theory of Effects and Resources: Adjunction Models and Polarised Calculi", Curien, Fiore, and Munch-Maccagnoni, 2016

that can be observed internally. Indeed, to use a solvable expression in a way that yields a chosen behavior, it suffices to observe that intermediate result, and then execute another program with the chosen behavior if the expected intermediate result was observed.

The central role of unsolvability In the call-by-name λ-calculus, the unsolvable expressions are exactly the operationally irrelevant ones. They are completely useless for writing actual programs, but are very useful for many theoretical purposes because they are a much more resilient notion of "undefined" than "being non-terminating". Quoting from [AccPao12] (itself quoting from [Wad76]):

> [...] only those expressions without normal forms which are in fact unsolvable can be regarded as being "undefined" (or better now: "totally undefined"); by contrast, all other expressions without normal forms are at least partially defined. Essentially the reason is that unsolvability is preserved by application and composition [...] which [...] is not true in general for the property of failing to have a normal form.

This leads to unsolvability being a central notion when studying λ-definability, λ-theories, the observational equivalence, or Böhm trees. When studying λ-theories (i.e. congruences on the λ-calculus that contain β-reduction), this manifests as the fact that any λ-theory that equates all expressions without a normal form is inconsistent (i.e. it is a trivial theory that identifies all expressions), while there are consistent λ-theories that equate all unsolvable expression. When studying λ-definability [dVri16] (i.e. encodings of partial recursive functions in the λ-calculus) the partiality of the function is represented by mapping inputs for which it is undefined to some "undefined" expressions of the λ-calculus. While it is possible to define "undefined" as meaning "having no normal form", the corresponding encoding is not compositional: the encoding of the composition of two partial functions can not be not encoded as the composition of the encodings. Defining "undefined" as meaning unsolvable instead allows for the definition of a compositional encoding.

Unary operational completeness In some programming languages, operational relevance and solvability are equivalent. With the intuition given above for solvability, this corresponds to saying that any (external) result of a program can be observed internally, i.e. can be used as an intermediate result. This can be though of as being a sort of internal completeness, which we call *unary*[7] *operational completeness*.

A programming language that does not have unary operational completeness (e.g. one where the result of a program can be an uncatchable exception) can be thought of as having either too many operationally relevant expressions or too few solvable expressions. There

[7] We call this unary operational completeness because it does not imply binary operational completeness, i.e. the equivalence between the corresponding binary notions: external and internal separability. See Part C.

[AccPao12] "Call-by-Value Solvability, Revisited", Accattoli and Paolini, 2012

[Wad76] "The Relation Between Computational and Denotational Properties for Scott's D_{infty}-Models of the Lambda-Calculus", Wadsworth, 1976

[dVri16] "On Undefined and Meaningless in Lambda Definability", de Vries, 2016 are therefore two approaches to recovering unary operational completeness: the *restrictive*[8] [AbrOng93] approach restrict the notion of operational relevance; and the *expansive*[8] approach expands the notion of solvability. For example, a lack of unary operational completeness that are due to uncatchable exceptions being results can be treated either by making the uncatchable exceptions operationally irrelevant by no longer considering them as results, or by making them solvable by adding try-catch statements to the language.

Operational characterization of solvability For translations between two programming languages for which it holds, preservation of operational relevance or solvability can often be proven directly by looking at the image of reductions and normal forms through the translation, while preservation of operational irrelevance or unsolvability is often harder to prove. For example, if the translation simply embeds a programming language in its extension, operationally relevance is clearly preserved and solvability most likely is too, but this is not necessarily the case for operational irrelevance and unsolvability: the extension can add new ways of using or observing some previously operationally irrelevant or unsolvable expressions.

One way to prove that operational irrelevance or unsolvability are preserved is to use an *operational characterization of operational relevance* (resp. *solvability*), i.e. a reduction $\rightsquigarrow$ such that weak $\rightsquigarrow$-normalization, strong $\rightsquigarrow$-normalization, and operational relevance (resp. solvability) are equivalent. Given operational characterizations $\rightsquigarrow_1$ and $\rightsquigarrow_2$ of operational relevance (resp. solvability) in the source and target programming languages, to show that a translation preserves operational irrelevance (resp. unsolvability), it suffices to show that it sends infinite $\rightsquigarrow_1$ reduction sequences to infinite $\rightsquigarrow_2$ reduction sequences, which is often fairly easy.

0.2.3. Solvability in λ-calculi

In the untyped λ-calculus, the observational equivalence is defined as only observing expressionination, i.e. two expressions are observationally equivalent when replacing either by the other in an expressioninating (resp. diverging) program can not make the program diverge (resp. expressioninate). While this definition of observational equivalence could a priori identify too many expressions, it ends-up distinguishing any expressions we could want to use as inputs or outputs (e.g. Church encodings [Chu85] of natural numbers). A solvable expression is one that can be used to reach any expression (or equivalently any normal form), and an operationally relevant expression[9] is one that can be used to reach at least one normal form.

[8] These two words are used in [AbrOng93] to describe ways of rectifying a "poorness of fit" between a language and its model. Here, we have no model, but we can think of the language equipped with its observational preorder as being a sort of initial model. Since the observational preorder respects extenal observations, the intuition of operational relevance (resp. solvability) being about external (resp. internal) results casts operational relevance (resp. solvability) as slightly more on the semantic (resp. syntactic).

[9] In the litterature, the notion of operational relevance is mostly used informally, and formal notions of what we would call operational relevance are often called solvability.

[Chu85] *The Calculi of Lambda Conversion. (AM-6) (Annals of Mathematics Studies)*, Church, 1985

Those notions of course depend on the reduction $\rightsquigarrow$ used to evaluate the expressions, so we make this dependency explicit: given a reduction $\rightsquigarrow$, we write $\approx_{\rightsquigarrow}$ for the induced observational equivalence, and call $\rightsquigarrow$-solvability (resp. $\rightsquigarrow$-operational relevance) the induced notions of solvability and operational relevance. There are five main reductions that appear in the litterature: $\rhd_N$, $\overset{h}{\mathrel{-\!\!\rhd}}_N$, $\mathrel{-\!\!\rhd}_N$, $\rhd_V$, and $\mathrel{-\!\!\rhd}_V$. The reduction $\mathrel{-\!\!\rhd}_N$ (resp. $\mathrel{-\!\!\rhd}_V$) is the strong call-by-name (resp. call-by-value) reduction, i.e. the call-by-name (resp. call-by-value) reduction that can reduce anywhere in the expression; the reduction $\rhd_N$ (resp. $\rhd_V$) is the call-by-name (resp. call-by-value) operational reduction[10] that more closely models how expressions are evaluated in a real-world call-by-name (resp. call-by-value) programming language; and the reduction $\overset{h}{\mathrel{-\!\!\rhd}}_N$ is a call-by-name reduction such that

$$\rhd_N \subsetneq \overset{h}{\mathrel{-\!\!\rhd}}_N \subsetneq \mathrel{-\!\!\rhd}_N$$

called the (call-by-name) head reduction.

Call-by-name solvability In call-by-name, the observational equivalence $\approx_{\rhd_N}$ induced by the call-by-name operational reduction $\rhd_N$ is Abramsky's one [Abr90] (in the so-called lazy λ-calculus); the observational equivalence $\approx_{\overset{h}{\mathrel{-\!\!\rhd}}_N}$ induced by the head reduction $\overset{h}{\mathrel{-\!\!\rhd}}_N$ is Wadsworth' one [Wad76], and the observational equivalence $\approx_{\mathrel{-\!\!\rhd}_N}$ induced by the call-by-name strong reduction $\mathrel{-\!\!\rhd}_N$ is Morris' one [Mor69][11]. It is well-known that there are strict inclusions[12] [DezGio01; Bar84; IntManPol17]

$$\approx_{\rhd_N} \subsetneq \approx_{\mathrel{-\!\!\rhd}_N} \subsetneq \approx_{\overset{h}{\mathrel{-\!\!\rhd}}_N}$$

The 6 call-by-name notions of $\rightsquigarrow$-solvability and $\rightsquigarrow$-operational relevance induced by the 3 call-by-name reductions we consider are related as depicted in Figure 0.2.1, where equivalent notions are placed in the same node, and implications between non-equivalent notions are depicted by arrows $\Rightarrow$. Note that both notions have an operational characterization: the stronger notion is operationally characterized by the head reduction $\overset{h}{\mathrel{-\!\!\rhd}}_N$, while the weaker one is operationally characterized by the operational reduction $\rhd_N$. Also note that using either the head reduction $\overset{h}{\mathrel{-\!\!\rhd}}_N$ or the strong reduction $\mathrel{-\!\!\rhd}_N$ yields a calculus that has unary operational completeness, but that using the operational reduction $\rhd_N$ does not.

The lack of unary completeness when using the operational reduction $\rhd_N$ is due to all λ-

[10] In call-by-value, to get a deterministic reduction, we need to further restrict to either left-to-right or right-to-left evaluation (depending on whether we want to evaluate functions or their arguments first). Both restrictions work for our purposes.

[11] And can alternatively be defined by observing normal forms modulo η (or equivalently $\beta\eta$-normal forms) [Mor69].

[12] The strictness of the inclusions can be understood as stemming from a differences of strength between their respective versions of η-conversion on Böhm trees [IntManPol17].

[Abr90] "The lazy lambda calculus", Abramsky, 1990

[Wad76] "The Relation Between Computational and Denotational Properties for Scott's D_{infty}-Models of the Lambda-Calculus", Wadsworth, 1976

[Mor69] "Lambda Calculus Models of Programming Languages", Morris, 1969

[DezGio01] "From Böhm's Theorem to Observational Equivalences: an Informal Account", Dezani-Ciancaglini and Giovannetti, 2001

[Bar84] *The lambda calculus: its syntax and semantics*, Barendregt, 1984

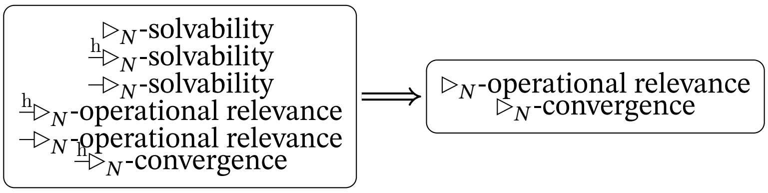


Figure 0.2.1: Notions of $\rightsquigarrow$-solvability and $\rightsquigarrow$-operational relevance in call-by-name

abstractions being $\rhd_N$-operationally relevant while some of them are $\rhd_N$-unsolvable[13]. The notion of order of an expression (which is more or less its arity) allows to relate both notions more precisely: $\rhd_N$-operationally irrelevant expressions are exactly $\rhd_N$-unsolvable expressions of order 0.

Trying to restore unary operational completeness using the restrictive approach would mean preventing (at least) some λ-abstractions from being $\rhd_N$-operationally relevant, e.g. by replacing the reduction by the strong reduction $-\!\!\rhd_N$ or the head reduction $\overset{h}{-\!\!\rhd}_N$. Using the expansive approach would mean adding a new construction that allows testing whether an expression is a λ-abstraction, e.g. an if-lambda conditional or a call-by-value let-expression. We could not find such an extension in the literature, and do not study it directly either[14].

Call-by-value solvability

[13] For example, given a (closed) $\rhd_N$-diverging expression T (e.g. $\Omega \overset{\text{def}}{=} \delta\delta$ where $\delta \overset{\text{def}}{=} \lambda y.xx$), the expression $\lambda x.T$ is $\rhd_N$-operationally relevant (because it is $\rhd_N$-normal) but $\rhd_N$-unsolvable (because whenever it is given an argument, it $\rhd_N$-diverges).

[14] However, the embeddings of the call-by-value λ-calculus into our polarized λ-calculus described in can be understood as a way of adding such an operation.

0.3. Content

Parts A and B and currently being cleaned up and should be available in their entirety soon. The first two chapters of part C should follow shortly thereafter. The last chapter will most likely not be part of the official thesis, but should eventually appear.

0.4. Notations

Reduction sequences A reduction $\rightsquigarrow$ on a set $\mathbf{X}$ is defined as being a subsets of the Cartesian square of $\mathbf{X}$, i.e. $\rightsquigarrow \subseteq \mathbf{X} \times \mathbf{X}$. We say that O $\rightsquigarrow$-reduces to O', and write $O \rightsquigarrow O'$, when $(O, O') \in \rightsquigarrow$. We say that O is $\rightsquigarrow$-reducible (resp. $\rightsquigarrow$-normal), and write $O \rightsquigarrow$ (resp. $O \not\rightsquigarrow$) when there exists (resp. does not exist) O' such that $O \rightsquigarrow O'$, i.e. when O is (resp. is not) in the domain of $\rightsquigarrow$. More generally, we write $O_0 \rightsquigarrow_1 O_1 \rightsquigarrow_2 O_2 \rightsquigarrow_3 \dots \rightsquigarrow_n O_n$ for $\forall k \in \{1, \dots, n\}, O_{k-1} \rightsquigarrow O_k$, and any missing object should be understood as being quantified existentially, e.g. $O \rightsquigarrow_1 \rightsquigarrow_2 O''$ stands for $\exists O', O \rightsquigarrow_1 O' \rightsquigarrow_2 O''$. We write $\rightsquigarrow^=$ (resp. $\rightsquigarrow^+$, $\rightsquigarrow^*$) for the reflexive (resp. transitive, reflexive transitive) closure of $\rightsquigarrow$. We write $O \rightsquigarrow^{\circledast} O'$ for $O \rightsquigarrow^* O' \not\rightsquigarrow$, $O \rightsquigarrow^{\circledast}$ for the existence of a finite maximal $\rightsquigarrow$-reduction sequence starting at O, and $O \rightsquigarrow^{\omega}$ for the existence of an infinite $\rightsquigarrow$-reduction sequence $O \rightsquigarrow O' \rightsquigarrow O'' \rightsquigarrow \dots$ starting at O. The inverse (as a binary relation) of a reduction $\rightsquigarrow$ is denoted by reflecting the symbol along a vertical line: $O \rightsquigarrow O'$ is equivalent to $O' \leftsquigarrow O$.

Main reductions We use four tip symbols for reductions: $\rhd$ for β-reduction, $\boldsymbol{\Sigma}$ for σ-reduction, $\dagger$ for η-expansion, and $\Rrightarrow$ for an arbitrary reduction. Each symbol is combined with a vertical line to denote the operational variant of the reduction (i.e. the one relevant to study evaluation), and with a tail to denote its equational variant (i.e. the one that can reduce anywhere in the expression and is relevant to the study of the equational theory):

	β-reduction	σ-reduction	η-expansion	Arbitrary
Top-level	$\rhd$	$\boldsymbol{\Sigma}$	$\dagger$	$\Rrightarrow$
Operational	$\mid\!\rhd$	$\mid\!\boldsymbol{\Sigma}$	$\mid\!\dagger$	$\mid\!\Rrightarrow$
Strong	$\multimap\!\rhd$	$\multimap\!\boldsymbol{\Sigma}$	$\multimap\!\dagger$	$\rightsquigarrow\!\Rrightarrow$

Unions of some of these reductions are denoted by superimposing the symbols, e.g. the strong $\beta\sigma$-reduction is $\multimap\!\rhd\!\boldsymbol{\Sigma} = \multimap\!\rhd \cup \multimap\!\boldsymbol{\Sigma}$, the strong $\beta\eta$-reduction is $\multimap\!\dagger\!\rhd = \multimap\!\rhd \cup \multimap\!\dagger$, and the strong β-reduction combined with the strong η-expansion is $\multimap\!\dagger\!\rhd = \multimap\!\rhd \cup \multimap\!\dagger$.

Some other closures of $\rhd$ will be used often, and they will be denoted by $\multimap\!\rhd$ with symbols on the tail: **t** for **t**op-level, **o** for **o**perational, **h** for **h**ead, **a** for **a**head, **lo** for **l**eftmost **o**utermost, **s** for **s**trong, and $\neg$ for "and not". For example, $\xrightarrow{\text{h}}\!\rhd$ is the head reduction, and $\xrightarrow{\text{s}\neg\text{h}}\!\rhd$ (or $\xrightarrow{\neg\text{h}}\!\rhd$) is the non-head reduction.

Closure of reductions under contexts More generally, given an arbitrary set of contexts $\mathbf{X}$ (i.e. expressions with a hole $\square$) and an arbitrary reduction $\rightsquigarrow$, we call *closure*[15] *of* $\rightsquigarrow$ *under* $\mathbf{X}$ the reduction

$$\mathbf{X}\boxed{\rightsquigarrow} \stackrel{\text{def}}{=} \{(\mathbb{K}\boxed{O}, \mathbb{K}\boxed{O'}) \mid \mathbb{K} \in \mathbf{X} \text{ and } (O, O') \in \rightsquigarrow\}$$

(where $\mathbb{K}\boxed{O}$ denotes the result of plugging O in the hole $\square$ of the context $\mathbb{K}$) that allows $\rightsquigarrow$ reductions under contexts $\mathbb{K} \in \mathbf{X}$. When the reduction $\rightsquigarrow$ is denoted by one of the four tip symbols ($\rhd$, $\boldsymbol{\Sigma}$, $\dagger$, or $\Rrightarrow$), we also denote this closure by using the symbol for the corresponding strong reduction (i.e. $\multimap\!\rhd$, $\multimap\!\boldsymbol{\Sigma}$, $\multimap\!\dagger$, or $\rightsquigarrow\!\Rrightarrow$) and placing $\mathbf{X}$ over its tail, e.g.

$$\xrightarrow{\mathbf{X}}\!\rhd \stackrel{\text{ntn}}{=} \mathbf{X}\boxed{\rhd} \quad \text{and} \quad \overset{\mathbf{X}}{\rightsquigarrow}\!\Rrightarrow \stackrel{\text{ntn}}{=} \mathbf{X}\boxed{\Rrightarrow}$$

[15] For some sets $\mathbf{X}$, the induced operation is not really a closure because it is not idempotent. For it to be idempotent, it suffices for $\mathbf{X}$ to be closed under composition.

The previously given notations are instances of this, e.g. $\rightsquigarrow\!\rhd$ is $\overset{\mathbf{X}}{\rightsquigarrow}\!\rhd$ with $\mathbf{X}$ left implicit because it is the set of all contexts $\mathbf{K}$, and the symbols above the tail of $\mathrel{-\!\!\rhd}$ denote sets of contexts, e.g. $\overset{\mathrm{a}}{\mathrel{-\!\!\rhd}}$ is the closure $\overset{\mathbf{A}}{\mathrel{-\!\!\rhd}}$ of $\rhd$ under the set $\mathbf{A}$ of ahead contexts. Note that the negation symbol over tails denotes a set difference on contexts, e.g.

$$\xrightarrow{\mathrm{s}\neg\mathrm{o}}\!\!\rhd \;=\; \xrightarrow{\mathbf{K} \setminus \mathbf{O}}\!\!\rhd$$

(and not a set difference on the reductions[16]).

Subscripts should be though of as commuting with closures when it makes sense, e.g. $\mathrel{-\!\!\rhd}_{\mathsf{let}}$ denotes the contextual closure of $\rhd_{\mathsf{let}}$:

$$\mathrel{-\!\!\rhd}_{\mathsf{let}} = (\mathbf{K}_{\mathrm{N}}\,\boxed{\rhd})_{\mathsf{let}} = \mathbf{K}_{\mathrm{N}}\,\boxed{\rhd_{\mathsf{let}}}$$

[16] For example, we have

$$\xrightarrow{\mathrm{s}\neg\mathrm{o}}\!\!\rhd \neq \overset{\mathrm{s}}{\mathrel{-\!\!\rhd}} \setminus \overset{\mathrm{o}}{\mathrel{-\!\!\rhd}} = \mathrel{-\!\!\rhd} \setminus \rhd$$

in the λ-calculus because there can be several ways to reduce an expression to another expression:

$$(\lambda y.y)V \lhd (\lambda x.(\lambda y.y)x)V \xrightarrow{\mathrm{s}\neg\mathrm{o}}\!\!\rhd (\lambda x.x)V$$

where the $\lhd$ reduction reduces the outer redex and the $\xrightarrow{\mathrm{s}\neg\mathrm{o}}\!\!\rhd$ one reduces the inner redex. The equation $\xrightarrow{\mathrm{s}\neg\mathrm{o}}\!\!\rhd = \overset{\mathrm{s}}{\mathrel{-\!\!\rhd}} \setminus \overset{\mathrm{o}}{\mathrel{-\!\!\rhd}}$ would hold if we though of the reductions as being multisets that count the numbers of ways in which the reduction can happen (or used labeled transitions to allow distinguishing them), but we do not.

0.5. Table of contents

Part A.

Introduction to L calculi

Part A is an introduction to the untyped $\bar{\lambda}\mu\tilde{\mu}$-calculus [CurHer00], and more generally to calculi that look like it, which we call L-calculi. Through the Curry-Howard correspondence, the simply-typed $\bar{\lambda}\mu\tilde{\mu}$-calculus corresponds to Gentzen's sequent calculus for classical logic in the same way that the λ-calculus corresponds to natural deduction. Most introductions to $\bar{\lambda}\mu\tilde{\mu}$ focus on this correspondence, and sometimes mention the similarity with abstract machines. Here, we focus on the parts that are relevant to using $\bar{\lambda}\mu\tilde{\mu}$ to study the untyped λ-calculus, and in particular on the correspondence between the reductions of the call-by-name (resp. call-by-value) λ-calculus and the reductions of the call-by-name (resp. call-by-value) intuitionistic fragment of $\bar{\lambda}\mu\tilde{\mu}$.

It is well-known that the operational (i.e. weak head) reduction of the call-by-name λ-calculus $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ is refined by the reduction of the Krivine abstract machine [Kri07], that makes the search for the redex explicit. The intuitionistic call-by-name fragment $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ of $\bar{\lambda}\mu\tilde{\mu}$ extends this refinement to its contextual closure, the strong reduction, that can reduce anywhere in the expression. To make understanding the call-by-name (resp. call-by-value) fragment $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ (resp. $\mathrm{Li}^{\rightarrow}_{\mathrm{v}}$) of $\bar{\lambda}\mu\tilde{\mu}$ easier, we introduce a new λ-like syntax $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ (resp. $\underline{\lambda}^{\rightarrow}_{\mathrm{v}}$) for it. In this new syntax $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ (resp. $\underline{\lambda}^{\rightarrow}_{\mathrm{v}}$), the reductions of the μ binder of $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ (resp. $\mathrm{Li}^{\rightarrow}_{\mathrm{v}}$) appear as a natural generalizations of the redex searching reductions of abstract machines, and of some of Regnier's σ-reductions [Reg94].

While some advantages of using L-calculi are immediately apparent (e.g. the symmetry, and the built-in classical logic), many of their advantages only become relevant in larger calculi (e.g. those in Part B) or when studying more complex properties (e.g. those in Part C). The reader that has yet to be convinced of the usefulness of L-calculi should therefore not expect to be convinced after reading just Part A.

Content Chapter I describes the following calculi (in left-to-right order), translations[17] between them, and their properties:

$$\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}} \underset{\lceil\cdot\rceil}{\overset{\lfloor\cdot\rfloor}{\rightleftarrows}} \underline{\lambda}^{\rightarrow}_{\mathrm{n}} \underset{\dot{\leftarrow}}{\overset{\dot{\rightarrow}}{\rightleftarrows}} \mathrm{Li}^{\rightarrow}_{\mathrm{n}} \subsetneq \mathrm{L}^{\rightarrow}_{\mathrm{n}}$$

Chapter II describes their call-by-value counterparts:

$$\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{V}} \underset{\lceil\cdot\rceil}{\overset{\lfloor\cdot\rfloor}{\rightleftarrows}} \underline{\lambda}^{\rightarrow}_{\mathrm{v}} \underset{\dot{\leftarrow}}{\overset{\dot{\rightarrow}}{\rightleftarrows}} \mathrm{Li}^{\rightarrow}_{\mathrm{v}} \subsetneq \mathrm{L}^{\rightarrow}_{\mathrm{v}}$$

Contribution The contribution of this part is mainly pedagogical: it provides a detailed introduction to $\bar{\lambda}\mu\tilde{\mu}$ from a new angle. Technical contributions include:

[17] Translations are represented by arrows with a hook ↪ when they are injective, with two heads ↠ when they are surjective, and with both when they are bijective.

[CurHer00] "The duality of computation", Curien and Herbelin, 2000

[Kri07] "A call-by-name lambda-calculus machine", Krivine, 2007

[Reg94] "Une équivalence sur les lambda-termes", Regnier, 1994

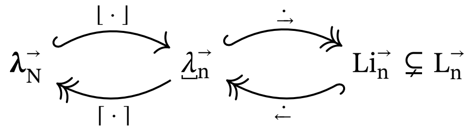
- defining the call-by-name (resp. call-by-value) λ-calculi with focus $\underline{\lambda}_{\mathrm{n}}^{\to}$ (resp. $\underline{\lambda}_{\mathrm{v}}^{\to}$) as an alternative syntax for the call-by-name (resp. call-by-value) intuitionistic fragment $\mathrm{Li}_{\mathrm{n}}^{\to}$ (resp. $\mathrm{Li}_{\mathrm{v}}^{\to}$) of $\bar{\lambda}\mu\tilde{\mu}$; and

- giving a detailed description of the action of focus-inserting and focus-erasing translations $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ on reduction sequences.

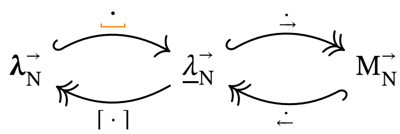
I. Pure call-by-name calculi

Summary

The goal of this chapter is to recall the pure untyped call-by-name λ-calculus $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ [Bar84], the pure untyped call-by-name L calculus $\mathrm{L}_{\mathrm{n}}^{\rightarrow}$ (i.e. the call-by-name fragment of $\bar{\lambda}\mu\tilde{\mu}$ [CurHer00]), and its intuitionistic fragment $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$; to introduce the pure untyped call-by-name λ-calculus with focus $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ as an alternative syntax to $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$; and to relate them via translations[1]:



In order to make the introduction of concepts more progressive, after recalling $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$, we introduce the pure untyped call-by-name λ-calculus with top-level focus $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ and recall the Krivine abstract machine $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$ [Kri07], which are simpler versions of $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ and $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ respectively, and are related to $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ in a similar way:



In both cases, the translations $\underrightarrow{\cdot}$ and $\underleftarrow{\cdot}$ are inverses, so that up to syntax $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ and $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$ (resp. $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ and $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$) are identical. Both the $\underbracket{\cdot}$ translation from $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ to $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ and the $\lfloor \cdot \rfloor$ translation from $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ to $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ add markers $\underbracket{\cdot}$ to make explicit where the focus is, i.e. which subexpression we are currently trying to reduce, while the $\lceil \cdot \rceil$ translation erases these markers. This allows to refine an operational reduction step into three simpler steps: moving the focus downwards until a redex is found, reducing the redex, and moving the focused back to the top of the expression. When looking at several successive operational reduction steps, time can be gained by not going back to the top of the expression between two steps, but instead refocusing [DanNie04], i.e. continuing the search for the next redex from where the previous redex was reduced. In $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ / $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$, the strong reduction step can also be refined in a similar way, with focus movement replaced by a more general reduction called $\mathrel{-\!\!\rhd}_{\mu}$, which also generalizes (some of) Regnier's σ-reductions [Reg94].

[1] Translations are represented by arrows with a hook $\hookrightarrow$ when they are injective, with two heads $\twoheadrightarrow$ when they are surjective, and with both when they are bijective.

[Bar84] *The lambda calculus: its syntax and semantics*, Barendregt, 1984
[CurHer00] “The duality of computation”, Curien and Herbelin, 2000
[Kri07] “A call-by-name lambda-calculus machine”, Krivine, 2007
[DanNie04] “Refocusing in Reduction Semantics”, Danvy and Nielsen, 2004
[Reg94] “Une équivalence sur les lambda-termes”, Regnier, 1994

Table of contents

I.1. A pure call-by-name λ-calculus: $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$

Syntax

We recall the pure untyped call-by-name λ-calculus [Bar84], which we will call $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$, in Figure I.1.1. This is the standard λ-calculus with a few minor changes to the syntax. First, we added N at all the places where polarity annotations will be needed later, e.g. to differentiate for example positive expressions T_+ from negative ones T_- or positive variables x^+ from negative ones x^-. For now, those annotations are mostly useless[2] (and there is no real difference between N as a subscript and N as a superscript) but we nevertheless keep them to prepare the reader for the polarized calculi. Secondly, we have let-expressions $\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}$, even though they behave exactly like β-redexes $(\lambda x^{\mathrm{N}}.U_{\mathrm{N}})T_{\mathrm{N}}$, because when translating from $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ to another calculus, the translation of $\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}$ is sometimes simpler than that of $(\lambda x^{\mathrm{N}}.U_{\mathrm{N}})T_{\mathrm{N}}$. Finally, while it is common to only refer to the objects of study as terms, we also call them values and expressions. In general, given a calculus described by a BNF grammar, we call *expressions* T (resp. *values* V, *terms* t) the elements of the syntax generated by the start non-terminal symbol (resp. same non-terminal symbols as variables x, any non-terminal symbol). In $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$, the BNF grammar only has only one non-terminal symbol, and all three names therefore denote the same objects. As is usual, application is considered to be left-associative, i.e. $T_{\mathrm{N}}U_{\mathrm{N}}^1U_{\mathrm{N}}^2$ stands for $(T_{\mathrm{N}}U_{\mathrm{N}}^1)U_{\mathrm{N}}^2$. We write $\mathbf{T}_{\mathrm{N}}$ for the set of all expressions T_{N}.

Figure I.1.1: Syntax of $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$

Expressions / values:

$$T_{\mathrm{N}}, U_{\mathrm{N}}, V_{\mathrm{N}}, W_{\mathrm{N}} ::= x^{\mathrm{N}} \mid \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}} \mid \lambda x^{\mathrm{N}}.T_{\mathrm{N}} \mid T_{\mathrm{N}}U_{\mathrm{N}}$$

Contexts

Contexts of $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ are denoted by $\mathbb{K}_{\mathrm{N}}$, and are generated by the BNF grammar given in Figure I.1.2.

Figure I.1.2: Contexts in $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$

Contexts:

$$\mathbb{K}_{\mathrm{N}} ::= \square \mid \mathsf{let}\ x^{\mathrm{N}} := \mathbb{K}_{\mathrm{N}}\ \mathsf{in}\ T_{\mathrm{N}} \mid \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ \mathbb{K}_{\mathrm{N}} \mid \lambda x^{\mathrm{N}}.\mathbb{K}_{\mathrm{N}} \mid \mathbb{K}_{\mathrm{N}}T_{\mathrm{N}} \mid T_{\mathrm{N}}\mathbb{K}_{\mathrm{N}}$$

[2] Except when looking at translations between several calculi, or skim-reading, where they serve as a reminder of which calculus we are in.

[Bar84] *The lambda calculus: its syntax and semantics*, Barendregt, 1984

The result of *plugging* a term T_{N} (resp. a context $\mathbb{K}^0_{\mathrm{N}}$) in a context $\mathbb{K}_{\mathrm{N}}$, i.e. the non-capture-avoiding[3] substitution of $\square$ by T_{N} (resp. $\mathbb{K}^0_{\mathrm{N}}$) in $\mathbb{K}_{\mathrm{N}}$, is denoted by $\mathrm{plug}(\mathbb{K}_{\mathrm{N}}, T_{\mathrm{N}})$ or $\mathbb{K}_{\mathrm{N}}\boxed{T_{\mathrm{N}}}$ (resp. $\mathrm{plug}(\mathbb{K}_{\mathrm{N}}, \mathbb{K}^0_{\mathrm{N}})$ or $\mathbb{K}_{\mathrm{N}}\boxed{\mathbb{K}^0_{\mathrm{N}}}$).

The weak head contexts, which we prefer calling *operational contexts* (because they allow defining the operational semantics) or *stacks* (because they correspond to stacks in abstract machines and L calculi), are defined in Figure I.1.3.

Figure I.1.3: Operational contexts in $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$

Operational contexts / stacks / weak head contexts:

$$\mathbf{O}_{\mathrm{N}} = \mathbf{S}_{\mathrm{N}} = \mathring{\mathbf{S}}_{\mathrm{N}} \ni \mathbb{O}_{\mathrm{N}}, \mathbb{S}_{\mathrm{N}}, \mathring{\mathbb{S}}_{N} ::= \square \mid \mathbb{O}_{\mathrm{N}} T_{\mathrm{N}}$$

Substitutions and disubstitutions

We write $\mathrm{FV}(T_{\mathrm{N}})$ for the set of all free variables of T_{N}, and we say that a variable is *fresh* with respect to an expression when it is neither free nor bound in it. We write $T_{\mathrm{N}}[V_{\mathrm{N}}/x^{\mathrm{N}}]$ for the usual capture avoiding substitution of x^{N} by V_{N}, denote arbitrary substitutions by σ and write $T_{\mathrm{N}}[\sigma]$ for the result of applying a substitution σ to a given expression T_{N}.

When studying the behavior of terms (see e.g.), we often want to close them via a substitution σ, and then give them arguments via a stack $\mathbb{S}_{\mathrm{N}}$. We therefore give a name to the combination of a substitutions and a stack:

Definition I.1.1

A *disubstitution* φ is a pair $\varphi = (\sigma, \mathbb{S}_{\mathrm{N}})$ that consists of a substitution σ and a stack $\mathbb{S}_{\mathrm{N}}$. We write $T_{\mathrm{N}}[\varphi]$ for $\mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}[\sigma]}$.

We call these disubstitutions because they correspond to substitutions that act on both the usual variables x and on a stack variable $\star$ in L-calculi (see). We call disubstitutivity the property of being closed under disubstitutions:

Definition I.1.2

A reduction $\rightsquigarrow$ of $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ is said to be:

- *substitutive* when for any substitution σ and terms T_{N} and T'_{N}, we have

$$T_{\mathrm{N}} \rightsquigarrow T'_{\mathrm{N}} \quad \Rightarrow \quad T_{\mathrm{N}}[\sigma] \rightsquigarrow T'_{\mathrm{N}}[\sigma]$$

[3] Contrary to substitutions where variable capture was avoided by renaming bound variables on the fly, e.g. $(\lambda x^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}})[x^{\mathrm{N}}/y^{\mathrm{N}}] = (\lambda z^{\mathrm{N}}.z^{\mathrm{N}}y^{\mathrm{N}})[x^{\mathrm{N}}/y^{\mathrm{N}}] = \lambda z^{\mathrm{N}}.z^{\mathrm{N}}x^{\mathrm{N}}$, plugging does not rename anything and allows variable capture: $(\lambda x^{\mathrm{N}}.x^{\mathrm{N}}\square)\boxed{x^{\mathrm{N}}} = \lambda x^{\mathrm{N}}.x^{\mathrm{N}}x^{\mathrm{N}}$.

- *closed under stacks* when for any stack $\mathbb{S}_N$ and terms T_N and T'_N, we have
$$T_N \rightsquigarrow T'_N \quad \Rightarrow \quad \mathbb{S}_N\boxed{T_N} \rightsquigarrow \mathbb{S}_N\boxed{T'_N}$$
- *disubstitutive* when for any disubstitution φ and terms T_N and T'_N, we have
$$T_N \rightsquigarrow T'_N \quad \Rightarrow \quad T_N[\varphi] \rightsquigarrow T'_N[\varphi]$$

Fact I.1.3

A reduction $\rightsquigarrow$ is disubstitutive if and only if it is substitutive and closed under stacks.

Proof

$\boxed{\Rightarrow}$ Take $\varphi = (\sigma, \square)$ and $\varphi = (\mathrm{Id}, \mathbb{S}_N)$. $\boxed{\Leftarrow}$ Immediate.

β-reduction

The top-level reduction $\triangleright$ is defined in Figure I.1.4. It is the usual one (if one thinks of $\mathsf{let}\ x^N := T_N\ \mathsf{in}\ U_N$ as being a notation for $(\lambda x^N.U_N)T_N$).

Figure I.1.4: Top-level reduction

$$\mathsf{let}\ x^N := T_N\ \mathsf{in}\ U_N \triangleright_{\mathsf{let}} U_N[T_N/x^N]$$
$$(\lambda x^N.T_N)U_N \triangleright_{\rightarrow} T_N[U_N/x^N]$$

$$\triangleright \stackrel{\text{def}}{=} \triangleright_{\mathsf{let}} \cup \triangleright_{\rightarrow}$$

The two closures of the top-level β-reduction we are interested in for now are its operational and strong closures:

Definition I.1.4: Operational and strong reductions

The *operational reduction* $\triangleright$ is defined as the operational closure of the top-level β-reduction $\triangleright$, and the *strong reduction* $\multimap\!\!\triangleright$ as the contextual closure of $\triangleright$:

$$\triangleright \stackrel{\text{def}}{=} \mathbf{O}_N\boxed{\triangleright} \quad \text{and} \quad \multimap\!\!\triangleright \stackrel{\text{def}}{=} \mathbf{K}_N\boxed{\triangleright}$$

We write $\xrightarrow{\neg o}\!\triangleright$ for the closure of the top-level β-reduction $\triangleright$ under the set of non-operational contexts $\mathbf{K}_N \setminus \mathbf{O}_N$:

$$\xrightarrow{\neg o}\!\triangleright \stackrel{\text{def}}{=} (\mathbf{K}_N \setminus \mathbf{O}_N)\boxed{\triangleright}$$

The operational reduction $\triangleright$ is often called the weak head reduction, but we prefer calling it the operational reduction because its main characteristic is that it induces a small-step operational semantics for the calculus, i.e. it represents evaluation. The strong reduction $\multimap\!\!\triangleright$

should be understood as defining an equational theory $\mathrel{\lhd\!\!-\!\!\rhd}^* = (\mathrel{-\!\!\rhd} \cup \mathrel{\lhd\!\!-})^*$ for the calculus, and it being directed helps when relating it to the operational reduction $\rhd$ (e.g. via the factorization $\mathrel{-\!\!\rhd}^* = \rhd^* \overset{\neg o}{\mathrel{-\!\!\rhd}}{}^*$). The reductions have the properties announced in Figure **??** (see Section .2 for details).

σ-reductions

Regnier's σ-reductions [Reg94] allow commuting redexes in a way that preserves most properties of the expression:

$$\begin{array}{rcll} (\lambda x^{\mathrm{N}}.U_{\mathrm{N}})T_{\mathrm{N}}V_{\mathrm{N}} & \rightsquigarrow_\sigma & (\lambda x^{\mathrm{N}}.U_{\mathrm{N}}V_{\mathrm{N}})T_{\mathrm{N}} & \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } V_{\mathrm{N}} \\ (\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}} & \rightsquigarrow_\sigma & \lambda y^{\mathrm{N}}.(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}} & \text{if } y^{\mathrm{N}} \text{ fresh w.r.t. } U_{\mathrm{N}} \end{array}$$

Replacing $(\lambda x^{\mathrm{N}}.U_{\mathrm{N}})T_{\mathrm{N}}$ by $\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}$ in these yields

$$\begin{array}{rcll} (\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}})V_{\mathrm{N}} & \rightsquigarrow_\sigma & \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V_{\mathrm{N}} & \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } V_{\mathrm{N}} \\ \mathsf{let}\ x^{\mathrm{N}} := U_{\mathrm{N}}\ \mathsf{in}\ \lambda y^{\mathrm{N}}.T_{\mathrm{N}} & \rightsquigarrow_\sigma & \lambda y^{\mathrm{N}}.\mathsf{let}\ x^{\mathrm{N}} := U_{\mathrm{N}}\ \mathsf{in}\ T_{\mathrm{N}} & \text{if } y^{\mathrm{N}} \text{ fresh w.r.t. } U_{\mathrm{N}} \end{array}$$

We only use the first of these two σ-reductions and denote it by a backwards Σ as shown in Figure I.1.5.

Figure I.1.5: Top-level σ-reduction

$$(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}})V_{\mathrm{N}} \ \reflectbox{Σ}\ \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V_{\mathrm{N}} \quad \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } V_{\mathrm{N}}$$

Definition I.1.5

We write $\reflectbox{$\Sigma$}$ for the closure of $\reflectbox{$\Sigma$}$ under the set of simple stacks $\mathring{\mathbf{S}}_{\mathrm{N}}$[a], and $-\!\reflectbox{$\Sigma$}$ for the contextual closure of $\reflectbox{$\Sigma$}$:

$$\reflectbox{Σ} \stackrel{\text{def}}{=} \mathring{\mathbf{S}}_{\mathrm{N}}\boxed{\reflectbox{Σ}} \quad \text{and} \quad -\!\reflectbox{Σ} \stackrel{\text{def}}{=} \mathbf{K}_{\mathrm{N}}\boxed{\reflectbox{Σ}}$$

[a] While we have $\mathring{\mathbf{S}}_{\mathrm{N}} = \mathbf{S}_{\mathrm{N}} = \mathbf{O}_{\mathrm{N}}$ in $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$, in general, we only have $\mathring{\mathbf{S}}_{\mathrm{N}} \subseteq \mathbf{S}_{\mathrm{N}} \subseteq \mathbf{O}_{\mathrm{N}}$.

In accordance with our convention of denoting unions of reductions by superimposing their symbols, we use the notations

$$\rhd\!\!\!\reflectbox{Σ} \stackrel{\text{ntn}}{=} \rhd \cup \reflectbox{Σ} \quad \text{and} \quad -\!\rhd\!\!\!\reflectbox{Σ} \stackrel{\text{ntn}}{=} \mathrel{-\!\!\rhd} \cup -\!\reflectbox{Σ}$$

In the call-by-name λ-calculus, σ-reductions are somewhat superfluous because they only relate expressions that have a common reduct:

$$(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}})V_{\mathrm{N}} \rhd (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}])V_{\mathrm{N}} \lhd \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V_{\mathrm{N}}$$

They are however very useful to make the call-by-value λ-calculus behave well on open expressions [AccGue16; AccPao12; PaoRon99], and to understand the $\rhd_\mu$ reduction of L calculi

[Reg94] "Une équivalence sur les lambda-termes", Regnier, 1994
[AccGue16] "Open Call-by-Value", Accattoli and Guerrieri, 2016
[AccPao12] "Call-by-Value Solvability, Revisited", Accattoli and Paolini, 2012
[PaoRon99] "Call-by-value Solvability", Paolini and Ronchi Della Rocca, 1999

(which can be though of as being a generalization of $\ltimes$), which is why is nevertheless examine them in the call-by-name λ-calculus.

The first thing to note is that extending $\triangleright$ by $\ltimes$ yields a reduction $\bowtie = \triangleright \cup \ltimes$ that is not deterministic:

$$\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}} V_{\mathrm{N}} \Join (\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}) V_{\mathrm{N}} \triangleright_{\mathsf{let}} (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}]) V_{\mathrm{N}}$$

This also happens in call-by-value, where we would really like to use $\bowtie$ to evaluate open expressions. A very common choice to avoid this problem is to simply not add $\ltimes$ to the operational reduction, and to only add σ-reductions $\multimap\!\ltimes$ in the strong reduction $\multimap\!\triangleright$ when looking at the equational theory. This of course leads to complications, e.g. requiring distinguishing σ-reduction from operational reduction in many lemmas and theorems. The reduction $\triangleright_{\mu}$ of $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ and $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ takes the opposite approach to recover determinism: it prevents the $\triangleright_{\mathsf{let}}$ reduction above by disallowing the reduction of let-expressions under non-trivial operational contexts and keeps $\ltimes$ as part of the operational reduction!

More precisely, since $\ltimes$ can not be added directly to $\triangleright$ without breaking determinism, we first restrict $\triangleright$ and only then extend it with $\ltimes$:

Definition I.1.6

The reductions $\vartriangleright\!\!\vdash$ and $\bowtie\!\!\vdash$ are defined by

$$\vartriangleright\!\!\vdash \ \stackrel{\text{def}}{=}\ \triangleright_{\rightarrow} \cup \triangleright_{\mathsf{let}} \qquad \text{and} \qquad \bowtie\!\!\vdash \ \stackrel{\text{def}}{=}\ \vartriangleright\!\!\vdash \cup \ltimes$$

The difference between $\triangleright$ and $\vartriangleright\!\!\vdash$ is that $\triangleright$ allows all reductions of the shape

$$(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}) V_{\mathrm{N}}^{1} \dots V_{\mathrm{N}}^{q} \triangleright (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}]) V_{\mathrm{N}}^{1} \dots V_{\mathrm{N}}^{q}$$

while $\vartriangleright\!\!\vdash$ only allows those of the shape

$$\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}} \triangleright_{\mathsf{let}} U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}]$$

i.e. those where the operational contexts $\mathbb{O}_{\mathrm{N}} = \square V_{\mathrm{N}}^{1} \dots V_{\mathrm{N}}^{q}$ under which the reduction happens is trivial. In particular, the $\triangleright_{\mathsf{let}}$ reduction of the aformentioned critical pair is not allowed by $\vartriangleright\!\!\vdash$, which allows it to be deterministic:

Fact I.1.7: Determinism of $\bowtie\!\!\vdash$

The reduction $\bowtie\!\!\vdash$ reduction is deterministic.

Proof

Both $\vartriangleright\!\!\vdash$ and $\ltimes$ are deterministic, and they have disjoint domains.

Furthermore, the forbidden reductions

$$(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}) V_{\mathrm{N}}^{1} \dots V_{\mathrm{N}}^{q} \triangleright_{\mathsf{let}} (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}]) V_{\mathrm{N}}^{1} \dots V_{\mathrm{N}}^{q}$$

can be simulated by

$$\begin{aligned}(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}})V^1_{\mathrm{N}} \dots V^q_{\mathrm{N}} &\ltimes (\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V^1_{\mathrm{N}})V^2_{\mathrm{N}} \dots V^q_{\mathrm{N}} \\ &\ltimes^* \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V^1_{\mathrm{N}} \dots V^q_{\mathrm{N}} \\ &>_{\mathsf{let}} (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}])V^1_{\mathrm{N}} \dots V^q_{\mathrm{N}}\end{aligned}$$

In fact, the reductions $\triangleright$ and $\bowtie$ have the same notion of normal form, and induce the same notion of (big-step) evaluation:

Fact I.1.8: Equivalence between $\triangleright^{\circledast}$ and $\bowtie^{\circledast}$

- The $\bowtie$-normal expressions are exactly the $\triangleright$-normal expressions:

$$T_{\mathrm{N}} \not\bowtie \Leftrightarrow T_{\mathrm{N}} \not\triangleright$$

- The $\ltimes$ steps can be postponed at the cost of strengthening $>_{\mathsf{let}}$ to $\triangleright_{\mathsf{let}}$:

$$T_{\mathrm{N}} \bowtie^* T'_{\mathrm{N}} \Leftrightarrow T_{\mathrm{N}} \triangleright^* \ltimes^* T'_{\mathrm{N}}$$

- Evaluating with $\bowtie$ or $\triangleright$ yields the same result:

$$T_{\mathrm{N}} \bowtie^{\circledast} T'_{\mathrm{N}} \Leftrightarrow T_{\mathrm{N}} \triangleright^{\circledast} T'_{\mathrm{N}}$$

Proof sketch (See page 186 for details)

Immediate.

η-expansion

Another well-known and useful relation on λ-terms is η-expansion (and its symmetric, η-reduction) that relates any expressions T_{N} to a λ-abstraction $\lambda x^{\mathrm{N}}.T_{\mathrm{N}}x^{\mathrm{N}}$ that has the same functional behavior, i.e. that behaves the same once given an argument. The η-expansions for $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ are defined in Figure I.1.6, where $\wr_{\rightarrow}$ is the standard η-expansion for functions.

Figure I.1.6: Top-level η-expansion

$$\begin{aligned}T_{\mathrm{N}} &\wr_{\rightarrow} \lambda x^{\mathrm{N}}.T_{\mathrm{N}}x^{\mathrm{N}} && \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } T_{\mathrm{N}} \\ T_{\mathrm{N}} &\wr_{\mathsf{let}} \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ x^{\mathrm{N}}\end{aligned}$$

We write $\mathrel{-\!\!\wr}$ for the contextual closure of $\wr$, $\mathrel{-\!\!\wr\!\!\triangleright}$ for $\mathrel{-\!\!\triangleright} \cup \mathrel{-\!\!\wr}$, $\mathrel{-\!\!\wr\!\!\ltimes}$ for $\mathrel{-\!\!\ltimes} \cup \mathrel{-\!\!\wr}$, $\mathrel{-\!\!\wr\!\!\bowtie}$ for $\mathrel{-\!\!\triangleright} \cup \mathrel{-\!\!\ltimes} \cup \mathrel{-\!\!\wr}$, and $\approx_{\beta\eta\sigma}$ or $\mathrel{\bowtie\!\!\wr\!\!\bowtie}^*$ for the $\beta\eta\sigma$-equivalence:

$$\approx_{\beta\eta\sigma} \stackrel{\mathrm{def}}{=} \mathrel{\bowtie\!\!\wr\!\!\bowtie}^* = (\mathrel{-\!\!\triangleright} \cup \mathrel{\triangleleft\!\!-} \cup \mathrel{-\!\!\wr} \cup \mathrel{\wr\!\!-} \cup \mathrel{-\!\!\ltimes} \cup \mathrel{\rtimes\!\!-})^*$$

The η-expansion for let-expressions $\wr_{\mathsf{let}}$ is less common, most likely because it is contained in $<_{\mathsf{let}}$ (in call-by-name):

$$T_{\mathrm{N}} <_{\mathsf{let}} \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ x^{\mathrm{N}}$$

There are other reasonable definitions of η-expansion, but all of them are contained in the $\beta\eta\sigma$-equivalence induced by this definition of η-expansion. For example, we have

$$\mathbb{K}_{\mathrm{N}}\boxed{V_{\mathrm{N}}} \approx_{\beta\eta\sigma} \mathsf{let}\; x^{\mathrm{N}} := V_{\mathrm{N}}\; \mathsf{in}\; \mathbb{K}_{\mathrm{N}}\boxed{x^{\mathrm{N}}} \quad \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } \mathbb{K}_{\mathrm{N}}$$

because

$$\mathbb{K}_{\mathrm{N}}\boxed{V_{\mathrm{N}}} \lhd_{\mathsf{let}} \mathsf{let}\; x^{\mathrm{N}} := V_{\mathrm{N}}\; \mathsf{in}\; \mathbb{K}_{\mathrm{N}}\boxed{x^{\mathrm{N}}}$$

and

$$\mathbb{O}_{\mathrm{N}}\boxed{T_{\mathrm{N}}} \approx_{\beta\eta\sigma} \mathsf{let}\; x^{\mathrm{N}} := T_{\mathrm{N}}\; \mathsf{in}\; \mathbb{O}_{\mathrm{N}}\boxed{x^{\mathrm{N}}} \quad \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } \mathbb{O}_{\mathrm{N}}$$

because

$$\mathbb{O}_{\mathrm{N}}\boxed{T_{\mathrm{N}}} \dashv_{\mathsf{let}} \mathbb{O}_{\mathrm{N}}\boxed{\mathsf{let}\; x^{\mathrm{N}} := T_{\mathrm{N}}\; \mathsf{in}\; x^{\mathrm{N}}} \multimap^{*} \mathsf{let}\; x^{\mathrm{N}} := T_{\mathrm{N}}\; \mathsf{in}\; \mathbb{O}_{\mathrm{N}}\boxed{T_{\mathrm{N}}}$$

In call-by-name, all terms are values, so the first $\approx_{\beta\eta\sigma}$-equivalence implies the second, but in call-by-value and polarized settings, neither implies the other.

I.2. A pure call-by-name λ-calculus with toplevel focus: $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$

Abstract machines use a subset of operational contexts called stacks. In general, stacks $\mathbb{S}_{\mathrm{N}}$ form a possibly strict subset of operational contexts $\mathbb{O}_{\mathrm{N}}$, but in $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ they are exactly the same. To avoid forming intuitions that do not generalize to subsequent calculi, we call operational contexts $\mathbb{O}_{\mathrm{N}}$ stacks $\mathbb{S}_{\mathrm{N}}$ in this section. We also completely ignore let-expressions in this section because our goal is to make the comprehension of L calculi easier, and adding let-expressions at this point would not help in that regard.

Searching for the next redex

In $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$, to implement the $\triangleright_{\rightarrow}$-reduction of a term T_{N}, a machine needs to decompose it as

$$T_{\mathrm{N}} = \mathbb{S}_{\mathrm{N}}\boxed{(\lambda x^{\mathrm{N}}.U_{\mathrm{N}})V_{\mathrm{N}}}$$

Figure I.2.1: The $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ calculus

Figure I.2.1.a: Syntax

Stacks:	Configurations:
$\mathbb{S}_{\mathrm{N}} ::= \square$	$C_{\mathrm{N}} ::= \underline{T_{\mathrm{N}}}$
$\mid \mathbb{S}_{\mathrm{N}} T_{\mathrm{N}}$	$\mid C_{\mathrm{N}} T_{\mathrm{N}}$

Figure I.2.1.b: Expanded descriptions

Stacks (expanded):	Configurations (expanded):
$\mathbb{S}_{\mathrm{N}} ::= \square T_{\mathrm{N}}^{1} \ldots T_{\mathrm{N}}^{k}$	$C_{\mathrm{N}} ::= \underline{T_{\mathrm{N}}} U_{\mathrm{N}}^{1} \ldots U_{\mathrm{N}}^{k}$

Figure I.2.1.c: Operational reduction

$$\mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}U_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}U_{\mathrm{N}}}$$

$$\mathbb{S}_{\mathrm{N}}\boxed{(\underline{\lambda x^{\mathrm{N}}.T_{\mathrm{N}}})U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\rightarrow} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}}$$

$$\overset{\mathrm{M}}{\triangleright} \overset{\mathrm{def}}{=} \overset{\mathrm{M}}{\triangleright}_{\rightarrow} \cup \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$$

Figure I.2.1.d: Disubstitutions

Stacks:	Configurations:
$\check{\mathbb{S}}_{\mathrm{N}}[\sigma, \mathbb{S}_{\mathrm{N}}] \overset{\mathrm{def}}{=} \mathbb{S}_{\mathrm{N}}\boxed{\check{\mathbb{S}}_{\mathrm{N}}}$	$C_{\mathrm{N}}[\sigma, \mathbb{S}_{\mathrm{N}}] \overset{\mathrm{def}}{=} \mathbb{S}_{\mathrm{N}}\boxed{C_{\mathrm{N}}}$

The $\underline{\lambda}^{\to}_{\mathrm{N}}$ calculus defined in Figure I.2.1 makes the computation of that decomposition explicit: a configuration $C_{\mathrm{N}} = \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}}$ represents the expression $\mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}}$ in which the machine is currently looking at the subexpression T_{N}. Initially, the machine is looking at the whole term, i.e. it starts from $\underline{T_{\mathrm{N}}}$. It then moves to the left of applications with

$$\mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}U_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}U_{\mathrm{N}}}$$

until it reaches a λ-abstraction, at which point it reduces the β-redex with

$$\mathbb{S}_{\mathrm{N}}\boxed{\underline{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})}U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\to} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}}$$

For example, the reduction

$$I_{\mathrm{N}}T_{\mathrm{N}}U_{\mathrm{N}} \triangleright_{\to} I_{\mathrm{N}}T_{\mathrm{N}}U_{\mathrm{N}}$$

of $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ becomes

$$\underline{I_{\mathrm{N}}T_{\mathrm{N}}U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \underline{I_{\mathrm{N}}T_{\mathrm{N}}}U_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \underline{I_{\mathrm{N}}}T_{\mathrm{N}}U_{\mathrm{N}} \triangleright_{\to} T_{\mathrm{N}}U_{\mathrm{N}}$$

in $\underline{\lambda}^{\to}_{\mathrm{N}}$. Note that the “move” reduction steps $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$ are invisible in the original calculus, while the “reduce” reduction step $\overset{\mathrm{M}}{\triangleright}_{\to}$ corresponds exactly to the reduction reduction step $\triangleright_{\to}$ in $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$.

Simulation

A top-level reduction

$$(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}} \triangleright_{\to} T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]$$

in $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ becomes

$$\underline{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \underline{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})}U_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}_{\to} \underline{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}$$

in $\underline{\lambda}^{\to}_{\mathrm{N}}$, and an operational reduction

$$\mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}} \triangleright_{\to} \mathbb{S}_{\mathrm{N}}\boxed{T'_{\mathrm{N}}}$$

induced by $T_{\mathrm{N}} \triangleright_{\to} T'_{\mathrm{N}}$ in $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ becomes

$$\underline{\mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright}^{*}_{\mathrm{m}} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}\overset{\mathrm{M}}{\triangleright}_{\to} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T'_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleleft}^{*}_{\mathrm{m}} \underline{\mathbb{S}_{\mathrm{N}}\boxed{T'_{\mathrm{N}}}}$$

in $\underline{\lambda}^{\to}_{\mathrm{N}}$, where the reduction sequences

$$\underline{\mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright}^{*}_{\mathrm{m}} \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}} \quad \text{and} \quad \mathbb{S}_{\mathrm{N}}\boxed{\underline{T'_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleleft}^{*}_{\mathrm{m}} \underline{\mathbb{S}_{\mathrm{N}}\boxed{T'_{\mathrm{N}}}}$$

just correspond to moving downwards through $\mathbb{S}_{\mathrm{N}}$ and do not depend on what is plugged in $\mathbb{S}_{\mathrm{N}}$, and the $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}\overset{\mathrm{M}}{\triangleright}_{\to}$ reduction steps correspond to the actual reduction $T_{\mathrm{N}} \triangleright_{\to} T'_{\mathrm{N}}$.

Refocusing

A reduction sequence

$$\mathbb{S}^{1}_{\mathrm{N}}\boxed{T_{\mathrm{N}}V_{\mathrm{N}}} \triangleright_{\to} \mathbb{S}^{1}_{\mathrm{N}}\boxed{T'_{\mathrm{N}}} = \mathbb{S}^{2}_{\mathrm{N}}\boxed{U_{\mathrm{N}}W_{\mathrm{N}}} \triangleright_{\to} \mathbb{S}^{2}_{\mathrm{N}}\boxed{U'_{\mathrm{N}}}$$

in $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ induced by

$$T_{\mathrm{N}}V_{\mathrm{N}} \triangleright T'_{\mathrm{N}} \quad \text{and} \quad U_{\mathrm{N}}W_{\mathrm{N}} \triangleright U'_{\mathrm{N}}$$

can be simulated step by step in $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ as

$$\begin{array}{ccccccc}
\mathbb{S}^1_{\mathrm{N}}\boxed{T_{\mathrm{N}}V_{\mathrm{N}}} & & \mathbb{S}^1_{\mathrm{N}}\boxed{T'_{\mathrm{N}}} = \mathbb{S}^2_{\mathrm{N}}\boxed{U_{\mathrm{N}}W_{\mathrm{N}}} & & \mathbb{S}^2_{\mathrm{N}}\boxed{U'_{\mathrm{N}}} \\
\nabla^{\leq}_{\exists *} & & \nabla^{\leq}_{\exists *} \quad \nabla^{\leq}_{\exists *} & & \nabla^{\leq}_{\exists *} \\
\mathbb{S}^1_{\mathrm{N}}\boxed{T_{\mathrm{N}}V_{\mathrm{N}}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & \mathbb{S}^1_{\mathrm{N}}\boxed{T'_{\mathrm{N}}} \quad \mathbb{S}^2_{\mathrm{N}}\boxed{U_{\mathrm{N}}W_{\mathrm{N}}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & \mathbb{S}^2_{\mathrm{N}}\boxed{U'_{\mathrm{N}}}
\end{array}$$

Moving the focus back to the top of the term between the two reduction steps is inefficient: instead of computing the decomposition $\mathbb{S}^2_{\mathrm{N}}\boxed{U_{\mathrm{N}}W_{\mathrm{N}}}$ from $\mathbb{S}^2_{\mathrm{N}}\boxed{U_{\mathrm{N}}W_{\mathrm{N}}}$, we could compute it from $\mathbb{S}^1_{\mathrm{N}}\boxed{T'_{\mathrm{N}}}$, which is called refocusing [DanNie04]. This amounts to simplifying the reduction sequence $\overset{\mathrm{M}}{\triangleleft}{}^*_{\mathrm{m}}$ induced by one step with the reduction sequence $\overset{\mathrm{M}}{\triangleright}{}^*_{\mathrm{m}}$ induced by the next step (using determinism of $\overset{\mathrm{M}}{\triangleright}$), which yields the shorter reduction sequence

$$\begin{array}{ccccccc}
\mathbb{S}^1_{\mathrm{N}}\boxed{T_{\mathrm{N}}V_{\mathrm{N}}} & & & & & & \mathbb{S}^2_{\mathrm{N}}\boxed{U'_{\mathrm{N}}} \\
\nabla^{\leq}_{\exists *} & & & & & & \nabla^{\leq}_{\exists *} \\
\mathbb{S}^1_{\mathrm{N}}\boxed{T_{\mathrm{N}}V_{\mathrm{N}}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & \mathbb{S}^1_{\mathrm{N}}\boxed{T'_{\mathrm{N}}} & \overset{\mathrm{M}}{\triangleright}{}^*_{\mathrm{m}} & \mathbb{S}^2_{\mathrm{N}}\boxed{U_{\mathrm{N}}W_{\mathrm{N}}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & \mathbb{S}^2_{\mathrm{N}}\boxed{U'_{\mathrm{N}}}
\end{array}$$

For example, for any terms T^1_{N} and T^2_{N},

$$\begin{array}{ccccccc}
(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}I_{\mathrm{N}}x^{\mathrm{N}})T^1_{\mathrm{N}}T^2_{\mathrm{N}} & & I_{\mathrm{N}}I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} = I_{\mathrm{N}}I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & & I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} = I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & & T^1_{\mathrm{N}}T^2_{\mathrm{N}} \\
\nabla^{\leq}_{\exists *} & & \nabla^{\leq}_{\exists *} \quad \nabla^{\leq}_{\exists *} & & \nabla^{\leq}_{\exists *} \quad \nabla^{\leq}_{\exists *} & & \nabla^{\leq}_{\exists *} \\
(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}I_{\mathrm{N}}x^{\mathrm{N}})T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & I_{\mathrm{N}}I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} \quad I_{\mathrm{N}}I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} \quad I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & T^1_{\mathrm{N}}T^2_{\mathrm{N}}
\end{array}$$

simplifies to

$$\begin{array}{ccccccccc}
(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}I_{\mathrm{N}}x^{\mathrm{N}})T^1_{\mathrm{N}}T^2_{\mathrm{N}} & & & & & & & & T^1_{\mathrm{N}}T^2_{\mathrm{N}} \\
\nabla^{\leq}_{\exists *} & & & & & & & & \nabla^{\leq}_{\exists *} \\
(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}I_{\mathrm{N}}x^{\mathrm{N}})T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & I_{\mathrm{N}}I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} & I_{\mathrm{N}}I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} = I_{\mathrm{N}}T^1_{\mathrm{N}}T^2_{\mathrm{N}} & \overset{\mathrm{M}}{\triangleright}_{\rightarrow} & T^1_{\mathrm{N}}T^2_{\mathrm{N}}
\end{array}$$

Properties of reductions

Disubstitutions of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ are defined just like in $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$:

Definition I.2.1

A *disubstitution* φ is a pair $\varphi = (\sigma, \mathbb{S}_{\mathrm{N}})$ composed of a substitution σ and a stack $\mathbb{S}_{\mathrm{N}}$. Given a configuration C_{N} (resp. stack $\mathbb{S}^{\vee}_{\mathrm{N}}$), we write $C_{\mathrm{N}}[\varphi]$ (resp. $\mathbb{S}^{\vee}_{\mathrm{N}}[\varphi]$) or $C_{\mathrm{N}}[\sigma, \mathbb{S}_{\mathrm{N}}]$ (resp. $\mathbb{S}^{\vee}_{\mathrm{N}}[\sigma, \mathbb{S}_{\mathrm{N}}]$) for $\mathbb{S}_{\mathrm{N}}\boxed{C_{\mathrm{N}}[\sigma]}$ (resp. $\mathbb{S}_{\mathrm{N}}\boxed{\mathbb{S}^{\vee}_{\mathrm{N}}[\sigma]}$).

As announced in Figure **??**, $\overset{\mathrm{M}}{\triangleright}$ is deterministic, substitutive, and disubstitutive (see Section .2 for details).

[DanNie04] "Refocusing in Reduction Semantics", Danvy and Nielsen, 2004

I.3. A pure call-by-name abstract machine: $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$

The inside-out syntax

When implementing an abstract machine, representing $\mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}}$ as a tree with a marked position is suboptimal because most operations will require traversing $\mathbb{S}_{\mathrm{N}}$, and hence takes a time linear in the depth of the hole $\square$ in $\mathbb{S}_{\mathrm{N}}$. It is more efficient to use a zipper [Hue97], i.e. to represent $\mathbb{S}_{\mathrm{N}}$ and T_{N} independently, and to represent $\mathbb{S}_{\mathrm{N}}$ in an "inside-out" fashion. More precisely, a stack $\square$ is representend by $\star$, and $\mathbb{S}_{\mathrm{N}}\boxed{\square V_{\mathrm{N}}}$ by $V_{\mathrm{N}} \cdot S_{\mathrm{N}}$ (where S_{N} is the inside-out representation of $\mathbb{S}_{\mathrm{N}}$), so that a stack

$$\mathbb{S}_{\mathrm{N}} = ((\square V^1_{\mathrm{N}}) \dots) V^k_{\mathrm{N}} = \left(\left(\square\boxed{\square V^k_{\mathrm{N}}}\right)\dots\right)\boxed{\square V^1_{\mathrm{N}}}$$

is represented by

$$S_{\mathrm{N}} = V^1_{\mathrm{N}} \cdot (\dots \cdot (V^k_{\mathrm{N}} \cdot \star))$$

or

$$S_{\mathrm{N}} = V^1_{\mathrm{N}} \cdot \dots \cdot V^k_{\mathrm{N}} \cdot \star$$

with the convention that $\cdot$ is right associative. Note that the arguments appear in the order in which they will (possibly) be needed by the computation, and that $\star$ represents the *outside* of the context. An expression with an underlined subexpressions $\mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}}$ is then represented by a pair $(T_{\mathrm{N}}, S_{\mathrm{N}})$, which we call a *configuration*[4] and denote by $\langle T_{\mathrm{N}} | S_{\mathrm{N}} \rangle$, where T_{N} is the focused subexpression, and S_{N} is the inside-out representation of $\mathbb{S}_{\mathrm{N}}$. This yields the $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ calculus described in Figure I.3.1, a variant of the Krivine abstract machine [Kri07] that uses substitutions instead of environments and closures. Just like in the Krivine abstract machine, $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$-reducing a term is a constant time operation thanks to the inside-out representation, but the use of substitutions in $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ makes $\overset{\mathrm{M}}{\triangleright}_{\rightarrow}$-reducing a term linear in the number of free occurrences of the variable, and hence less efficient than in the Krivine abstrat machine.

An example reduction sequence is given in the right column of Figure I.3.2, with the corresponding reduction sequence in the left column.

As announced in Figure **??**, $\overset{\mathrm{M}}{\triangleright}$ is deterministic, substitutive, and disubstitutive (see Section .2 for details).

Disubstitutions

In $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$, we also consider substitutions that act on $\star^{\mathrm{N}}$ (in addition to the usual variables x^{N}), which we call disubstitutions to avoid any confusion with the usual definition of substitutions:

[4] These are also sometimes called a command. In this document, we only use "configuration" for abstract machines, and keep "command" for L calculi.

[Hue97] "The Zipper", Huet, 1997

[Kri07] "A call-by-name lambda-calculus machine", Krivine, 2007

Figure I.3.1: The $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$ calculus

Figure I.3.1.a: Syntax

Stacks:
$$S_{\mathrm{N}} ::= \star^{\mathrm{N}} \mid T_{\mathrm{N}} \cdot S_{\mathrm{N}}$$

Configurations:
$$C_{\mathrm{N}} ::= \langle T_{\mathrm{N}} | S_{\mathrm{N}} \rangle$$

Figure I.3.1.b: Expanded descriptions

Stacks (expanded):
$$S_{\mathrm{N}} ::= T_{\mathrm{N}}^{1} \cdot \ldots \cdot T_{\mathrm{N}}^{q} \cdot \star^{\mathrm{N}}$$

Configurations (expanded):
$$C_{\mathrm{N}} ::= \langle T_{\mathrm{N}} | U_{\mathrm{N}}^{1} \cdot \ldots \cdot U_{\mathrm{N}}^{q} \cdot \star^{\mathrm{N}} \rangle$$

Figure I.3.1.c: Operational reduction

$$\langle T_{\mathrm{N}} U_{\mathrm{N}} | S_{\mathrm{N}} \rangle \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \langle T_{\mathrm{N}} | U_{\mathrm{N}} \cdot S_{\mathrm{N}} \rangle$$
$$\langle \lambda x^{\mathrm{N}}.T_{\mathrm{N}} | U_{\mathrm{N}} \cdot S_{\mathrm{N}} \rangle \overset{\mathrm{M}}{\triangleright}_{\rightarrow} \langle T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}] | S_{\mathrm{N}} \rangle$$

$$\overset{\mathrm{M}}{\triangleright} \overset{\mathrm{def}}{=} \overset{\mathrm{M}}{\triangleright}_{\rightarrow} \cup \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$$

Figure I.3.1.d: Disubstitutions

Expressions:
$$x^{\mathrm{N}}[\varphi] \overset{\mathrm{def}}{=} \varphi(x^{\mathrm{N}})$$
$$(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})[\varphi] \overset{\mathrm{def}}{=} \lambda x^{\mathrm{N}}.T_{\mathrm{N}}[\varphi] \quad \text{if } x^{\mathrm{N}} \text{ fresh w.r.t. } \varphi$$
$$(T_{\mathrm{N}} U_{\mathrm{N}}) \overset{\mathrm{def}}{=} (T_{\mathrm{N}}[\varphi])(U_{\mathrm{N}}[\varphi])$$

Stacks:
$$\star^{\mathrm{N}}[\varphi] \overset{\mathrm{def}}{=} \varphi(\star^{\mathrm{N}})$$
$$(T_{\mathrm{N}} \cdot S_{\mathrm{N}})[\varphi] \overset{\mathrm{def}}{=} (T_{\mathrm{N}}[\varphi]) \cdot (S_{\mathrm{N}}[\varphi])$$

Configurations:
$$\langle T_{\mathrm{N}} | S_{\mathrm{N}} \rangle[\varphi] \overset{\mathrm{def}}{=} \langle T_{\mathrm{N}}[\varphi] | S_{\mathrm{N}}[\varphi] \rangle$$

Figure I.3.1.e: Disubstitutions $\star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$

Expressions:

$$T_{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}] = T_{\mathrm{N}}$$

Stacks:

$$\star^{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}] = S_{\mathrm{N}}$$

$$(T_{\mathrm{N}} \cdot S^{\vee}_{\mathrm{N}})[S_{\mathrm{N}}/\star^{\mathrm{N}}] = T_{\mathrm{N}} \cdot (S^{\vee}_{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}])$$

Configurations:

$$\langle T_{\mathrm{N}} | S^{\vee}_{\mathrm{N}} \rangle [S_{\mathrm{N}}/\star^{\mathrm{N}}] = \langle T_{\mathrm{N}} | S^{\vee}_{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}] \rangle$$

Figure I.3.1.f: Disubstitutions (simplified)

Expressions:

$$T_{\mathrm{N}}[\sigma, S_{\mathrm{N}}/\star^{\mathrm{N}}] = T_{\mathrm{N}}[\sigma]$$

Stacks:

$$S^{\vee}_{\mathrm{N}}[\sigma, S_{\mathrm{N}}/\star^{\mathrm{N}}] = S^{\vee}_{\mathrm{N}}[\sigma][S_{\mathrm{N}}/\star^{\mathrm{N}}]$$

Configurations:

$$\langle T_{\mathrm{N}} | S^{\vee}_{\mathrm{N}} \rangle [\sigma, S_{\mathrm{N}}/\star^{\mathrm{N}}] = \langle T_{\mathrm{N}}[\sigma] | S^{\vee}_{\mathrm{N}}[\sigma][S_{\mathrm{N}}/\star^{\mathrm{N}}] \rangle$$

Definition I.3.1: Disubstitutions

A *disubstitution* φ is a function of the shape $\varphi = \sigma, \star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$, i.e. it is a substitution σ extended by $\star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$ for some stack S_{N}.

One way to understand this operation is to think of $\star^{\mathrm{N}}$ as meaning "outside", so that $C_{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}]$ means replacing the "outside" of C_{N} by S_{N}. The action of disubstitutions on terms is described in Figure I.3.1d, and the special case $\varphi = \star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$ (i.e. $\varphi = \mathrm{Id}, \star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$) is described in Figure I.3.1e.

Since expressions T_{N} can never contain $\star^{\mathrm{N}}$, the action of a disubstitution $\varphi = \sigma, \star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$ can be expressed in terms of the action of the subsitution σ and of the disubstitution $\star^{\mathrm{N}} \mapsto S_{\mathrm{N}}$:

Fact I.3.2

The equations given in Figure I.3.1f always hold.

Proof

The equation on expressions is proven by induction on T_{N}. The equation on stacks is prove by induction on $S^{\vee}_{\mathrm{N}}$, using the equation on terms. The equation on configura-

Figure I.3.2: Example of reductions in $\underline{\lambda}^{\to}_{\mathrm{N}}$ and $\mathrm{M}^{\to}_{\mathrm{N}}$

$$
\begin{array}{cc}
\underbrace{((\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}})I_{\mathrm{N}})I_{\mathrm{N}}} & \langle((\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}})I_{\mathrm{N}})I_{\mathrm{N}} \| \star^{\mathrm{N}}\rangle \\
\equiv_{\trianglerighteq} & \equiv_{\trianglerighteq} \\
\underbrace{((\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}})I_{\mathrm{N}})}I_{\mathrm{N}} & \langle(\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}})I_{\mathrm{N}} \| I_{\mathrm{N}} \cdot \star^{\mathrm{N}}\rangle \\
\equiv_{\trianglerighteq} & \equiv_{\trianglerighteq} \\
(\underbrace{(\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}})}I_{\mathrm{N}})I_{\mathrm{N}} & \langle\lambda x^{\mathrm{N}}.\lambda y^{\mathrm{N}}.x^{\mathrm{N}}y^{\mathrm{N}} \| I_{\mathrm{N}} \cdot I_{\mathrm{N}} \cdot \star^{\mathrm{N}}\rangle \\
\downarrow_{\trianglerighteq} & \downarrow_{\trianglerighteq} \\
\underbrace{(\lambda y^{\mathrm{N}}.I_{\mathrm{N}}y^{\mathrm{N}})}I_{\mathrm{N}} & \langle\lambda y^{\mathrm{N}}.I_{\mathrm{N}}y^{\mathrm{N}} \| I_{\mathrm{N}} \cdot \star^{\mathrm{N}}\rangle \\
\downarrow_{\trianglerighteq} & \downarrow_{\trianglerighteq} \\
\underbrace{I_{\mathrm{N}}I_{\mathrm{N}}} & \langle I_{\mathrm{N}}I_{\mathrm{N}} \| \star^{\mathrm{N}}\rangle \\
\equiv_{\trianglerighteq} & \equiv_{\trianglerighteq} \\
\underbrace{I_{\mathrm{N}}}I_{\mathrm{N}} & \langle I_{\mathrm{N}} \| I_{\mathrm{N}} \cdot \star^{\mathrm{N}}\rangle \\
\downarrow_{\trianglerighteq} & \downarrow_{\trianglerighteq} \\
\underbrace{I_{\mathrm{N}}} & \langle I_{\mathrm{N}} \| \star^{\mathrm{N}}\rangle
\end{array}
$$

tions immediately follows from the equations on expressions and stacks.

Ambiguity of the ambiant calculus

There is sometimes a slight ambiguity on which calculus an expression T_{N} lives: it could live in $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$, $\underline{\lambda}^{\to}_{\mathrm{N}}$, or $\mathrm{M}^{\to}_{\mathrm{N}}$. Most of the time, this ambiguity is unimportant, but it sometimes needs to be resolved:

Remark I.3.3

Translating the of disubstitutions on expressions T_{N} described in Figure I.3.1d to $\underline{\lambda}^{\to}_{\mathrm{N}}$ would yield

$$T_{\mathrm{N}}[\sigma, \mathbb{S}_{\mathrm{N}}] = T_{\mathrm{N}}[\sigma] \quad \text{in } \underline{\lambda}^{\to}_{\mathrm{N}}$$

which would clash with

$$T_{\mathrm{N}}[\sigma, \mathbb{S}_{\mathrm{N}}] = \mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}[\sigma]} \quad \text{in } \boldsymbol{\lambda}^{\to}_{\mathrm{N}}$$

This mismatch would not be that problematic because it can be trivially resolved by making the ambiant calculus explicit. Furthermore, since the action of disubstitutions on expressions is uninteresting in $\underline{\lambda}^{\to}_{\mathrm{N}}$ and $\mathrm{M}^{\to}_{\mathrm{N}}$ (because they act like substitutions), we could simply take the convention that when writing $T_{\mathrm{N}}[\varphi]$, both T_{N} and φ live in $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$. We nevertheless avoided redefining the action of disubstitutions on expressions in Figure I.2.1d to avoid unnecessary confusion.

Remark I.3.4

The $_{\text{M}}$ in $\overset{\text{M}}{\triangleright}$ is redundant (i.e. we could denote $\overset{\text{M}}{\triangleright}$ by $\triangleright$) because $\triangleright$ only reduces expressions, while $\overset{\text{M}}{\triangleright}$ only reduces configurations, so that any reduction

$$T_{\text{N}} \triangleright T'_{\text{N}} \quad (\text{resp. } C_{\text{N}} \overset{\text{M}}{\triangleright} C'_{\text{N}})$$

necessarily happens in $\boldsymbol{\lambda}_{\text{N}}^{\rightarrow}$ (resp. $\underline{\lambda}_{\text{N}}^{\rightarrow}$ or $\text{M}_{\text{N}}^{\rightarrow}$). The remaining ambiguity between $\underline{\lambda}_{\text{N}}^{\rightarrow}$ and $\text{M}_{\text{N}}^{\rightarrow}$ is not problematic because those two calculi are basically the same (as will be shown in Section I.4).

We nevertheless keep writing $\overset{\text{M}}{\triangleright}$ for the reduction of $\underline{\lambda}_{\text{N}}^{\rightarrow}$ or $\text{M}_{\text{N}}^{\rightarrow}$ because the dinstinction between expressions T_{N} and configurations C_{N} may not be immediate for large terms, e.g.

$$\mathbb{S}_{\text{N}}^{1} \boxed{\ldots \boxed{\mathbb{S}_{\text{N}}^{q} \boxed{T_{\text{N}} U_{\text{N}}^{1} \ldots U_{\text{N}}^{r}}}} \quad \text{vs} \quad \mathbb{S}_{\text{N}}^{1} \boxed{\ldots \boxed{\mathbb{S}_{\text{N}}^{q} \boxed{C_{\text{N}} U_{\text{N}}^{1} \ldots U_{\text{N}}^{r}}}}$$

In $\underline{\lambda}_{\text{n}}^{\rightarrow}$ and $\text{Li}_{\text{n}}^{\rightarrow}$, the operational reduction will be denoted by $\triangleright$, and this will not lead to any semblance of ambiguity because we use lower cases letters to denote terms of $\underline{\lambda}_{\text{n}}^{\rightarrow}$ and $\text{Li}_{\text{n}}^{\rightarrow}$.

I.4. Equivalence between $\underline{\lambda}^{\to}_{\mathrm{N}}$ and $\mathrm{M}^{\to}_{\mathrm{N}}$

Inside-out and outside-out descriptions

Figure I.4.1: Syntax of $\underline{\lambda}^{\to}_{\mathrm{N}}$ and $\mathrm{M}^{\to}_{\mathrm{N}}$

Figure I.4.1.a: Syntax of $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and outside-out description of $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

Stacks:

$$\mathbf{S}_{\mathrm{N}} \ni \mathbb{S}_{\mathrm{N}} ::= \square \mid \mathbb{S}_{\mathrm{N}} T_{\mathrm{N}}$$

Configurations:

$$\mathbf{C}_{\mathrm{N}} \ni C_{\mathrm{N}} ::= \underline{T_{\mathrm{N}}} \mid C_{\mathrm{N}} T_{\mathrm{N}}$$

Stacks (outside-out):

$$\mathbf{S}_{\mathrm{N}} \ni S_{\mathrm{N}} ::= \star^{\mathrm{N}} \mid S_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$$

Configurations (outside-out):

$$\mathbf{C}_{\mathrm{N}} \ni C_{\mathrm{N}} ::= \langle T_{\mathrm{N}} | \star^{\mathrm{N}} \rangle \mid C_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$$

Figure I.4.1.b: Inside-out description of $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and syntax of $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

Stacks (inside-out):

$$\mathbb{S}_{\mathrm{N}} ::= \square \mid \mathbb{S}_{\mathrm{N}} \boxed{\square T_{\mathrm{N}}}$$

Configurations (inside-out):

$$\mathbf{C}_{\mathrm{N}} \ni C_{\mathrm{N}} ::= \mathbb{S}_{\mathrm{N}} \boxed{\underline{T_{\mathrm{N}}}}$$

Stacks:

$$S_{\mathrm{N}} ::= \star^{\mathrm{N}} \mid T_{\mathrm{N}} \cdot S_{\mathrm{N}}$$

Configurations:

$$C_{\mathrm{N}} ::= \langle T_{\mathrm{N}} | S_{\mathrm{N}} \rangle$$

Figure I.4.1.c: Expanded descriptions of $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

Stacks (expanded):

$$\mathbb{S}_{\mathrm{N}} ::= \square T^1_{\mathrm{N}} \dots T^k_{\mathrm{N}}$$

Configurations (expanded):

$$C_{\mathrm{N}} ::= \underline{T_{\mathrm{N}}} U^1_{\mathrm{N}} \dots U^k_{\mathrm{N}}$$

Stacks (expanded):

$$S_{\mathrm{N}} ::= T^1_{\mathrm{N}}$$

Configurations (expanded):

$$C_{\mathrm{N}} ::= \langle T_{\mathrm{N}} | U^1_{\mathrm{N}} \cdot \ldots \cdot U^q_{\mathrm{N}} \cdot \star^{\mathrm{N}} \rangle$$

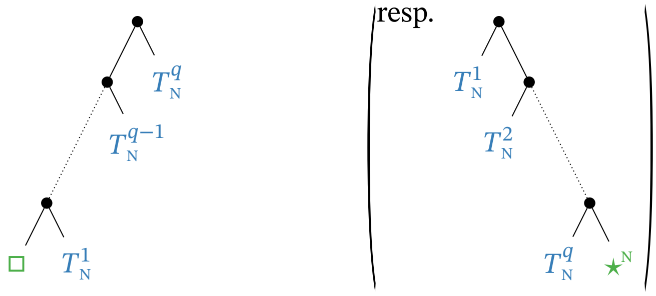
The right column of Figure I.4.1b and the left column of Figure I.4.1a recall the syntaxes of $\mathrm{M}^{\to}_{\mathrm{N}}$ and $\underline{\lambda}^{\to}_{\mathrm{N}}$ respectively. Though $\underline{\lambda}^{\to}_{\mathrm{N}}$ and $\mathrm{M}^{\to}_{\mathrm{N}}$ represent the same objects, they represent them in structurally different ways. Indeed, in $\underline{\lambda}^{\to}_{\mathrm{N}}$ (resp. $\mathrm{M}^{\to}_{\mathrm{N}}$) a stack

$$\mathbb{S}_{\mathrm{N}} = \square T^1_{\mathrm{N}} \dots T^q_{\mathrm{N}} \quad (\text{resp. } S_{\mathrm{N}} = T^1_{\mathrm{N}} \cdot \ldots \cdot T^q_{\mathrm{N}} \cdot \star^{\mathrm{N}})$$

is implicitly parenthesized as

$$\mathbb{S}_{\mathrm{N}} = ((\square T^1_{\mathrm{N}}) \dots) T^q_{\mathrm{N}} \quad (\text{resp. } S_{\mathrm{N}} = T^1_{\mathrm{N}} \cdot (\ldots \cdot (T^q_{\mathrm{N}} \cdot \star^{\mathrm{N}})))$$

i.e. its parse tree is a left (resp. right) comb:



Taking the structure of stacks in $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ as reference, stacks of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ are therefore *inside-out*, and we call stacks of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ (which are exactly stacks of $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$) *outside-out* by opposition. To make the difference in structure more apparent, we give an outside-out description of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ in the right column of Figure I.4.1a and an inside-out description of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ in the left column of Figure I.4.1b, using an operation that substitutes $\star^{\mathrm{N}}$ by a stack S_{N} and plugging.

The difference between inside-out and outside-out descriptions is fairly inconsequential here because both are clearly equivalent to the expanded descriptions given in Figure I.4.1c. However, in more complex calculi, expanded descriptions become unusable. Since we only study $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ and $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ as a stepping stone towards more complex calculi, we therefore avoid using expanded descriptions.

Translations

Figure I.4.2 defines translations

$$\underset{\rightarrow}{\cdot} : \underline{\lambda}^{\rightarrow}_{\mathrm{N}} \to \mathrm{M}^{\rightarrow}_{\mathrm{N}} \quad \text{and} \quad \underset{\leftarrow}{\cdot} : \mathrm{M}^{\rightarrow}_{\mathrm{N}} \to \underline{\lambda}^{\rightarrow}_{\mathrm{N}}$$

It is immediate that the translation $\underset{\rightarrow}{\cdot}$ maps outside-out expressions (resp. stacks, configurations) of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ to outside-out expressions (resp. stacks, configurations) of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$, and that $\underset{\leftarrow}{\cdot}$ maps inside-out expressions (resp. stacks, configurations) of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ to inside-out expressions (resp. stacks, configurations) of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$. To consider them as translations between $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ and $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ (i.e. between outside-out $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ and inside-out $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$), we therefore need to show that:

- all inside-out expressions (resp. stacks, configurations) of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ are outside-out expressions (resp. stacks, configurations) of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$; and that
- all outside-out expressions (resp. stacks, configurations) of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ are inside-out expressions (resp. stacks, configurations) of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$.

This holds thanks to the following fact:

Fact I.4.1

- In $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$, for any stacks $\mathbb{S}^1_{\mathrm{N}}$ and $\mathbb{S}^2_{\mathrm{N}}$ (resp. stack $\mathbb{S}_{\mathrm{N}}$ and configuration C_{N}), $\mathbb{S}^2_{\mathrm{N}}\boxed{\mathbb{S}^1_{\mathrm{N}}}$ is a stack (resp. $\mathbb{S}_{\mathrm{N}}\boxed{C_{\mathrm{N}}}$ is a configuration).

Figure I.4.2: Translations $\underrightarrow{\cdot} : \mathrm{M}^{\rightarrow}_{\mathrm{N}} \to \underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ and $\underleftarrow{\cdot} : \underline{\lambda}^{\rightarrow}_{\mathrm{N}} \to \mathrm{M}^{\rightarrow}_{\mathrm{N}}$

Figure I.4.2.a: Definition of $\underrightarrow{\cdot}$ and outside-out description of $\underleftarrow{\cdot}$

Terms:

$$\underrightarrow{T_{\mathrm{N}}} \stackrel{\text{def}}{=} T_{\mathrm{N}}$$

Stacks:

$$\underrightarrow{\square} \stackrel{\text{def}}{=} \star^{\mathrm{N}}$$

$$\underrightarrow{\mathbb{S}_{\mathrm{N}} T_{\mathrm{N}}} = \underrightarrow{(\square T_{\mathrm{N}})\boxed{\mathbb{S}_{\mathrm{N}}}} \stackrel{\text{def}}{=} \underrightarrow{\mathbb{S}_{\mathrm{N}}}[\underrightarrow{T_{\mathrm{N}}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$$

Configurations (outside-out):

$$\underrightarrow{\underbracket{T_{\mathrm{N}}}} \stackrel{\text{def}}{=} \langle \underrightarrow{T_{\mathrm{N}}} | \star^{\mathrm{N}} \rangle$$

$$\underrightarrow{C_{\mathrm{N}} T_{\mathrm{N}}} \stackrel{\text{def}}{=} \underrightarrow{C_{\mathrm{N}}}[\underrightarrow{T_{\mathrm{N}}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$$

Terms:

$$\underleftarrow{T_{\mathrm{N}}} = T_{\mathrm{N}}$$

Stacks:

$$\underleftarrow{\star^{\mathrm{N}}} = \square$$

$$\underleftarrow{S_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]} = (\square \underleftarrow{T_{\mathrm{N}}})\boxed{\underleftarrow{S_{\mathrm{N}}}} = \mathbb{S}_{\mathrm{N}} T_{\mathrm{N}}$$

Configurations:

$$\underleftarrow{\langle T_{\mathrm{N}} | \star^{\mathrm{N}} \rangle} = \underbracket{\underleftarrow{T_{\mathrm{N}}}}$$

$$\underleftarrow{C_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]} = (\square \underleftarrow{T_{\mathrm{N}}})\boxed{\underleftarrow{C_{\mathrm{N}}}} = C_{\mathrm{N}} T_{\mathrm{N}}$$

Figure I.4.2.b: Inside-out description of $\underrightarrow{\cdot}$ and definition of $\underleftarrow{\cdot}$

Terms:

$$\underrightarrow{T_{\mathrm{N}}} = T_{\mathrm{N}}$$

Stacks:

$$\underrightarrow{\square} = \star^{\mathrm{N}}$$

$$\underrightarrow{\mathbb{S}_{\mathrm{N}}\boxed{\square T_{\mathrm{N}}}} = \underrightarrow{T_{\mathrm{N}}} \cdot \underrightarrow{\mathbb{S}_{\mathrm{N}}}$$

Configurations:

$$\underrightarrow{\mathbb{S}_{\mathrm{N}}\boxed{\underbracket{T_{\mathrm{N}}}}} = \langle \underrightarrow{T_{\mathrm{N}}} | \underrightarrow{\mathbb{S}_{\mathrm{N}}} \rangle$$

Terms:

$$\underleftarrow{T_{\mathrm{N}}} \stackrel{\text{def}}{=} T_{\mathrm{N}}$$

Stacks:

$$\underleftarrow{\star^{\mathrm{N}}} \stackrel{\text{def}}{=} \square$$

$$\underleftarrow{T_{\mathrm{N}} \cdot S_{\mathrm{N}}} \stackrel{\text{def}}{=} \underleftarrow{S_{\mathrm{N}}}\boxed{\square \underleftarrow{T_{\mathrm{N}}}}$$

Configurations:

$$\underleftarrow{\langle T_{\mathrm{N}} | S_{\mathrm{N}} \rangle} \stackrel{\text{def}}{=} \underleftarrow{S_{\mathrm{N}}}\boxed{\underbracket{\underleftarrow{T_{\mathrm{N}}}}}$$

Figure I.4.2.c: Expanded description of $\underrightarrow{\cdot}$ and $\underleftarrow{\cdot}$

Terms:

$$\underrightarrow{T_{\mathrm{N}}} = T_{\mathrm{N}}$$

Stacks:

$$\underrightarrow{\square T^1_{\mathrm{N}} \dots T^q_{\mathrm{N}}} = T^1_{\mathrm{N}} \cdot \dots \cdot T^q_{\mathrm{N}} \cdot \star^{\mathrm{N}}$$

Configurations:

$$\underrightarrow{\underbracket{T_{\mathrm{N}}} U^1_{\mathrm{N}} \dots U^q_{\mathrm{N}}} = \langle T_{\mathrm{N}} | U^1_{\mathrm{N}} \cdot \dots \cdot U^q_{\mathrm{N}} \cdot \star^{\mathrm{N}} \rangle$$

Terms:

$$\underleftarrow{T_{\mathrm{N}}} = T_{\mathrm{N}}$$

Stacks:

$$\underleftarrow{T^1_{\mathrm{N}} \cdot \dots \cdot T^q_{\mathrm{N}} \cdot \star^{\mathrm{N}}} = \square T^1_{\mathrm{N}} \dots T^q_{\mathrm{N}}$$

Configurations:

$$\underleftarrow{\langle T_{\mathrm{N}} | U^1_{\mathrm{N}} \cdot \dots \cdot U^q_{\mathrm{N}} \cdot \star^{\mathrm{N}} \rangle} \stackrel{\text{def}}{=} \underbracket{T_{\mathrm{N}}} U^1_{\mathrm{N}} \dots U^q_{\mathrm{N}}$$

- In $M_N^{\rightarrow}$, for any stacks S_N^1 and S_N^2 (resp. stack S_N and configuration C_N), $S_N^1[S_N^2/\star^N]$ is a stack (resp. $C_N[S_N/\star^N]$ is a configuration).

Proof

- By induction on $\mathbb{S}_N^2$ (resp. $\mathbb{S}_N$).
- By induction on S_N^1 (resp. C_N).

Fact I.4.2

The translation $\underrightarrow{\cdot}$ maps expressions (resp. stacks, configurations) of $\underline{\lambda}_N^{\rightarrow}$ to expressions (resp. stacks, configurations) of $M_N^{\rightarrow}$, and the translation $\underleftarrow{\cdot}$ maps (resp. stacks, configurations) of $M_N^{\rightarrow}$ to expressions (resp. stacks, configurations) of $\underline{\lambda}_N^{\rightarrow}$.

Proof

By the previous fact.

Proving that $\underrightarrow{\cdot}$ and $\underleftarrow{\cdot}$ are inverses amounts to proving that the translations distribute over plugging and stubstitutions of $\star^N$ by a stack, which in turn relies on these operations inducing a monoid structure on stacks, and an action of that monoid on configurations:

Fact I.4.3

In $\underline{\lambda}_N^{\rightarrow}$ (resp. $M_N^{\rightarrow}$), the set of stacks $\mathbf{S}_N$ has a monoid structure

$$(\mathbf{S}_N, \bigcirc_\square, \square) \quad (\text{resp. } (\mathbf{S}_N, \bigcirc_\star, \star^N))$$

where

$$\mathbb{S}_N^2 \bigcirc_\square \mathbb{S}_N^1 \stackrel{\text{def}}{=} \mathbb{S}_N^2 \boxed{\mathbb{S}_N^1} \quad (\text{resp. } S_N^1 \bigcirc_\star S_N^2 \stackrel{\text{def}}{=} S_N^1[S_N^2/\star^N])$$

and this monoid acts on configurations on the left (resp. on the right) via

$$\mathbb{S}_N \bullet_\square C_N \stackrel{\text{def}}{=} \mathbb{S}_N \boxed{C_N} \quad (\text{resp. } C_N \bullet_\star S_N \stackrel{\text{def}}{=} C_N[S_N/\star^N])$$

In other words:

- (mon-unit) for any stack $\mathbb{S}_N$ (resp. S_N), we have
$$\square\boxed{\mathbb{S}_N} = \mathbb{S}_N = \mathbb{S}_N\boxed{\square} \quad (\text{resp. } \star^N[S_N/\star^N] = S_N = S_N[\star^N/\star^N])$$
- (mon-accoc) for any stacks $\mathbb{S}_N^1$, $\mathbb{S}_N^2$, and $\mathbb{S}_N^3$ (resp. S_N^1, S_N^2, and S_N^3), we have
$$\mathbb{S}_N^3\boxed{\mathbb{S}_N^2\boxed{\mathbb{S}_N^1}} = \left(\mathbb{S}_N^3\boxed{\mathbb{S}_N^2}\right)\boxed{\mathbb{S}_N^1} \quad (\text{resp. } S_N^1[S_N^2/\star^N][S_N^3/\star^N] = S_N^1[S_N^2[S_N^3/\star^N]/\star^N])$$
- (act-unit) for any configuration C_N, we have
$$\square\boxed{C_N} = C_N \quad (\text{resp. } C_N = C_N[\star^N/\star^N])$$

- (act-assoc) for any configuration C^1_{N} and stacks $\mathbb{S}^2_{\mathrm{N}}$ and $\mathbb{S}^3_{\mathrm{N}}$ (resp. S^2_{N} and S^3_{N}), we have

$$\mathbb{S}^3_{\mathrm{N}} \boxed{\mathbb{S}^2_{\mathrm{N}} \boxed{C^1_{\mathrm{N}}}} = \left(\mathbb{S}^3_{\mathrm{N}} \boxed{\mathbb{S}^2_{\mathrm{N}}}\right) \boxed{C^1_{\mathrm{N}}} \quad (\text{resp. } C^1_{\mathrm{N}}[S^2_{\mathrm{N}}/\star^{\mathrm{N}}][S^3_{\mathrm{N}}/\star^{\mathrm{N}}] = C^1_{\mathrm{N}}[S^2_{\mathrm{N}}[S^3_{\mathrm{N}}/\star^{\mathrm{N}}]/\star^{\mathrm{N}}])$$

Proof sketch (See page 187 for details)

By a few inductions.

Fact I.4.4

- For any stacks $\mathbb{S}_{\mathrm{N}}$ and $\mathbb{S}^0_{\mathrm{N}}$ (resp. stack $\mathbb{S}_{\mathrm{N}}$ and configuration C_{N}) of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$, we have

$$\underrightarrow{\mathbb{S}^2_{\mathrm{N}} \boxed{\mathbb{S}^1_{\mathrm{N}}}} = \underrightarrow{\mathbb{S}^1_{\mathrm{N}}}[\underrightarrow{\mathbb{S}^2_{\mathrm{N}}}/\star^{\mathrm{N}}] \quad \left(\text{resp. } \underrightarrow{\mathbb{S}_{\mathrm{N}} \boxed{C_{\mathrm{N}}}} = \underrightarrow{C_{\mathrm{N}}}[\underrightarrow{\mathbb{S}_{\mathrm{N}}}/\star^{\mathrm{N}}]\right)$$

- For any stacks S_{N} and S^0_{N} (resp. stack S_{N} and configuration C_{N}) of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$, we have

$$\underleftarrow{S^1_{\mathrm{N}}[S^2_{\mathrm{N}}/\star^{\mathrm{N}}]} = \underleftarrow{S^2_{\mathrm{N}}} \boxed{\underleftarrow{S^1_{\mathrm{N}}}} \quad \left(\text{resp. } \underleftarrow{C_{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}]} = \underleftarrow{S_{\mathrm{N}}} \boxed{\underleftarrow{C_{\mathrm{N}}}}\right)$$

Proof

By induction on $\mathbb{S}_{\mathrm{N}}$ / S_{N} (resp. $\mathbb{S}^2_{\mathrm{N}}$ / S^2_{N}), using the previous fact.

Fact I.4.5

The translation $\underrightarrow{\cdot}$ and $\underleftarrow{\cdot}$ are are each other's inverse:

- For any expression T_{N} (resp. stack $\mathbb{S}_{\mathrm{N}}$, configuration C_{N}) of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$, we have

$$\underleftarrow{\underrightarrow{T_{\mathrm{N}}}} = T_{\mathrm{N}} \quad \left(\text{resp. } \underleftarrow{\underrightarrow{\mathbb{S}_{\mathrm{N}}}} = \mathbb{S}_{\mathrm{N}}, \quad \underleftarrow{\underrightarrow{C_{\mathrm{N}}}} = C_{\mathrm{N}}\right)$$

- For any expression T_{N} (resp. stack S_{N}, configuration C_{N}) of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$, we have

$$T_{\mathrm{N}} = \underrightarrow{\underleftarrow{T_{\mathrm{N}}}} \quad \left(\text{resp. } S_{\mathrm{N}} = \underrightarrow{\underleftarrow{S_{\mathrm{N}}}}, \quad C_{\mathrm{N}} = \underrightarrow{\underleftarrow{C_{\mathrm{N}}}}\right)$$

Proof

By induction on the term, using the previous fact.

We write $\rightleftharpoons$ for equality through these translations:

Definition I.4.6

We write $t_1 \rightleftharpoons t_2$ to state that $\underrightarrow{t_1} = t_2$, or equivalently that $t_1 = \underleftarrow{t_2}$. Whenever we write, $t_1 \rightleftharpoons t_2$, we implicitly assume that t_1 lives in $\underline{\lambda}_{N}^{\to}$ and that t_2 lives in $M_{N}^{\to}$.

With this notation, Fact I.4.4 can be reformulated as the compatibility of the operations with $\rightleftharpoons$:

Fact I.4.7

We have

$$\mathbb{S}_{N}^{\langle 1} \rightleftharpoons S_{N}^{\rangle 1} \text{ and } \mathbb{S}_{N}^{\langle 2} \rightleftharpoons S_{N}^{\rangle 2} \quad \Rightarrow \quad \mathbb{S}_{N}^{\langle 2}\boxed{\mathbb{S}_{N}^{\langle 1}} \rightleftharpoons S_{N}^{\rangle 1}[S_{N}^{\rangle 2}/\star^{N}]$$

and

$$\mathbb{S}_{N}^{\langle} \rightleftharpoons S_{N}^{\rangle} \text{ and } C_{N}^{\langle} \rightleftharpoons C_{N}^{\rangle} \quad \Rightarrow \quad \mathbb{S}_{N}^{\langle}\boxed{C_{N}^{\langle}} \rightleftharpoons C_{N}^{\rangle}[S_{N}^{\rangle}/\star^{N}]$$

Proof

By Fact I.4.4.

Remark I.4.8

Every notion will be defined in both $\underline{\lambda}_{N}^{\to}$ and $M_{N}^{\to}$, and shown to be compatible with $\rightleftharpoons$ (i.e. to be the same in $\underline{\lambda}_{N}^{\to}$ and $M_{N}^{\to}$). We will often also give an equivalent outside-out (resp. inside-out) description in $M_{N}^{\to}$ (resp. $\underline{\lambda}_{N}^{\to}$), but will leave the proof of the equivalence implicit[a]. From a technical perspective, the alternative descriptions are completely superfluous, and the reader should feel free to ignore them, but we nevertheless keep them because we believe that they may have some pedagogical value.

[a] The outside-out (resp. inside-out) description in $M_{N}^{\to}$ (resp. $\underline{\lambda}_{N}^{\to}$) will be defined as being exactly the definition in $\underline{\lambda}_{N}^{\to}$ (resp. $M_{N}^{\to}$) transported through $\rightleftharpoons$, so that the equivalence between the outside-out (resp. inside-out) description in $M_{N}^{\to}$ (resp. $\underline{\lambda}_{N}^{\to}$) and the definition in $M_{N}^{\to}$ (resp. $\underline{\lambda}_{N}^{\to}$) will immediately follow from compatibility of the definition with $\rightleftharpoons$.

For example, if we define a unary operation $f_{\langle}(C_{N})$ in $\underline{\lambda}_{N}^{\to}$ and the corresponding operation $f_{\rangle}(C_{N})$ in $M_{N}^{\to}$, we will show that

$$C_{N}^{1} \rightleftharpoons C_{N}^{2} \quad \Rightarrow \quad f_{\langle}(C_{N}^{1}) \rightleftharpoons f_{\rangle}(C_{N}^{2}) \qquad (1)$$

The outside-out (resp. inside-out) description $f_{\rangle}^{\rightleftharpoons}$ (resp. $f_{\langle}^{\rightleftharpoons}$) in $M_{N}^{\to}$ (resp. $\underline{\lambda}_{N}^{\to}$) will be defined by starting from the equalities

$$f_{\langle}^{\rightleftharpoons}(C_{N}^{2}) = \underrightarrow{f_{\langle}(\underleftarrow{C_{N}^{2}})} \quad \left(\text{resp. } f_{\rangle}^{\rightleftharpoons}(C_{N}^{1}) = \underleftarrow{f_{\rangle}(\underrightarrow{C_{N}^{1}})}\right)$$

and possibly simplifying the right hand side. By (1), we therefore immediately get

$$f_{\langle}^{\rightleftharpoons} = f_{\rangle} \quad (\text{resp. } f_{\rangle}^{\rightleftharpoons} = f_{\langle})$$

Substitutions

Figure I.4.3 recalls the action of substitutions on stacks and configurations of $\underline{\lambda}_{N}^{\to}$ and $M_{N}^{\to}$, and gives alternative descriptions. The translations are extended to substitutions pointwise:

Figure I.4.3: The action of substitutions on terms of $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ and $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$

Figure I.4.3.a: Definition in $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ (left) and outside-out description in $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$

Stacks:

$$\square[\sigma] \stackrel{\mathrm{def}}{=} \square$$
$$(\mathbb{S}_{\mathrm{N}} T_{\mathrm{N}})[\sigma] \stackrel{\mathrm{def}}{=} (\mathbb{S}_{\mathrm{N}}[\sigma])(T_{\mathrm{N}}[\sigma])$$

Configurations:

$$\underline{T_{\mathrm{N}}}[\sigma] \stackrel{\mathrm{def}}{=} \underline{T_{\mathrm{N}}[\sigma]}$$
$$(C_{\mathrm{N}} T_{\mathrm{N}})[\sigma] \stackrel{\mathrm{def}}{=} (C_{\mathrm{N}}[\sigma])(T_{\mathrm{N}}[\sigma])$$

Stacks:

$$\star^{\mathrm{N}}[\sigma] = \star^{\mathrm{N}}$$
$$(S_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}])[\sigma] = S_{\mathrm{N}}[\sigma][(T_{\mathrm{N}}[\sigma]) \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$$

Configurations:

$$\langle T_{\mathrm{N}} | \star^{\mathrm{N}} \rangle[\sigma] = \langle T_{\mathrm{N}}[\sigma] | \star^{\mathrm{N}} \rangle[\sigma]$$
$$(C_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}])[\sigma] = C_{\mathrm{N}}[\sigma][(T_{\mathrm{N}}[\sigma]) \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$$

Figure I.4.3.b: Inside-out description in $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ (left) and definition in $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$ (right)

Stacks:

$$\square[\sigma] = \square$$
$$(\mathbb{S}_{\mathrm{N}} \boxed{\square T_{\mathrm{N}}})[\sigma] = (\mathbb{S}_{\mathrm{N}}[\sigma]) \boxed{\square (T_{\mathrm{N}}[\sigma])}$$

Configurations:

$$(\mathbb{S}_{\mathrm{N}} \boxed{\underline{T_{\mathrm{N}}}})[\sigma] = (\mathbb{S}_{\mathrm{N}}[\sigma]) \boxed{\underline{T_{\mathrm{N}}[\sigma]}}$$

Stacks:

$$\star^{\mathrm{N}}[\sigma] \stackrel{\mathrm{def}}{=} \star^{\mathrm{N}}$$
$$(T_{\mathrm{N}} \cdot S_{\mathrm{N}})[\sigma] \stackrel{\mathrm{def}}{=} (T_{\mathrm{N}}[\sigma]) \cdot (S_{\mathrm{N}}[\sigma])$$

Configurations:

$$\langle T_{\mathrm{N}} | S_{\mathrm{N}} \rangle[\sigma] \stackrel{\mathrm{def}}{=} \langle T_{\mathrm{N}}[\sigma] | S_{\mathrm{N}}[\sigma] \rangle$$

Figure I.4.3.c: Expanded descriptions in $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ (top) and $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$ (bottom)

Stacks:

$$(\square T_{\mathrm{N}}^{1} \dots T_{\mathrm{N}}^{q})[\sigma] = \square(T_{\mathrm{N}}^{1}[\sigma]) \dots (T_{\mathrm{N}}^{q}[\sigma])$$

Configurations:

$$(\underline{T_{\mathrm{N}}^{0}} T_{\mathrm{N}}^{1} \dots T_{\mathrm{N}}^{q})[\sigma] = \underline{T_{\mathrm{N}}^{0}[\sigma]}(T_{\mathrm{N}}^{1}[\sigma]) \dots (T_{\mathrm{N}}^{q}[\sigma])$$

Stacks:

$$(T_{\mathrm{N}}^{1} \cdot \dots \cdot T_{\mathrm{N}}^{q} \cdot \star^{\mathrm{N}})[\sigma] = (T_{\mathrm{N}}^{1}[\sigma]) \cdot \dots \cdot (T_{\mathrm{N}}^{q}[\sigma]) \cdot \star^{\mathrm{N}}$$

Configurations:

$$(\langle T_{\mathrm{N}}^{0} | T_{\mathrm{N}}^{1} \cdot \dots \cdot T_{\mathrm{N}}^{q} \cdot \star^{\mathrm{N}} \rangle)[\sigma] = \langle T_{\mathrm{N}}^{0} | (T_{\mathrm{N}}^{1}[\sigma]) \cdot \dots \cdot (T_{\mathrm{N}}^{q}[\sigma]) \cdot \star^{\mathrm{N}} \rangle$$

Definition I.4.9: Extension of $\rightleftharpoons$ to substitutions

Given a substitution σ in $\underline{\lambda}_{\mathrm{N}}^{\to}$ (resp. $\mathrm{M}_{\mathrm{N}}^{\to}$), we write $\underset{\to}{\sigma}$ (resp. $\underset{\leftarrow}{\sigma}$) for the substitution of $\mathrm{M}_{\mathrm{N}}^{\to}$ (resp. $\underline{\lambda}_{\mathrm{N}}^{\to}$) defined by

$$\underset{\to}{\sigma}(x^{\mathrm{N}}) = \underset{\longrightarrow}{\sigma(x^{\mathrm{N}})} \quad \left(\text{resp. } \underset{\leftarrow}{\sigma}(x^{\mathrm{N}}) = \underset{\longleftarrow}{\sigma(x^{\mathrm{N}})}\right)$$

Given two substitutions, $\sigma_{<}$ in $\underline{\lambda}_{\mathrm{N}}^{\to}$ and $\sigma_{>}$ in $\mathrm{M}_{\mathrm{N}}^{\to}$, we write $\sigma_{<} \rightleftharpoons \sigma_{>}$ for $\underset{\longrightarrow}{\sigma_{<}} = \sigma_{>}$, or equivalently for $\sigma_{<} = \underset{\longleftarrow}{\sigma_{>}}$.

Remark I.4.10

Since expressions are the same in $\underline{\lambda}_{\mathrm{N}}^{\to}$ and $\mathrm{M}_{\mathrm{N}}^{\to}$, we have

$$\sigma_{<} \rightleftharpoons \sigma_{>} \quad \Leftrightarrow \quad \sigma_{<} = \sigma_{>}$$

We nevertheless use the notation $\sigma_{<} \rightleftharpoons \sigma_{>}$ because this will no longer be the case in $\underline{\lambda}_{\mathrm{n}}^{\to}$ and $\mathrm{Li}_{\mathrm{n}}^{\to}$.

Translations distribute over the action of substitutions, i.e. substitutions are compatible with $\rightleftharpoons$:

Fact I.4.11: Compatibility of substitutions with $\rightleftharpoons$

We have

$$\sigma_{<} \rightleftharpoons \sigma_{>} \text{ and } \mathbb{S}_{\mathrm{N}}^{<} \rightleftharpoons S_{\mathrm{N}}^{>} \quad \Rightarrow \quad \mathbb{S}_{\mathrm{N}}^{<}[\sigma_{<}] \rightleftharpoons S_{\mathrm{N}}^{>}[\sigma_{>}]$$

and

$$\sigma_{<} \rightleftharpoons \sigma_{>} \text{ and } C_{\mathrm{N}}^{<} \rightleftharpoons C_{\mathrm{N}}^{>} \quad \Rightarrow \quad C_{\mathrm{N}}^{<}[\sigma_{<}] \rightleftharpoons C_{\mathrm{N}}^{>}[\sigma_{>}]$$

Proof

By induction on $\mathbb{S}_{\mathrm{N}}^{<}$ / $C_{\mathrm{N}}^{<}$.

Disubstitutions

The translations are extended to disubstitutions in the expected way:

Definition I.4.12: Extension of $\rightleftharpoons$ to disubstitutions

Given a disubstitution $\varphi = (\sigma, \mathbb{S}_{N})$ in $\underline{\lambda}_{N}^{\rightarrow}$ (resp. $\varphi = \sigma, \star^{N} \mapsto S_{N}$ in $M_{N}^{\rightarrow}$), we write $\underrightarrow{\sigma}$ (resp. $\underleftarrow{\sigma}$) for the substitution of $M_{N}^{\rightarrow}$ (resp. $\underline{\lambda}_{N}^{\rightarrow}$) defined by

$$\underrightarrow{(\sigma, \mathbb{S}_{N})} = \underrightarrow{\sigma}, \star^{N} \mapsto \underrightarrow{\mathbb{S}_{N}} \quad \left(\text{resp. } \underleftarrow{\sigma, \star^{N} \mapsto S_{N}} = (\underleftarrow{\sigma}, \underleftarrow{S_{N}})\right)$$

Given two disubstitutions, $\varphi_{\langle}$ in $\underline{\lambda}_{N}^{\rightarrow}$ and $\varphi_{\rangle}$ in $M_{N}^{\rightarrow}$, we write $\varphi_{\langle} \rightleftharpoons \varphi_{\rangle}$ for $\underrightarrow{\varphi_{\langle}} = \varphi_{\rangle}$, or equivalently for $\varphi_{\langle} = \underleftarrow{\varphi_{\rangle}}$.

The translations distribute over the translations:

Fact I.4.13: Compatibility of disubstitutions with $\rightleftharpoons$

We have

$$\varphi_{\langle} \rightleftharpoons \varphi_{\rangle} \text{ and } \mathbb{S}_{N}^{\langle} \rightleftharpoons S_{N}^{\rangle} \quad \Rightarrow \quad \mathbb{S}_{N}^{\langle}[\varphi_{\langle}] \rightleftharpoons S_{N}^{\rangle}[\varphi_{\rangle}]$$

and

$$\varphi_{\langle} \rightleftharpoons \varphi_{\rangle} \text{ and } C_{N}^{\langle} \rightleftharpoons C_{N}^{\rangle} \quad \Rightarrow \quad C_{N}^{\langle}[\varphi_{\langle}] \rightleftharpoons C_{N}^{\rangle}[\varphi_{\rangle}]$$

Proof

This holds for substitutions σ by Fact I.4.11 and for disubstitutions of the shape $\varphi_{\langle} = (\mathrm{Id}, \mathbb{S}_{N})$ (resp. $\varphi_{\rangle} = \star^{N} \mapsto S_{N}$) by Fact I.4.7. We can therefore conclude by Fact I.3.2.

Reductions

The definitions of the operational reduction $\overset{M}{\triangleright}$ of $\underline{\lambda}_{N}^{\rightarrow}$ and $M_{N}^{\rightarrow}$ are recalled in Figure I.4.4a. These two definitions correspond to each other through $\rightleftharpoons$:

Fact I.4.14: Compatibility of $\overset{M}{\triangleright}$ with $\rightleftharpoons$

We have

$$C_{N}^{\langle} \rightleftharpoons C_{N}^{\rangle} \text{ and } C_{N}^{\langle\prime} \rightleftharpoons C_{N}^{\rangle\prime} \quad \Rightarrow \quad (C_{N}^{\langle} \overset{M}{\triangleright} C_{N}^{\langle\prime} \Leftrightarrow C_{N}^{\rangle} \overset{M}{\triangleright} C_{N}^{\rangle\prime})$$

Proof

By compatibility of disubstitutions with $\rightleftharpoons$ (Fact I.4.13).

Figure I.4.4: Operational reduction in $\underline{\lambda}^{\to}_{\mathrm{N}}$ and $\mathrm{M}^{\to}_{\mathrm{N}}$

Figure I.4.4.a: Definition in $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

$$\mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}U_{\mathrm{N}}}$$

$$\mathbb{S}_{\mathrm{N}}\boxed{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\to} \mathbb{S}_{\mathrm{N}}\boxed{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}$$

$$\overset{\mathrm{M}}{\triangleright} \overset{\mathrm{def}}{=} \overset{\mathrm{M}}{\triangleright}_{\to} \cup \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$$

$$\langle T_{\mathrm{N}}U_{\mathrm{N}} | S_{\mathrm{N}}\rangle \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} \langle T_{\mathrm{N}} | U_{\mathrm{N}} \cdot S_{\mathrm{N}}\rangle$$

$$\langle \lambda x^{\mathrm{N}}.T_{\mathrm{N}} | U_{\mathrm{N}} \cdot S_{\mathrm{N}}\rangle \overset{\mathrm{M}}{\triangleright}_{\to} \langle T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}] | S_{\mathrm{N}}\rangle$$

$$\overset{\mathrm{M}}{\triangleright} \overset{\mathrm{def}}{=} \overset{\mathrm{M}}{\triangleright}_{\to} \cup \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$$

Figure I.4.4.b: Outside-out description in $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

$$\frac{}{\underbrace{T_{\mathrm{N}}U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright} \underbrace{T_{\mathrm{N}}}U_{\mathrm{N}}}$$

$$\frac{}{\underbrace{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})}U_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright} \underbrace{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}}$$

$$\frac{C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright} C'_{\mathrm{N}}}{C_{\mathrm{N}}U_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright} C'_{\mathrm{N}}U_{\mathrm{N}}}$$

$$\frac{}{\langle T_{\mathrm{N}}U_{\mathrm{N}} | \star^{\mathrm{N}}\rangle \overset{\mathrm{M}}{\triangleright} \langle T_{\mathrm{N}} | U_{\mathrm{N}} \cdot \star^{\mathrm{N}}\rangle}$$

$$\frac{}{\langle \lambda x^{\mathrm{N}}.T_{\mathrm{N}} | U_{\mathrm{N}} \cdot \star^{\mathrm{N}}\rangle \overset{\mathrm{M}}{\triangleright} \langle T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}] | \star^{\mathrm{N}}\rangle}$$

$$\frac{C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright} C'_{\mathrm{N}}}{C_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}}/\star^{\mathrm{N}}] \overset{\mathrm{M}}{\triangleright} C'_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}}/\star^{\mathrm{N}}]}$$

Figure I.4.4.c: Inside-out description in $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

$$\frac{}{\mathbb{S}_{\mathrm{N}}\boxed{\underbrace{T_{\mathrm{N}}U_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright} \mathbb{S}_{\mathrm{N}}\boxed{\underbrace{T_{\mathrm{N}}}U_{\mathrm{N}}}}$$

$$\frac{}{\mathbb{S}_{\mathrm{N}}\boxed{\underbrace{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})}U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright} \mathbb{S}_{\mathrm{N}}\boxed{\underbrace{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}}}$$

$$\frac{}{\langle T_{\mathrm{N}}U_{\mathrm{N}} | S_{\mathrm{N}}\rangle \overset{\mathrm{M}}{\triangleright} \langle T_{\mathrm{N}} | U_{\mathrm{N}} \cdot S_{\mathrm{N}}\rangle}$$

$$\frac{}{\langle \lambda x^{\mathrm{N}}.T_{\mathrm{N}} | U_{\mathrm{N}} \cdot S_{\mathrm{N}}\rangle \overset{\mathrm{M}}{\triangleright} \langle T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}] | S_{\mathrm{N}}\rangle}$$

Figures I.4.4b and I.4.4c give inside-out and outside-out descriptions via inferrence rules, which are of course equivalent:

Fact I.4.15

In $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ (resp. $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$), the definition of $\overset{\mathrm{M}}{\triangleright}$ (Figure I.4.4a) is equivalent to its outside-out description (Figure I.4.4b), and to its inside-out description (Figure I.4.4c).

Proof

- definition $\Leftrightarrow$ inside-out Trivial.
- definition $\Leftrightarrow$ outside-out Thinking of $\mathbb{S}_{\mathrm{N}}$ (resp. S_{N}) as being an outside-out stack, the $\Rightarrow$ implication holds by induction on the stack, and the $\Leftarrow$ holds by induction on the derivation.

The fact that stacks should be inside-out in Figure I.4.4c would be clearer if we were to give a similar inside-out description of the strong reduction $\multimap\!\!\triangleright$. For example, we would have

$$\frac{\dfrac{U_{\mathrm{N}} \multimap\!\!\triangleright U'_{\mathrm{N}}}{\mathbb{S}_{\mathrm{N}}\boxed{\square U_{\mathrm{N}}} \multimap\!\!\triangleright \mathbb{S}_{\mathrm{N}}\boxed{\square U'_{\mathrm{N}}}}}{\mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}} U_{\mathrm{N}}} \multimap\!\!\triangleright \mathbb{S}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}} U'_{\mathrm{N}}}}$$

An inside-out description of $\multimap\!\!\triangleright$ in $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ and $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ is given in ⚠. We do not given one in $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ and $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$ because these calculi are not the right setting to study the strong reduction $\multimap\!\!\triangleright$.

I.5. Translations between $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ and $\underline{\lambda}^{\to}_{\mathrm{N}}$

Focus insertion and erasure

We start by defining the focus-erasing translation $\lceil \cdot \rceil$ from $\underline{\lambda}^{\to}_{\mathrm{N}}$ to $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ in Figure I.5.1 that removes the underlinement in C_{N}, and the corresponding translation from $\mathrm{M}^{\to}_{\mathrm{N}}$ to $\boldsymbol{\lambda}^{\to}_{\mathrm{N}}$, which we denote by the same symbol.

Figure I.5.1: The focus-erasing translations $\lceil \cdot \rceil : \underline{\lambda}^{\to}_{\mathrm{N}} \to \boldsymbol{\lambda}^{\to}_{\mathrm{N}}$ and $\lceil \cdot \rceil : \mathrm{M}^{\to}_{\mathrm{N}} \to \boldsymbol{\lambda}^{\to}_{\mathrm{N}}$

Figure I.5.1.a: Definition in $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and outside-out description in $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

$\lceil \underline{T_{\mathrm{N}}} \rceil \overset{\mathrm{def}}{=} T_{\mathrm{N}}$	$\lceil \langle T_{\mathrm{N}} \mid \star^{\mathrm{N}} \rangle \rceil \overset{\mathrm{def}}{=} T_{\mathrm{N}}$
$\lceil C_{\mathrm{N}} T_{\mathrm{N}} \rceil \overset{\mathrm{def}}{=} \lceil C_{\mathrm{N}} \rceil T_{\mathrm{N}}$	$\lceil C_{\mathrm{N}}[T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}] \rceil \overset{\mathrm{def}}{=} \lceil C_{\mathrm{N}} \rceil [T_{\mathrm{N}} \cdot \star^{\mathrm{N}} / \star^{\mathrm{N}}]$

Figure I.5.1.b: Inside-out description in $\underline{\lambda}^{\to}_{\mathrm{N}}$ (left) and definition in $\mathrm{M}^{\to}_{\mathrm{N}}$ (right)

$\lceil \mathbb{S}_{\mathrm{N}} \boxed{T_{\mathrm{N}}} \rceil \overset{\mathrm{def}}{=} \mathbb{S}_{\mathrm{N}} \boxed{T_{\mathrm{N}}}$	$\lceil \langle T_{\mathrm{N}} \mid S_{\mathrm{N}} \rangle \rceil \overset{\mathrm{def}}{=} \underleftarrow{\mathbb{S}_{\mathrm{N}}} \boxed{T_{\mathrm{N}}}$

These two translations of course correspond to each other through $\rightleftharpoons$:

Fact I.5.1

We have

$$C^{<}_{\mathrm{N}} \rightleftharpoons C^{>}_{\mathrm{N}} \quad \Rightarrow \quad \lceil C^{<}_{\mathrm{N}} \rceil = \lceil C^{>}_{\mathrm{N}} \rceil$$

Proof

Immediate.

Erasing the underlinement preserves disubsitutions:

Fact I.5.2

For any configuration C_{N} and disubstitution φ, $\lceil C_{\mathrm{N}}[\varphi] \rceil = \lceil C_{\mathrm{N}} \rceil [\varphi]$.

Proof

Immediate.

The focus-erasing translation is a left inverse of $\underline{\cdot}$ and $\langle - \mid \star^{\mathrm{N}} \rangle$:

Fact I.5.3

For any expression T_{N}, we have

$$\lceil \underline{T_{N}} \rceil = T_{N} \quad \text{and} \quad \lceil \langle T_{N} | \star^{N} \rangle \rceil = T_{N}$$

Proof

Immediate.

Composing these two maps in the opposite order yields the identity only up to $\overset{M}{\triangleright}_{m}$ reductions:

Fact I.5.4

For any configuration C_{N} of $\lambda_{N}^{\to}$ (resp. $M_{N}^{\to}$), we have

$$\lceil \underline{C_{N}} \rceil \overset{M}{\triangleright}_{m}^{*} C_{N} \quad (\text{resp. } \langle \lceil C_{N} \rceil | \star^{N} \rangle \overset{M}{\triangleright}_{m}^{*} C_{N})$$

i.e. for any stack $\mathbb{S}_{N}$ (resp. S_{N}) and term T_{N} of $\lambda_{N}^{\to}$ (resp. $M_{N}^{\to}$), we have

$$\underline{\mathbb{S}_{N} \boxed{T_{N}}} \overset{M}{\triangleright}_{m}^{*} \mathbb{S}_{N} \underline{\boxed{T_{N}}} \quad \left(\text{resp. } \left\langle \overleftarrow{S_{N}} \boxed{T_{N}} \middle| \star^{N} \right\rangle \overset{M}{\triangleright}_{m}^{*} \langle T_{N} | S_{N} \rangle \right)$$

Proof

We have

$$\left\lceil \mathbb{S}_{N} \underline{\boxed{T_{N}}} \right\rceil = \underline{\mathbb{S}_{N} \boxed{T_{N}}} \overset{M}{\triangleright}_{m}^{*} \mathbb{S}_{N} \underline{\boxed{T_{N}}} \quad \left(\text{resp. } \langle \lceil \langle T_{N} | S_{N} \rangle \rceil | \star^{N} \rangle = \left\langle \overleftarrow{S_{N}} \boxed{T_{N}} \middle| \star^{N} \right\rangle \overset{M}{\triangleright}_{m}^{*} \langle T_{N} | S_{N} \rangle \right)$$

where the equality is given by Fact I.5.2, and the $\overset{M}{\triangleright}_{m}^{*}$ reduction sequence is obtained by induction on $\mathbb{S}_{N}$ (resp. S_{N}).

Reductions through focus erasure

Since $\overset{M}{\triangleright}_{m}$ reductions only move focus, they are erased by $\lceil \cdot \rceil$:

Fact I.5.5

If $C_{N} \overset{M}{\triangleright}_{m} C'_{N}$ then $\lceil C_{N} \rceil = \lceil C'_{N} \rceil$.

Proof

By Fact I.5.2.

Conversely, two configurations whose image by $\lceil \cdot \rceil$ are equal are related by $\triangleright_{m}$ steps:

Fact I.5.6

If $\lceil C_{\mathrm{N}} \rceil = \lceil C'_{\mathrm{N}} \rceil$ then either $C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}{}^{*}_{\mathrm{m}} C'_{\mathrm{N}}$ or $C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleleft}{}^{*}_{\mathrm{m}} C'_{\mathrm{N}}$.

Proof

Applying Fact I.5.4 to both C_{N} and C'_{N} yields

$$C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleleft}{}^{j}_{\mathrm{m}} \underline{\lceil C_{\mathrm{N}} \rceil} = \underline{\lceil C'_{\mathrm{N}} \rceil} \overset{\mathrm{M}}{\triangleright}{}^{k}_{\mathrm{m}} C'_{\mathrm{N}}$$

By determinism of $\overset{\mathrm{M}}{\triangleright}$, we can simplify this to get either $j = 0$ or $k = 0$.

The $\overset{\mathrm{M}}{\triangleright}_{\rightarrow}$ reductions of $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ are preserved by focus erasure:

Fact I.5.7

If $C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}_{\rightarrow} C'_{\mathrm{N}}$ then $\lceil C_{\mathrm{N}} \rceil \triangleright_{\rightarrow} \lceil C'_{\mathrm{N}} \rceil$.

Proof

By Fact I.5.2.

Reductions through focus insertion

A top-level reduction $\triangleright_{\rightarrow}$ in $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ becomes $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}\overset{\mathrm{M}}{\triangleright}_{\rightarrow}$ in $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$:

Fact I.5.8

If $T_{\mathrm{N}} \triangleright_{\rightarrow} T'_{\mathrm{N}}$ then $\underline{T_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}\overset{\mathrm{M}}{\triangleright}_{\rightarrow} \underline{T'_{\mathrm{N}}}$.

Proof

We have

$$\underline{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}_{\mathrm{m}} (\underline{\lambda x^{\mathrm{N}}.T_{\mathrm{N}}})U_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}_{\rightarrow} \underline{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}$$

The search for the redex is represented by a sequence of $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$ reductions. The $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$ reduction can find redexes under all operational contexts:

Fact I.5.9

For any operational contexts $\mathbb{O}_{\mathrm{N}}$ and expression T_{N}, $\underline{\mathbb{O}_{\mathrm{N}}\boxed{T_{\mathrm{N}}}} \overset{\mathrm{M}}{\triangleright}{}^{*}_{\mathrm{m}} \mathbb{O}_{\mathrm{N}}\boxed{\underline{T_{\mathrm{N}}}}$.

Proof

Since all operational contexts $\mathbb{O}_{\mathrm{N}}$ are stacks $\mathbb{S}_{\mathrm{N}}$, this is just Fact I.5.4.

We can therefore simulate $\triangleright_{\rightarrow}$ steps as follows:

Fact I.5.10

If $T_{\mathrm{N}} \triangleright_{\rightarrow} T'_{\mathrm{N}}$ then $\underline{T_{\mathrm{N}}} \overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow} \overset{M}{\triangleleft}{}^{*}_{m} \underline{T'_{\mathrm{N}}}$.

Proof

We have

$$\underline{\mathbb{O}_{\mathrm{N}} \boxed{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}}}} \overset{M}{\triangleright}{}^{*}_{m} \mathbb{O}_{\mathrm{N}} \boxed{\underline{(\lambda x^{\mathrm{N}}.T_{\mathrm{N}})U_{\mathrm{N}}}} \overset{M}{\triangleright}_{m} \overset{M}{\triangleright}_{\rightarrow} \mathbb{O}_{\mathrm{N}} \boxed{\underline{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}} \overset{M}{\triangleleft}{}^{*}_{m} \underline{\mathbb{O}_{\mathrm{N}} \boxed{T_{\mathrm{N}}[U_{\mathrm{N}}/x^{\mathrm{N}}]}}$$

where the $\overset{M}{\triangleright}{}^{*}_{m}$ and $\overset{M}{\triangleleft}{}^{*}_{m}$ reduction sequences are by Fact I.5.9.

We then look at sequences of reductions $\triangleright^{*}_{\rightarrow}$. The fact that the abstract machine does not need to go back to the top of the expression, sometimes called refocusing [DanNie04], can be expressed as follows:

Fact I.5.11

If $T_{\mathrm{N}} \triangleright^{k}_{\rightarrow} T'_{\mathrm{N}}$ then $\underline{T_{\mathrm{N}}} \left(\overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow}\right)^{k} \overset{M}{\triangleleft}{}^{*}_{m} \underline{T'_{\mathrm{N}}}$.

Proof

The previous fact gives us

$$\underline{T_{\mathrm{N}}} \left(\overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow} \overset{M}{\triangleleft}{}^{*}_{m}\right)^{k} \underline{T'_{\mathrm{N}}}$$

which can be rewritten (for $k \geq 1$) as

$$\underline{T_{\mathrm{N}}} \overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow} \left(\overset{M}{\triangleleft}{}^{*}_{m} \overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow}\right)^{k-1} \overset{M}{\triangleleft}{}^{*}_{m} \underline{T'_{\mathrm{N}}}$$

Since $\overset{M}{\triangleright}_{m}$ is deterministic, this implies

$$\underline{T_{\mathrm{N}}} \overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow} \left(\left(\overset{M}{\triangleleft}{}^{*}_{m} \cup \overset{M}{\triangleright}{}^{*}_{m}\right) \triangleright_{\rightarrow}\right)^{k-1} \overset{M}{\triangleleft}{}^{*}_{m} \underline{T'_{\mathrm{N}}}$$

and since $\overset{M}{\triangleright}_{m}$ and $\overset{M}{\triangleright}_{\rightarrow}$ have disjoint domains, we get

$$\underline{T_{\mathrm{N}}} \overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow} \left(\overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow}\right)^{k-1} \overset{M}{\triangleleft}{}^{*}_{m} \underline{T'_{\mathrm{N}}}$$

i.e.

$$\underline{T_{\mathrm{N}}} \left(\overset{M}{\triangleright}{}^{*}_{m} \overset{M}{\triangleright}_{\rightarrow}\right)^{k} \overset{M}{\triangleleft}{}^{*}_{m} \underline{T'_{\mathrm{N}}}$$

Since $\overset{M}{\triangleright}_{m}$ steps are erased by $\lceil \cdot \rceil$, the above implication is an equivalence, so that:

[DanNie04] "Refocusing in Reduction Semantics", Danvy and Nielsen, 2004

Proposition I.5.12

For any configurations C_{N} and C'_{N}, we have

$$\lceil C_{\mathrm{N}} \rceil \triangleright^{\circledast} \quad \Leftrightarrow \quad C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}^{\circledast}$$

In particular, for any expressions T_{N} and T'_{N}, we have

$$T_{\mathrm{N}} \triangleright^{\circledast} \quad \Leftrightarrow \quad \underline{T_{\mathrm{N}}} \overset{\mathrm{M}}{\triangleright}^{\circledast}$$

Proof

- $\boxed{\Rightarrow}$ Suppose that $T_{\mathrm{N}} \triangleright^{\circledast} T'_{\mathrm{N}}$. By the previous fact, we have

$$\underline{T_{\mathrm{N}}} \, (\overset{\mathrm{M}}{\triangleright}{}^{*}_{\mathrm{m}} \overset{\mathrm{M}}{\triangleright}_{\rightarrow})^{k} \, C'_{\mathrm{N}} \overset{\mathrm{M}}{\triangleleft}{}^{*}_{\mathrm{m}} \underline{T'_{\mathrm{N}}}$$

  for some C'_{N}. Since $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$ is strongly normalizing (because the depth of the expression minus the depth of $\underline{\;\cdot\;}$ in it strictly decreases at each $\overset{\mathrm{M}}{\triangleright}_{\mathrm{m}}$ step), we can find C''_{N} such that

$$\underline{T_{\mathrm{N}}} \, (\overset{\mathrm{M}}{\triangleright}{}^{*}_{\mathrm{m}} \overset{\mathrm{M}}{\triangleright}_{\rightarrow})^{k} \, C'_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}{}^{\circledast}_{\mathrm{m}} C''_{\mathrm{N}} \overset{\mathrm{M}}{\triangleleft}{}^{\circledast}_{\mathrm{m}} \underline{T'_{\mathrm{N}}}$$

  It now suffices to show that $C''_{\mathrm{N}} \overset{\mathrm{M}}{\not\triangleright}_{\rightarrow}$. By Fact I.5.5, we have

$$\lceil C''_{\mathrm{N}} \rceil = \lceil \underline{T'_{\mathrm{N}}} \rceil = T'_{\mathrm{N}}$$

  so that having $C''_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}_{\rightarrow}$ would contradict the hypothesis $T'_{\mathrm{N}} \not\triangleright_{\rightarrow}$ by Fact I.5.7.

- $\boxed{\Leftarrow}$ Suppose that $C_{\mathrm{N}} \overset{\mathrm{M}}{\triangleright}^{\circledast} C'_{\mathrm{N}}$. By Facts I.5.5 and I.5.7, we have $\lceil C_{\mathrm{N}} \rceil \triangleright^{*} \lceil C'_{\mathrm{N}} \rceil$. Since $C'_{\mathrm{N}} \overset{\mathrm{M}}{\not\triangleright}$, Fact I.5.10 allows to conclude that $\lceil C'_{\mathrm{N}} \rceil \not\triangleright_{\rightarrow}$.

I.6. A pure call-by-name λ-calculus with focus: $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$

In this section, we introduce the pure call-by-name λ-calculus with focus $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$, which is the λ-like syntax for the calculus we are really interested in: $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$. While it is suboptimal from a technical standpoint, we expect the $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ syntax to make understanding how $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ computes easier.

Section I.6.1 refines $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ to allow decomposing the strong reduction, Section I.6.2 add lets expressions, and Section I.6.3 describes the actual $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ calculus.

I.6.1. The simple fragment of the naive $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ calculus

Decomposing the strong reduction

As we have seen in Section I.5, the operational reduction of $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ and $\mathrm{M}^{\rightarrow}_{\mathrm{N}}$ refines that of $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ by making the implicit $\triangleright_{\mathrm{m}}$ steps explicit. This has the unfortunate consequence of damaging the relationship between the operational reduction $\triangleright$ and the strong reduction $\dashrightarrow\!\triangleright$. For example, the expression $(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}})V_{\mathrm{N}}$ (where $I_{\mathrm{N}} = \lambda y^{\mathrm{N}}.y^{\mathrm{N}}$) of $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ is represented by $\underline{(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}})V_{\mathrm{N}}}$

Figure I.6.1: Example of strong reduction in subterms of abstract machines

Figure I.6.1.a: Example in $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$

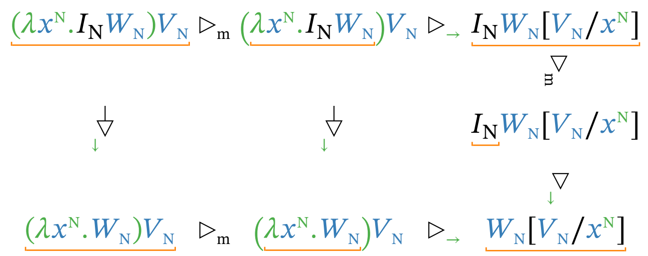
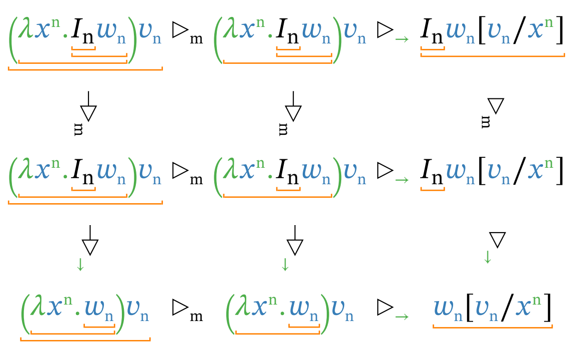


Figure I.6.1.b: Example in $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$



in $\underline{\lambda}_N^{\rightarrow}$ and reduces as shown in Figure I.6.1a, where the inner reduction $I_N W_N \triangleright W_N$ can not be refined as two steps $\triangleright_{\rightarrow} \triangleright_m$ because $I_N W_N$ has no underlined subterm.

A naive attempt at modifying $\underline{\lambda}_N^{\rightarrow}$ to allow to explicitly moving the focus in subterms can be found in Figure I.6.2. Just like in $\underline{\lambda}_N^{\rightarrow}$, commands c_n (which correspond to configurations of $\underline{\lambda}_N^{\rightarrow}$) are computations that can be reduced / evaluated by the operational reduction $\triangleright$, while an expression t_n is only part of a computation meant to be combined with a stack $\mathbb{S}_n$ (or an evaluation context $\mathbb{e}_n$ in the full calculus) to form a command. Of course, any expression t_n can be turned into a command by making it interact with the trivial stack $\square$, which yields $\underline{t_n}$. The distinction between commands and simple commands will become relevant later when we add let-expressions (and the distinction between stacks and simple stacks will become

Figure I.6.2: The simple fragment of the naive call-by-name λ-calculus with focus $\underline{\lambda}_n^{\rightarrow}$

Figure I.6.2.a: Syntax (naive)

Expressions / values:

$$t_n, u_n, v_n, w_n ::= x^n \mid c_n \mid \lambda x^n . c_n$$

Stacks / simple stacks:

$$\mathbb{S}_n, \mathring{\mathbb{S}}_n ::= \square \mid \mathbb{S}_n t_n$$

Commands:

$$c_n ::= \mathring{c}_n$$

Simple commands:

$$\mathring{c}_n ::= \underline{t_n} \mid \mathring{c}_n t_n$$

Figure I.6.2.b: Expanded description of commands and stacks (naive)

Simple commands:

$$\mathring{c}_n ::= \underline{t_n} u_n^1 \dots u_n^q$$

Stacks / simple stacks:

$$\mathbb{S}_n, \mathring{\mathbb{S}}_n ::= \square u_n^1 \dots u_n^q$$

Figure I.6.2.c: Operational reduction (naive)

$$\mathbb{S}_n \boxed{\underline{\mathring{c}_n}} \triangleright_m \mathbb{S}_n \boxed{\mathring{c}_n}$$

$$\mathbb{S}_n \boxed{\underline{(\lambda x^n . \mathring{c}_n)} t_n} \triangleright_{\rightarrow} \mathbb{S}_n \boxed{\mathring{c}_n [t_n / x^n]}$$

$$\triangleright \stackrel{\text{def}}{=} \triangleright_m \cup \triangleright_{\rightarrow}$$

Figure I.6.2.d: Expanded description of the operational reduction (naive)

$$\underline{\underline{t_n} v_n^1 \dots v_n^q} w_n^1 \dots w_n^r \triangleright_m \underline{t_n} v_n^1 \dots v_n^q w_n^1 \dots w_n^r$$

$$\underline{(\lambda x^n . \underline{t_n} v_n^1 \dots v_n^q)} w_n^0 w_n^1 \dots w_n^r \triangleright_{\rightarrow} \underline{t_n [w_n^0 / x^n]} (v_n^1 [w_n^0 / x^n]) \dots (v_n^q [w_n^0 / x^n]) w_n^1 \dots w_n^r$$

relevant in call-by-value).

The main difference with $\underline{\lambda}^{\rightarrow}_{\mathrm{N}}$ is that some subterms are now also represented with commands, which allows the strong reduction $\twoheadrightarrow\!\!\rhd$ to move focus in subterms: the expression $\underline{(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}})V_{\mathrm{N}}}$ of $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ can be represented in $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ by

$$\underline{\big(\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}\big)v_{\mathrm{n}}} \quad \text{with} \quad I_{\mathrm{n}} = \lambda y^{\mathrm{n}}.\underline{y^{\mathrm{n}}}$$

and reduces as shown in Figure I.6.1b.

Focus erasure in place of focus movement

Since subcommands now already have their own focused subterm, the reduction now erases the underlinement $\underline{\cdot}$ instead of moving it. For example, the reduction

$$\underline{(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}})V_{\mathrm{N}}} \rhd_{\mathrm{m}} (\underline{\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}}})V_{\mathrm{N}} \rhd_{\rightarrow} \underline{I_{\mathrm{N}}W_{\mathrm{N}}[V_{\mathrm{N}}/x^{\mathrm{N}}]}$$

becomes

$$\underline{\big(\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}\big)v_{\mathrm{n}}} \rhd_{\mathrm{m}} \big(\underline{\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}}\big)v_{\mathrm{n}} \rhd_{\rightarrow} \underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]}$$

where the first step erases the focus under $\big(\underline{\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}}\big)v_{\mathrm{n}}$ and the second step erases the focus under $\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}$ and reduces the β-redex. The result $\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]}$ is the body $\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}$ of the function $\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}$ with the substitution $x^{\mathrm{n}} \mapsto v_{\mathrm{n}}$ applied to it. Erasing the focus, instead of moving it, is a way of ensuring the focus in subcommands can be moved by $\twoheadrightarrow\!\!\rhd_{\mathrm{m}}$ independently of what happens above: if we decide to first apply the reduction

$$\underline{\big(\lambda x^{\mathrm{n}}.\underline{\underline{I_{\mathrm{n}}}w_{\mathrm{n}}}\big)v_{\mathrm{n}}} \twoheadrightarrow\!\!\rhd_{\mathrm{m}} \underline{\big(\lambda x^{\mathrm{n}}.\underline{I_{\mathrm{n}}w_{\mathrm{n}}}\big)v_{\mathrm{n}}}$$

then the body is now $\underline{I_{\mathrm{n}}w_{\mathrm{n}}}$ and after performing the same two $\rhd_{\mathrm{m}}\rhd_{\rightarrow}$ steps, we get $\underline{I_{\mathrm{n}}w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]}$ as expected.

I.6.2. The naive $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ calculus

Stack deferrals

We now add let-expressions to our naive $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$, which yields Figure I.6.3. The most important difference is that stacks $\mathbb{S}_{\mathrm{n}}$ can now be moved by the operational reduction $\rhd$ via the defer operation. Indeed, in the simple fragment, we only had simple commands for which defer only plugs the simple command in the stack:

$$\mathsf{defer}(\underline{t_{\mathrm{n}}}\vec{v}_n, \square\vec{w}_n) = (\square\vec{w}_n)\boxed{\underline{t_{\mathrm{n}}}\vec{v}_n} = \underline{t_{\mathrm{n}}}\vec{v}_n\vec{w}_n$$

Having let-expressions allows us to form non-simple commands for which defer moves the stack to the body of the let expression when this body is a simple command

$$\mathsf{defer}\begin{pmatrix}\mathsf{let}\ x^{\mathrm{n}} = \underline{t_{\mathrm{n}}}\ \mathsf{in}, \square\vec{w}_n \\ \underline{u_{\mathrm{n}}}\vec{v}_n\end{pmatrix} = \begin{array}{l}\mathsf{let}\ x^{\mathrm{n}} = \underline{t_{\mathrm{n}}}\ \mathsf{in} \\ \mathsf{defer}(\underline{u_{\mathrm{n}}}\vec{v}_n, \square\vec{w}_n)\end{array} = \begin{array}{l}\mathsf{let}\ x^{\mathrm{n}} = \underline{t_{\mathrm{n}}}\ \mathsf{in} \\ (\square\vec{w}_n)\boxed{\underline{u_{\mathrm{n}}}\vec{v}_n}\end{array} = \begin{array}{l}\mathsf{let}\ x^{\mathrm{n}} = \underline{t_{\mathrm{n}}}\ \mathsf{in} \\ \underline{u_{\mathrm{n}}}\vec{v}_n\vec{w}_n\end{array}$$

Figure I.6.3: The call-by-name λ-calculus with focus $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ (naive)

Figure I.6.3.a: Syntax (naive)

Expressions / values:

$$t_{\mathrm{n}}, u_{\mathrm{n}}, v_{\mathrm{n}}, w_{\mathrm{n}} ::= x^{\mathrm{n}} \mid c_{\mathrm{n}} \mid \lambda x^{\mathrm{n}}.c_{\mathrm{n}}$$

Stacks / simple stacks:

$$\mathbb{S}_{\mathrm{n}}, \mathring{\mathbb{S}}_{n} ::= \square \mid \mathbb{S}_{\mathrm{n}} t_{\mathrm{n}}$$

Commands:

$$c_{\mathrm{n}} ::= \mathring{c}_{\mathrm{n}} \mid \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}}$$

Simple commands:

$$\mathring{c}_{\mathrm{n}} ::= t_{\mathrm{n}} \mid \mathring{c}_{\mathrm{n}} t_{\mathrm{n}}$$

Figure I.6.3.b: Stack deferral (naive)

$$\mathsf{defer}(\mathring{c}_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \stackrel{\mathrm{def}}{=} \mathbb{S}_{\mathrm{n}}\boxed{\mathring{c}_{\mathrm{n}}}$$

$$\mathsf{defer}(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \stackrel{\mathrm{def}}{=} \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \quad \text{if } x^{\mathrm{n}} \text{ fresh w.r.t. } \mathbb{S}_{\mathrm{n}}$$

Figure I.6.3.c: Operational reduction (naive)

$$\mathbb{S}_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}} \triangleright_{\mu} \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}})$$

$$\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}} \triangleright_{\mathsf{let}} c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}]$$

$$\mathbb{S}_{\mathrm{n}}\boxed{\underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})t_{\mathrm{n}}}} \triangleright_{\rightarrow} \mathsf{defer}(c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}], \mathbb{S}_{\mathrm{n}})$$

$$\triangleright \stackrel{\mathrm{def}}{=} \triangleright_{\mathrm{m}} \cup \triangleright_{\mathsf{let}} \cup \triangleright_{\rightarrow}$$

and further down if that body is again a let-expression

$$\mathsf{defer}\begin{pmatrix} \mathsf{let}\ x_1^{\mathrm{n}} = \underline{t_{\mathrm{n}}^1}\ \mathsf{in}, \square\overrightarrow{w_n} \\ \vdots \\ \mathsf{let}\ x_q^{\mathrm{n}} = \underline{t_{\mathrm{n}}^q}\ \mathsf{in} \\ \underline{u_{\mathrm{n}}}\overrightarrow{v_n} \end{pmatrix} = \ldots = \begin{array}{l} \mathsf{let}\ x_1^{\mathrm{n}} = \underline{t_{\mathrm{n}}^1}\ \mathsf{in} \\ \vdots \\ \mathsf{let}\ x_q^{\mathrm{n}} = \underline{t_{\mathrm{n}}^q}\ \mathsf{in} \\ \mathsf{defer}(\underline{u_{\mathrm{n}}}\overrightarrow{v_n}, \square\overrightarrow{w_n}) \end{array} = \begin{array}{l} \mathsf{let}\ x_1^{\mathrm{n}} = \underline{t_{\mathrm{n}}^1}\ \mathsf{in} \\ \vdots \\ \mathsf{let}\ x_q^{\mathrm{n}} = \underline{t_{\mathrm{n}}^q}\ \mathsf{in} \\ (\square\overrightarrow{w_n})\boxed{\underline{u_{\mathrm{n}}}\overrightarrow{v_n}} \end{array} = \begin{array}{l} \mathsf{let}\ x_1^{\mathrm{n}} = \underline{t_{\mathrm{n}}^1}\ \mathsf{in} \\ \vdots \\ \mathsf{let}\ x_q^{\mathrm{n}} = \underline{t_{\mathrm{n}}^q}\ \mathsf{in} \\ \underline{u_{\mathrm{n}}}\overrightarrow{v_n}\overrightarrow{w_n} \end{array}$$

$\triangleright_{\mu}$ as a generalization of $\triangleright_{\mathrm{m}}$ and $\ltimes$

The $\triangleright_{\mu}$ reduction therefore does two things: it erases the underline and moves the stack:

$$\underline{\begin{pmatrix} \mathsf{let}\ x_1^{\mathrm{n}} = \underline{t_{\mathrm{n}}^1}\ \mathsf{in} \\ \vdots \\ \mathsf{let}\ x_q^{\mathrm{n}} = \underline{t_{\mathrm{n}}^q}\ \mathsf{in} \\ \underline{u_{\mathrm{n}}}\overrightarrow{v_n} \end{pmatrix}}\overrightarrow{w_n} \triangleright_{\mu} \begin{array}{l} \mathsf{let}\ x_1^{\mathrm{n}} = \underline{t_{\mathrm{n}}^1}\ \mathsf{in} \\ \vdots \\ \mathsf{let}\ x_q^{\mathrm{n}} = \underline{t_{\mathrm{n}}^q}\ \mathsf{in} \\ \underline{u_{\mathrm{n}}}\overrightarrow{v_n}\overrightarrow{w_n} \end{array}$$

The reduction $\rhd_{\mathrm{m}}$ of the simple fragment of $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ corresponds to the case where there are $q = 0$ nested let-expressions, and the $\ltimes$ reduction of $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ corresponds to

$$\underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}})}w_{\mathrm{n}} \rhd_{\mu} \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}w_{\mathrm{n}}$$

i.e. to the case where there is $q = 1$ let-expression whose body is underlined (i.e. $\Box\overrightarrow{v_n} = \Box$) and the moved stack contains a single value (i.e. $\Box\overrightarrow{w_n} = \Box w_{\mathrm{n}}^1$). The reduction $\rhd_{\mu}$ can therefore be though of as a combination of the reduction $\rhd_{\mathrm{m}}$ that simply moves focus and of a strenghened variant of $\ltimes$ that can move several arguments at once, and move them through several let-expressions at once. This strengthening ensures that $\twoheadrightarrow\!\!\rhd_{\mu}$ steps commute with each other. Indeed, being able to move several values allows for

$$\begin{array}{ccc}
\underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}})v_{\mathrm{n}}}w_{\mathrm{n}} & \rhd_{\mu} & \underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}})}v_{\mathrm{n}}w_{\mathrm{n}} \\
\downarrow\!\!\rhd_{\mu} & & \rhd_{\mu} \\
\underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}v_{\mathrm{n}}})}w_{\mathrm{n}} & \rhd_{\mu} & \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}v_{\mathrm{n}}w_{\mathrm{n}}
\end{array}$$

and being able to move through let-expressions allows for

$$\begin{array}{ccc}
\underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{\mathsf{let}\ y^{\mathrm{n}} := \underline{u_{\mathrm{n}}}\ \mathsf{in}\ \underline{v_{\mathrm{n}}}})}w_{\mathrm{n}} & \rhd_{\mu} & \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{(\mathsf{let}\ y^{\mathrm{n}} := \underline{u_{\mathrm{n}}}\ \mathsf{in}\ \underline{v_{\mathrm{n}}})}w_{\mathrm{n}} \\
\downarrow\!\!\rhd_{\mu} & & \downarrow\!\!\rhd_{\mu} \\
\underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \mathsf{let}\ y^{\mathrm{n}} := \underline{u_{\mathrm{n}}}\ \mathsf{in}\ \underline{v_{\mathrm{n}}})}w_{\mathrm{n}} & \rhd_{\mu} & \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \mathsf{let}\ y^{\mathrm{n}} := \underline{u_{\mathrm{n}}}\ \mathsf{in}\ \underline{v_{\mathrm{n}}}w_{\mathrm{n}}
\end{array}$$

See ⚠ for a more formal statement.

Underlines as potential places of interaction

One way to think about the

$$\mathbb{S}_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}} \rhd_{\mu} \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}})$$

reduction is that it moves the stack $\mathbb{S}_{\mathrm{n}} = \Box\overrightarrow{w_n}$ at the next point in c_{n} where it might interact with an expression. In non-simple commands $c_{\mathrm{n}} = \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}}^0$, that next point of interaction of the stack with an expression is necessarily in the subcommand c_{n}^0, while for simple commands $c_{\mathrm{n}} = \underline{t_{\mathrm{n}}}\overrightarrow{v_n}$, the stack may interact with an expression that comes from reducing $\underline{t_{\mathrm{n}}}\overrightarrow{v_n}$ so we leave it here. Leaving it here does not mean that it will interact here, only that it will follow $\Box\overrightarrow{v_n}$ around until all values of $\Box\overrightarrow{v_n}$ have been consumed. For example, if t_{n} is a non-simple command, it will be moved together with $\Box\overrightarrow{v_n}$:

$$\underline{(\mathsf{let}\ x^{\mathrm{n}} := \underline{u_{\mathrm{n}}^1}\ \mathsf{in}\ \underline{u_{\mathrm{n}}^2})}\overrightarrow{v_n}\overrightarrow{w_n} \rhd_{\mu} (\mathsf{let}\ x^{\mathrm{n}} := \underline{u_{\mathrm{n}}^1}\ \mathsf{in}\ \underline{u_{\mathrm{n}}^2})\overrightarrow{v_n}\overrightarrow{w_n} \rhd_{\mu} \mathsf{let}\ x^{\mathrm{n}} := \underline{u_{\mathrm{n}}^1}\ \mathsf{in}\ \underline{u_{\mathrm{n}}^2}\overrightarrow{v_n}\overrightarrow{w_n}$$

Note that this interpretation of underlines marking points of interaction also works for let-expressions: a non-simple command

$$\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}}$$

is an interaction between a term t_{n} and an evaluation context

$$\mathbb{e}_{\mathrm{n}} = \mathsf{let}\ x^{\mathrm{n}} := \Box\ \mathsf{in}\ c_{\mathrm{n}}$$

Reducing let-expressions

Note that

$$\mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}} \quad \text{and} \quad \underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})}v_{\mathrm{n}}$$

do not reduce in the same way. Indeed, while $\underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})}v_{\mathrm{n}}$ can be reduced under an arbitrary stack $\mathbb{S}_{\mathrm{n}}$, $\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}$ can not:

$$\begin{aligned} \mathbb{S}_{\mathrm{n}}\boxed{\underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})}v_{\mathrm{n}}} &\rhd_{\rightarrow} \mathsf{defer}(c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}], \mathbb{S}_{\mathrm{n}}) && \text{for any } \mathbb{S}_{\mathrm{n}}, \text{ while} \\ \mathbb{S}_{\mathrm{n}}\boxed{\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}} &\rhd_{\mathsf{let}} \mathsf{defer}(c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}], \mathbb{S}_{\mathrm{n}}) && \text{only for } \mathbb{S}_{\mathrm{n}} = \square \end{aligned}$$

(and for $\mathbb{S}_{\mathrm{n}} \neq \square$, $\mathbb{S}_{\mathrm{n}}\boxed{\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}}$ is not even in the syntax). This somewhat surprising weakness of $\rhd_{\mathsf{let}}$ compensates for the strength of defer (and hence of the $\rhd_{\mu}$ reduction) on let-expressions:

$$\begin{aligned} \mathsf{defer}(\underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})}v_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) &= \underline{(\lambda x^{\mathrm{n}}.\mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}))}v_{\mathrm{n}} && \text{only for } \mathbb{S}_{\mathrm{n}} = \square, \text{ while} \\ \mathsf{defer}(\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) &= \mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) && \text{for any } \mathbb{S}_{\mathrm{n}} \end{aligned}$$

Indeed, the expected reduction of a let-expression under a stack can be simulated via

$$\mathbb{S}_{\mathrm{n}}\boxed{\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}} \rhd_{\mu} \mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \rhd_{\mathsf{let}} \mathsf{defer}(c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}], \mathbb{S}_{\mathrm{n}})$$

The difference between $\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}$ and $\underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})}v_{\mathrm{n}}$ is therefore that defer understands that $\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}$ will reduce without interacting with the surrounding stack, but has no such knowledge for $\underline{(\lambda x^{\mathrm{n}}.c_{\mathrm{n}})}v_{\mathrm{n}}$[5]. This implies that $\mathbb{S}_{\mathrm{n}}\boxed{\underline{\mathsf{let}\, x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\, \mathsf{in}\, c_{\mathrm{n}}}}$ is $\rhd_{\mu}$ reducible, and since we want $\rhd$ to be deterministic (and $\rhd_{l_1}$ and $\rhd_{l_2}$ to have disjoint domains for $l_1 \neq l_2$), it can not be $\rhd_{\mathsf{let}}$-reducible, which is why $\rhd_{\mathsf{let}}$ is weaker than expected.

Undesirable strong reductions

In this naive version of $\lambda_{\mathrm{n}}^{\rightarrow}$, the strong reduction $\twoheadrightarrow\!\!\rhd$[6] is somewhat unsatisfying because we do not have

$$\left.\begin{array}{l} c_{\mathrm{n}} \rhd c'_{\mathrm{n}}, \text{ and} \\ \mathbb{k}_{\mathrm{n}}\boxed{c_{\mathrm{n}}} \text{ is in the syntax} \end{array}\right\} \Rightarrow \mathbb{k}_{\mathrm{n}}\boxed{c_{\mathrm{n}}} \twoheadrightarrow\!\!\rhd \mathbb{k}_{\mathrm{n}}\boxed{c'_{\mathrm{n}}}$$

but only

$$\left.\begin{array}{l} c_{\mathrm{n}} \rhd c'_{\mathrm{n}}, \\ \mathbb{k}_{\mathrm{n}}\boxed{c_{\mathrm{n}}} \text{ is in the syntax, and} \\ \mathbb{k}_{\mathrm{n}}\boxed{c'_{\mathrm{n}}} \text{ is in the syntax} \end{array}\right\} \Rightarrow \mathbb{k}_{\mathrm{n}}\boxed{c_{\mathrm{n}}} \twoheadrightarrow\!\!\rhd \mathbb{k}_{\mathrm{n}}\boxed{c'_{\mathrm{n}}}$$

---

[5]Note that $\rhd_{\mu}$ sometimes failing to recognize that a term will not interact with the surrounding stack is perfectly reasonable: "interacting with the surrounding stack" is an undecidable property (e.g. because a closed term interacts with the surrounding stack if and only if it terminates), so that $\rhd_{\mu}$ is necessarily an approximation.

[6]The strong reduction $\twoheadrightarrow\!\!\rhd$ can be defined either by

$$t \twoheadrightarrow\!\!\rhd t' \overset{\text{def}}{\equiv} \exists \mathbb{k}_{\mathrm{n}}, \exists c_{\mathrm{n}}, \exists c'_{\mathrm{n}}, \left\{\begin{array}{l} c_{\mathrm{n}} \rhd c'_{\mathrm{n}}, \\ t = \mathbb{k}_{\mathrm{n}}\boxed{c_{\mathrm{n}}}, \text{ and} \\ t' = \mathbb{k}_{\mathrm{n}}\boxed{c'_{\mathrm{n}}} \end{array}\right.$$

or equivalently by the expected inferrence rules.

For example, the reduction

$$c_{\mathrm{n}} = \big(\underline{\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}}\big)v_{\mathrm{n}} \twoheadrightarrow_{\mu} \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}v_{\mathrm{n}} = c'_{\mathrm{n}}$$

is not preserved under $\mathbb{k}_{\mathrm{n}} = \square w_{\mathrm{n}}$ because $(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}v_{\mathrm{n}})w_{\mathrm{n}}$ is not in the syntax:

$$\begin{array}{ccc}
\mathbb{k}_{\mathrm{n}}\boxed{c_{\mathrm{n}}} & & \mathbb{k}_{\mathrm{n}}\boxed{c'_{\mathrm{n}}} \\
\| & & \| \\
(\square w_{\mathrm{n}})\boxed{\big(\underline{\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}}\big)v_{\mathrm{n}}} & & (\square w_{\mathrm{n}})\boxed{\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}v_{\mathrm{n}}} \\
\| & & \| \\
\big(\underline{\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}}\big)v_{\mathrm{n}}w_{\mathrm{n}} & \not\twoheadrightarrow_{\mu} & (\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \underline{u_{\mathrm{n}}}v_{\mathrm{n}})w_{\mathrm{n}}
\end{array}$$

This is due to a reduction

$$\mathbb{s}^2_{\mathrm{n}}\boxed{\mathbb{s}^1_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}}} \twoheadrightarrow_{\mu} \mathbb{s}^2_{\mathrm{n}}\boxed{\mathsf{defer}(c_{\mathrm{n}}, \mathbb{s}^1_{\mathrm{n}})}$$

only being valid when there is no potential interaction between $\mathsf{defer}(c_{\mathrm{n}}, \mathbb{s}^1_{\mathrm{n}})$ and $\mathbb{s}^2_{\mathrm{n}}$, because if there is, the the syntax requires making it with $\underline{\,\cdot\,}$, i.e. writing $\mathbb{s}^2_{\mathrm{n}}\boxed{\underline{\mathsf{defer}(c_{\mathrm{n}}, \mathbb{s}^1_{\mathrm{n}})}}$. In other words, since $\twoheadrightarrow_{\mu}$ erases $\underline{\,\cdot\,}$, it must ensure that there is no potential interaction at that $\underline{\,\cdot\,}$ by deferring the whole stack $\mathbb{s}^2_{\mathrm{n}}\boxed{\mathbb{s}^1_{\mathrm{n}}}$.

This can also be understood as stemming from the fact that simple commands are not stable under $\triangleright_{\mu}$: $\mathbb{s}^1_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}}$ is a simple command and can hence be plugged in $\mathbb{s}^2_{\mathrm{n}}$ while remaining within the syntax, while $\mathsf{defer}(c_{\mathrm{n}}, \mathbb{s}^1_{\mathrm{n}})$ is simple only if c_{n} is, and $\mathbb{s}^2_{\mathrm{n}}\boxed{\mathsf{defer}(c_{\mathrm{n}}, \mathbb{s}^1_{\mathrm{n}})}$ is therefore in the syntax only if $\mathbb{s}^2_{\mathrm{n}} = \square$ or c_{n} is simple.

I.6.3. The $\lambda^{\rightarrow}_{\mathrm{n}}$ calculus

Explicit command boundaries

The pure untyped call-by-name λ-calculus with focus $\lambda^{\rightarrow}_{\mathrm{n}}$ is defined in Figure I.6.4. It deals with the aforementioned quirks of the strong reduction by explicitly marking the top of commands with a constructor $\mathsf{com}^{\mathrm{n}}$, i.e. replacing

$$c_{\mathrm{n}} ::= \mathring{c}_{\mathrm{n}} \quad \text{by} \quad c_{\mathrm{n}} ::= \mathsf{com}^{\mathrm{n}}(\mathring{c}_{\mathrm{n}}),$$

and restricting $\triangleright$ by only allowing it to reduce objects that have $\mathsf{com}^{\mathrm{n}}$ above them. This prevents the problematic $\twoheadrightarrow_{\mu}$ reductions because there is no $\mathsf{com}^{\mathrm{n}}$ around $\mathbb{s}^1_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}}$ in $\mathsf{com}^{\mathrm{n}}\big(\mathbb{s}^2_{\mathrm{n}}\boxed{\mathbb{s}^1_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}}}\big)$, and it can therefore not be reduced on its own. Adding the $\mathsf{com}^{\mathrm{n}}$ yields an invalid term $\mathsf{com}^{\mathrm{n}}\big(\mathbb{s}^2_{\mathrm{n}}\boxed{\mathsf{com}^{\mathrm{n}}(\mathbb{s}^1_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}})}\big)$, unless we also add a $\underline{\,\cdot\,}$, which yields $\mathsf{com}^{\mathrm{n}}\big(\mathbb{s}^2_{\mathrm{n}}\boxed{\underline{\mathsf{com}^{\mathrm{n}}(\mathbb{s}^1_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}})}}\big)$ for which the problematic is disallowed by the extra $\underline{\,\cdot\,}$ (and $\mathsf{com}^{\mathrm{n}}$).

The need for $\mathsf{com}^{\mathrm{n}}$ was to be expected. Indeed, the calculus $\lambda^{\rightarrow}_{\mathrm{n}}$ was built as an outside-out representation of $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ (defined in the next section), and commands $\langle t_{\mathrm{n}} | e_{\mathrm{n}}\rangle$ of $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ have both an explicit marker $|$ for the point of potential interaction between t_{n} and e_{n}; and an explicit marker $\langle\cdot\rangle$ for the top of the command. It therefore makes sense for commands $\mathsf{com}^{\mathrm{n}}(\mathbb{e}_{\mathrm{n}}\boxed{\underline{t_{\mathrm{n}}}})$ of $\lambda^{\rightarrow}_{\mathrm{n}}$ to have both an explicit marker $\underline{\,\cdot\,}$ for the point of interaction between $\mathbb{e}_{\mathrm{n}}$ and t_{n}; and an explicit marker $\mathsf{com}^{\mathrm{n}}$ for the top of the command.

Figure I.6.4: The pure call-by-name λ-calculus with focus $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$

Figure I.6.4.a: Syntax

Expressions / values:

$$t_{\mathrm{n}}, u_{\mathrm{n}}, v_{\mathrm{n}}, w_{\mathrm{n}} ::= x^{\mathrm{n}} \mid \mathsf{ctot}^{\mathrm{n}}(c_{\mathrm{n}}) \mid \lambda x^{\mathrm{n}}.c_{\mathrm{n}}$$

Stacks / simple stacks:

$$\mathbb{S}_{\mathrm{n}}, \mathring{\mathbb{S}}_{n} ::= \square \mid \mathbb{S}_{\mathrm{n}} t_{\mathrm{n}}$$

Commands:

$$c_{\mathrm{n}} ::= \mathsf{com}^{\mathrm{n}}(\mathring{c}_{\mathrm{n}}) \mid \mathsf{com}^{\mathrm{n}}(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}})$$

Incomplete simple commands:

$$\mathring{c}_{\mathrm{n}} ::= \mathsf{instk}^{\mathrm{n}}(\underline{t_{\mathrm{n}}}) \mid \mathring{c}_{\mathrm{n}} t_{\mathrm{n}}$$

Figure I.6.4.b: Notations

Evaluation contexts:

$$\mathscr{e}_{\mathrm{n}} ::= \mathbb{S}_{\mathrm{n}}\boxed{\mathsf{instk}^{\mathrm{n}}(\square)} \mid \mathsf{let}\ x^{\mathrm{n}} := \square\ \mathsf{in}\ c_{\mathrm{n}}$$

Simple commands:

$$c_{\mathrm{n}}^{\mathrm{simple}} ::= \mathsf{com}^{\mathrm{n}}(\mathring{c}_{\mathrm{n}})$$

Terms:

$$t ::= t_{\mathrm{n}} \mid \mathring{c}_{\mathrm{n}} \mid c_{\mathrm{n}}$$

Figure I.6.4.c: Deferred stacks

$$\mathsf{defer}(\mathring{c}_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \stackrel{\mathrm{def}}{=} \mathbb{S}_{\mathrm{n}}\boxed{\mathring{c}_{\mathrm{n}}}$$

$$\mathsf{defer}(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \stackrel{\mathrm{def}}{=} \mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}})$$

Figure I.6.4.d: Deferred stacks (in evaluation contexts)

$$\mathsf{defer}(\mathring{\mathbb{S}}_{n}, \mathbb{S}_{\mathrm{n}}) \stackrel{\mathrm{def}}{=} \mathbb{S}_{\mathrm{n}}\boxed{\mathring{\mathbb{S}}_{n}}$$

$$\mathsf{defer}(\mathsf{let}\ x^{\mathrm{n}} := \square\ \mathsf{in}\ c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}) \stackrel{\mathrm{def}}{=} \mathsf{let}\ x^{\mathrm{n}} := \square\ \mathsf{in}\ \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}})$$

Figure I.6.4.e: Operational reduction

$$\mathsf{com}^{\mathrm{n}}\left(\mathbb{S}_{\mathrm{n}}\boxed{\underline{c_{\mathrm{n}}}}\right) \rhd_{\mu} \mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}})$$

$$\mathsf{com}^{\mathrm{n}}(\mathsf{let}\ x^{\mathrm{n}} := \underline{t_{\mathrm{n}}}\ \mathsf{in}\ c_{\mathrm{n}}) \rhd_{\mathsf{let}} c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}]$$

$$\mathsf{com}^{\mathrm{n}}\left(\mathbb{S}_{\mathrm{n}}\boxed{(\underline{\lambda x^{\mathrm{n}}.c_{\mathrm{n}}})t_{\mathrm{n}}}\right) \rhd_{\rightarrow} \mathsf{defer}(c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}], \mathbb{S}_{\mathrm{n}})$$

$$\rhd \stackrel{\mathrm{def}}{=} \rhd_{\mu} \cup \rhd_{\mathsf{let}} \cup \rhd_{\rightarrow}$$

Figure I.6.4.f: Top-level η-expansion

$$t_{\mathrm{n}} \fatsemi_{\mu} \mathsf{ctot}^{\mathrm{n}}(\underline{t_{\mathrm{n}}})$$

$$\mathscr{e}_{\mathrm{n}} \fatsemi_{\mathsf{let}} \mathsf{let}\ x^{\mathrm{n}} := \square\ \mathsf{in}\ \mathscr{e}_{\mathrm{n}}\boxed{\underline{x^{\mathrm{n}}}} \quad \text{if } x^{\mathrm{n}} \text{ fresh w.r.t. } \mathscr{e}_{\mathrm{n}}$$

$$t_{\mathrm{n}} \fatsemi_{\rightarrow} \lambda x^{\mathrm{n}}.\underline{t_{\mathrm{n}}}x^{\mathrm{n}} \quad \text{if } x^{\mathrm{n}} \text{ fresh w.r.t. } t_{\mathrm{n}}$$

$$\fatsemi \stackrel{\mathrm{def}}{=} \fatsemi_{\mu} \cup \fatsemi_{\mathsf{let}} \cup \fatsemi_{\rightarrow}$$

Coercions

In addition to explicit command boundaries $\mathsf{com}^{\mathrm{n}}$, $\lambda_{\mathrm{n}}^{\to}$ has an explicit coercion $\mathsf{ctot}^{\mathrm{n}}$ from command c_{n} to expessions t_{n} (hence the name $c$to$t$) and an explicit marker $\mathsf{instk}^{\mathrm{n}}$ around underlines $\underline{\cdot}$ that will be placed within a stack $\mathbb{S}_{\mathrm{n}}$. Both $\mathsf{com}^{\mathrm{n}}$ and $\mathsf{instk}^{\mathrm{n}}$ are often left implicit because the former is only relevant when definining $\twoheadrightarrow$, and the latter is only relevant when studying translations between $\lambda_{\mathrm{n}}^{\to}$ and $\mathrm{Li}_{\mathrm{n}}^{\to}$. In particular, we will often denote simple commands by $\mathring{c}_{\mathrm{n}}$ instead of $\mathsf{com}^{\mathrm{n}}(\mathring{c}_{\mathrm{n}})$. While $\mathsf{ctot}^{\mathrm{n}}$ could also often be left implicit, not distinguishing the command c_{n} from the expression $\mathsf{ctot}^{\mathrm{n}}(c_{\mathrm{n}})$ may lead to some confusion, and we therefore keep $\mathsf{ctot}^{\mathrm{n}}$ explicit for the sake of clarity.

Evaluation contexts

Evaluation contexts $\mathbb{e}_{\mathrm{n}}$ form a superset of stacks that are not required to define $\lambda_{\mathrm{n}}^{\to}$. Commands are exactly terms of the shape $\mathsf{com}^{\mathrm{n}}(\mathbb{e}_{\mathrm{n}}\boxed{t_{\mathrm{n}}})$ (see ⚠), and evaluation contexts are therefore useful whenever this inside-out description of commands is, e.g. when studying at translations between $\lambda_{\mathrm{n}}^{\to}$ and $\mathrm{Li}_{\mathrm{n}}^{\to}$.

The difference between a stack and an evaluation context is that an a stack $\mathbb{S}_{\mathrm{n}}$ can be deferred so that something else is computed first, while a non-stack evaluation context $\mathbb{e}_{\mathrm{n}}$ can not. For example, placing $t_{\mathrm{n}} = \mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\,\mathsf{in}\,\underline{x^{\mathrm{n}}}$ under $\mathbb{S}_{\mathrm{n}} = \square w_{\mathrm{n}}$ results in $\mathbb{S}_{\mathrm{n}}$ being moved

$$\mathbb{S}_{\mathrm{n}}\boxed{t_{\mathrm{n}}} = \underline{\left(\mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\,\mathsf{in}\,\underline{x^{\mathrm{n}}}\right)}w_{\mathrm{n}} \rhd_{\mu} \mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\,\mathsf{in}\,\underline{x^{\mathrm{n}}}w_{\mathrm{n}} = \mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\,\mathsf{in}\,\mathbb{S}_{\mathrm{n}}\boxed{x^{\mathrm{n}}}$$

while placing it inside $\mathbb{e}_{\mathrm{n}} = \mathsf{let}\, y^{\mathrm{n}} := \square\,\mathsf{in}\, c_{\mathrm{n}}$ results in this let-expressiong being evaluated

$$\mathbb{e}_{\mathrm{n}}\boxed{t_{\mathrm{n}}} = \mathsf{let}\, y^{\mathrm{n}} := \underline{\left(\mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\,\mathsf{in}\,\underline{x^{\mathrm{n}}}\right)}\,\mathsf{in}\, c_{\mathrm{n}} \rhd_{\mathrm{let}} c_{\mathrm{n}}\left[\underline{\mathsf{let}\, x^{\mathrm{n}} := \underline{v_{\mathrm{n}}}\,\mathsf{in}\,\underline{x^{\mathrm{n}}}}/y^{\mathrm{n}}\right] = c_{\mathrm{n}}[t_{\mathrm{n}}/y^{\mathrm{n}}]$$

Disubstitution

Disubstitutions of $\lambda_{\mathrm{n}}^{\to}$ are defined just like in $\lambda_{\mathrm{n}}^{\to}$, with plugging replaced by defer:

Definition I.6.1

A *disubstitution* φ is a pair $\varphi = (\sigma, \mathbb{S}_{\mathrm{n}})$ composed of a substitution σ and a stack $\mathbb{S}_{\mathrm{n}}$. The action of a disubstitution $\varphi = (\sigma, \mathbb{S}_{\mathrm{n}})$ on commands (resp. evaluations contexts) is defined by

$$c_{\mathrm{n}}[\varphi] \overset{\mathrm{def}}{=} \mathsf{defer}(c_{\mathrm{n}}[\sigma], \mathbb{S}_{\mathrm{n}}) \quad (\text{resp. } \mathbb{e}_{\mathrm{n}} \overset{\mathrm{def}}{=} \mathsf{defer}(\mathbb{e}_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}))$$

and its action on expressions[a] is defined by

$$t_{\mathrm{n}}[\varphi] \overset{\mathrm{def}}{=} t_{\mathrm{n}}[\sigma]$$

The composition $\varphi_1[\varphi_2]$ of two disubstitutions is defined by

$$(\sigma_1, \mathbb{S}_{\mathrm{n}}^1)[\sigma_2, \mathbb{S}_{\mathrm{n}}^2] \overset{\mathrm{def}}{=} \left(\sigma_1[\sigma_2], \mathbb{S}_{\mathrm{n}}^2\boxed{\mathbb{S}_{\mathrm{n}}^1[\sigma_2]}\right)$$

[a] See ⚠.

Fact I.6.2

The set of disubstitutions φ_{n} has a monoid structure

$$(\varphi_{\mathrm{n}}, \bigcirc, (\mathrm{Id}_{\mathcal{V}}, \square)) \quad \text{where} \quad \varphi_2 \bigcirc \varphi_1 \stackrel{\text{def}}{=} \varphi_1[\varphi_2]$$

and this monoid acts on commands, expressions, and evaluation contexts via

$$\varphi \bullet t \stackrel{\text{def}}{=} t[\varphi]$$

In particular, defer is associative: for any command c_{n} and stacks $\mathbb{S}_{\mathrm{n}}^1$ and $\mathbb{S}_{\mathrm{n}}^2$, we have

$$\mathsf{defer}(\mathsf{defer}(c_{\mathrm{n}}, \mathbb{S}_{\mathrm{n}}^1), \mathbb{S}_{\mathrm{n}}^2) = \mathsf{defer}(c_{\mathrm{n}}, \mathsf{defer}(\mathbb{S}_{\mathrm{n}}^1, \mathbb{S}_{\mathrm{n}}^2))$$

Proof

⚠

Reductions

We write $\mathrel{-\!\!\triangleright}$ for the contextual closure of $\triangleright$, $\mathrel{-\!\!\rbrace}$ for the contextual closure of $\rbrace$, $\mathrel{\lbrace\!\!\!-\!\!\triangleright}$ for $\mathrel{-\!\!\triangleright} \cup \mathrel{\lbrace\!\!-}$ and $\mathrel{\triangleleft\!\!\lbrace\!\!\rbrace\!\!\triangleright}$ for $\mathrel{\lbrace\!\!\!-\!\!\triangleright} \cup \mathrel{\triangleleft\!\!-\!\!\!\rbrace}$ (i.e. $\mathrel{-\!\!\triangleright} \cup \mathrel{\triangleleft\!\!-} \cup \mathrel{-\!\!\rbrace} \cup \mathrel{\lbrace\!\!-}$). The η-expansion for functions is the usual one with a conversion $\underline{\text{-}}$ from expressions to commands added, and the other η-expansions are easier to understand in the $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ where they look natural, and can safely be ignored for now. The reductions have the properties announced in Figure **??** (see Section .2 for details).

I.7. Translations between $\boldsymbol{\lambda}^{\rightarrow}_{\mathrm{N}}$ and $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$

I.8. A pure call-by-name intuitionistic L calculus: $\mathrm{Li}_{\mathrm{n}}^{\to}$

In this section, we recall the intuitionistic call-by-name fragments of $\bar{\lambda}\mu\tilde{\mu}$ [CurHer00], which we call $\mathrm{Li}_{\mathrm{n}}^{\to}$.

I.8.1. From the $\mathrm{M}_{\mathrm{N}}^{\to}$ abstract machine to the $\mathrm{Li}_{\mathrm{n}}^{\to}$ calculus

Decomposing the strong reduction

Just like in $\underline{\lambda}_{\mathrm{N}}^{\to}$, the strong reduction $\twoheadrightarrow$ is unsatisfying in $\mathrm{M}_{\mathrm{N}}^{\to}$ because it can not be decomposed like $\triangleright$ can, as shown in Figure I.8.1a. The naive $\mathrm{Li}_{\mathrm{n}}^{\to}$ calculus defined in Figure I.8.2 fixes this, as shown in Figure I.8.1b (where $I_{\mathrm{n}} = \lambda y^{\mathrm{n}}.\langle y^{\mathrm{n}}|\star^{\mathrm{n}}\rangle$), in the same way that $\underline{\lambda}_{\mathrm{n}}^{\to}$ did: by representing the body of λ-abstraction by configurations / commands.

Pattern matching stacks

In the actual simple fragment of the $\mathrm{Li}_{\mathrm{n}}^{\to}$ calculus described in Figure I.8.3, the stack $\star^{\mathrm{n}}$ is thought of as being a stack variable, and λ-abstractions $\lambda x^{\mathrm{n}}.c_{\mathrm{n}}$ are denoted by $\mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}}$ to emphasize that $\star^{\mathrm{n}}$ is bound in $\mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}}$, and hence that the disubstitution $\star^{\mathrm{n}} \mapsto s_{\mathrm{n}}$ acts trivially on $\mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}}$:

$$(\mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}})[s_{\mathrm{n}}/\star^{\mathrm{n}}] = \mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}}$$

This allows for a more succinct description of the $\triangleright_{\to}$ reduction

$$\langle \mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}} | v_{\mathrm{n}} \cdot s_{\mathrm{n}} \rangle \triangleright_{\to} c_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}, s_{\mathrm{n}}/\star^{\mathrm{n}}]$$

which can be though of as pattern-maching the stack $v_{\mathrm{n}} \cdot s_{\mathrm{n}}$ against the stack pattern $x^{\mathrm{n}} \cdot \star^{\mathrm{n}}$ and applying the unifier $x^{\mathrm{n}} \mapsto v_{\mathrm{n}}, \star^{\mathrm{n}} \mapsto s_{\mathrm{n}}$ to the command c_{n}.

Stack variable names

Note that just like the same name x can be used for several unrelated value variables x^{n} in the same expression, the name $\star$ can be used for several unrelated stack variables. For example,

$$\underline{I_{\mathrm{n}}}I_{\mathrm{n}} = (\lambda x^{\mathrm{n}}.\underline{x^{\mathrm{n}}})(\lambda x^{\mathrm{n}}.\underline{x^{\mathrm{n}}})$$

stands for

$$(\lambda x_1^{\mathrm{n}}.\underline{x_1^{\mathrm{n}}})(\lambda x_2^{\mathrm{n}}.\underline{x_2^{\mathrm{n}}})$$

and similarly

$$\langle I_{\mathrm{n}} | I_{\mathrm{n}} \cdot \star^{\mathrm{n}} \rangle = \langle \mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).\langle x^{\mathrm{n}} | \star^{\mathrm{n}} \rangle | (\mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).\langle x^{\mathrm{n}} | \star^{\mathrm{n}} \rangle) \cdot \star^{\mathrm{n}} \rangle$$

stands for

$$\langle \mu(x_1^{\mathrm{n}} \cdot \star_1{}^{\mathrm{n}}).\langle x_1^{\mathrm{n}} | \star_1{}^{\mathrm{n}} \rangle | (\mu(x_2^{\mathrm{n}} \cdot \star_2{}^{\mathrm{n}}).\langle x_2^{\mathrm{n}} | \star_2{}^{\mathrm{n}} \rangle) \cdot \star_0{}^{\mathrm{n}} \rangle$$

The difference is that we have infinitely many value variables available, and can therefore always rename the bound ones to avoid such name clashes, but only one stack variable in $\mathrm{Li}_{\mathrm{n}}^{\to}$ and can therefore not avoid such clashes. The stack variable $\star^{\mathrm{n}}$ can alternatively be thought of as being the 0 de Bruijn index for stack variables. The difference between the

[CurHer00] "The duality of computation", Curien and Herbelin, 2000

Figure I.8.1: Example of strong reduction in subterms of abstract machines

Figure I.8.1.a: Example in $\mathrm{M}_{\mathrm{N}}^{\vec{}}$

$$\begin{array}{ccccc}
\langle(\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}})V_{\mathrm{N}}|\star^{\mathrm{N}}\rangle & \triangleright_{\mathrm{m}} & \langle\lambda x^{\mathrm{N}}.I_{\mathrm{N}}W_{\mathrm{N}}|V_{\mathrm{N}}\cdot\star^{\mathrm{N}}\rangle & \triangleright_{\rightarrow} & \langle I_{\mathrm{N}}W_{\mathrm{N}}[V_{\mathrm{N}}/x^{\mathrm{N}}]|\star^{\mathrm{N}}\rangle \\
 & & & & \triangledown_{\exists} \\
\Downarrow & & \Downarrow & & \langle I_{\mathrm{N}}|W_{\mathrm{N}}[V_{\mathrm{N}}/x^{\mathrm{N}}]\cdot\star^{\mathrm{N}}\rangle \\
 & & & & \triangledown_{\downarrow} \\
\langle(\lambda x^{\mathrm{n}}.\langle w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mathrm{m}} & \langle\lambda x^{\mathrm{n}}.\langle w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle W_{\mathrm{N}}[V_{\mathrm{N}}/x^{\mathrm{N}}]|\star^{\mathrm{N}}\rangle
\end{array}$$

Figure I.8.1.b: Example in naive $\mathrm{Li}_{\mathrm{n}}^{\vec{}}$

$$\begin{array}{ccccc}
\langle(\lambda x^{\mathrm{n}}.\langle I_{\mathrm{n}}w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mathrm{m}} & \langle\lambda x^{\mathrm{n}}.\langle I_{\mathrm{n}}w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle I_{\mathrm{n}}w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]|\star\rangle \\
\Downarrow_{\exists} & & \Downarrow_{\exists} & & \triangledown_{\exists} \\
\langle(\lambda x^{\mathrm{n}}.\langle I_{\mathrm{n}}|w_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mathrm{m}} & \langle\lambda x^{\mathrm{n}}.\langle I_{\mathrm{n}}|w_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle I_{\mathrm{n}}|w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]\cdot\star^{\mathrm{n}}\rangle \\
\Downarrow_{\downarrow} & & \Downarrow_{\downarrow} & & \triangledown_{\downarrow} \\
\langle(\lambda x^{\mathrm{n}}.\langle w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mathrm{m}} & \langle\lambda x^{\mathrm{n}}.\langle w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]|\star^{\mathrm{n}}\rangle
\end{array}$$

Figure I.8.1.c: Example in $\mathrm{Li}_{\mathrm{n}}^{\vec{}}$

$$\begin{array}{ccccc}
\langle(\mu(x^{\mathrm{n}}\cdot\star^{\mathrm{n}}).\langle I_{\mathrm{n}}w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mu} & \langle\mu(x^{\mathrm{n}}\cdot\star^{\mathrm{n}}).\langle I_{\mathrm{n}}w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle I_{\mathrm{n}}w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]|\star\rangle \\
\Downarrow_{\mu} & & \Downarrow_{\mu} & & \triangledown_{\mu} \\
\langle(\mu(x^{\mathrm{n}}\cdot\star^{\mathrm{n}}).\langle I_{\mathrm{n}}|w_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mu} & \langle\mu(x^{\mathrm{n}}\cdot\star^{\mathrm{n}}).\langle I_{\mathrm{n}}|w_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle I_{\mathrm{n}}|w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]\cdot\star^{\mathrm{n}}\rangle \\
\Downarrow_{\downarrow} & & \Downarrow_{\downarrow} & & \triangledown_{\downarrow} \\
\langle(\mu(x^{\mathrm{n}}\cdot\star^{\mathrm{n}}).\langle w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle)v_{\mathrm{n}}|\star^{\mathrm{n}}\rangle & \triangleright_{\mu} & \langle\mu(x^{\mathrm{n}}\cdot\star^{\mathrm{n}}).\langle w_{\mathrm{n}}|\star^{\mathrm{n}}\rangle|v_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle & \triangleright_{\rightarrow} & \langle w_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}]|\star^{\mathrm{n}}\rangle
\end{array}$$

two interpretations is only relevant when looking at the inclusion of $\mathrm{Li}_{\mathrm{n}}^{\vec{}}$ into $\mathrm{L}_{\mathrm{n}}^{\vec{}}$, and will be discussed in Section I.10.

Binding the stack variable

Since we think of $\star^{\mathrm{n}}$ as a variable, one can add a binder $\mu\star^{\mathrm{n}}.c_{\mathrm{n}}$ for it, with the reduction

$$\langle\mu\star^{\mathrm{n}}.c_{\mathrm{n}}|s_{\mathrm{n}}\rangle \triangleright_{\mu} c_{\mathrm{n}}[s_{\mathrm{n}}/\star^{\mathrm{n}}]$$

and define $t_{\mathrm{n}}u_{\mathrm{n}}$ as a notation for $\mu\star^{\mathrm{n}}.\langle t_{\mathrm{n}}|u_{\mathrm{n}}\cdot\star^{\mathrm{n}}\rangle$. Indeed, with this notation, the reduction

$$\langle t_{\mathrm{n}}u_{\mathrm{n}}|s_{\mathrm{n}}\rangle \triangleright_{\mathrm{m}} \langle t_{\mathrm{n}}|u_{\mathrm{n}}\cdot s_{\mathrm{n}}\rangle$$

Figure I.8.2: The simple fragment of the naive $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ calculus

Figure I.8.2.a: Syntax (naive)

Terms / values:
$$t_{\mathrm{n}}, u_{\mathrm{n}}, v_{\mathrm{n}}, w_{\mathrm{n}} ::= x^{\mathrm{n}} \mid t_{\mathrm{n}} u_{\mathrm{n}} \mid \lambda x^{\mathrm{n}}.c_{\mathrm{n}}$$

Stacks:
$$s_{\mathrm{n}} ::= \star^{\mathrm{n}} \mid t_{\mathrm{n}} \cdot s_{\mathrm{n}}$$

Commands:
$$c_{\mathrm{n}} ::= \langle t_{\mathrm{n}} | s_{\mathrm{n}} \rangle$$

Figure I.8.2.b: Operational reduction (naive)

$$\langle t_{\mathrm{n}} u_{\mathrm{n}} | s_{\mathrm{n}} \rangle \rhd_{\mathrm{m}} \langle t_{\mathrm{n}} | u_{\mathrm{n}} \cdot s_{\mathrm{n}} \rangle$$
$$\langle \lambda x^{\mathrm{n}}.\langle t_{\mathrm{n}} | \overrightarrow{w_n} \cdot \star^{\mathrm{n}} \rangle | v_{\mathrm{n}}^0 \cdot \overrightarrow{v_n} \cdot \star^{\mathrm{n}} \rangle \rhd_{\rightarrow} \langle t_{\mathrm{n}}[v_{\mathrm{n}}^0/x^{\mathrm{n}}] | \overrightarrow{w_n}[v_{\mathrm{n}}^0/x^{\mathrm{n}}] \cdot \overrightarrow{v_n} \cdot \star^{\mathrm{n}} \rangle$$

$$\rhd \stackrel{\mathrm{def}}{=} \rhd_{\mathrm{m}} \cup \rhd_{\rightarrow}$$

Figure I.8.3: The simple fragment of the $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ calculus

Figure I.8.3.a: Syntax

Terms / values:
$$t_{\mathrm{n}}, u_{\mathrm{n}}, v_{\mathrm{n}}, w_{\mathrm{n}} ::= x^{\mathrm{n}} \mid \mu \star^{\mathrm{n}}.c_{\mathrm{n}} \mid \mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}}$$

Stacks:
$$s_{\mathrm{n}} ::= \star^{\mathrm{n}} \mid t_{\mathrm{n}} \cdot s_{\mathrm{n}}$$

Commands:
$$c_{\mathrm{n}} ::= \langle t_{\mathrm{n}} | s_{\mathrm{n}} \rangle$$

Figure I.8.3.b: Operational reduction

$$\langle \mu \star^{\mathrm{n}}.c_{\mathrm{n}} | s_{\mathrm{n}} \rangle \rhd_{\mu} c_{\mathrm{n}}[s_{\mathrm{n}}/\star^{\mathrm{n}}]$$
$$\langle \mu(x^{\mathrm{n}} \cdot \star^{\mathrm{n}}).c_{\mathrm{n}} | v_{\mathrm{n}} \cdot s_{\mathrm{n}} \rangle \rhd_{\rightarrow} c_{\mathrm{n}}[v_{\mathrm{n}}/x^{\mathrm{n}}, s_{\mathrm{n}}/\star^{\mathrm{n}}]$$

$$\rhd \stackrel{\mathrm{def}}{=} \rhd_{\mu} \cup \rhd_{\rightarrow}$$

is a special case of $\rhd_{\mu}$:
$$\langle t_{\mathrm{n}} u_{\mathrm{n}} | s_{\mathrm{n}} \rangle = \langle \mu \star^{\mathrm{n}}.\langle t_{\mathrm{n}} | u_{\mathrm{n}} \cdot \star^{\mathrm{n}} \rangle | s_{\mathrm{n}} \rangle \rhd_{\mu} \langle t_{\mathrm{n}} | u_{\mathrm{n}} \cdot s_{\mathrm{n}} \rangle$$

Remark I.8.1

For this particular calculus, removing $t_{\mathrm{n}}u_{\mathrm{n}}$ from the syntax and adding $\mu\star^{\mathrm{n}}.c_{\mathrm{n}}$ does not change the calculus much. However, in L calculi with more constructors, more expressions can be expressed with $\mu\star^{\mathrm{n}}.c_{\mathrm{n}}$, so that the calculus with $\mu\star^{\mathrm{n}}.c_{\mathrm{n}}$ ends up being simpler. More precisely, without $\mu\star^{\mathrm{n}}.c_{\mathrm{n}}$, every stack constructor (e.g. $u_{\mathrm{n}} \cdot s_{\mathrm{n}}$) needs to have an associated expression constructor (e.g. $t_{\mathrm{n}}u_{\mathrm{n}}$), while with $\mu\star^{\mathrm{n}}.c_{\mathrm{n}}$, the expression constructors can be defined as notation (e.g. $t_{\mathrm{n}}u_{\mathrm{n}} \overset{\mathrm{ntn}}{=} \mu\star^{\mathrm{n}}.\langle t_{\mathrm{n}} | s_{\mathrm{n}} \cdot \star^{\mathrm{n}}\rangle$). The gain is therefore linear in the number of stack constructors. Here, nothing is gained because there is only one stack constructor, but for larger calculi such as those in Chapter IV, the gain is non-negligible.

I.8.2. The $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ calculus

Let-expressions and $\tilde{\mu}$

The full $\mathrm{Li}^{\rightarrow}_{\mathrm{n}}$ calculus, described in Figure I.8.4, is its simple fragment extended by commands $\langle t_{\mathrm{n}} | \tilde{\mu}x^{\mathrm{n}}.c_{\mathrm{n}}\rangle$ that represent let-expressions $\mathsf{let}\ x^{\mathrm{n}} := t_{\mathrm{n}}\ \mathsf{in}\ c_{\mathrm{n}}$ (and $\tilde{\mu}x^{\mathrm{n}}.c_{\mathrm{n}}$ that represents $\mathsf{let}\ x^{\mathrm{n}} := \square\ \mathsf{in}\ c_{\mathrm{n}}$). Their reduction is exactly what one would expect:

$$\langle t_{\mathrm{n}} | \tilde{\mu}x^{\mathrm{n}}.c_{\mathrm{n}}\rangle \rhd_{\tilde{\mu}} c_{\mathrm{n}}[t_{\mathrm{n}}/x^{\mathrm{n}}]$$

Note that commands could be defined without refering to evaluationg contexts e_{n}:

$$c_{\mathrm{n}} ::= \langle t_{\mathrm{n}} | s_{\mathrm{n}}\rangle \mid \langle t_{\mathrm{n}} | \tilde{\mu}x^{\mathrm{n}}.c_{\mathrm{n}}\rangle$$

Indeed, $\tilde{\mu}x^{\mathrm{n}}.c_{\mathrm{n}}$ can only appear inside contexts of the shape $\langle t_{\mathrm{n}} | \square\rangle$. It is nevertheless useful to have $\tilde{\mu}x^{\mathrm{n}}.c_{\mathrm{n}}$ be a term on its own because it makes the calculus more symmetric and makes expressing some definitions nicer (e.g. η-expansion).

Coercions

⚠

Disubstitutions

Disubstitutions have the properties announced in ⚠ (see Section .1 for details).

Reductions

The reductions have the properties announced in Figure **??** (see Section .2 for details).

Figure I.8.4: The $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ calculus

Figure I.8.4.a: Syntax

Terms / values:
$$t_n, u_n, v_n, w_n ::= x^n \mid \mu\star^n.c_n \mid \mu(x^n \cdot \star^n).c_n$$

Stacks:
$$s_n ::= \star^n \mid t_n \cdot s_n$$

Commands:
$$c_n ::= \langle t_n | e_n \rangle$$

Evaluation contexts:
$$e_n ::= \mathsf{stk}^n(s_n) \mid \tilde{\mu}x^n.c_n$$

Figure I.8.4.b: Notations

$$t_n u_n \overset{\text{ntn}}{=} \mu\star^n.\langle t_n | u_n \cdot \star^n \rangle$$

Figure I.8.4.c: Operational reduction

$$\langle \mu\star^n.c_n | s_n \rangle \rhd_\mu c_n[s_n/\star^n]$$
$$\langle t_n | \tilde{\mu}x^n.c_n \rangle \rhd_{\tilde{\mu}} c_n[t_n/x^n]$$
$$\langle \mu(x^n \cdot \star^n).c_n | v_n \cdot s_n \rangle \rhd_{\rightarrow} c_n[v_n/x^n, s_n/\star^n]$$

$$\rhd \overset{\text{def}}{=} \rhd_\mu \cup \rhd_{\tilde{\mu}} \cup \rhd_{\rightarrow}$$

Figure I.8.4.d: η-expansion

$$t_n \,\S_\mu\, \mu\star^n.\langle t_n | \star^n \rangle$$
$$e_n \,\S_{\mathrm{let}}\, \tilde{\mu}x^n.\langle x^n | e_n \rangle \quad \text{if } x^n \text{ fresh w.r.t. } e_n$$
$$t_n \,\S_{\rightarrow}\, \mu(x^n \cdot \star^n).\langle t_n | x^n \cdot \star^n \rangle \quad \text{if } x^n \text{ fresh w.r.t. } t_n$$

$$\S \overset{\text{def}}{=} \S_\mu \cup \S_{\tilde{\mu}} \cup \S_{\rightarrow}$$

I.9. Equivalence between $\underline{\lambda}^{\rightarrow}_{\mathrm{n}}$ and $\mathbf{Li}^{\rightarrow}_{\mathrm{n}}$

I.10. A pure call-by-name classical L calculus: $\mathsf{L}_{\mathrm{n}}^{\rightarrow}$

I.11. Simply-typed L calculi

II. Pure call-by-value calculi

II.1. A pure call-by-value λ-calculus: $\boldsymbol{\lambda}_{\mathrm{V}}^{\rightarrow}$

II.2. A pure call-by-value λ-calculus with focus: $\underline{\lambda}_{\mathrm{v}}^{\rightarrow}$

II.3. A pure call-by-value intuitionistic L calculus: $\mathrm{Li}^{\rightarrow}_{\mathrm{v}}$

II.4. A pure call-by-value classical L calculus: $\mathsf{L}_{\mathrm{v}}^{\rightarrow}$

Part B.

Untyped polarized calculi

Introduction

III. Pure polarized calculi

III.1. Relative expresiveness of call-by-name and call-by-value

III.2. A pure polarized λ-calculus: $\boldsymbol{\lambda}_{\mathrm{P}}^{\to\Uparrow\Downarrow}$

III.3. A pure polarized λ-calculus with focus: $\underline{\lambda}_{\mathrm{p}}^{\to\Uparrow\Downarrow}$

III.4. A pure polarized intuitionistic L-calculus: $\mathsf{Li}_{\mathrm{p}}^{\to\Uparrow\Downarrow}$

III.5. A pure polarized classical L-calculus: $\mathsf{L}_{\mathrm{p}}^{\to\Uparrow\Downarrow}$

IV. Polarized calculi with pairs and sums

IV.1. A polarized λ-calculus with pairs and sums: $\boldsymbol{\lambda}_{\mathrm{P}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$

IV.2. CBPV as a subcalculus of $\lambda_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$

Call-by-push-value (CBPV) [Lev01; Lev04; Lev06] is a well-known calculus that subsumes both call-by-name and call-by-value (including in the presence of side effects). It does so by decomposing Moggi's computation monad [Mog89] as an adjunction. Typed models of LJ_p^η (i.e. of $Li_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$) have been shown to generalize that of CBPV in [CurFioMun16]. In this section, we explain how $\lambda_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$ can be though of as being CBPV "completed" by adding positive expressions, and in Section IV.6, we will explain how the CBPV abstract machine relates to $Li_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$.

IV.2.1. CBPV

Syntax

We recall the syntax of Call-by-Push-Value (CBPV) in Figure IV.2.1a[1], with a few minor differences: we only have binary sum and negative pairs (and not those of arbitrary finite arity), we write $(V_{pv}, W_{pv})^{pv}$ for a pair instead of $\langle V, W\rangle$, and we add pv subscripts and superscripts. In CBPV, $\mathsf{return}(V_{pv})$ is sometimes called $\mathsf{produce}(V_{pv})$, and application $T_{pv} V_{pv}$ and (resp. projection $T_{pv} i$) are sometimes written in the reverse order $V_{pv}\,`\,T_{pv}$ (resp. $i\,`\,T_{pv}$).

Operational semantics

We recall the big-step operational semantics of CBPV Figure IV.2.1d[2], where $T_{pv} \Downarrow R_{pv}$ stands for "the computation T_{pv} terminates and its result is R_{pv}". Results form a subset of the set of computations, and their grammar is described in Figure IV.2.1c.

Complex values

Figure IV.2.1b[3] extends CBPV with complex values such as $\mathsf{pm}\ x^{pv}\ \mathsf{as}[(y^{pv}, z^{pv})^{pv}. y^{pv}]$. These are useful when looking at the semantics of CBPV, but not suitable for operational semantics because "they detract from the rigid sequential nature of the language, because they can be evaluated at any time" [Lev06]. Adding complex values has no effect on what computations can be expressed (see Proposition 14 of [Lev06]), because CBPV with complex values can be translated to CBPV without complex values (see Figure 13 of [Lev06]).

[1] This figure corresponds to figure 3.1 of [Lev01], figure 2.1 of [Lev04], figure 2 of [Lev06].
[2] This figure corresponds to Figure 4 of [Lev06].
[3] This figure corresponds to Figure 12 of [Lev06].

[Lev01] "Call-by-push-value", Levy, 2001
[Lev04] *Call-By-Push-Value: A Functional/Imperative Synthesis*, Levy, 2004
[Lev06] "Call-by-push-value: Decomposing call-by-value and call-by-name", Levy, 2006
[Mog89] "Computational Lambda-Calculus and Monads", Moggi, 1989
[CurFioMun16] "A Theory of Effects and Resources: Adjunction Models and Polarised Calculi", Curien, Fiore, and Munch-Maccagnoni, 2016

Figure IV.2.1: Call-by-Push-Value

Figure IV.2.1.a: Syntax

Values:

$$
\begin{aligned}
V_{pv} ::=\ & x^{pv} \\
\mid\ & (V_{pv}, W_{pv})^{pv} \\
\mid\ & (1, V_{pv})^{pv} \mid (2, V_{pv})^{pv} \\
\mid\ & \mathsf{thunk}(T_{pv})
\end{aligned}
$$

Expressions / computations:

$$
\begin{aligned}
T_{pv}, U_{pv} ::=\ & V_{pv} \mid \mathsf{let}\ x^{pv}\ \mathsf{be}\ V_{pv}.U_{pv} \\
\mid\ & \lambda x^{pv}.T_{pv} \mid T_{pv} V_{pv} \\
\mid\ & \lambda^{pv}[1.T_{pv} \mid 2.U_{pv}] \mid T_{pv} 1 \mid T_{pv} 2 \\
\mid\ & \mathsf{return}(V_{pv}) \mid T_{pv}\ \mathsf{to}\ x^{pv}.U_{pv} \\
\mid\ & \mathsf{pm}\ V_{pv}\ \mathsf{as}[(x^{pv}, y^{pv})^{pv}.U_{pv}] \\
\mid\ & \mathsf{pm}\ V_{pv}\ \mathsf{as}[(1, x_1^{pv})^{pv}.U^1_{pv} \mid (2, x_2^{pv})^{pv}.U^2_{pv}] \\
\mid\ & \mathsf{force}(V_{pv})
\end{aligned}
$$

Figure IV.2.1.b: Syntax with complex values

Complex values:

$$
\begin{aligned}
V_{cv}, W_{cv} ::=\ & x^{pv} \mid \mathsf{let}\ x^{pv}\ \mathsf{be}\ V_{cv}.W_{cv} \\
\mid\ & (V_{cv}, W_{cv})^{pv} \mid \mathsf{pm}\ V_{cv}\ \mathsf{as}[(x^{pv}, y^{pv})^{pv}.W_{cv}] \\
\mid\ & (1, V_{cv})^{pv} \mid (2, V_{cv})^{pv} \mid \mathsf{pm}\ V_{cv}\ \mathsf{as}[(1, x_1^{pv})^{pv}.W^1_{cv} \mid (2, x_2^{pv})^{pv}.W^2_{cv}] \\
\mid\ & \mathsf{thunk}(T_{cv})
\end{aligned}
$$

Expressions / computations (with complex values):

T_{cv}, U_{cv} ::= (Same production rules as T_{pv} with all occurrences of V_{pv} replaced by V_{cv}. See Figure IV.2.2.)

Figure IV.2.1.c: Syntax of results

Results:

$$R_{pv} ::= \mathsf{return}(V_{pv}) \mid \lambda x^{pv}.T_{pv} \mid \lambda^{pv}[1.T^1_{pv} \mid 2.T^2_{pv}]$$

Figure IV.2.1.d: Big-step operational semantics

$$\frac{T_{pv}[V_{pv}/x^{pv}] \Downarrow R_{pv}}{\mathsf{let}\ x^{pv}\ \mathsf{be}\ V_{pv}.T_{pv} \Downarrow R_{pv}}$$

$$\frac{}{\lambda x^{pv}.T_{pv} \Downarrow \lambda x^{pv}.T_{pv}} \qquad \frac{T_{pv} \Downarrow \lambda x^{pv}.U_{pv} \qquad U_{pv}[V_{pv}/x^{pv}] \Downarrow R_{pv}}{T_{pv}\,V_{pv} \Downarrow R_{pv}}$$

$$\frac{}{\lambda^{pv}[1.T^1_{pv} \mid 2.T^2_{pv}] \Downarrow \lambda^{pv}[1.T^1_{pv} \mid 2.T^2_{pv}]} \qquad \frac{T_{pv} \Downarrow \lambda^{pv}[1.U^1_{pv} \mid 2.U^2_{pv}] \qquad U^i_{pv} \Downarrow R_{pv}}{T_{pv}\,i \Downarrow R_{pv}}$$

$$\frac{}{\mathsf{return}(V_{pv}) \Downarrow \mathsf{return}(V_{pv})} \qquad \frac{T_{pv} \Downarrow \mathsf{return}(V_{pv}) \qquad U_{pv}[V_{pv}/x^{pv}] \Downarrow R_{pv}}{T_{pv}\ \mathsf{to}\ x^{pv}.U_{pv} \Downarrow R_{pv}}$$

$$\frac{T_{pv} \Downarrow R_{pv}}{\mathsf{force}(\mathsf{thunk}(T_{pv})) \Downarrow R_{pv}}$$

$$\frac{T_{pv}[V_{pv}/x^{pv}, W_{pv}/y^{pv}] \Downarrow R_{pv}}{\mathsf{pm}(V_{pv}, W_{pv})^{pv}\ \mathsf{as}[(x^{pv}, y^{pv})^{pv}.T_{pv}] \Downarrow R_{pv}}$$

$$\frac{T^i_{pv}[V_{pv}/x^{pv}_i] \Downarrow R_{pv}}{\mathsf{pm}(i, V_{pv})^{pv}\ \mathsf{as}[(1, x^{pv}_1)^{pv}.T^1_{pv} \mid (2, x^{pv}_2)^{pv}.T^2_{pv}] \Downarrow R_{pv}}$$

IV.2.2. Embedding CBPV into $\lambda_{\mathrm{P}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$

Embedding values and computations

Figure IV.2.2: Syntax of $\boldsymbol{\lambda}_{\mathrm{P}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$ (left) and CBPV (right)

Positive values:

$V_+, W_+ ::= x^+$
$\mid (V_+ \otimes W_+)$
$\mid \iota_1(V_+) \mid \iota_2(V_+)$
$\mid \mathsf{box}(V_-)$

Negative values / expressions:

$V_-, W_-, T_-, U_- ::= x^- \mid \mathsf{let}\ x^+ := T_+\ \mathsf{in}\ U_- \mid \mathsf{let}\ x^- := T_-\ \mathsf{in}\ U_-$
$\mid \lambda x^+.T_- \mid T_- V_+$
$\mid (T_- \& U_-) \mid \pi_1(T_-) \mid \pi_2(T_-)$
$\mid \mathsf{freeze}(T_+)$
$\mid \mathsf{match}\ T_+\ \mathsf{with}[(x^+ \otimes y^+).U_-]$
$\mid \mathsf{match}\ T_+\ \mathsf{with}[\iota_1(x_1^+).U_\varepsilon^1 \mid \iota_2(x_2^+).U_-^2]$
$\mid \mathsf{match}\ T_+\ \mathsf{with}[\mathsf{box}(x^-).U_-]$

Positive expressions:

$T_+, U_+ ::= V_+$
$\mid \mathsf{let}\ x^+ := T_+\ \mathsf{in}\ U_+ \mid \mathsf{let}\ x^- := T_-\ \mathsf{in}\ U_+$
$\mid \mathsf{unfreeze}(T_-)$
$\mid \mathsf{match}\ T_+\ \mathsf{with}[(x^+ \otimes y^+).U_+]$
$\mid \mathsf{match}\ T_+\ \mathsf{with}[\iota_1(x_1^+).U_+^1 \mid \iota_2(x_2^+).U_+^2]$
$\mid \mathsf{match}\ T_+\ \mathsf{with}[\mathsf{box}(x^+).U_+]$

Values:

$V_{\mathrm{pv}} ::= x^{\mathrm{pv}}$
$\mid (V_{\mathrm{pv}}, W_{\mathrm{pv}})^{\mathrm{pv}}$
$\mid (1, V_{\mathrm{pv}})^{\mathrm{pv}} \mid (2, V_{\mathrm{pv}})^{\mathrm{pv}}$
$\mid \mathsf{thunk}(T_{\mathrm{pv}})$

Expressions / computations:

$T_{\mathrm{pv}}, U_{\mathrm{pv}} ::= V_{\mathrm{pv}} \mid \mathsf{let}\ x^{\mathrm{pv}}\ \mathsf{be}\ V_{\mathrm{pv}}.U_{\mathrm{pv}}$
$\mid \lambda x^{\mathrm{pv}}.T_{\mathrm{pv}} \mid T_{\mathrm{pv}} V_{\mathrm{pv}}$
$\mid \lambda^{\mathrm{pv}}[1.T_{\mathrm{pv}} \mid 2.U_{\mathrm{pv}}] \mid T_{\mathrm{pv}}1 \mid T_{\mathrm{pv}}2$
$\mid \mathsf{return}(V_{\mathrm{pv}}) \mid T_{\mathrm{pv}}\ \mathsf{to}\ x^{\mathrm{pv}}.U_{\mathrm{pv}}$
$\mid \mathsf{pm}\ V_{\mathrm{pv}}\ \mathsf{as}[(x^{\mathrm{pv}}, y^{\mathrm{pv}})^{\mathrm{pv}}.U_{\mathrm{pv}}]$
$\mid \mathsf{pm}\ V_{\mathrm{pv}}\ \mathsf{as}[(1, x_1^{\mathrm{pv}})^{\mathrm{pv}}.U_{\mathrm{pv}}^1 \mid (2, x_2^{\mathrm{pv}})^{\mathrm{pv}}.U_{\mathrm{pv}}^2]$
$\mid \mathsf{force}(V_{\mathrm{pv}})$

Complex values:

$V_{\mathrm{cv}}, W_{\mathrm{cv}} ::= x^{\mathrm{pv}} \mid (V_{\mathrm{cv}}, W_{\mathrm{cv}})^{\mathrm{pv}} \mid (1, V_{\mathrm{cv}})^{\mathrm{pv}} \mid (2, V_{\mathrm{cv}})^{\mathrm{pv}} \mid \mathsf{thunk}(T_{\mathrm{pv}})$
$\mid \mathsf{let}\ x^{\mathrm{pv}}\ \mathsf{be}\ V_{\mathrm{cv}}.W_{\mathrm{cv}}$
$\mid$
$\mid \mathsf{pm}\ V_{\mathrm{cv}}\ \mathsf{as}[(x^{\mathrm{pv}}, y^{\mathrm{pv}})^{\mathrm{pv}}.W_{\mathrm{cv}}]$
$\mid \mathsf{pm}\ V_{\mathrm{cv}}\ \mathsf{as}[(1, x_1^{\mathrm{pv}})^{\mathrm{pv}}.W_{\mathrm{cv}} \mid (2, x_2^{\mathrm{pv}})^{\mathrm{pv}}.W_{\mathrm{cv}}]$

The syntaxes of $\lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$ and CBPV are shown side by side in Figure IV.2.2, with expressions and values that correspond to each other placed on the same line, and things that are present in one calculus but not the other highlighted. Values V_{pv} of CBPV correspond to positive values V_+ of $\lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$, and expressions T_{pv} of CBPV correspond to negative expressions T_- of $\lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$. For shifts (see Figure IV.2.3), $\mathsf{thunk}(T_{pv})$ corresponds to $\mathsf{box}(T_-)$, $\mathsf{force}(V_{pv})$ to $\mathsf{unbox}(V_+)$, and $\mathsf{return}(V_{pv})$ to $\mathsf{freeze}(\mathsf{val}(V_+))$ (i.e. the restriction of the general $\mathsf{freeze}(T_+)$ to values). The "inverse" $T_{pv}\ \mathsf{to}\ x^{pv}.U_{pv}$ of $\mathsf{return}(V_{pv})$ corresponds to $\mathsf{let}\ x^+ := \mathsf{unfreeze}(T_-)\ \mathsf{in}\ U_-$. The values types A_{pv} and computation type $\underline{B}_{pv}$ of CBPV (which are not described here) correspond to positive types A_+ and negative types B_- respectively, with $\mathsf{F}^{pv}(A_{pv})$ corresponding to $\Uparrow A_+$ and $\mathsf{U}^{pv}(\underline{B}_{pv})$ to $\Downarrow B_-$[4]. More precisely, the translation

$$\underline{\cdot}_P : \text{CBPV} \to \lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$$

described in Figure IV.2.4 is an embedding:

Fact IV.2.1

The translation $\underline{\cdot}_P : \text{CBPV} \to \lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$ is injective.

Proof

By induction on the syntax.

Fact IV.2.2

The translation $\underline{\cdot}_P$ is substitutive: for any computation T_{pv} (resp. value V_{pv}), variable x^{pv}, and value W_{pv}, we have

$$\underline{T_{pv}[W_{pv}/x^{pv}]}_P = \underline{T_{pv}}_P\left[\underline{W_{pv}}_P/x^+\right] \quad \left(\text{resp. } \underline{V_{pv}[W_{pv}/x^{pv}]}_P = \underline{V_{pv}}_P\left[\underline{W_{pv}}_P/x^+\right]\right)$$

Proof

By induction on the syntax of T_{pv} (resp. V_{pv}).

Differences between CBPV and $\lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$

With these correspondances in mind, there are two main differences between CBPV and $\lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$:

- There are no negative variables x^- in CBPV, and hence no $\mathsf{let}\ x^- := V_-\ \mathsf{in}\ U_\varepsilon$. The only other use of negative variables in $\lambda_P^{\to\&\Uparrow\otimes\oplus\Downarrow}$, namely $\mathsf{match}\ T_+\ \mathsf{with}[\mathsf{box}(x^-).U_-]$, is

[4]For this correspondance, one can remember that U^{pv} unfortunately *does not* corresponds to the Upshift $\Uparrow$, or notice that both $\supset$ and $\Rightarrow$ are common symbols for implication, and that applying the same rotation to both of them yields U^{pv} and $\Downarrow$.

Figure IV.2.3: Shifts in $\boldsymbol{\lambda}_{\mathrm{p}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$ (left) and CBPV (right)

Figure IV.2.4: Embedding $\underline{\cdot}_{\mathrm{p}}$ of CBPV into $\boldsymbol{\lambda}_{\mathrm{p}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$

Values:

$$\underline{\cdot}_{\mathrm{p}} : \mathbf{V}_{\mathrm{pv}} \to \mathbf{V}_{+}$$

$$\underline{x^{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} x^{+}$$

$$\underline{(V_{\mathrm{pv}}, W_{\mathrm{pv}})^{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \left(\underline{V_{\mathrm{pv}}}_{\mathrm{p}} \otimes \underline{W_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

$$\underline{(i, V_{\mathrm{pv}})^{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \iota_i\left(\underline{V_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

$$\underline{\mathsf{thunk}(T_{\mathrm{pv}})}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{box}\left(\underline{T_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

Expressions:

$$\underline{\cdot}_{\mathrm{p}} : \mathbf{T}_{\mathrm{pv}} \to \mathbf{T}_{-}$$

$$\underline{\mathsf{pm}\ V_{\mathrm{pv}}\ \mathsf{as}\left[(x^{\mathrm{pv}}, y^{\mathrm{pv}})^{\mathrm{pv}}.T_{\mathrm{pv}}\right]}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{match}\ \underline{V_{\mathrm{pv}}}_{\mathrm{p}}\ \mathsf{with}\left[(x^{+} \otimes y^{+}).\underline{T_{\mathrm{pv}}}_{\mathrm{p}}\right]$$

$$\underline{\mathsf{pm}\ V_{\mathrm{pv}}\ \mathsf{as}\left[(1, x_1^{\mathrm{pv}})^{\mathrm{pv}}.T^1_{\mathrm{pv}} \mid (2, x_2^{\mathrm{pv}})^{\mathrm{pv}}.T^2_{\mathrm{pv}}\right]}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{match}\ \underline{V_{\mathrm{pv}}}_{\mathrm{p}}\ \mathsf{with}\left[\iota_1(x_1^{+}).\underline{V_{\mathrm{pv}}}_{\mathrm{p}} \mid \iota_2(x_2^{+}).\underline{T^2_{\mathrm{pv}}}_{\mathrm{p}}\right]$$

$$\underline{\mathsf{force}(V_{\mathrm{pv}})}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{unbox}\left(\underline{V_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

$$\underline{\mathsf{let}\ x^{\mathrm{pv}}\ \mathsf{be}\ V_{\mathrm{pv}}.T_{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{let}\ x^{+} := \underline{V_{\mathrm{pv}}}_{\mathrm{p}}\ \mathsf{in}\ \underline{T_{\mathrm{pv}}}_{\mathrm{p}}$$

$$\underline{\lambda x^{\mathrm{pv}}.T_{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \lambda x^{+}.\underline{T_{\mathrm{pv}}}_{\mathrm{p}}$$

$$\underline{T_{\mathrm{pv}} V_{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \underline{T_{\mathrm{pv}}}_{\mathrm{p}}\,\underline{V_{\mathrm{pv}}}_{\mathrm{p}}$$

$$\underline{\lambda^{\mathrm{pv}}\left[1.T^1_{\mathrm{pv}} \mid 2.T^2_{\mathrm{pv}}\right]}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \left(\underline{T^1_{\mathrm{pv}}}_{\mathrm{p}} \,\&\, \underline{T^2_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

$$\underline{T_{\mathrm{pv}}\, i}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \pi_i\left(\underline{T_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

$$\underline{\mathsf{return}(V_{\mathrm{pv}})}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{freeze}\left(\underline{V_{\mathrm{pv}}}_{\mathrm{p}}\right)$$

$$\underline{T_{\mathrm{pv}}\ \mathsf{to}\ x^{\mathrm{pv}}.U_{\mathrm{pv}}}_{\mathrm{p}} \stackrel{\mathrm{def}}{=} \mathsf{let}\ x^{+} := \mathsf{unfreeze}\left(\underline{T_{\mathrm{pv}}}_{\mathrm{p}}\right)\ \mathsf{in}\ \underline{U_{\mathrm{pv}}}_{\mathrm{p}}$$

restricted to the cases where T_+ is a value $T_+ = V_+$ and $U_- = x^-$, i.e. to $\mathsf{unbox}(V_+)$, and is denoted by $\mathsf{force}(V_{pv})$.

- There are no non-value positive expressions T_+ in CBPV (without complex values), which corresponds to replacing T_+ by V_+ everywhere in the syntax of $\boldsymbol{\lambda}_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$. Since $\mathsf{unfreeze}(T_-)$ is a positive expression, it is no longer expressible, and is therefore replaced by $T_{pv}\ \mathsf{to}\ x^{pv}.U_{pv}$ which corresponds to its composition with a let-expression $\mathsf{let}\ x^+ := \mathsf{unfreeze}(T_+)\ \mathsf{in}\ U_-$.

Complex values and positive expressions

Complex values V_{cv} are very similar to positive expressions T_+, but neither of the set of complex values nor the set of positive expressions is contained in the other:

- $\mathsf{unfreeze}(T_-)$ corresponds to no complex value; and
- $(((\lambda x^{pv}.\mathsf{return}(x^{pv}))V_{pv}), W_{pv})^{pv}$ is a complex value, while $(((\lambda x^+.\mathsf{freeze}(x^+))V_+)\otimes W_+)$ is not a positive term (because $(\lambda x^+.\mathsf{freeze}(x^+))V_+$ is not a value).

Complex values can nevertheless be represented by positive terms via let-expansions, e.g.

$$(((\lambda x^{pv}.\mathsf{return}(x^{pv}))V_{pv}), W_{pv})^{pv} \quad \text{corresponds to} \quad \mathsf{let}\ y^+ := (\lambda x^+.\mathsf{freeze}(x^+))V_+\ \mathsf{in}(y^+ \otimes W_+)$$

More generally, if V_{cv} corresponds to T_+, and W_{cv} to U_+, then the complex value

$$(V_{cv}, W_{cv})^{pv} \quad \text{corresponds to} \quad \mathsf{let}\ x^+ := T_+\ \mathsf{in}\ \mathsf{let}\ y^+ := U_+\ \mathsf{in}(x^+ \otimes y^+)$$

Expanding CBPV with positive terms (i.e. to $\boldsymbol{\lambda}_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$) has the same advantages as extending it with complex values (i.e. it makes it better suited for semantic endeavors), but avoids the complications of the operational semantics induced by complex values: the choice of when to evaluate complex values is pushed to the "user" through the need for let-expression to express some complex values. Of course, in an actual programming language, we would want to be able to write $(T_+ \otimes U_+)$, but this could be a notation for $\mathsf{let}\ x^+ := T_+\ \mathsf{in}\ \mathsf{let}\ y^+ := U_+\ \mathsf{in}(x^+ \otimes y^+)$, and can therefore be ignored for theoretical purposes.

Preservation of operational semantics

In $\boldsymbol{\lambda}_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$, we have a small-step operational semantics $\triangleright$, which induces a big-step operational semantics given by

$$T_\varepsilon \Downarrow T'_\varepsilon \overset{\text{def}}{=} T_\varepsilon \triangleright^* T'_\varepsilon \not\triangleright$$

Through the translation $\underline{\cdot}_p$, the big-step operational semantics of CBPV corresponds exactly to the one of $\boldsymbol{\lambda}_p^{\to\&\Uparrow\otimes\oplus\Downarrow}$:

Proposition IV.2.3

For any closed expression T_{pv}, $T_{pv} \Downarrow R_{pv}$ if and only if $\underline{T_{pv}}_p \Downarrow \underline{R_{pv}}_p$.

Proof

- ⇒ We have $\underline{T_{\mathrm{pv}}}_{\mathrm{P}} \triangleright^* \underline{R_{\mathrm{pv}}}_{\mathrm{P}}$ by induction on the derivation of $T_{\mathrm{pv}} \Downarrow R_{\mathrm{pv}}$ and Fact IV.2.2, and $\underline{R_{\mathrm{pv}}}_{\mathrm{P}} \not\triangleright$ by case analysis on the syntax of R_{pv}.
- ⇐ By induction on the length of the reduction $\underline{T_{\mathrm{pv}}}_{\mathrm{P}} \triangleright^* \underline{R_{\mathrm{pv}}}_{\mathrm{P}}$.

IV.3. A polarized λ-calculus with focus: $\underline{\lambda}_{\mathrm{p}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$

IV.4. A polarized intuitionistic L calculus: $\mathrm{Li}_{\mathrm{p}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$

IV.5. A polarized classical L calculus: $\mathsf{L}_{\mathrm{p}}^{\to \& \Uparrow \otimes \oplus \Downarrow}$

IV.6. The CBPV abstract machine as a subcalculus of $\underline{\lambda}_{\mathrm{p}}^{\to\&\Uparrow\otimes\oplus\Downarrow}$

V. Polarized calculi with arbitrary constructors

V.1. A (classical) polarized L-calculus: $\mathsf{L}_{\mathrm{p}}^{\vec{\tau}}$

V.1.1. Syntax

Type formers

Everything starts with a (finite) set of positive type formers $\tau_+^1, \dots, \tau_+^n$ and negative type formers $\tau_-^1, \dots, \tau_-^m$ that generate positive types A_+ and negative types A_- as described in Figure V.1.1a. With the usual type formers, this yields Figure V.1.1e. For binary type formers (e.g. $\rightarrow$), we often use the infix notation (e.g. we write $A_+ \rightarrow B_-$ for $\rightarrow(A_+, B_-)$). Even though the notation $\tau_\varepsilon^j(\vec{A})$ may suggested it, the type formers τ_ε^j do not take arbitrary sequence $\vec{A}$ of arguments: the arity of each τ_ε^j is fixed (e.g. $\rightarrow(A_-)$ would be invalid), and the polarity of each argument is also fixed (e.g. $\rightarrow(A_+, B_+)$ would also be invalid). A more precise notation would be

$$\tau_\varepsilon^j\left(A^1_{\mathsf{pol}(\tau_\varepsilon^j,1)}, \dots, A^{\mathsf{ar}(\tau_\varepsilon^j)}_{\mathsf{pol}(\tau_\varepsilon^j,\mathsf{ar}(\tau_\varepsilon^j))}\right)$$

where $\mathsf{ar}(\tau_\varepsilon^j)$ is the arity of τ_ε^j, and $\mathsf{pol}(\tau_\varepsilon^j, k)$ is the polarity of the k^{th} argument of τ_ε^j. In other words, when we write $\tau_\varepsilon^j(\vec{A})$, the length and shape of $\vec{A}$ depends on τ_ε^j, but we do not make this dependence explicit in the notations.

Value and stack constructors

We denote by a and call argument any value v or stack s, and write $\vec{a}$ for an arbitrary list $a_1, \dots, a_q$ of arguments. We denote by χ[1] and call variable any value variable x^ε or stack variable α^ε, and write $\vec{\chi}$ for an arbitrary list $\chi_1, \dots, \chi_q$ of variables.

As depicted in Figure V.1.2, each positive type former τ_+^j (resp. negative type former τ_-^j) has $l_j^+ \in \mathbb{N}$ (positive) value constructors $\mathfrak{v}_1^{\tau_+^j}, \dots, \mathfrak{v}_{l_j^+}^{\tau_+^j}$ (resp. $l_j^- \in \mathbb{N}$ (negative) stack constructors $\mathfrak{s}_1^{\tau_-^j}, \dots, \mathfrak{s}_{l_j^-}^{\tau_-^j}$), which can be applied to suitable arguments to form positive values $\mathfrak{v}_k^{\tau_+^j}(\vec{a})$ (resp. negative stacks $\mathfrak{s}_k^{\tau_-^j}(\vec{a})$)[2], and a positive stack (resp. negative value)

$$\tilde{\mu}\begin{bmatrix} \mathfrak{v}_1^{\tau_+^j}(\overrightarrow{\chi_1}).\, c_1 \\ \vdots \\ \mathfrak{v}_{l_j^+}^{\tau_+^j}(\overrightarrow{\chi_{l_j^+}}).\, c_{l_j^+} \end{bmatrix} \quad \left(\text{resp. } \mu\left\langle \begin{matrix} \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{\chi_1}).\, c_1 \\ \vdots \\ \mathfrak{s}_{l_j^-}^{\tau_-^j}(\overrightarrow{\chi_{l_j^-}}).\, c_{l_j^-} \end{matrix} \right\rangle\right)$$

that matches over all values (resp. stacks) formed using these constructors. This stack (resp. value) is often denoted by

$$\tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{\chi_1}).c_1 | \dots | \mathfrak{v}_{l_j^+}^{\tau_+^j}(\overrightarrow{\chi_{l_j^+}}).c_{l_j^+}\right] \quad \left(\text{resp. } \mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{\chi_1}).c_1 | \dots | \mathfrak{s}_{l_j^-}^{\tau_-^j}(\overrightarrow{\chi_{l_j^-}}).c_{l_j^-} \right\rangle\right)$$

When $l_j^+ = 1$ (resp. $l_j^- = 1$), we sometimes write these without the $[\cdot]$ (resp. $<\cdot>$), e.g.

[1]Mnemonic: the symbol for variables χ looks like the symbol for value variables x and is from the Greek alphabet just like the symbol for stack variables α.

[2]Note that even though we write $\mathfrak{v}_k^{\tau_+^j}(\vec{a})$ (resp. $\mathfrak{s}_k^{\tau_+^j}(\vec{a})$), the length and shape of $\vec{a}$ depend on τ_+^j and $\mathfrak{v}_k^{\tau_+^j}$(resp. τ_-^j and $\mathfrak{s}_k^{\tau_-^j}$), just like we wrote write $\tau_\varepsilon^j(\vec{A})$ even though the length and shape of $\vec{A}$ could depend on τ_ε^j.

Figure V.1.1: Types generated by a set of type formers $\vec{\tau}$

Figure V.1.1.a: Types generated by $\vec{\tau} = \tau^1_+ \ldots \tau^n_+ \tau^1_- \ldots \tau^m_-$

Positive types:

$$A_+, B_+ ::= \tau^1_+(\vec{A}) \mid \vdots \mid \tau^n_+(\vec{A})$$

Negative types:

$$A_-, B_- ::= \tau^1_-(\vec{A}) \mid \vdots \mid \tau^m_-(\vec{A})$$

Figure V.1.1.b: Types generated by $\vec{\tau} = \rightarrow_-$

Positive types:

$A_+, B_+ ::=$ (none)

Negative types:

$$A_-, B_- ::= A_- \rightarrow_- B_-$$

Figure V.1.1.c: Types generated by $\vec{\tau} = \rightarrow_+ \Downarrow$

Positive types:

$$A_+, B_+ ::= \Downarrow A_-$$

Negative types:

$$A_-, B_- ::= A_+ \rightarrow_+ B_+$$

Figure V.1.1.d: Types generated by $\vec{\tau} = \rightarrow \Downarrow \Uparrow$

Positive types:

$$A_+, B_+ ::= \mid \Downarrow A_-$$

Negative types:

$$A_-, B_- ::= A_+ \rightarrow B_- \mid \Uparrow A_+$$

Figure V.1.1.e: Types generated by $\vec{\tau} = \rightarrow \Downarrow \Uparrow \neg_- \neg_+ \otimes ⅋ \oplus \& 1 \bot 0 \top$

Positive types:

$$A_+, B_+ ::= \mid \Downarrow A_- \mid \neg_+(A_-) \mid A_+ \otimes B_+ \mid A_+ \oplus B_+ \mid 1 \mid 0$$

Negative types:

$$A_-, B_- ::= A_+ \rightarrow B_- \mid \Uparrow A_+ \mid \neg_-(A_+) \mid A_- ⅋ B_- \mid A_- \& B_- \mid \bot \mid \top$$

Figure V.1.2: Examples of value and stack constructors

Figure V.1.2.a: Examples of value constructors and value pattern-matchings

Positive type former	Value constructors	Value pattern match
$\Downarrow$	$\mathfrak{v}_1^{\Downarrow}(v_-) = \{v_-\}$	$\tilde{\mu}\{x^-\}.c$
$\neg_+$	$\mathfrak{v}_1^{\neg_+}(s_-) = \neg_+(s_-)$	$\tilde{\mu}\neg_+(\alpha^-).\alpha c$
$\otimes$	$\mathfrak{v}_1^{\otimes}(v_+, w_+) = (v_+ \otimes w_+)$	$\tilde{\mu}(x^+ \otimes y^+).c$
$\oplus$	$\mathfrak{v}_1^{\oplus}(v_+) = \iota_1(v_+)$ $\mathfrak{v}_2^{\oplus}(v_+) = \iota_2(v_+)$	$\tilde{\mu}\left[\begin{array}{l}\iota_1(x_1^+).\,c_1 \\ \iota_2(x_2^+).\,c_2\end{array}\right]$
1	$\mathfrak{v}_1^{1}() = ()$	$\tilde{\mu}().c$
0	(none)	$\tilde{\mu}[]$

Figure V.1.2.b: Examples of stack constructors and stack pattern-matchings

Negative type former	Stack constructors	Stack pattern match
$\rightarrow$	$\mathfrak{s}_1^{\rightarrow}(v_+, s_-) = v_+ \cdot s_-$	$\mu(x^+ \cdot \alpha^-).c$
$\rightarrow_-$	$\mathfrak{s}_1^{\rightarrow_-}(v_-, s_-) = v_- \mathbin{\bar{\cdot}} s_-$	$\mu(x^- \mathbin{\bar{\cdot}} \alpha^-).c$
$\rightarrow_+$	$\mathfrak{s}_1^{\rightarrow_+}(v_+, s_+) = v_+ \mathbin{\dot{+}} s_+$	$\mu(x^+ \mathbin{\dot{+}} \alpha^+).c$
$\Uparrow$	$\mathfrak{s}_1^{\Uparrow}(s_+) = \{s_+\}$	$\mu\{x^+\}.c$
$\neg_-$	$\mathfrak{s}_1^{\neg_-}(v_+) = \neg_-(v_+)$	$\mu\neg_-(x^+).xc$
$\parr$	$\mathfrak{s}_1^{\parr}(s_-^1, s_-^2) = (s_-^1 \parr s_-^2)$	$\mu(\alpha^- \parr \beta^-).c$
$\&$	$\mathfrak{s}_1^{\&}(s_-) = \pi_1 \cdot s_-$ $\mathfrak{s}_2^{\&}(s_-) = \pi_2 \cdot s_-$	$\mu\left\langle\begin{array}{l}(\pi_1 \cdot \alpha_1^-).\,c_1 \\ (\pi_2 \cdot \alpha_2^-).\,c_2\end{array}\right\rangle$
$\bot$	$\mathfrak{s}_1^{\bot}() = \tilde{()}$	$\mu\tilde{()}.c$
$\top$	(none)	$\mu<>$

writing

$$\tilde{\mu}\{x^+\}.c \text{ for } \tilde{\mu}[\{x^+\}.c] \quad (\text{resp. } \mu\{\alpha^-\}.c \text{ for } \mu<\{\alpha^-\}.c>)$$

To simplify notations, we sometimes assume that constructors take value arguments before stack arguments:

Definition V.1.1

A constructor $\mathfrak{v}_k^{\tau_+^j}$ (resp. $\mathfrak{s}_k^{\tau_-^j}$) is said to be *vs-sorted* when its value arguments are on the left of its stack arguments, i.e. when

$$\mathfrak{v}_k^{\tau_+^j}(\vec{a}) = \mathfrak{v}_k^{\tau_+^j}(\vec{v}, \vec{s}) \quad \left(\text{resp. } \mathfrak{s}_k^{\tau_-^j}(\vec{a}) = \mathfrak{s}_k^{\tau_-^j}(\vec{v}, \vec{s})\right)$$

Replacing a constructor $\mathfrak{v}_k^{\tau_+^j}$ (resp. $\mathfrak{s}_k^{\tau_-^j}$) by another one that takes its arguments in another order changes nothing for our purposes, and we therefore assume that all constructors are vs-sorted when convenient.

Syntax

The syntax of $L_p^{\vec{\tau}}$ is given in Figure V.1.3a, and the result of instanciating it with $\vec{\tau} = \rightarrow\Downarrow\Uparrow\neg\_\neg_+\otimes⅋\oplus\&1\bot 0\top$ is given in Figure V.1.4a. The polarities ε on commands $\langle\cdot|\cdot\rangle^\varepsilon$, and $+$ and $-$ on the coercions val^+ and stk^- are there to ensure that the induced grammar of fragments (see ⚠) is non-ambiguous, but are superfluous in the grammar of terms (i.e. removing them does not make the grammar of terms ambiguous). The coercions val^+ and stk^- are often left implicit[3].

V.1.2. Reductions

Definitions

The operational reduction $\triangleright$ (which is also the top-level β-reduction $\triangleright$ in $L_p^{\vec{\tau}}$) is defined defined in Figure V.1.3c, and the top-level η-expansion $\mathrel{\rotatebox{90}{$\triangleleft$}}$ is defined in Figure V.1.3d. The result of instanciating these with $\vec{\tau} = \rightarrow\Downarrow\Uparrow\neg\_\neg_+\otimes⅋\oplus\&1\bot 0\top$ is described in Figure V.2.3b and Figure V.2.3c respectively. The strong reduction $\twoheadrightarrow\!\!\triangleright$ is defined as the contextual closure $\mathcal{K}\,\boxed{\triangleright}$ of the operational reduction $\triangleright$, and the η-expansion $\dashv$ as the contextual closure $\mathcal{K}\,[\,]$ of the top-level η-expansion. The reduction $\xrightarrow{\neg o}\!\triangleright$ is defined as the closure $(\mathcal{K} \setminus \{\square\})\boxed{\triangleright}$ of the operational reduction $\triangleright$ under non-trivial contexts. Alternative definitions of these closures via inferrence rules can be found in ⚠.

[3] We only use these coercions when defining the η-expansions $\rfloor_\mu$ and $\rfloor_{\tilde{\mu}}$ (see Remark V.1.2), and for everything else, we leave these coercions implicit.

Figure V.1.3: The $L_p^{\vec{\tau}}$ calculus

Figure V.1.3.a: Syntax

Positive values:

$$\begin{aligned} v_+, w_+ ::= {} & x^+ \\ & \mid \mathfrak{v}_1^{\tau_+^1}(\vec{a}) \mid \dots \mid \mathfrak{v}_{l_1^+}^{\tau_+^1}(\vec{a}) \\ & \mid \vdots \qquad \mid \ddots \mid \vdots \\ & \mid \mathfrak{v}_1^{\tau_+^n}(\vec{a}) \mid \dots \mid \mathfrak{v}_{l_n^+}^{\tau_+^n}(\vec{a}) \end{aligned}$$

Positive stacks / evaluation contexts:

$$\begin{aligned} s_+, e_+ ::= {} & \alpha^+ \mid \tilde{\mu} x^+.c \\ & \mid \tilde{\mu}\Big[\mathfrak{v}_1^{\tau_+^1}(\overrightarrow{\chi_1}).c_1 \mid \dots \mid \mathfrak{v}_{l_1^+}^{\tau_+^1}(\overrightarrow{\chi_{l_1^+}}).c_{l_1^+}\Big] \\ & \mid \vdots \\ & \mid \tilde{\mu}\Big[\mathfrak{v}_1^{\tau_+^n}(\overrightarrow{\chi_1}).c_1 \mid \dots \mid \mathfrak{v}_{l_n^+}^{\tau_+^n}(\overrightarrow{\chi_{l_n^+}}).c_{l_n^+}\Big] \end{aligned}$$

Positive expressions:

$$t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mu\alpha^+.c$$

Negative values / expressions:

$$\begin{aligned} v_-, w_-, t_-, u_- ::= {} & x^- \mid \mu\alpha^-.c \\ & \mid \mu\Big\langle \mathfrak{s}_1^{\tau_-^1}(\overrightarrow{\chi_1}).c_1 \mid \dots \mid \mathfrak{s}_{l_1^-}^{\tau_-^1}(\overrightarrow{\chi_{l_1^-}}).c_{l_1^-}\Big\rangle \\ & \mid \vdots \\ & \mid \mu\Big\langle \mathfrak{s}_1^{\tau_-^m}(\overrightarrow{\chi_1}).c_1 \mid \dots \mid \mathfrak{s}_{l_m^-}^{\tau_-^m}(\overrightarrow{\chi_{l_m^-}}).c_{l_m^-}\Big\rangle \end{aligned}$$

Negative stacks:

$$\begin{aligned} s_- ::= {} & \alpha^- \\ & \mid \mathfrak{s}_1^{\tau_-^1}(\vec{a}) \mid \dots \mid \mathfrak{s}_{l_1^-}^{\tau_-^1}(\vec{a}) \\ & \mid \vdots \qquad \mid \ddots \mid \vdots \\ & \mid \mathfrak{s}_1^{\tau_-^m}(\vec{a}) \mid \dots \mid \mathfrak{s}_{l_m^-}^{\tau_-^m}(\vec{a}) \end{aligned}$$

Negative evaluation contexts:

$$e_- ::= \mathsf{stk}^-(s_-) \mid \tilde{\mu} x^-.c$$

Commands:

$$c ::= \langle t_+ | e_+ \rangle^+ \mid \langle t_- | e_- \rangle^-$$

Figure V.1.3.b: Notations

Polarities: $\varepsilon ::= + \mid -$

Arguments: $a ::= v_\varepsilon \mid s_\varepsilon$

Variables: $\chi ::= x^\varepsilon \mid \alpha^\varepsilon$

Term: $\mathfrak{t} ::= t_\varepsilon \mid v_\varepsilon \mid e_\varepsilon \mid s_\varepsilon \mid c$

Figure V.1.3.c: Operational reduction

$$
\begin{aligned}
\langle \mu\alpha^{\varepsilon}.c \,|\, s_{\varepsilon}\rangle^{\varepsilon} &\rhd_{\mu}\ c[s_{\varepsilon}/\alpha^{\varepsilon}] \\
\langle v_{\varepsilon} \,|\, \tilde{\mu}x^{\varepsilon}.c\rangle^{\varepsilon} &\rhd_{\tilde{\mu}}\ c[v_{\varepsilon}/x^{\varepsilon}] \\
\left\langle \mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{\chi_1}).c_1 \,|\ldots|\, \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{\chi_l}).c_l \right\rangle \middle|\, \mathfrak{s}_k^{\tau_-^j}(\vec{a})\right\rangle^{-} &\rhd_{\tau_-^j}\ c_k[\vec{a}/\overrightarrow{\chi_k}] \\
\left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{a}) \,\middle|\, \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{\chi_1}).c_1 \,|\ldots|\, \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{\chi_l}).c_l\right]\right\rangle^{+} &\rhd_{\tau_+^j}\ c_k[\vec{a}/\overrightarrow{\chi_k}]
\end{aligned}
$$

$$
\rhd \stackrel{\text{def}}{=} \rhd_{\tilde{\mu}} \cup \rhd_{\tilde{\mu}} \cup \left(\bigcup_j \rhd_{\tau_-^j}\right) \cup \left(\bigcup_j \rhd_{\tau_+^j}\right)
$$

Figure V.1.3.d: Top-level η-expansion

$$
\begin{aligned}
t_{\varepsilon} &\rightsquigarrow_{\mu}\ \mu\alpha^{\varepsilon}.\langle t_{\varepsilon} | \alpha^{\varepsilon}\rangle^{\varepsilon} && \text{if } \alpha^{\varepsilon} \text{ fresh w.r.t. } t_{\varepsilon} \\
e_{\varepsilon} &\rightsquigarrow_{\tilde{\mu}}\ \tilde{\mu}x^{\varepsilon}.\langle x^{\varepsilon} | e_{\varepsilon}\rangle^{\varepsilon} && \text{if } x^{\varepsilon} \text{ fresh w.r.t. } e_{\varepsilon} \\
v_{-} &\rightsquigarrow_{\tau_-^j}\ \mu\left\langle \begin{matrix} \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{\chi_1}).\left\langle v_- \middle| \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{\chi_1})\right\rangle^{-} \\ \vdots \\ \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{\chi_l}).\left\langle v_- \middle| \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{\chi_l})\right\rangle^{-} \end{matrix}\right\rangle && \text{if } \overrightarrow{\chi_1}, \ldots, \overrightarrow{\chi_l} \text{ fresh w.r.t. } v_{-} \\
s_{+} &\rightsquigarrow_{\tau_+^j}\ \tilde{\mu}\left[\begin{matrix} \mathfrak{v}_1^{\tau_+^j}(\overrightarrow{\chi_1}).\left\langle \mathfrak{v}_1^{\tau_+^j}(\overrightarrow{\chi_1}) \middle| s_+\right\rangle^{+} \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{\chi_l}).\left\langle \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{\chi_l}) \middle| s_+\right\rangle^{+} \end{matrix}\right] && \text{if } \overrightarrow{\chi_1}, \ldots, \overrightarrow{\chi_l} \text{ fresh w.r.t. } s_{+}
\end{aligned}
$$

$$
\rightsquigarrow \stackrel{\text{def}}{=} \rightsquigarrow_{\tilde{\mu}} \cup \rightsquigarrow_{\tilde{\mu}} \cup \left(\bigcup_j \rightsquigarrow_{\tau_-^j}\right) \cup \left(\bigcup_j \rightsquigarrow_{\tau_+^j}\right)
$$

Figure V.1.4: The $\mathrm{L}_{\mathrm{p}}^{\rightarrow\Downarrow\Uparrow\neg_{-}\neg_{+}\otimes\parr\oplus\&1\bot 0\top}$ calculus

Figure V.1.4.a: Syntax

Positive values:

$$
\begin{aligned}
v_+, w_+ ::= {} & x^+ \\
\mid {} & (v_+ \otimes w_+) \\
\mid {} & \iota_1(v_+) \mid \iota_2(v_+) \\
\mid {} & \{v_-\} \\
\mid {} & \neg_+(s_-) \\
\mid {} & ()
\end{aligned}
$$

Positive stacks / evaluation contexts:

$$
\begin{aligned}
s_+, e_+ ::= {} & \alpha^+ \mid \tilde{\mu}x^+.c \\
\mid {} & \tilde{\mu}(x^+ \otimes y^+).c \\
\mid {} & \tilde{\mu}[\iota_1(x_1^+).c_1 | \iota_2(x_2^+).c_2] \\
\mid {} & \tilde{\mu}\{x^-\}.c \\
\mid {} & \tilde{\mu}\neg_+(\alpha^-).\alpha c \\
\mid {} & \tilde{\mu}().c \\
\mid {} & \tilde{\mu}[]
\end{aligned}
$$

Positive expressions:

$$
t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mu\alpha^+.c
$$

Negative values / expressions:

$$
\begin{aligned}
v_-, w_-, t_-, u_- ::= {} & x^- \mid \mu\alpha^-.c \\
\mid {} & \mu(x^+ \cdot \alpha^-).c \\
\mid {} & \mu(\alpha^- \parr \beta^-).c \\
\mid {} & \mu\langle(\pi_1 \cdot \alpha_1^-).c_1 | (\pi_2 \cdot \alpha_2^-).c_2\rangle \\
\mid {} & \mu\{\alpha^+\}.c \\
\mid {} & \mu\neg_-(x^+).xc \\
\mid {} & \mu\tilde{()}.c \\
\mid {} & \mu<>
\end{aligned}
$$

Negative stacks:

$$
\begin{aligned}
s_- ::= {} & \alpha^- \\
\mid {} & v_+ \cdot s_- \\
\mid {} & (s_-^1 \parr s_-^2) \\
\mid {} & \pi_1 \cdot s_- \mid \pi_2 \cdot s_- \\
\mid {} & \{s_+\} \\
\mid {} & \neg_-(v_+) \\
\mid {} & \tilde{()}
\end{aligned}
$$

Negative evaluation contexts:

$$
e_- ::= \mathsf{stk}^-(s_-) \mid \tilde{\mu}x^-.c
$$

Commands:

$$
c ::= \langle t_+ | e_+ \rangle^+ \mid \langle t_- | e_- \rangle^-
$$

Figure V.1.4.b: Operational reduction

$$\langle \mu\alpha^{\varepsilon}.c | s_{\varepsilon}\rangle^{\varepsilon} \triangleright_{\mu} \; c[s_{\varepsilon}/\alpha^{\varepsilon}]$$
$$\langle v_{\varepsilon} | \tilde{\mu}x^{\varepsilon}.c\rangle^{\varepsilon} \triangleright_{\tilde{\mu}} \; c[v_{\varepsilon}/x^{\varepsilon}]$$
$$\langle \mu(x^{+}\cdot\alpha^{-}).c | v_{+}\cdot s_{-}\rangle^{-} \triangleright_{\to} \; c[v_{+}/x^{+}, s_{-}/\alpha^{-}]$$
$$\langle \mu\{\alpha^{+}\}.c | \{s_{+}\}\rangle^{-} \triangleright_{\Uparrow} \; c[s_{+}/\alpha^{+}]$$
$$\langle \mu\neg_{-}(x^{+}).x c | \neg_{-}(v_{+})\rangle^{-} \triangleright_{\neg_{-}} \; c[v_{+}/x^{+}]$$
$$\langle \mu(\alpha^{-}\parr\beta^{-}).c | (s_{-}^{1}\parr s_{-}^{2})\rangle^{-} \triangleright_{\parr} \; c[s_{-}^{1}/\alpha^{-}, s_{-}^{2}/\beta^{-}]$$
$$\langle \mu\langle(\pi_1\cdot\alpha_1^{-}).c^{1} | (\pi_2\cdot\alpha_2^{-}).c^{2}\rangle | \pi_i\cdot s_{-}\rangle^{-} \triangleright_{\&} \; c^{i}[s_{-}/\alpha_i^{-}]$$
$$\langle \mu\tilde{()}.c | \tilde{()}\rangle^{-} \triangleright_{\bot} \; c$$
($\triangleright_{\top}$ is trivial)
$$\langle \{v_{-}\} | \tilde{\mu}\{x^{-}\}.c\rangle^{+} \triangleright_{\Downarrow} \; c[v_{-}/x^{-}]$$
$$\langle \neg_{+}(s_{-}) | \tilde{\mu}\neg_{+}(\alpha^{-}).\alpha c\rangle^{+} \triangleright_{\neg_{+}} \; c[s_{-}/\alpha^{-}]$$
$$\langle (v_{+}\otimes w_{+}) | \tilde{\mu}(x^{+}\otimes y^{+}).c\rangle^{+} \triangleright_{\otimes} \; c[v_{+}/x^{+}, w_{+}/y^{+}]$$
$$\langle \iota_i(v_{+}) | \tilde{\mu}[\iota_1(x_1^{+}).c^{1} | \iota_2(x_2^{+}).c^{2}]\rangle^{+} \triangleright_{\oplus} \; c^{i}[v_{+}/x_i^{+}]$$
$$\langle () | \tilde{\mu}().c\rangle^{+} \triangleright_{1} \; c$$
($\triangleright_{0}$ is trivial)

$$\triangleright \stackrel{\text{def}}{=} \triangleright_{\tilde{\mu}} \cup \triangleright_{\tilde{\mu}} \cup \triangleright_{\to} \cup \triangleright_{\parr} \cup \triangleright_{\&} \cup \triangleright_{\Uparrow} \cup \triangleright_{\neg_{-}} \cup \triangleright_{\bot} \cup \triangleright_{\otimes} \cup \triangleright_{\oplus} \cup \triangleright_{\Downarrow} \cup \triangleright_{\neg_{+}} \cup \triangleright_{1}$$

Figure V.1.4.c: Top-level η-expansion

$t_\varepsilon \rightsquigarrow_{\mu}$	$\mu\alpha^\varepsilon.\langle t_\varepsilon \| \alpha^\varepsilon\rangle^\varepsilon$	if α^ε fresh w.r.t. t_ε
$e_\varepsilon \rightsquigarrow_{\tilde{\mu}}$	$\tilde{\mu}x^\varepsilon.\langle x^\varepsilon \| e_\varepsilon\rangle^\varepsilon$	if x^ε fresh w.r.t. e_ε
$v_- \rightsquigarrow_{\to}$	$\mu(x^+ \cdot \alpha^-).\langle v_- \| x^+ \cdot \alpha^-\rangle^-$	if x^+ and α^- fresh w.r.t. v_-
$v_- \rightsquigarrow_{\Uparrow}$	$\mu\{\alpha^+\}.\langle v_- \| \{\alpha^+\}\rangle^-$	if α^+ fresh w.r.t. v_-
$v_- \rightsquigarrow_{\neg_-}$	$\mu\neg_-(x^+).x\langle v_- \| \neg_-(x^+)\rangle^-$	if x^+ fresh w.r.t. v_-
$v_- \rightsquigarrow_{⅋}$	$\mu(\alpha^- ⅋ \beta^-).\langle v_- \| (\alpha^- ⅋ \beta^-)\rangle^-$	if α^- and β^- fresh w.r.t. v_-
$v_- \rightsquigarrow_{\&}$	$\mu\big\langle(\pi_1 \cdot \alpha_1^-).\langle v_- \| \alpha_1^-\rangle^- \| (\pi_2 \cdot \alpha_2^-).\langle v_- \| \alpha_2^-\rangle^-\big\rangle$	if α_1^- and α_2^- fresh w.r.t. v_-
$v_- \rightsquigarrow_{\bot}$	$\mu\tilde{()}.\langle v_- \| \tilde{()}\rangle^-$	
$v_- \rightsquigarrow_{\top}$	$\mu<>$	
$s_+ \rightsquigarrow_{\Downarrow}$	$\tilde{\mu}\{x^-\}.\langle \{x^-\} \| s_+\rangle^+$	if x^- fresh w.r.t. s_+
$s_+ \rightsquigarrow_{\neg_+}$	$\tilde{\mu}\neg_+(\alpha^-).\alpha\langle \neg_+(\alpha^-) \| s_+\rangle^+$	if α^- fresh w.r.t. s_+
$s_+ \rightsquigarrow_{\otimes}$	$\tilde{\mu}(x^+ \otimes y^+).\langle (x^+ \otimes y^+) \| s_+\rangle^+$	if x^+ and y^+ fresh w.r.t. s_+
$s_+ \rightsquigarrow_{\oplus}$	$\tilde{\mu}\big[\iota_1(x_1^+).\langle \iota_1(x_1^+) \| s_+\rangle^+ \| \iota_2(x_2^+).\langle \iota_2(x_2^+) \| s_+\rangle^+\big]$	if x_1^+ and x_2^+ fresh w.r.t. s_+
$s_+ \rightsquigarrow_{1}$	$\tilde{\mu}().\langle () \| s_+\rangle^+$	
$s_+ \rightsquigarrow_{0}$	$\tilde{\mu}[]$	

$$\rightsquigarrow \overset{\text{def}}{=} \rightsquigarrow_{\tilde{\mu}} \cup \rightsquigarrow_{\tilde{\mu}} \cup \rightsquigarrow_{\to} \cup \rightsquigarrow_{\Uparrow} \cup \rightsquigarrow_{\neg_-} \cup \rightsquigarrow_{⅋} \cup \rightsquigarrow_{\&} \cup \rightsquigarrow_{\bot} \cup \rightsquigarrow_{\top} \cup \rightsquigarrow_{\Downarrow} \cup \rightsquigarrow_{\neg_+} \cup \rightsquigarrow_{\otimes} \cup \rightsquigarrow_{\oplus} \cup \rightsquigarrow_{1} \cup \rightsquigarrow_{0}$$

Remark V.1.2

Note that the coercions val^+ (resp. stk^-) are what ensures that the syntax is closed under η-expansions. Indeed, if we removed val^+ (resp. stk^-), then we would have

$$v_+ \fatsemi_{\mu} \mu\alpha^+.\langle v_+ | \alpha^+ \rangle^+ \quad (\text{resp. } s_- \fatsemi_{\tilde{\mu}} \tilde{\mu}x^-.\langle x^- | s_- \rangle^-)$$

and hence

$$(v_+ \otimes w_+) \dashv\!\fatsemi ((\mu\alpha^+.\langle v_+ | \alpha^+ \rangle^+) \otimes w_+) \quad (\text{resp. } v_+ \cdot s_- \dashv\!\fatsemi v_+ \cdot (\tilde{\mu}x^-.\langle x^- | s_- \rangle))$$

With the coercions, this problem disapears because v_+ (resp. s_-) can not be η-expanded on its own, and $(\mathsf{val}^+(v_+) \otimes w_+)$ (resp. $v_+ \cdot \mathsf{stk}^-(s_-)$) is not within the syntax[a].

[a] Of course, one can also fix this by allowing expressions (resp. evaluation contexts) in value and stack constructors, see .

Remark V.1.3

We could also add coercions val^- (resp. stk^+), i.e. define negative expressions (resp. positive evaluation contexts) by

$$t_-, u_- \mathrel{::=} \mathsf{val}^-(v_-) \quad (\text{resp. } e_+ \mathrel{::=} \mathsf{stk}^+(s_+))$$

While this could be useful in future calculi, here it would be completely superfluous (because these coercions would be bijections), while requiring duplications in the definition of $\fatsemi$:

$$t_- \fatsemi_{\mu} \mu\alpha^-.\langle t_- | \alpha^- \rangle^- \quad (\text{resp. } e_+ \fatsemi \tilde{\mu}x^+.\langle x^+ | e_+ \rangle^+)$$

would need to be replaced by

$$v_- \fatsemi_{\mu} \mu\alpha^-.\langle v_- | \alpha^- \rangle^- \quad (\text{resp. } s_+ \fatsemi \tilde{\mu}x^+.\langle x^+ | s_+ \rangle^+)$$

to ensure that e.g.

$$\{v_-\} \dashv\!\fatsemi_{\mu} \{\mu\alpha^-.\langle v_- | \alpha^- \rangle^-\} \quad (\text{resp. } \{s_+\} \fatsemi \{\tilde{\mu}x^+.\langle x^+ | s_+ \rangle^+\})$$

while

$$t_+ \fatsemi_{\mu} \mu\alpha^+.\langle t_+ | \alpha^+ \rangle^+ \quad (\text{resp. } e_- \fatsemi \tilde{\mu}x^-.\langle x^- | e_- \rangle^-)$$

can not be made to act on v_+ (resp. s_-) if one wants η-expansion to preserve the syntax.

Normal forms, clashes and waiting commands

There are several kinds of $\triangleright$-normal forms:

Definition V.1.4

A command c is said to be:

- a *clash* when it is of one of the following shapes:

$$c = \left\langle \mathfrak{v}_k^{\tau_+^{j_1}}(\vec{a}) \middle| \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^{j_2}}(\vec{\chi_1}).c_1 | \dots | \mathfrak{v}_l^{\tau_+^{j_2}}(\vec{\chi_l}).c_l\right] \right\rangle^+ \quad \text{with } \tau_+^{j_1} \neq \tau_+^{j_2}\text{, or}$$

$$c = \left\langle \mu\left\langle \mathfrak{s}_1^{\tau_-^{j_1}}(\vec{\chi_1}).c_1 | \dots | \mathfrak{s}_l^{\tau_-^{j_1}}(\vec{\chi_l}).c_l \right\rangle \middle| \mathfrak{s}_k^{\tau_-^{j_2}}(\vec{a}) \right\rangle^- \quad \text{with } \tau_-^{j_1} \neq \tau_-^{j_2}$$

- *waiting* when it is of one of the following shapes:

$$c = \langle x^\varepsilon | \alpha^\varepsilon \rangle^\varepsilon, \quad c = \left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{a}) \middle| \alpha^+ \right\rangle^+, \quad c = \left\langle x^+ \middle| \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\vec{\chi_1}).c_1 | \dots | \mathfrak{v}_l^{\tau_+^j}(\vec{\chi_l}).c_l\right] \right\rangle^+,$$

$$c = \left\langle x^- \middle| \mathfrak{s}_k^{\tau_-^j}(\vec{a}) \right\rangle^-, \quad \text{or} \quad c = \left\langle \mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\vec{\chi_1}).c_1 | \dots | \mathfrak{s}_l^{\tau_-^j}(\vec{\chi_l}).c_l \right\rangle \middle| \alpha^- \right\rangle$$

Example V.1.5

The commands

$$\langle \iota_1(\upsilon_+) | \tilde{\mu}(x^+ \otimes y^+).c \rangle^+ \quad \text{and} \quad \left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{a}) \middle| \tilde{\mu}[] \right\rangle^+$$

are clashes.

These two definitions cover exactly all $\triangleright$-normal commands:

Fact V.1.6

A command is $\triangleright$-normal if and only if it is either a clash or a waiting command, and those two cases are mutually exclusive.

Proof

By case analysis on the command.

We now look at the effect of disubstitutions on $\triangleright$-normal commands. For clashes, disubstitutions have no effect:

Fact V.1.7

The set of clashes is disubstitutive: for any clash c and disubstitution φ, the command $c[\varphi]$ is a clash.

Proof

By case analysis on c.

Waiting commands are waiting on a particular variable, and until this variable is substituted by a non-variable, they remain waiting:

Definition V.1.8

A command c is said to be:

- *waiting for* x^- if it is of the shape $c = \langle x^- | \alpha^- \rangle^-$ or $c = \left\langle x^- \middle| \mathfrak{s}_k^{\tau_-^j}(\vec{a}) \right\rangle^-$;
- *waiting for* α^+ if it is of the shape $c = \langle x^+ | \alpha^+ \rangle^+$ or $c = \left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{a}) \middle| \alpha^+ \right\rangle^+$;
- *waiting for* α^- if it is of the shape $c = \left\langle \mu \left\langle \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{\chi_1}).c_1 | \ldots | \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{\chi_l}).c_l \right\rangle \middle| \alpha^- \right\rangle^-$;
- *waiting for* x^+ if it is of the shape $c = \left\langle x^+ \middle| \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{\chi_1}).c_1 | \ldots | \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{\chi_l}).c_l \right] \right\rangle^+$.

Fact V.1.9

If c is waiting for χ then there exists an argument a such that $c[a/\chi]$ is reducible.

Proof

Defining a as $\mu\star^-.c$, $\tilde{\mu}x^+.c$, $\mathfrak{s}_k^{\tau_-^j}(\vec{a})$ or $\mathfrak{v}_k^{\tau_+^j}(\vec{a})$ works when χ is x^-, α^+, α^- and x^+ respectively (with τ_ε^j being the type former of the displayed μ<...> or $\tilde{\mu}[\ldots]$).

Fact V.1.10

If c is waiting for χ then for any disubstitution φ such that $\varphi(\chi)$ is a variable, $c[\varphi]$ is waiting for $\varphi(\chi)$.

Proof

By case analysis on c.

Remark V.1.11

If c is waiting for χ then given a non-variable a, $c[a/\chi]$ may be:

- reducible, e.g.
$$\left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{a}) \middle| \alpha^+ \right\rangle^+ [\tilde{\mu}x^+.c/\alpha^+] = \left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{a}) \middle| \tilde{\mu}x^+.c \right\rangle^+$$
is reducible;
- a clash, e.g.
$$\left\langle \mathfrak{v}_k^{\tau_+^{j_1}}(\vec{a}) \middle| \alpha^+ \right\rangle^+ \left[\tilde{\mu}\left[\mathfrak{v}_1^{\tau_-^{j_2}}(\chi_1).c_1 | \ldots | \mathfrak{v}_l^{\tau_-^{j_2}}(\chi_l).c_l \right] / \alpha^+ \right]$$

is a clash when $j_1 \neq j_2$;

- waiting for another variable, e.g. $\langle x^- | \alpha^- \rangle^-$ is waiting for x^- and

$$\langle x^- | \alpha^- \rangle^- [\mu(y^+ \cdot \beta^-).c / x^-] = \langle \mu(y^+ \cdot \beta^-).c | \alpha^- \rangle^-$$

  is waiting for α^-.

This last case could make us want to say that $\langle x^- | \alpha^- \rangle^-$ waits for both x^- and α^- but this is not really the case because

$$\langle x^- | \alpha^- \rangle^- [\mu\beta^-.c / x^-] = \langle \mu\beta^-.c | \alpha^- \rangle^-$$

is not waiting for α^- in general (e.g. it $\triangleright$-diverges whenever c does).

Definition V.1.12

A command c is said to:

- *converge to* c', written $c \Downarrow c'$ or $c \triangleright^{\circledast} c'$, when $c \triangleright^* c' \not\triangleright$;
- *converge*, written $c \Downarrow$ or $c \triangleright^{\circledast}$, when there exists some c' it converges to;
- *diverge*, written $c \Uparrow$ or $c \triangleright^{\omega}$, when there is an infinite reduction sequence

$$c \triangleright c' \triangleright c'' \triangleright \ldots$$

  starting at c.

There are three possible outcomes for a command:

Fact V.1.13

Any command either diverges, converges to a clash, or converges to a waiting command, and those three cases are mutually exclusive.

Proof

By determinism of $\triangleright$, it either converges or diverges, and by Fact V.1.6, the $\triangleright$-normal command it converges to is either a clash or a waiting command.

Properties

Just like we distinguish substitution from disubstitutions, we distinguish substitutivity from disubstitutivity:

Definition V.1.14

A reduction $\rightsquigarrow$ of $\mathrm{L}_\mathrm{p}^{\vec{\tau}}$ is said to be *substitutive* (resp. *disubstitutive*) when for any terms t and $\vec{t}$, and substitution σ (resp. disubstitution φ), we have

$$t \rightsquigarrow t' \quad\Rightarrow\quad t[\sigma] \rightsquigarrow t'[\sigma] \quad (\text{resp. } t \rightsquigarrow t' \quad\Rightarrow\quad t[\varphi] \rightsquigarrow t'[\varphi])$$

The properties of the reductions are summarized in Figure V.1

Table V.1.: Properties of reductions in the $\mathrm{L}_\mathrm{p}^{\vec{\tau}}$ calculus

	$\triangleright$	$\mathrel{-\!\triangleright}$	$\overset{\circ}{\longrightarrow}\!\!\triangleright$	$\dashv$
Substitutive	✓	✓	✓	✓
Disubstitutive	✓	✓	✓	✓
Deterministic	✓	✗	✗	✗
Confluent	✓	✓	✓	✓
Postpones after $\triangleright$	✓	✓	✓	✓

The proofs of all of these properties are either trivial or routine, and are therefore relegated to ⚠ in the appendix. Confluence and postponement are proven in a standard way using a parallel reduction $\Rightarrow\!\!\triangleright$ [Tak95; Bar84]. The only slightly non-standard choice is the definition of $\Rightarrow\!\!\triangleright$:

Remark V.1.15

The most common definitions of the parallel reduction $\Rightarrow\!\!\triangleright$ contain rules two kinds of rules: those that simply combine reduction sequences on subterms such as

$$\frac{t_\varepsilon \Rightarrow\!\!\triangleright t'_\varepsilon \qquad e_\varepsilon \Rightarrow\!\!\triangleright e'_\varepsilon}{\langle t_\varepsilon | e_\varepsilon \rangle^\varepsilon \Rightarrow\!\!\triangleright \langle t'_\varepsilon | e'_\varepsilon \rangle^\varepsilon}\;, \qquad \frac{v_+ \Rightarrow\!\!\triangleright v'_+}{\iota_i(v_+) \Rightarrow\!\!\triangleright \iota_i(v_+)}\;, \qquad \text{and} \qquad \frac{c_1 \Rightarrow\!\!\triangleright c'_1 \qquad c_2 \Rightarrow\!\!\triangleright c'_2}{\tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \Rightarrow\!\!\triangleright \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c'_1 \\ \iota_2(x_2^+).\, c'_2\end{bmatrix}}$$

and those that add a reduction step such as

$$\frac{v_+ \Rightarrow\!\!\triangleright v'_+ \qquad c_1 \Rightarrow\!\!\triangleright c'_1 \qquad c_2 \Rightarrow\!\!\triangleright c'_2}{\left\langle \iota_i(v_+) \middle| \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \right\rangle^+ \Rightarrow\!\!\triangleright c'_i[v'_+/x_i^+]}$$

Let $\overset{\circ}{\Rightarrow}\!\!\triangleright$ be the restriction of $\Rightarrow\!\!\triangleright$ defined by $t \overset{\circ}{\Rightarrow}\!\!\triangleright t'$ meaning that there exists a derivation of $t \Rightarrow\!\!\triangleright t'$ whose last rule is not a step rule. With the usual definition of $\Rightarrow\!\!\triangleright$, the

[Tak95] "Parallel Reductions in λ-Calculus", Takahashi, 1995
[Bar84] *The lambda calculus: its syntax and semantics*, Barendregt, 1984

two following rules are admissible:

$$\frac{t \mathrel{\dot{\Rrightarrow}} t'}{t \Rrightarrow t'} \quad \text{and} \quad \frac{t \mathrel{\dot{\Rrightarrow}} t' \qquad t' \triangleright t''}{t \Rrightarrow t''}$$

For $L_{\mathrm{p}}^{\vec{\tau}}$, since there are four kinds of $\triangleright$ reductions (namely $\triangleright_{\mu}$, $\triangleright_{\tilde{\mu}}$, $\triangleright_{\tau_-^j}$, and $\triangleright_{\tau_+^j}$), the usual definition of $\Rrightarrow$ has four step rules. It is hence slightly easier to define $\Rrightarrow$ and $\dot{\Rrightarrow}$ by mutual induction by adding the two rules above in the definition, removing the step rules, and strenghening the other rules to remember that no step was taken at the top-level, e.g.

$$\frac{t_\varepsilon \Rrightarrow t'_\varepsilon \qquad e_\varepsilon \Rrightarrow e'_\varepsilon}{\langle t_\varepsilon | e_\varepsilon \rangle^\varepsilon \mathrel{\dot{\Rrightarrow}} \langle t'_\varepsilon | e'_\varepsilon \rangle^\varepsilon}\ , \quad \frac{v_+ \Rrightarrow v'_+}{\iota_i(v_+) \mathrel{\dot{\Rrightarrow}} \iota_i(v_+)}\ , \quad \text{and} \quad \frac{c_1 \Rrightarrow c'_1 \qquad c_2 \Rrightarrow c'_2}{\tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \mathrel{\dot{\Rrightarrow}} \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c'_1 \\ \iota_2(x_2^+).\, c'_2\end{bmatrix}}$$

The usual step rules are then derivable, e.g.

$$\dfrac{\dfrac{\dfrac{\dfrac{v_+ \Rrightarrow v'_+}{\iota_i(v_+) \mathrel{\dot{\Rrightarrow}} \iota_i(v'_+)}}{\iota_i(v_+) \Rrightarrow \iota_i(v'_+)} \qquad \dfrac{\dfrac{c_1 \Rrightarrow c'_1 \qquad c_2 \Rrightarrow c'_2}{\tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \mathrel{\dot{\Rrightarrow}} \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c'_1 \\ \iota_2(x_2^+).\, c'_2\end{bmatrix}}}{\tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \Rrightarrow \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c'_1 \\ \iota_2(x_2^+).\, c'_2\end{bmatrix}}}{\left\langle \iota_i(v_+) \middle| \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \right\rangle^+ \mathrel{\dot{\Rrightarrow}} \left\langle \iota_i(v'_+) \middle| \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c'_1 \\ \iota_2(x_2^+).\, c'_2\end{bmatrix} \right\rangle^+} \qquad \dfrac{}{\left\langle \iota_i(v'_+) \middle| \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c'_1 \\ \iota_2(x_2^+).\, c'_2\end{bmatrix} \right\rangle^+ \triangleright c'_i[v'_+/x_i^+]}}{\left\langle \iota_i(v_+) \middle| \tilde{\mu}\begin{bmatrix}\iota_1(x_1^+).\, c_1 \\ \iota_2(x_2^+).\, c_2\end{bmatrix} \right\rangle^+ \Rrightarrow c'_i[v'_+/x_i^+]}$$

This alternative definition of $\Rrightarrow$ also has the advantage of disentangling the part of $\Rrightarrow$ that depends on $\triangleright$ from the rest, which makes $\Rrightarrow$ parametric in $\triangleright$.

V.1.3. Well-typed and well-polarized terms

Well-typed terms

Simply typed $L_{\mathrm{p}}^{\vec{\tau}}$ is described in Figure V.1.5. Each type former τ_+^j (resp. τ_-^j) has logic rules $\left(\vdash \mathfrak{v}_k^{\tau_+^j}\right)$ (resp. $\left(\mathfrak{s}_k^{\tau_-^j} \vdash\right)$) that introduce its constructors, and $\left(\tau_+^j \vdash\right)$(resp. $\left(\vdash \tau_-^j\right)$) that introduces the correspond pattern match. Of course, in these logic rules, sequence of types given to τ_ε^j in the conclusions depends on the type in the premises, and if one wants the subformula property to hold, on should require $\{\vec{A}\} \subseteq \{\vec{B}\}$ in $\left(\mathfrak{s}_k^{\tau_-^j} \vdash\right)$ and $\left(\vdash \mathfrak{v}_k^{\tau_+^j}\right)$ and $\{\vec{A}, \vec{B}\} \subseteq \{\vec{C}\}$ in $\left(\vdash \tau_-^j\right)$ and $\left(\tau_+^j \vdash\right)$.

Figure V.1.5: Simply typed $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$

Figure V.1.5.a: Core rules

$$\frac{}{x^\varepsilon : A_\varepsilon \vdash \underline{x^\varepsilon : A_\varepsilon} \mid}\ (\vdash\textsc{ax}) \qquad \frac{}{\mid \underline{\alpha^\varepsilon : A_\varepsilon} \vdash \alpha^\varepsilon : A_\varepsilon}\ (\textsc{ax}\vdash)$$

$$\frac{c : (\Gamma \vdash \alpha^\varepsilon : A_\varepsilon, \Delta)}{\Gamma \vdash \underline{\mu\alpha^\varepsilon.c : A_\varepsilon} \mid \Delta}\ (\vdash\mu) \qquad \frac{c : (\Gamma, x^\varepsilon : A_\varepsilon \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}x^\varepsilon.c : A_\varepsilon} \vdash \Delta}\ (\tilde{\mu}\vdash)$$

$$\frac{\Gamma_1 \vdash \underline{t_\varepsilon : A_\varepsilon} \mid \Delta_1 \qquad \Gamma_2 \mid \underline{e_\varepsilon : A_\varepsilon} \vdash \Delta_2}{\langle t_\varepsilon | e_\varepsilon \rangle^\varepsilon : (\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2)}\ (\textsc{cut})$$

Figure V.1.5.b: Structural rules (commands)

$$\frac{c : (\Gamma \vdash \Delta)}{c : (\Gamma \vdash \alpha^\varepsilon : A_\varepsilon, \Delta)}\ (\vdash\textsc{w}c) \qquad \frac{c : (\Gamma \vdash \alpha_1^\varepsilon : A_\varepsilon, \alpha_2^\varepsilon : A_\varepsilon, \Delta)}{c[\beta^\varepsilon/\alpha_1^\varepsilon, \beta^\varepsilon/\alpha_2^\varepsilon] : (\Gamma \vdash \beta^\varepsilon : A_\varepsilon, \Delta)}\ (\vdash\textsc{c}c)$$

$$\frac{c : (\Gamma \vdash \Delta)}{c : (\Gamma, x^\varepsilon : A_\varepsilon \vdash \Delta)}\ (\textsc{w}c\vdash) \qquad \frac{c : (\Gamma, x_1^\varepsilon : A_\varepsilon, x_2^\varepsilon : A_\varepsilon \vdash \Delta)}{c[y^\varepsilon/x_1^\varepsilon, y^\varepsilon/x_2^\varepsilon] : (\Gamma, y^\varepsilon : A_\varepsilon \vdash \Delta)}\ (\textsc{c}c\vdash)$$

$$\frac{c : (\Gamma \vdash \Delta_1, \alpha_1^\varepsilon : A_\varepsilon, \alpha_2^\varepsilon : A_\varepsilon, \Delta_2)}{c : (\Gamma \vdash \Delta_1, \alpha_2^\varepsilon : A_\varepsilon, \alpha_1^\varepsilon : A_\varepsilon, \Delta_2)}\ (\vdash\textsc{p}c) \qquad \frac{c : (\Gamma_1, x_1^\varepsilon : A_\varepsilon, x_2^\varepsilon : A_\varepsilon, \Gamma_2 \vdash \Delta)}{c : (\Gamma_1, x_2^\varepsilon : A_\varepsilon, x_1^\varepsilon : A_\varepsilon, \Gamma_2 \vdash \Delta)}\ (\textsc{p}c\vdash)$$

Figure V.1.5.c: Structural rules (expressions)

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta}{\Gamma \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \alpha^\varepsilon : B_\varepsilon, \Delta}\ (\vdash\textsc{w}t) \qquad \frac{\Gamma \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \alpha_1^\varepsilon : B_\varepsilon, \alpha_2^\varepsilon : B_\varepsilon, \Delta}{\Gamma \vdash \underline{t_{\varepsilon_0}[\beta^\varepsilon/\alpha_1^\varepsilon, \beta^\varepsilon/\alpha_2^\varepsilon] : A_{\varepsilon_0}} \mid \beta^\varepsilon : B_\varepsilon, \Delta}\ (\vdash\textsc{c}t)$$

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta}{\Gamma, x^\varepsilon : B_\varepsilon \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta}\ (\textsc{w}t\vdash) \qquad \frac{\Gamma, x_1^\varepsilon : B_\varepsilon, x_2^\varepsilon : B_\varepsilon \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta}{\Gamma, x^\varepsilon : B_\varepsilon \vdash \underline{t_{\varepsilon_0}[x^\varepsilon/x_1^\varepsilon, x^\varepsilon/x_2^\varepsilon] : A_{\varepsilon_0}} \mid \Delta}\ (\textsc{c}t\vdash)$$

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta_1, \alpha_1^\varepsilon : B_\varepsilon, \alpha_2^\varepsilon : B_\varepsilon, \Delta_2}{\Gamma \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta_1, \alpha_2^\varepsilon : B_\varepsilon, \alpha_1^\varepsilon : B_\varepsilon, \Delta_2}\ (\vdash\textsc{p}t) \qquad \frac{\Gamma_1, x_1^\varepsilon : B_\varepsilon, x_2^\varepsilon : B_\varepsilon, \Gamma_2 \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta}{\Gamma_1, x_2^\varepsilon : B_\varepsilon, x_1^\varepsilon : B_\varepsilon, \Gamma_2 \vdash \underline{t_{\varepsilon_0} : A_{\varepsilon_0}} \mid \Delta}\ (\textsc{p}t\vdash)$$

Figure V.1.5.d: Structural rules (evaluation contexts)

$$\frac{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}}{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \alpha^{\varepsilon} : B_{\varepsilon}, \mathit{\Delta}} \ (\vdash\mathrm{we}) \qquad \frac{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \alpha_1^{\varepsilon} : B_{\varepsilon}, \alpha_2^{\varepsilon} : B_{\varepsilon}, \mathit{\Delta}}{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0}[\beta^{\varepsilon}/\alpha_1^{\varepsilon}, \beta^{\varepsilon}/\alpha_2^{\varepsilon}] : A_{\varepsilon_0}} \vdash \beta^{\varepsilon} : B_{\varepsilon}, \mathit{\Delta}} \ (\vdash\mathrm{ce})$$

$$\frac{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}}{\mathit{\Gamma}, x^{\varepsilon} : B_{\varepsilon} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}} \ (\mathrm{we}\vdash) \qquad \frac{\mathit{\Gamma}, x_1^{\varepsilon} : B_{\varepsilon}, x_2^{\varepsilon} : B_{\varepsilon} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}}{\mathit{\Gamma}, x^{\varepsilon} : B_{\varepsilon} \mid \underline{e_{\varepsilon_0}[x^{\varepsilon}/x_1^{\varepsilon}, x^{\varepsilon}/x_2^{\varepsilon}] : A_{\varepsilon_0}} \vdash \mathit{\Delta}} \ (\mathrm{ce}\vdash)$$

$$\frac{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}_1, \alpha_1^{\varepsilon} : B_{\varepsilon}, \alpha_2^{\varepsilon} : \varepsilon, \mathit{\Delta}_2}{\mathit{\Gamma} \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}_1, \alpha_2^{\varepsilon} : B_{\varepsilon}, \alpha_1^{\varepsilon} : \varepsilon, \mathit{\Delta}_2} \ (\vdash\mathrm{Pe}) \qquad \frac{\mathit{\Gamma}_1, x_1^{\varepsilon} : B_{\varepsilon}, x_2^{\varepsilon} : B_{\varepsilon}, \mathit{\Gamma}_2 \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}}{\mathit{\Gamma}_1, x_2^{\varepsilon} : B_{\varepsilon}, x_1^{\varepsilon} : B_{\varepsilon}, \mathit{\Gamma}_2 \mid \underline{e_{\varepsilon_0} : A_{\varepsilon_0}} \vdash \mathit{\Delta}} \ (\mathrm{Pe}\vdash)$$

Figure V.1.5.e: General shape of logic rules

$$\frac{\begin{array}{ccc} \mathit{\Gamma}_1 \vdash \underline{v^1_{\varepsilon_1} : A^1_{\varepsilon_1}} \mid \mathit{\Delta}_1 & \dots & \mathit{\Gamma}_q \vdash \underline{v^q_{\varepsilon_q} : A^q_{\varepsilon_q}} \mid \mathit{\Delta}_q \\ \mathit{\Gamma}_{q+1} \mid \underline{s^1_{\varepsilon_{q+1}} : A^{q+1}_{\varepsilon_{q+1}}} \vdash \mathit{\Delta}_{q+1} & \dots & \mathit{\Gamma}_{q+r} \mid \underline{s^r_{\varepsilon_{q+r}} : A^{q+r}_{\varepsilon_{q+r}}} \vdash \mathit{\Delta}_{q+r} \end{array}}{\mathit{\Gamma}_1, \dots, \mathit{\Gamma}_{q+r} \mid \underline{\mathfrak{s}_k^{\tau_-^j}\left(v^1_{\varepsilon_1}, \dots, v^q_{\varepsilon_q}, s^1_{\varepsilon_{q+1}}, \dots, s^r_{\varepsilon_{q+r}}\right) : \tau_-^j(\vec{B})} \vdash \mathit{\Delta}_1, \dots, \mathit{\Delta}_{q+r}} \ \left(\mathfrak{s}_k^{\tau_-^j}\vdash\right)$$

$$\frac{c_1 : \left(\mathit{\Gamma}, \vec{x_1} : \vec{A^1} \vdash \vec{\alpha_1} : \vec{B^1}, \mathit{\Delta}\right) \quad \dots \quad c_l : \left(\mathit{\Gamma}, \vec{x_l} : \vec{A^l} \vdash \vec{\alpha_l} : \vec{B^l}, \mathit{\Delta}\right)}{\mathit{\Gamma} \vdash \underline{\mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\vec{x_1}, \vec{\alpha_1}).c_1 | \dots | \mathfrak{s}_l^{\tau_-^j}(\vec{x_l}, \vec{\alpha_l}).c_l \right\rangle : \tau_-^j(\vec{C})} \mid \mathit{\Delta}} \ \left(\vdash\tau_-^j\right)$$

$$\frac{\begin{array}{ccc} \mathit{\Gamma}_1 \vdash \underline{v^1_{\varepsilon_1} : A^1_{\varepsilon_1}} \mid \mathit{\Delta}_1 & \dots & \mathit{\Gamma}_q \vdash \underline{v^q_{\varepsilon_q} : A^q_{\varepsilon_q}} \mid \mathit{\Delta}_q \\ \mathit{\Gamma}_{q+1} \mid \underline{s^1_{\varepsilon_{q+1}} : A^{q+1}_{\varepsilon_{q+1}}} \vdash \mathit{\Delta}_{q+1} & \dots & \mathit{\Gamma}_{q+r} \mid \underline{s^r_{\varepsilon_{q+r}} : A^{q+r}_{\varepsilon_{q+r}}} \vdash \mathit{\Delta}_{q+r} \end{array}}{\mathit{\Gamma}_1, \dots, \mathit{\Gamma}_q \vdash \underline{\mathfrak{v}_k^{\tau_+^j}\left(v^1_{\varepsilon_1}, \dots, v^q_{\varepsilon_q}, s^1_{\varepsilon_{q+1}}, \dots, s^r_{\varepsilon_{q+r}}\right) : \tau_+^j(\vec{B})} \mid \mathit{\Delta}_1, \dots, \mathit{\Delta}_q} \ \left(\vdash\mathfrak{v}_k^{\tau_+^j}\right)$$

$$\frac{c_1 : \left(\mathit{\Gamma}, \vec{x_1} : \vec{A^1} \vdash \vec{\alpha_1} : \vec{B^1}, \mathit{\Delta}\right) \quad \dots \quad c_l : \left(\mathit{\Gamma}, \vec{x_l} : \vec{A^l} \vdash \vec{\alpha_l} : \vec{B^l}, \mathit{\Delta}\right)}{\mathit{\Gamma} \mid \underline{\tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\vec{x_1}, \vec{\alpha_1}).c_1 | \dots | \mathfrak{v}_l^{\tau_+^j}(\vec{x_l}, \vec{\alpha_l}).c_l \right] : \tau_+^j(\vec{C})} \vdash \mathit{\Delta}} \ \left(\tau_+^j\vdash\right)$$

Figure V.1.5.f: Logic rules for multiplicative types

$$\frac{c : (\Gamma, x^+ : A_+ \vdash \alpha^- : B_-, \Delta)}{\Gamma \vdash \mu(x^+ \cdot \alpha^-).c : A_+ \to B_- \mid \Delta}\ (\vdash\to) \qquad \frac{\Gamma_1 \vdash v_+ : A_+ \mid \Delta_1 \qquad \Gamma_2 \mid s_- : B_- \vdash \Delta_2}{\Gamma_1, \Gamma_2 \mid v_+ \cdot s_- : A_+ \to B_- \vdash \Delta_1, \Delta_2}\ (\to\vdash)$$

$$\frac{c : (\Gamma \vdash \alpha^- : A_-, \beta^- : B_-, \Delta)}{\Gamma \vdash \mu(\alpha^- ⅋ \beta^-).c : A_- \,\&\, B_- \mid \Delta}\ (\vdash ⅋) \qquad \frac{\Gamma_1 \mid s_-^1 : A_-^1 \vdash \Delta_1 \qquad \Gamma_2 \mid s_-^2 : A_-^2 \vdash \Delta_2}{\Gamma_1, \Gamma_2 \mid (s_-^1 ⅋ s_-^2) : A_-^1 \,\&\, A_-^2 \vdash \Delta_1, \Delta_2}\ (⅋\vdash)$$

$$\frac{\Gamma_1 \vdash v_+^1 : A_+^1 \mid \Delta_1 \qquad \Gamma_2 \vdash v_+^2 : A_+^2 \mid \Delta_2}{\Gamma_1, \Gamma_2 \vdash (v_+^1 \otimes v_+^2) : A_+ \otimes B_+ \mid \Delta_1, \Delta_2}\ (\vdash\otimes) \qquad \frac{c : (\Gamma, x^+ : A_+, y^+ : B_+ \vdash \Delta)}{\Gamma \mid \tilde{\mu}(x^+ \otimes y^-).c : A_+ \otimes B_+ \vdash \Delta}\ (\otimes\vdash)$$

$$\frac{c : (\Gamma \vdash \Delta)}{\Gamma \vdash \mu\tilde{()}.c : \bot \mid \Delta}\ (\vdash\bot) \qquad \frac{}{\mid \tilde{()} : \bot \vdash}\ (\bot\vdash)$$

$$\frac{}{\vdash () : 1 \mid}\ (1\vdash) \qquad \frac{c : (\Gamma \vdash \Delta)}{\Gamma \mid \tilde{\mu}().c : 1 \vdash \Delta}\ (\vdash 1)$$

Figure V.1.5.g: Logic rules for additive types

$$\frac{c_1 : (\Gamma \vdash \alpha_1^- : A_-^1, \Delta) \qquad c_2 : (\Gamma \vdash \alpha_2^- : A_-^2, \Delta)}{\Gamma \vdash \mu\langle(\pi_1 \cdot \alpha_1^-).c_1 | (\pi_2 \cdot \alpha_2^-).c_2\rangle : A_-^1 \,\&\, A_-^2 \mid \Delta}\ (\vdash\&) \qquad \frac{\Gamma \mid s_- : A_-^i \vdash \Delta}{\Gamma \mid \pi_i \cdot s_- : A_-^1 \,\&\, A_-^2 \vdash \Delta}\ (\&\vdash)$$

$$\frac{\Gamma \vdash v_+ : A_+^i \mid \Delta}{\Gamma \vdash \iota_i(v_+) : A_+^1 \oplus A_+^2 \mid \Delta}\ (\vdash\oplus) \qquad \frac{c_1 : (\Gamma, x_1^+ : A_+^1 \vdash \Delta) \qquad c_2 : (\Gamma, x_2^+ : A_+^2 \vdash \Delta)}{\Gamma \mid \tilde{\mu}[\iota_1(x_1^+).c_1 | \iota_2(x_2^+).c_2] : A_+^1 \oplus A_+^2 \vdash \Delta}\ (\oplus\vdash)$$

$$\frac{}{\Gamma \vdash \mu\langle\rangle : \top \mid \Delta}\ (\vdash\top)$$

(No ($\top\vdash$) rule)

(No ($\vdash 0$) rule)

$$\frac{}{\Gamma \mid \tilde{\mu}[] : 0 \vdash \Delta}\ (0\vdash)$$

Figure V.1.5.h: Logic rules for shifts

$$\frac{c : (\Gamma \vdash \alpha^+ : A_+, \Delta)}{\Gamma \vdash \underline{\mu\{\alpha^+\}.c : \Uparrow A_+} \mid \Delta}\ (\vdash\Uparrow) \qquad \frac{\Gamma \mid \underline{s_+ : A_+} \vdash \Delta}{\Gamma \mid \underline{\{s_+\} : \Uparrow A_+} \vdash \Delta}\ (\Uparrow\vdash)$$

$$\frac{\Gamma \vdash \underline{v_- : A_-} \mid \Delta}{\Gamma \vdash \underline{\{v_-\} : \Downarrow A_-} \mid \Delta}\ (\vdash\Downarrow) \qquad \frac{c : (\Gamma, x^- : A_- \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}\{x^-\}.c : \Downarrow A_-} \vdash \Delta}\ (\Downarrow\vdash)$$

Figure V.1.5.i: Logic rules for negations

$$\frac{c : (\Gamma, x^+ : A_+ \vdash \Delta)}{\Gamma \vdash \underline{\mu\neg_-(x^+).xc : \neg_-(A_+)} \mid \Delta}\ (\vdash\neg_-) \qquad \frac{\Gamma \vdash \underline{v_+ : A_+} \mid \Delta}{\Gamma \mid \underline{\neg_-(v_+) : \neg_-(A_+)} \vdash \Delta}\ (\neg_-\vdash)$$

$$\frac{\Gamma \mid \underline{s_- : A_-} \vdash \Delta}{\Gamma \vdash \underline{\neg_+(s_-) : \neg_+(A_-)} \mid \Delta}\ (\vdash\neg_+) \qquad \frac{c : (\Gamma \vdash \alpha^- : A_-, \Delta)}{\Gamma \mid \underline{\tilde{\mu}\neg_+(\alpha^-).\alpha c : \neg_+(A_-)} \vdash \Delta}\ (\neg_+\vdash)$$

Definition V.1.16

An expression t_ε (resp. evaluation context e_ε) of $L_p^{\vec{\tau}}$ is said to be *of type* A_ε when there exists a derivation of

$$\Gamma \vdash \underline{t_\varepsilon : A_\varepsilon} \mid \Delta \quad (\text{resp. } \Gamma \mid \underline{e_\varepsilon : A_\varepsilon} \vdash \Delta)$$

in the type sytem described in Figure V.1.5, and *well-typed* when it is of type A_ε for some type A_ε. A command c is said to be *well-typed* when there exists a derivation of

$$c : (\Gamma \vdash \Delta)$$

in the type sytem described in Figure V.1.5. A term is said to be *ill-typed* when it is not well-typed.

Alternative presentations

Presentations of simply typed L-calculi sometimes use a syntax of preterms (i.e. possibly ill-typed terms) that treats polarities less rigidly, e.g. the syntax of preterms of [CurFioMun16, Figure 1, p. 4] allows pairs $(V_{\varepsilon_1} \otimes W_{\varepsilon_2})$ for values V_{ε_1} and W_{ε_2} of arbitrary polarities. Of course, extending the syntax of preterms without really changing the type system leaves the set of well-typed terms unchanged, e.g. extending simply typed $L_p^{\vec{\tau}}$ with these pairs and the rules

$$\frac{\Gamma_1 \vdash V_{\varepsilon_1} : A_+ \mid \Delta_1 \qquad \Gamma_2 \vdash W_{\varepsilon_2} : B_+ \mid \Delta_2}{\Gamma_1, \Gamma_2 \vdash (V_{\varepsilon_1} \otimes W_{\varepsilon_2}) : A_+ \otimes B_+} \quad \text{and} \quad \frac{c : (\Gamma, x^{\varepsilon_1} : A_+, y^{\varepsilon_2} : B_+ \vdash \Delta)}{\Gamma \mid \tilde{\mu}(x^{\varepsilon_1} \otimes y^{\varepsilon_2}).c : A_+ \otimes B_+ \vdash \Delta}$$

would not change anything since these rules can only be used with $\varepsilon_1 = \varepsilon_2 = +$. However, often, the typing rules also allow types of arbitrary polarities, e.g. the type system of [CurFioMun16, Figure 2, p. 5] has the rules

$$\frac{\Gamma_1 \vdash V_{\varepsilon_1} : A_{\varepsilon_1} \mid \Delta_1 \qquad \Gamma_2 \vdash W_{\varepsilon_2} : B_{\varepsilon_2} \mid \Delta_2}{\Gamma_1, \Gamma_2 \vdash (V_{\varepsilon_1} \otimes W_{\varepsilon_2}) : A_{\varepsilon_1} \otimes B_{\varepsilon_2}} \quad \text{and} \quad \frac{c : (\Gamma, x^{\varepsilon_1} : A_{\varepsilon_1}, y^{\varepsilon_2} : B_{\varepsilon_2} \vdash \Delta)}{\Gamma \mid \tilde{\mu}(x^{\varepsilon_1} \otimes y^{\varepsilon_2}).c : A_{\varepsilon_1} \otimes B_{\varepsilon_2} \vdash \Delta}$$

This corresponds to an instance of $L_p^{\vec{\tau}}$ with several instances of the type former indexed by the polarities of its argument, e.g. four type formers $\otimes$ indexed by the polarities ε_1 and ε_2 of the left and right arguments. These can often be though of as being combinations of a single type former with shifts, e.g. we have isomorphisms

$$A_- \overset{-,+}{\otimes} B_+ \cong \Downarrow A_- \overset{+,+}{\otimes} B_+, \quad A_+ \overset{+,-}{\otimes} B_- \cong A_+ \overset{+,+}{\otimes} \Downarrow B_+ \quad \text{and} \quad A_- \overset{-,-}{\otimes} B_- \cong \Downarrow A_- \overset{+,+}{\otimes} \Downarrow B_+$$

Conversely, while these presentations often do not define the shifts, they are often expressible:

$$\Uparrow A_+ \cong 1 \to A_+ \quad \text{and} \quad \Downarrow A_- \cong 1 \otimes A_-$$

Most presentations therefore end up being more or less equivalent.

[CurFioMun16] "A Theory of Effects and Resources: Adjunction Models and Polarised Calculi", Curien, Fiore, and Munch-Maccagnoni, 2016

Well-polarized terms

The type system in Figure V.1.5 can be weakened by replacing each type A_ε by its polarity ε, which yields the type system described in Figure .4.1 of the appendix. Terms that are well-typed in this weaker system are called well-polarized:

Definition V.1.17

A term of $L_p^{\vec{\tau}}$ is said to be *well-polarized* (resp. *ill-polarized*) when it is well-typed (resp. ill-typed) in the type system described in Figure .4.1.

Fact V.1.18

Well-typed terms are well-polarized.

Proof

By induction on the derivation, replacing each type A_ε by its polarity ε.

Many presentations of L-calculi in the litterature mostly focus on well-typed terms, and can hence choose a presentation that allows ill-polarized terms in the syntax for the sake of simplicity. Here, however, we want to study an untyped L-calculus, and must therefore reject the ill-polarized terms explicitly. The $L_p^{\vec{\tau}}$ calculus (or rather its instances for specific choices of $\vec{\tau}$) can be obtained from L-calculi of the litterature by restricting to well-polarized terms.

The rigid treatment of polarities in the syntax of $L_p^{\vec{\tau}}$ ensures that all terms are well-polarized:

Fact V.1.19

For any command c (resp. expression t_ε, evaluation context e_ε) of $L_p^{\vec{\tau}}$, we have

$$c : (\Gamma \vdash \Delta) \quad \left(\text{resp. } \Gamma \vdash \underline{t_\varepsilon : \varepsilon} \mid \Delta, \quad \Gamma \mid \underline{e_\varepsilon : \varepsilon} \vdash \Delta\right)$$

if and only if Γ and Δ map all free variables of c (resp. t_ε, e_ε) to their polarities, i.e.

$$\Gamma = \overrightarrow{x^+} : +, \overrightarrow{y^-} : - \quad \text{and} \quad \Delta = \overrightarrow{\alpha^+} : +, \overrightarrow{\beta^-} : -$$

with

$$\mathrm{FV}(c) \subseteq \{\overrightarrow{x^+}, \overrightarrow{y^+}, \overrightarrow{\alpha^+}, \overrightarrow{\beta^-}\} \quad \left(\text{resp. } \mathrm{FV}(t_\varepsilon) \subseteq \{\overrightarrow{x^+}, \overrightarrow{y^+}, \overrightarrow{\alpha^+}, \overrightarrow{\beta^-}\}, \quad \mathrm{FV}(e_\varepsilon) \subseteq \{\overrightarrow{x^+}, \overrightarrow{y^+}, \overrightarrow{\alpha^+}, \overrightarrow{\beta^-}\}\right)$$

In particular, all terms of $L_p^{\vec{\tau}}$ are well-polarized.

Proof

The $\Rightarrow$ implication is by induction on the typing derivation, and the $\Leftarrow$ implication it by induction on the syntax (using the structural rules to preserve the whole context through multiplicative rules).

V.2. Intuitionistic and minimalistic polarized L-calculi: $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$ and $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

In this section, we start the process of transforming $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ into a corresponding λ-calculus ($\boldsymbol{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ of Section V.5), by carving out its intuitionistic fragment $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$ and its minimalistic fragment $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$. In a typed setting, those fragments correspond to the restrictions of classical logic to minimal and intuitionistic logic respectively.

V.2.1. Intuitionistic and minimalistic fragments

Fragment definitions

It is well-known Gentzen's sequent calculus for classical logic can be restricted to minimal logic (resp. intuitionistic logic) by only considering sequents with exactly one (resp. at most one) succedent. We define the minimalistic[4] (resp. intuitionistic) fragment of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ similarly:

Definition V.2.1

Given a set of nice type formers $\vec{\tau}$, a term of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ is said to be *minimalistically well-typed* (resp. *intuitionistically well-typed*) when there is a derivation of its well-typedness that only contains sequents with exactly one (resp. at most one) succedent[a]. This yields the type system described in ⚠ for minimalistically well-typed terms.

[a] The number of conclusions of a sequent is the number of types on the right of the $\vdash$ symbol, so that sequents with one succedent are those of the shape

$$c:(\Gamma \vdash \alpha^{\varepsilon}:A_{\varepsilon}),\quad \Gamma \vdash \underline{t_{\varepsilon}:A_{\varepsilon}}\,|,\quad \text{or}\quad \Gamma \mid \underline{e_{\varepsilon_1}:A_{\varepsilon_1}} \vdash \alpha^{\varepsilon_2}:A_{\varepsilon_2}$$

and sequents with zero succedents are those of the shape

$$c:(\Gamma \vdash)\quad \text{or}\quad \Gamma \mid \underline{e_{\varepsilon}:A_{\varepsilon}} \vdash$$

These restrictions can also be applied to the trivial type system that only accounts for polarities:

Definition V.2.2

A term of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ is said to be *minimalistically well-polarized* (resp. *intuitionistically well-polarized*), or *minimalistic* (resp. *intuitionistic*), when there is a derivation of its well-polarization that only contains sequents with exactly one (resp. at most one) succedent. This yields the type system described in Figure V.2.1 for minimalistically well-typed terms. We call *minimalistic fragment* (resp. *intuitionistic fragment*) of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$, and denote by $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ (resp. $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$), the subset of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ that consists of all minimalistically (resp. intuitionistically) well-polarized terms.

[4] Since we use this adjective for many kinds of objects, including some that are equipped with a preorder (e.g. terms with the observational preorder), we use "minimalistic" instead of "minimal" to avoid any ambiguity.

Figure V.2.1: Well polarized $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

Figure V.2.1.a: Core rules

$$\frac{}{x^{\varepsilon}:\varepsilon \vdash \underline{x^{\varepsilon}:\varepsilon} \mid}\ (\vdash\textsc{ax}) \qquad \frac{}{\mid \underline{\alpha^{\varepsilon}:\varepsilon} \vdash \alpha^{\varepsilon}:\varepsilon}\ (\textsc{ax}\vdash)$$

$$\frac{c:(\Gamma \vdash \alpha^{\varepsilon}:\varepsilon)}{\Gamma \vdash \underline{\mu\alpha^{\varepsilon}.c:\varepsilon} \mid}\ (\vdash\mu) \qquad \frac{c:(\Gamma, x^{\varepsilon}:\varepsilon \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}{\Gamma \mid \underline{\tilde{\mu}x^{\varepsilon}.c:\varepsilon} \vdash \alpha^{\varepsilon_r}:\varepsilon_r}\ (\tilde{\mu}\vdash)$$

$$\frac{\Gamma_1 \vdash \underline{t_{\varepsilon}:\varepsilon} \mid \qquad \Gamma_2 \mid \underline{e_{\varepsilon}:\varepsilon} \vdash \alpha^{\varepsilon_r}:\varepsilon_r}{\langle t_{\varepsilon} | e_{\varepsilon}\rangle^{\varepsilon}:(\Gamma_1, \Gamma_2 \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}\ (\textsc{cut})$$

Figure V.2.1.b: Structural rules (commands)

(Inoperable ($\vdash$Wc) rule)

$$\frac{c:(\Gamma \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}{c:(\Gamma, x^{\varepsilon}:\varepsilon \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}\ (\textsc{w}c\vdash)$$

(Inoperable ($\vdash$Cc) rule)

$$\frac{c:(\Gamma, x_1^{\varepsilon}:\varepsilon, x_2^{\varepsilon}:\varepsilon \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}{c[y^{\varepsilon}/x_1^{\varepsilon}, y^{\varepsilon}/x_2^{\varepsilon}]:(\Gamma, y^{\varepsilon}:\varepsilon \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}\ (\textsc{c}c\vdash)$$

(Inoperable ($\vdash$Pc) rule)

$$\frac{c:(\Gamma_1, x_1^{\varepsilon}:\varepsilon, x_2^{\varepsilon}:\varepsilon, \Gamma_2 \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}{c:(\Gamma_1, x_2^{\varepsilon}:\varepsilon, x_1^{\varepsilon}:\varepsilon, \Gamma_2 \vdash \alpha^{\varepsilon_r}:\varepsilon_r)}\ (\textsc{p}c\vdash)$$

Figure V.2.1.c: Structural rules (expressions)

(Inoperable ($\vdash$Wt) rule)

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0}:\varepsilon_0} \mid}{\Gamma, x^{\varepsilon}:\varepsilon \vdash \underline{t_{\varepsilon_0}:\varepsilon_0} \mid}\ (\textsc{w}t\vdash)$$

(Inoperable ($\vdash$Ct) rule)

$$\frac{\Gamma, x_1^{\varepsilon}:\varepsilon, x_2^{\varepsilon}:\varepsilon \vdash \underline{t_{\varepsilon_0}:\varepsilon_0} \mid}{\Gamma, x^{\varepsilon}:\varepsilon \vdash \underline{t_{\varepsilon_0}[x^{\varepsilon}/x_1^{\varepsilon}, x^{\varepsilon}/x_2^{\varepsilon}]:\varepsilon_0} \mid}\ (\textsc{c}t\vdash)$$

(Inoperable ($\vdash$Pt) rule)

$$\frac{\Gamma_1, x_1^{\varepsilon}:\varepsilon, x_2^{\varepsilon}:\varepsilon, \Gamma_2 \vdash \underline{t_{\varepsilon_0}:\varepsilon_0} \mid}{\Gamma_1, x_2^{\varepsilon}:\varepsilon, x_1^{\varepsilon}:\varepsilon, \Gamma_2 \vdash \underline{t_{\varepsilon_0}:\varepsilon_0} \mid}\ (\textsc{p}t\vdash)$$

Figure V.2.1.d: Structural rules (evaluation contexts)

(Inoperable ($\vdash$We) rule)

$$\frac{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \alpha^{\varepsilon_r} : \varepsilon_r}{\Gamma, x^{\varepsilon} : \varepsilon \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \alpha^{\varepsilon_r} : \varepsilon_r}\ (\mathrm{W}e\vdash)$$

(Inoperable ($\vdash$Ce) rule)

$$\frac{\Gamma, x_1^{\varepsilon} : \varepsilon, x_2^{\varepsilon} : \varepsilon \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \alpha^{\varepsilon_r} : \varepsilon_r}{\Gamma, x^{\varepsilon} : \varepsilon \mid \underline{e_{\varepsilon_0}[x^{\varepsilon}/x_1^{\varepsilon}, x^{\varepsilon}/x_2^{\varepsilon}] : \varepsilon_0} \vdash \alpha^{\varepsilon_r} : \varepsilon_r}\ (\mathrm{C}e\vdash)$$

(Inoperable ($\vdash$Pe) rule)

$$\frac{\Gamma_1, x_1^{\varepsilon} : \varepsilon, x_2^{\varepsilon} : \varepsilon, \Gamma_2 \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}{\Gamma_1, x_2^{\varepsilon} : \varepsilon, x_1^{\varepsilon} : \varepsilon, \Gamma_2 \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}\ (\mathrm{P}e\vdash)$$

Figure V.2.1.e: General shape of logic rules (assuming vs-sorted constructors)

$$\frac{\Gamma_1 \vdash \underline{v_{\varepsilon_1}^1 : \varepsilon_1} \mid \quad \ldots \quad \Gamma_q \vdash \underline{v_{\varepsilon_q}^q : \varepsilon_q} \mid \qquad \Gamma_{q+1} \mid \underline{s_{\varepsilon_{q+1}}^1 : \varepsilon_{q+1}} \vdash \alpha^{\varepsilon_r} : \varepsilon_r}{\Gamma_1, \ldots, \Gamma_{q+r} \mid \underline{\mathfrak{s}_k^{\tau_-^j}(v_{\varepsilon_1}^1, \ldots, v_{\varepsilon_q}^q, s_{\varepsilon_{q+1}}) : -} \vdash \alpha^{\varepsilon_r} : \varepsilon_r}\ \left(\mathfrak{s}_k^{\tau_-^j}\vdash\right)$$

$$\frac{c_1 : (\Gamma, \overrightarrow{x_1} : \overrightarrow{\varepsilon_1} \vdash \alpha_1^{\varepsilon_r} : \varepsilon_r) \quad \ldots \quad c_l : (\Gamma, \overrightarrow{x_l} : \overrightarrow{\varepsilon_l} \vdash \alpha_l^{\varepsilon_r} : \varepsilon_r)}{\Gamma \vdash \underline{\mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{x_1}, \alpha_1).c_1 | \ldots | \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{x_l}, \alpha_l).c_l \right\rangle : -} \mid}\ (\vdash\tau_-^j)$$

$$\frac{\Gamma_1 \vdash \underline{v_{\varepsilon_1}^1 : \varepsilon_1} \mid \quad \ldots \quad \Gamma_q \vdash \underline{v_{\varepsilon_q}^q : \varepsilon_q} \mid}{\Gamma_1, \ldots, \Gamma_q \vdash \underline{\mathfrak{v}_k^{\tau_+^j}(v_{\varepsilon_1}^1, \ldots, v_{\varepsilon_q}^q, s_{\varepsilon_{q+1}}^1, \ldots, s_{\varepsilon_{q+r}}^r) : +} \mid}\ \left(\vdash\mathfrak{v}_k^{\tau_+^j}\right)$$

$$\frac{c_1 : (\Gamma, \overrightarrow{x_1} : \overrightarrow{\varepsilon_1} \vdash \alpha_r^{\varepsilon_r} : \varepsilon_r) \quad \ldots \quad c_l : (\Gamma, \overrightarrow{x_l} : \overrightarrow{\varepsilon_l} \vdash \alpha_r^{\varepsilon_r} : \varepsilon_r)}{\Gamma \mid \underline{\widetilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}, \overrightarrow{\alpha_1}).c_1 | \ldots | \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}, \overrightarrow{\alpha_l}).c_l\right] : +} \vdash \alpha_r^{\varepsilon_r} : \varepsilon_r}\ (\tau_+^j\vdash)$$

Figure V.2.1.f: Logic rules for multiplicative types

$$\frac{c:(\Gamma, x^+:+ \vdash \alpha^-:-)}{\Gamma \vdash \underline{\mu(x^+\cdot\alpha^-).c : -}\mid}\ (\vdash\to) \qquad \frac{\Gamma_1 \vdash \underline{v_+:+}\mid \qquad \Gamma_2 \mid \underline{s_-:-} \vdash \alpha^\varepsilon:\varepsilon}{\Gamma_1,\Gamma_2 \mid \underline{v_+\cdot s_- : -} \vdash \alpha^\varepsilon:\varepsilon}\ (\to\vdash)$$

(Inoperable ($\vdash\parr$) rule) (Inoperable ($\parr\vdash$) rule)

$$\frac{\Gamma_1 \vdash \underline{v_+^1:+}\mid \qquad \Gamma_2 \vdash \underline{v_+^2:+}\mid}{\Gamma_1,\Gamma_2 \vdash \underline{(v_+^1\otimes v_+^2):+}\mid}\ (\vdash\otimes) \qquad \frac{c:(\Gamma,\alpha^+:+, y^+:+ \vdash \alpha^\varepsilon:\varepsilon)}{\Gamma \mid \underline{\tilde{\mu}(x^+\otimes y^-).c:+} \vdash \alpha^\varepsilon:\varepsilon}\ (\otimes\vdash)$$

(Inoperable ($\vdash\bot$) rule) (Inoperable ($\bot\vdash$) rule)

$$\frac{}{\vdash \underline{():+}\mid}\ (1\vdash) \qquad \frac{c:(\Gamma \vdash \alpha^\varepsilon:\varepsilon)}{\Gamma \mid \underline{\tilde{\mu}().c:+} \vdash \alpha^\varepsilon:\varepsilon}\ (\vdash 1)$$

Figure V.2.1.g: Logic rules for additive types

$$\frac{c_1:(\Gamma \vdash \alpha_1^-:-) \qquad c_2:(\Gamma \vdash \alpha_2^-:-)}{\Gamma \vdash \underline{\mu\langle(\pi_1\cdot\alpha_1^-).c_1 \mid (\pi_2\cdot\alpha_2^-).c_2\rangle : -}\mid}\ (\vdash\&) \qquad \frac{\Gamma \mid \underline{s_-:-} \vdash \alpha^\varepsilon:\varepsilon}{\Gamma \mid \underline{\pi_i\cdot s_-:-} \vdash \alpha^\varepsilon:\varepsilon}\ (\&\vdash)$$

$$\frac{\Gamma \vdash \underline{v_+:+}\mid}{\Gamma \vdash \underline{\iota_i(v_+):+}\mid}\ (\vdash\oplus) \qquad \frac{c_1:(\Gamma, x_1^+:+ \vdash \alpha^\varepsilon:\varepsilon) \qquad c_2:(\Gamma, x_2^+:+ \vdash \alpha^\varepsilon:\varepsilon)}{\Gamma \mid \underline{\tilde{\mu}[\iota_1(x_1^+).c_1 \mid \iota_2(x_2^+).c_2]:+} \vdash \alpha^\varepsilon:\varepsilon}\ (\oplus\vdash)$$

$$\frac{}{\Gamma \vdash \underline{\mu\langle\rangle : -}\mid}\ (\vdash\top)$$

(No ($\top\vdash$) rule)

(No ($\vdash 0$) rule)

$$\frac{}{\Gamma \mid \underline{\tilde{\mu}[]:+} \vdash \alpha^\varepsilon:\varepsilon}\ (0\vdash)$$

Figure V.2.1.h: Logic rules for shifts

$$\frac{c:(\Gamma \vdash \alpha^+:+)}{\Gamma \vdash \underline{\mu\{\alpha^+\}.c : -}\mid}\ (\vdash\Uparrow) \qquad \frac{\Gamma \mid \underline{s_+:+} \vdash \alpha^\varepsilon:\varepsilon}{\Gamma \mid \underline{\{s_+\}:-} \vdash \alpha^\varepsilon:\varepsilon}\ (\Uparrow\vdash)$$

$$\frac{\Gamma \vdash \underline{v_-:-}\mid}{\Gamma \vdash \underline{\{v_-\}:+}\mid}\ (\vdash\Downarrow) \qquad \frac{c:(\Gamma, x^-:- \vdash \alpha^\varepsilon:\varepsilon)}{\Gamma \mid \underline{\tilde{\mu}\{x^-\}.c:+} \vdash \alpha^\varepsilon:\varepsilon}\ (\Downarrow\vdash)$$

Figure V.2.1.i: Logic rules for negations

(Inoperable ($\vdash \neg_-$) rule)	(Inoperable ($\neg_- \vdash$) rule)
(Inoperable ($\vdash \neg_+$) rule)	(Inoperable ($\neg_+ \vdash$) rule)

Inoperable rules

The restriction prevent the use of some some rules:

Definition V.2.3

A typing rule of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ is said to be *operable in* $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ (resp. *operable in* $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$) when there exists a derivation that a term is minimalistically (resp. intuitionistically) well-polarized that uses an instance of that typing rule.

Example V.2.4

The core rules, the left structural rules, and the logic rules for $\to$, $\Downarrow$, $\Uparrow$, $\otimes$, $\oplus$, $\&$, 1, 0, and $\top$ are operable in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, while the right structural rules and the logic rules for $\neg_-$, $\neg_+$, $⅋$, and $\bot$ are not.

Note that removing type formers whose typing rules are not operable does not change the calculus, e.g.

$$\mathrm{Lm}_{\mathrm{p}}^{\to\Downarrow\Uparrow\neg_-\neg_+\otimes⅋\oplus\&1\bot0\top} = \mathrm{Lm}_{\mathrm{p}}^{\to\Downarrow\Uparrow\otimes\oplus\&10\top}$$

Operability of logic rules in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ can be fully characterized via fairly simple criteria:

Fact V.2.5

For logic rules, we have:

$$\begin{aligned}
\left(\mathfrak{s}_k^{\tau_-^j}\vdash\right) \text{ is operable in } \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} &\iff \mathfrak{s}_k^{\tau_-^j} \text{ has a single stack argument}\\
\left(\vdash\tau_-^j\right) \text{ is operable in } \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} &\iff \forall k, \mathfrak{s}_k^{\tau_-^j} \text{ has a single stack argument}\\
\left(\vdash\mathfrak{v}_k^{\tau_+^j}\right) \text{ is operable in } \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} &\iff \mathfrak{v}_k^{\tau_+^j} \text{ has a no stack argument}\\
\left(\tau_+^j\vdash\right) \text{ is operable in } \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} &\iff \forall k, \mathfrak{v}_k^{\tau_+^j} \text{ has a no stack argument}
\end{aligned}$$

Proof

Proofs that rules are not operable and the $\Rightarrow$ implications are by case analysis on the number of succedents in each sequent of the rule. Proofs that rules are operable and the $\Leftarrow$ implications simply exhibit a derivation in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ that uses the rule. For $\left(\vdash\tau_-^j\right)$ and $\left(\vdash\mathfrak{v}_k^{\tau_+^j}\right)$, any derivation that

$$\mathfrak{s}_k^{\tau_-^j}(\vec{x}, \alpha^\varepsilon, \vec{y}) \quad \text{and} \quad \mathfrak{v}_k^{\tau_+^j}(\vec{x})$$

are minimalistically well-polarized works, and for $\left(\tau_-^j\vdash\right)$ and $\left(\tau_+^j\vdash\right)$, any derivation that

$$\mu\left\langle\begin{matrix}\mathfrak{s}_1^{\tau_-^j}(\overrightarrow{x_1},\alpha_1^{\varepsilon_1},\overrightarrow{y_1}).\langle z_1^{\varepsilon_1}|\alpha_1^{\varepsilon_1}\rangle^{\varepsilon_1}\\ \vdots \\ \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{x_l},\alpha_l^{\varepsilon_l},\overrightarrow{y_l}).\langle z_l^{\varepsilon_l}|\alpha_l^{\varepsilon_l}\rangle^{\varepsilon_l}\end{matrix}\right\rangle \quad\text{and}\quad \tilde{\mu}\left[\begin{matrix}\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\langle y_1^{\varepsilon}|\alpha^{\varepsilon}\rangle^{\varepsilon}\\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}).\langle y_l^{\varepsilon}|\alpha^{\varepsilon}\rangle^{\varepsilon}\end{matrix}\right]$$

are minimalistically well-polarized works.

Operability of rules in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$ is a bit more complex:

Example V.2.6

All rules that were operable in $\mathrm{Lm}_\mathrm{p}^{\vec{\tau}}$ are still operable in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$. Among rules that were inoperable in $\mathrm{Lm}_\mathrm{p}^{\vec{\tau}}$, the right contraction rules, the right permutation rules, the weakening rule for terms ($\vdash \mathrm{w}t$), and the logic rule ($\vdash ⅋$) remain inoperable in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$, while the left logic rules ($\bot\vdash$), ($\neg_-\vdash$), and ($\neg_+\vdash$) become operable in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$. The operability of the remaining rules depends on the ability to instanciate enough of the premises with sequents that have no succedents. The rule ($\vdash\bot$) (resp. ($\vdash\neg_-$)) is always operable in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$ because this ability is provided by the corresponding left logic rule ($\bot\vdash$) (resp. ($\neg_-\vdash$))[a]. The rules ($\vdash \mathrm{w}c$), ($\vdash \mathrm{w}e$), ($⅋\vdash$), and ($\vdash\neg_+$) may be operable in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$ or not depending on what type formers are in $\vec{\tau}$: none are operable in $\mathrm{Li}_\mathrm{p}^{⅋\neg+}$, the first two are operable in $\mathrm{Li}_\mathrm{p}^{⅋\neg+0}$ while the latter two are not, and all four are operable in $\mathrm{Li}_\mathrm{p}^{⅋\neg+\bot}$ and in $\mathrm{Li}_\mathrm{p}^{\Uparrow⅋\neg+0}$.

[a] Indeed, the η-expansion of x^-

$$\mu\tilde{()}.\langle x^-|\tilde{()}\rangle^-, \quad (\text{resp. } \mu\neg_-(y^+).y\langle x^-|\neg_-(y^+)\rangle^-)$$

is always in $\mathrm{Li}_\mathrm{p}^{\vec{\tau}}$ (assuming that $\bot\in\vec{\tau}$ (resp. $\neg_-\in\vec{\tau}$)).

This characterization can most likely be generalized to arbitrary type formers by defining

$$\mathbf{X} = \left\{\varepsilon\in\{+,\ -\}\middle|\text{the judgement } \Gamma \mid \underline{s_\varepsilon : \varepsilon} \vdash \ \text{ is derivable in } \mathrm{Li}_\mathrm{p}^{\vec{\tau}} \text{ for some } s_\varepsilon \text{ and } \Gamma\right\}$$

(which is such that $-\ \in\mathbf{X}$ implies $+\in\mathbf{X}$ because we can form $\tilde{\mu}x^+.\langle y^-|s_-\rangle^-$) and using conditions such as "$\mathfrak{s}_k^{\tau_-^j}$ has at most one stack argument whose polarity is not in $\mathbf{X}$", but we have no use for such a characterization, and therefore do not work out the details here.

Inclusions

We of course have inclusions:

Fact V.2.7

For any $\vec{\tau}$, we have

$$\mathrm{Lm}_\mathrm{p}^{\vec{\tau}} \subseteq \mathrm{Li}_\mathrm{p}^{\vec{\tau}} \subsetneq \mathrm{L}_\mathrm{p}^{\vec{\tau}}$$

Proof

The inclusions are immediate. We have $\mathrm{Li}_\mathrm{p}^{\vec\tau} \not\supseteq \mathrm{L}_\mathrm{p}^{\vec\tau}$ because for $\alpha^\varepsilon \neq \beta^\varepsilon$,

$$\mathrm{Li}_\mathrm{p}^{\vec\tau} \not\ni \langle \mu\alpha^\varepsilon.\langle x^\varepsilon | \beta^\varepsilon\rangle^\varepsilon | \beta^\varepsilon \rangle^\varepsilon \in \mathrm{L}_\mathrm{p}^{\vec\tau}$$

The first inclusion may be an equality or not depending on $\vec\tau$:

Fact V.2.8

The following are equivalent:

- (i) there exists a derivation of well-polarization which is valid in $\mathrm{Li}_\mathrm{p}^{\vec\tau}$ but not in $\mathrm{Lm}_\mathrm{p}^{\vec\tau}$;
- (ii) there exists a derivation of well-polarization which is valid in $\mathrm{Li}_\mathrm{p}^{\vec\tau}$ but not in $\mathrm{Lm}_\mathrm{p}^{\vec\tau}$, and whose conclusion is of the shape

  $$c : (\Gamma \vdash) \quad \text{or} \quad \Gamma \mid \underline{e_\varepsilon : \varepsilon} \vdash$$

  i.e. has no succedent;
- (iii) there exists a stack s_ε in $\mathrm{Li}_\mathrm{p}^{\vec\tau}$ such that $\Gamma \mid \underline{s_\varepsilon : \varepsilon} \vdash$ is derivable for some Γ;
- (iv) at least one of the following holds:
  - (a) there exists a stack constructor $\mathfrak{s}_k^{\tau_-^j}$ with zero stack arguments (e.g. $\neg_-(v_+)$ or $\tilde{()}$); or
  - (b) there exists a postitive type former τ_+^j whose value constructors $\mathfrak{v}_k^{\tau_+^j}$ all have exactly one stack arguments (e.g. $\neg_+$ or 0).
- (v) there exists a stack s_ε in $\mathrm{Li}_\mathrm{p}^{\vec\tau}$ of the shape

  $$s_\varepsilon = \mathfrak{s}_k^{\tau_-^j}(\vec{x}) \quad (\text{e.g. } \neg_-(x^+) \quad \text{or} \quad \tilde{()})$$

  or

  $$s_\varepsilon = \tilde\mu\begin{bmatrix} \mathfrak{v}_1^{\tau_+^j}(\vec{x_1}, \alpha_1^{\varepsilon_1}, \vec{y_1}).\langle z_1^{\varepsilon_1} | \alpha_1^{\varepsilon_1}\rangle^{\varepsilon_1} \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\vec{x_l}, \alpha_l^{\varepsilon_l}, \vec{y_l}).\langle z_l^{\varepsilon_l} | \alpha_l^{\varepsilon_l}\rangle^{\varepsilon_l} \end{bmatrix} \quad (\text{e.g. } \tilde\mu\neg_+(\alpha^-).\alpha\langle x^- | \alpha^-\rangle^- \quad \text{or} \quad \tilde\mu[])$$

Furthermore, if all positive type formers in $\vec\tau$ have at least one constructor (i.e. there are no copies of 0), then these are also equivalent to:

- (vi) $\mathrm{Lm}_\mathrm{p}^{\vec\tau} \subsetneq \mathrm{Li}_\mathrm{p}^{\vec\tau}$.

In particular, for $\vec\tau \subseteq \{\rightarrow \Downarrow \Uparrow \neg_- \neg_+ \otimes ⅋ \oplus \& 1 \bot \top\}$[a], we have

$$\mathrm{Lm}_\mathrm{p}^{\vec\tau} \subsetneq \mathrm{Li}_\mathrm{p}^{\vec\tau} \quad \Leftrightarrow \quad \vec\tau \cap \{\neg_- \neg_+ \bot\} \neq \emptyset$$

[a] Note the absence of 0.

Proof sketch (See page 189 for details)

The implications (i) ⇒ (ii) ⇒ (iii) ⇒ (iv) ⇒ (v) ⇒ (i) ⇐ (iv) are either immediate or by induction on the derivation, and in the particular case, the implication (v) ⇒ (iv) is immediate.

Straightforwardly minimalistic type formers

The restriction to sequents with exactly one (resp. at most one) conclusion is mostly used in systems in which "classical" type formers have already been removed, and for an arbitrary set of type formers $\vec{\tau}$, there might be some subtleties that Definition V.2.2 fails to consider[5]. However, for some sets of type formers, no such subtleties arise:

Definition V.2.9

A negative type former is said to be *straightforwardly minimalistic* when all its rules are operable in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, and a positive type former is said to be *straightforwardly minimalistic* when all its rules are operable in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and it has at least one constructor (i.e. it is not a copy of 0).

Example V.2.10

The type formers $\rightarrow$, $\Downarrow$, $\Uparrow$, $\otimes$, $\oplus$, $\&$, 1, and $\top$ are straightforwardly minimalistic, while $\neg_-$, $\neg_+$, $⅋$, 0, and $\bot$ are not.

The restriction to type formers that are operable in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ is fairly natural: we remove unwanted type formers before applying the restriction. The rejection of 0 is a bit harder to justify, but is not completely unheard of[6], not completely arbitrary[7], and fairly harmless:

---

[5]For example, negations can not be used in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ while some sequents with negations are provable in minimal logic according to the ncatlab page on minimal logic.

[6]The type former 0 needs to be removed to ensure that the teleological version of **ILL** is faithful [Gir11, p. 217].

[7]One could restrict

$$\frac{}{\Gamma \mid \underline{\tilde{\mu}[\,]:0} \vdash \Delta}\ (0\vdash) \quad \text{to} \quad \frac{}{\mid \underline{\tilde{\mu}[\,]:0} \vdash}$$

This would have no effect in $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ since the full rule would be derivable by composing the restricted rule with weakening rules, but this would make the rule $(0\vdash)$ inoperable in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$.

Furthermore, this restriction can be seen as an instance of a natural and systematic transformation of additive rules that allows them to have different contexts in their premises and takes their unions in the conclusions, e.g. replacing

$$\frac{c_1:(\Gamma, x_1^+ : A_+^1 \vdash \Delta) \qquad c_2:(\Gamma, x_2^+ : A_+^2 \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}[\iota_1(x_1^+).c_1 | \iota_2(x_2^+).c_2] : A_+^1 \oplus A_+^2} \vdash \Delta}\ (\oplus\vdash) \quad \text{by} \quad \frac{c_1:(\Gamma_1, x_1^+ : A_+^1 \vdash \Delta_1) \qquad c_2:(\Gamma_2, x_2^+ : A_+^2 \vdash \Delta_2)}{\Gamma_1 \cup \Gamma_2 \mid \underline{\tilde{\mu}[\iota_1(x_1^+).c_1 | \iota_2(x_2^+).c_2] : A_+^1 \oplus A_+^2} \vdash \Delta_1 \cup \Delta_2}$$

For other additive types, this yields a more general rule which is derivable in $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ by composing the normal rules with weakening rules, but for $(0\vdash)$, since there are no premises, we get the neutral element for context union, i.e. the empty context.

Fact V.2.11

Given a set of straightforwardly minimalistic type formers $\vec{\tau}$, for any term t, we have

$$t' \in \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}0} \quad \Leftrightarrow \quad \exists t \in \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}, t \dashv_0^* t'$$

i.e. terms of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}0}$ are exactly those of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ with some positive stacks replaced by $\tilde{\mu}[]$.

Proof

The $\Rightarrow$ implication is by induction on the derivation that $t' \in \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}0}$, and the $\Leftarrow$ implication follows from $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \subseteq \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}0}$ and closure of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}0}$ under $\dashv_0$, which is proven by induction on the derivation of $t \dashv_0 t'$.

Restricting to straightforwardly minimalistic type formers forces all commands and evaluation contexts to have at least one free stack variable (which is crucial for):

Fact V.2.12

Given a set of straightforwardly minimalistic type formers $\vec{\tau}$, for any evaluation context e_ε (resp. command c) of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$, we have

$$|\mathsf{FV}_{\mathcal{S}}(e_\varepsilon)| \geq 1 \quad (\text{resp. } |\mathsf{FV}_{\mathcal{S}}(c)| \geq 1)$$

In particular, there are no derivations whose conclusion is of the shape

$$\Gamma \mid \underline{e_\varepsilon : \varepsilon} \vdash \quad (\text{resp. } c : (\Gamma \vdash))$$

Proof

By induction on the derivation that e_ε (resp. c) is well-polarized. The restriction on derivations follows by Fact V.1.19.

Note that this property fails if $0 \in \vec{\tau}$:

$$\Gamma \mid \underline{\tilde{\mu}[] : +} \vdash \alpha^\varepsilon : \varepsilon \quad \text{but} \quad \mathsf{FV}_{\mathcal{S}}(\tilde{\mu}[]) = \emptyset$$

By forbidding these judgements, the restriction to straightforwardly minimalistic type formers erases the distinction between $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$:

Fact V.2.13

For any set of straightforwardly minimalistic type formers $\vec{\tau}$, we have $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} = \mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$.

Proof

By the previous fact.

V.2.2. A syntax for the minimalistic fragment

Characterization of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ via free stack variables

The $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus can be characterized as a subcalculus of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ as follows:

Proposition V.2.14

Given a set of straightforwardly minimalistic type formers $\vec{\tau}$, a term t of $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ is in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, if and only if all of the following hold:

- for any subexpression t_ε of t, $|\mathsf{FV}_{\mathcal{S}}(t_\varepsilon)| = 0$;
- for any sub-evaluation-context e_ε of t, $|\mathsf{FV}_{\mathcal{S}}(e_\varepsilon)| = 1$; and
- for any subcommand c of t, $|\mathsf{FV}_{\mathcal{S}}(c)| = 1$ (and this last condition on commands is redundant).

Proof

- $\boxed{\Rightarrow}$ The $\leq$ inequalities are given by Fact V.1.19 and the $\geq$ inequalities by Fact V.2.12.
- $\boxed{\Leftarrow}$ It suffices to remove all right weakening rules in the derivation. More precisely, we show by induction on the derivation that

$$\Gamma \vdash \underline{t_\varepsilon : \varepsilon} \mid \Delta, \quad (\text{resp. } \Gamma \mid \underline{e_\varepsilon : \varepsilon} \vdash \Delta, \quad c : (\Gamma \vdash \Delta))$$

implies

$$\Gamma \vdash_{\mathrm{m}} \underline{t_\varepsilon : \varepsilon} \mid, \quad (\text{resp. } \Gamma \mid \underline{e_\varepsilon : \varepsilon} \vdash_{\mathrm{m}} \alpha^{\varepsilon_\star} : \varepsilon_\star, \quad c : (\Gamma \vdash_{\mathrm{m}} \alpha^{\varepsilon_\star} : \varepsilon_\star))$$

for some $\alpha^{\varepsilon_\star}$. For right weakening rules, we simply apply the induction hypothesis to the premise, and for other rules, we apply the induction hypothesis to the premises and then reapply the same rule. This works for the $(\vdash \tau_-^j)$ (resp. $(\vdash \mu)$) rule because the $|\mathsf{FV}_{\mathcal{S}}(\mu{<}...{>})| = 0$ (resp. $|\mathsf{FV}_{\mathcal{S}}(\mu\alpha^\varepsilon.c)| = 0$) hypothesis ensures that the free stack variables it binds are exactly those that are free in the subcommands, and for the $(\tau_+^j \vdash)$ rule of types with more than one constructor because the condition $|\mathsf{FV}_{\mathcal{S}}(\tilde{\mu}[...])| = 1$ ensures that all the variables $\alpha^{\varepsilon_\star}$ given by the induction hypothesis are the same.
- The condition on commands is redundant because

$$|\mathsf{FV}_{\mathcal{S}}(\langle t_\varepsilon | e_\varepsilon \rangle^\varepsilon)| = |\mathsf{FV}_{\mathcal{S}}(t_\varepsilon) \cup \mathsf{FV}_{\mathcal{S}}(e_\varepsilon)| = |\emptyset \cup \mathsf{FV}_{\mathcal{S}}(e_\varepsilon)| = |\mathsf{FV}_{\mathcal{S}}(e_\varepsilon)| = 1$$

Output polarities

The inferrence rules that define $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ can be seen as production rules of a general grammar whose non-terminal symbols are the judgements, and whose terminal symbols are parentheses and rule names: an inferrence rule

$$\frac{\text{first premise} \qquad \dots \qquad \text{last premise}}{\text{conclusion}}\ \textsc{name}$$

becomes a production rule

$$\text{conlusion} \to \text{name}((\text{first premise}), \dots, (\text{last premise}))$$

This grammar is not really a syntax (i.e. it is not context-free) because there are infinitely many distinct judgements. By Proposition V.2.14, we can discard Γ. This is not sufficient because $\alpha^{\varepsilon_\star}$ ranges over infinitely many names, and we can not discard it because $\mu\beta^{\varepsilon}.\langle x^{\varepsilon_\star} | \alpha^{\varepsilon_\star} \rangle^{\varepsilon_\star}$ being in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ depends on whether $\alpha^{\varepsilon_\star} = \beta^{\varepsilon}$ or not. Instead, we switch to a presentation where stack variable names α are replace by de Bruijn indices $\star_0$, $\star_1$, and so on, and we write $\star$ for $\star_0$. With this presentation, judgements are of the shape

$$\Gamma \vdash \underline{t_\varepsilon : \varepsilon} \,|, \quad \Gamma \,|\, \underline{e_\varepsilon : \varepsilon} \vdash \star^{\varepsilon_\star} : \varepsilon_\star, \quad \text{or} \quad c : (\Gamma \vdash \star^{\varepsilon_\star} : \varepsilon_\star)$$

and $\mu\beta^{\varepsilon}.\langle x^{\varepsilon_\star} | \star^{\varepsilon_\star} \rangle^{\varepsilon_\star}$ is in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ if and only if $\beta^{\varepsilon} = \star^{\varepsilon_\star}$. By erasing Γ and replacing it by $\textcircled{?}$, we get a finite set of judgements:

$$\begin{array}{cc}
\textcircled{?} \vdash \underline{t_+ : +} \,| & \textcircled{?} \vdash \underline{t_- : -} \,| \\
c : (\textcircled{?} \vdash \star^{+} : +) & c : (\textcircled{?} \vdash \star^{-} : -) \\
\textcircled{?} \,|\, \underline{e_+ : +} \vdash \star^{+} : + & \textcircled{?} \,|\, \underline{e_+ : +} \vdash \star^{-} : - \\
\textcircled{?} \,|\, \underline{e_- : -} \vdash \star^{+} : + & \textcircled{?} \,|\, \underline{e_- : -} \vdash \star^{-} : -
\end{array}$$

We introduce concise notations that allow making explicit which one of these judgements holds for the term under consideration:

Definition V.2.15

Given an evaluation context e_ε (resp. command c) of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, we say that it has *output polarity* $\varepsilon_\star$ when there exists a derivation of

$$c : (\Gamma \vdash \star^{\varepsilon_\star} : \varepsilon_\star) \quad (\text{resp. } \Gamma \,|\, \underline{e_\varepsilon : \varepsilon} \vdash \star^{\varepsilon_\star} : \varepsilon_\star)$$

for some Γ. We write $e_{\varepsilon \rightsquigarrow \varepsilon_\star}$ (resp. $c_{\rightsquigarrow \varepsilon_\star}$) for evaluation contexts e_ε (resp. commands c) of output polarity $\varepsilon_\star$. We call the polarity ε of a term t_ε (resp. evaluation context e_ε) its *interaction polarity*, and sometimes also call the interaction polarity ε of an evaluation context $e_{\varepsilon \rightsquigarrow \varepsilon_\star}$ its *input polarity*.

A BNF grammar for $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

Figure V.2.2: The $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus

Figure V.2.2.a: Syntax

Negative values / expressions:

$$
\begin{aligned}
v_-, w_- ::=\ & x^- \mid \mu\star^-.c_{\leadsto -}\\
\mid\ & \mu\left\langle \mathfrak{s}_1^{\tau^1_-}(\overrightarrow{x_1}, \star^{\varepsilon_{1,1}}).c^1_{\leadsto\varepsilon_{1,1}} \mid \dots \mid \mathfrak{s}_{l_1^-}^{\tau^1_-}(\overrightarrow{x_{l_1^-}}, \star^{\varepsilon_{1,l_1^-}}).c^{l_1^-}_{\leadsto\varepsilon_{1,l_1^-}} \right\rangle\\
\mid\ & \vdots\\
\mid\ & \mu\left\langle \mathfrak{s}_1^{\tau^m_-}(\overrightarrow{x_1}, \star^{\varepsilon_{m,1}}).c^1_{\leadsto\varepsilon_{m,1}} \mid \dots \mid \mathfrak{s}_{l_m^-}^{\tau^m_-}(\overrightarrow{x_{l_m^-}}, \star^{\varepsilon_{m,l_m^-}}).c^{l_m^-}_{\leadsto\varepsilon_{m,l_m^-}} \right\rangle
\end{aligned}
$$

Negative stacks:

$$
\begin{aligned}
s_{-\leadsto\varepsilon} ::=\ & \star^-\}_{\varepsilon=-}\\
\mid\ & \mathfrak{s}_1^{\tau^1_-}(\vec{v}, s_{\varepsilon_{1,1}\leadsto\varepsilon}) \mid \dots \mid \mathfrak{s}_{l_1^-}^{\tau^1_-}(\vec{v}, s_{\varepsilon_{1,l_1^-}\leadsto\varepsilon})\\
\mid\ & \vdots \qquad \mid \ddots \mid \vdots\\
\mid\ & \mathfrak{s}_1^{\tau^m_-}(\vec{v}, s_{\varepsilon_{m,1}\leadsto\varepsilon}) \mid \dots \mid \mathfrak{s}_{l_m^-}^{\tau^m_-}(\vec{v}, s_{\varepsilon_{m,l_m^-}\leadsto\varepsilon})
\end{aligned}
$$

Negative evaluation contexts:

$$
e_{-\leadsto\varepsilon} ::= \mathsf{stk}^-(s_{-\leadsto\varepsilon}) \mid \tilde{\mu}x^-.c_{\leadsto\varepsilon}
$$

Positive values:

$$
\begin{aligned}
v_+, w_+ ::=\ & x^+\\
\mid\ & \mathfrak{v}_1^{\tau^1_+}(\vec{v}) \mid \dots \mid \mathfrak{v}_{l_1^+}^{\tau^1_+}(\vec{v})\\
\mid\ & \vdots \qquad \mid \ddots \mid \vdots\\
\mid\ & \mathfrak{v}_1^{\tau^n_+}(\vec{v}) \mid \dots \mid \mathfrak{v}_{l_n^+}^{\tau^n_+}(\vec{v})
\end{aligned}
$$

Positive stacks / evaluation contexts:

$$
\begin{aligned}
s_{+\leadsto\varepsilon}, e_{+\leadsto\varepsilon} ::=\ & \star^+\}_{\varepsilon=+} \mid \tilde{\mu}x^+.c_{\leadsto\varepsilon}\\
\mid\ & \tilde{\mu}\left[\mathfrak{v}_1^{\tau^1_+}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon} \mid \dots \mid \mathfrak{v}_{l_1^+}^{\tau^1_+}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\leadsto\varepsilon}\right]\\
\mid\ & \vdots\\
\mid\ & \tilde{\mu}\left[\mathfrak{v}_1^{\tau^n_+}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon} \mid \dots \mid \mathfrak{v}_{l_n^+}^{\tau^n_+}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\leadsto\varepsilon}\right]
\end{aligned}
$$

Positive expressions:

$$
t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mu\star^+.c_{\leadsto +}
$$

Commands:

$$
c_{\leadsto\varepsilon} ::= \langle t_+ | e_{+\leadsto\varepsilon}\rangle^+ \mid \langle t_- | e_{-\leadsto\varepsilon}\rangle^-
$$

Figure V.2.2.b: Operational reduction

$$
\begin{aligned}
\langle \mu\star^{\varepsilon}.c_{\leadsto\varepsilon} \,|\, s_\varepsilon\rangle^{\varepsilon} &\rhd_\mu\ c_{\leadsto\varepsilon}[s_\varepsilon/\star^\varepsilon]\\
\langle v_{\varepsilon_1} \,|\, \tilde\mu x^{\varepsilon_1}.c_{\leadsto\varepsilon_2}\rangle^{\varepsilon_1} &\rhd_{\tilde\mu}\ c_{\leadsto\varepsilon_2}[v_{\varepsilon_1}/x^{\varepsilon_1}]\\
\left\langle \mu \left\langle \mathfrak{s}_1^{\tau_-^j}(\vec{x_1}, \star^{\varepsilon_{j,1}}).c^1_{\leadsto\varepsilon_{j,1}} \,|\, \dots \,|\, \mathfrak{s}_l^{\tau_-^j}(\vec{x_l},\star^{\varepsilon_{j,l}}).c^l_{\leadsto\varepsilon_{j,l}}\right\rangle \,\middle|\, \mathfrak{s}_k^{\tau_-^j}(\vec{v}, s_{\varepsilon_{j,k}\leadsto\varepsilon})\right\rangle^- &\rhd_{\tau_-^j}\ c^k_{\leadsto\varepsilon_{j,k}}[\vec{v}/\vec{x_k}, s_{\varepsilon_{j,k}\leadsto\varepsilon}/\star^{\varepsilon_{j,k}}]\\
\left\langle \mathfrak{v}_k^{\tau_+^j}(\vec{v}) \,\middle|\, \tilde\mu\left[\mathfrak{v}_1^{\tau_+^j}(\vec{x_1}).c^1_{\leadsto\varepsilon} \,|\, \dots \,|\, \mathfrak{v}_l^{\tau_+^j}(x_l).c^l_{\leadsto\varepsilon}\right]\right\rangle^+ &\rhd_{\tau_+^j}\ c^k_{\leadsto\varepsilon}[\vec{v}/\vec{x_k}]
\end{aligned}
$$

$$\rhd \overset{\text{def}}{=} \rhd_{\tilde\mu} \cup \rhd_{\tilde\mu} \cup \left(\textstyle\bigcup_j \rhd_{\tau_-^j}\right) \cup \left(\textstyle\bigcup_j \rhd_{\tau_+^j}\right)$$

Figure V.2.2.c: Top-level η-expansion

$$
\begin{aligned}
t_\varepsilon &\rbrace_\mu\ \mu\star^\varepsilon.\langle t_\varepsilon \,|\, \star^\varepsilon\rangle^\varepsilon\\
e_{\varepsilon_1\leadsto\varepsilon_2} &\rbrace_{\tilde\mu}\ \tilde\mu x^{\varepsilon_1}.\langle x^{\varepsilon_1} \,|\, e_{\varepsilon_1\leadsto\varepsilon_2}\rangle^{\varepsilon_1} && \text{if } x^{\varepsilon_1} \text{ fresh w.r.t. } e_{\varepsilon_1\leadsto\varepsilon_2}\\
v_- &\rbrace_{\tau_-^j}\ \mu\left\langle \begin{array}{c} \mathfrak{s}_1^{\tau_-^j}(\vec{x_1},\star^{\varepsilon_{j,1}}).\left\langle v_- \,\middle|\, \mathfrak{s}_1^{\tau_-^j}(\vec{x_1},\star^{\varepsilon_{j,1}})\right\rangle^- \\ \vdots \\ \mathfrak{s}_l^{\tau_-^j}(\vec{x_l},\star^{\varepsilon_{j,l}}).\left\langle v_- \,\middle|\, \mathfrak{s}_l^{\tau_-^j}(\vec{x_l},\star^{\varepsilon_{j,l}})\right\rangle^- \end{array}\right\rangle && \text{if } \vec{x_1}, \dots, \vec{x_l} \text{ fresh w.r.t. } v_-\\
s_{+\leadsto\varepsilon} &\rbrace_{\tau_+^j}\ \tilde\mu\left[\begin{array}{c} \mathfrak{v}_1^{\tau_+^j}(\vec{x_1}).\left\langle \mathfrak{v}_1^{\tau_+^j}(\vec{x_1}) \,\middle|\, s_{+\leadsto\varepsilon}\right\rangle^+ \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\vec{x_l}).\left\langle \mathfrak{v}_l^{\tau_+^j}(\vec{x_l}) \,\middle|\, s_{+\leadsto\varepsilon}\right\rangle^+ \end{array}\right] && \text{if } \vec{x_1}, \dots, \vec{x_l} \text{ fresh w.r.t. } s_{+\leadsto\varepsilon}
\end{aligned}
$$

$$\rbrace \overset{\text{def}}{=} \rbrace_\mu \cup \rbrace_{\tilde\mu} \cup \left(\bigcup_j \rbrace_{\tau_-^j}\right) \cup \left(\bigcup_j \rbrace_{\tau_+^j}\right)$$

Figure V.2.3: The $\mathrm{Lm}_{\mathrm{p}}^{\to\Downarrow\Uparrow\otimes\oplus\&10\top}$ calculus

Figure V.2.3.a: Syntax

Negative values / expressions:

$$
\begin{aligned}
v_-, w_- ::= {} & x^- \mid \mu\star^-.c_{\rightsquigarrow -} \\
\mid {} & \mu(x^+ \cdot \star^-).c_{\rightsquigarrow -} \\
\mid {} & \mu\langle(\pi_1 \cdot \star^-).c^1_{\rightsquigarrow -} | (\pi_2 \cdot \star^-).c^2_{\rightsquigarrow -}\rangle \\
\mid {} & \mu\{\alpha^+\}.c \\
\mid {} & \mu\langle\rangle
\end{aligned}
$$

Negative stacks:

$$
\begin{aligned}
s_{-\rightsquigarrow\varepsilon} ::= {} & \star^-\}_{\varepsilon=-} \\
\mid {} & v_+ \cdot s_{-\rightsquigarrow\varepsilon} \\
\mid {} & \pi_1 \cdot s_{-\rightsquigarrow\varepsilon} \mid \pi_2 \cdot s_{-\rightsquigarrow\varepsilon} \\
\mid {} & \{s_{+\rightsquigarrow\varepsilon}\}
\end{aligned}
$$

Negative evaluation contexts:

$$
e_{-\rightsquigarrow\varepsilon} ::= \mathsf{stk}^-(s_{-\rightsquigarrow\varepsilon}) \mid \tilde{\mu}x^-.c_{\rightsquigarrow\varepsilon}
$$

Positive values:

$$
\begin{aligned}
v_+, w_+ ::= {} & x^+ \\
\mid {} & (v_+ \otimes w_+) \\
\mid {} & \iota_1(v_+) \mid \iota_2(v_+) \\
\mid {} & \{v_-\} \\
\mid {} & ()
\end{aligned}
$$

Positive stacks / evaluation contexts:

$$
\begin{aligned}
s_{+\rightsquigarrow\varepsilon}, e_{+\rightsquigarrow\varepsilon} ::= {} & \star^+\}_{\varepsilon=+} \mid \tilde{\mu}x^+.c_{\rightsquigarrow\varepsilon} \\
\mid {} & \tilde{\mu}(x^+ \otimes y^+).c_{\rightsquigarrow\varepsilon} \\
\mid {} & \tilde{\mu}[\iota_1(x_1^+).c^1_{\rightsquigarrow\varepsilon} | \iota_2(x_2^+).c^2_{\rightsquigarrow\varepsilon}] \\
\mid {} & \tilde{\mu}\{x^-\}.c_{\rightsquigarrow\varepsilon} \\
\mid {} & \tilde{\mu}().c_{\rightsquigarrow\varepsilon}
\end{aligned}
$$

Positive expressions:

$$
t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mu\star^+.c_{\rightsquigarrow +}
$$

Commands:

$$
c_{\rightsquigarrow\varepsilon} ::= \langle t_+ | e_{+\rightsquigarrow\varepsilon}\rangle^+ \mid \langle t_- | e_{-\rightsquigarrow\varepsilon}\rangle^-
$$

Figure V.2.3.b: Operational reduction

$$\langle \mu \star^{\varepsilon}.c_{\leadsto\varepsilon} | s_{\varepsilon} \rangle^{\varepsilon} \rhd_{\mu} \; c_{\leadsto\varepsilon}[s_{\varepsilon}/\star^{\varepsilon}]$$

$$\langle v_{\varepsilon_1} | \tilde{\mu} x^{\varepsilon_1}.c_{\leadsto\varepsilon_2} \rangle^{\varepsilon_1} \rhd_{\tilde{\mu}} \; c_{\leadsto\varepsilon_2}[v_{\varepsilon_1}/x^{\varepsilon_1}]$$

$$\langle \mu(x^{+} \cdot \star^{-}).c_{\leadsto-} | v_{+} \cdot s_{-\leadsto\varepsilon} \rangle^{-} \rhd_{\to} \; c_{\leadsto-}[v_{+}/x^{+}, s_{-\leadsto\varepsilon}/\star^{-}]$$

$$\langle \mu\{\star^{+}\}.c_{\leadsto+} | \{s_{+\leadsto\varepsilon}\} \rangle^{-} \rhd_{\Uparrow} \; c_{\leadsto+}[s_{+\leadsto\varepsilon}/\star^{+}]$$

$$\langle \mu \langle (\pi_1 \cdot \star^{-}).c^{1}_{\leadsto-} | (\pi_2 \cdot \star^{-}).c^{2}_{\leadsto-} \rangle | \pi_i \cdot s_{-\leadsto\varepsilon} \rangle^{-} \rhd_{\&} \; c^{i}_{\leadsto-}[s_{-\leadsto\varepsilon}/\star^{-}]$$

($\rhd_{\top}$ is trivial)

$$\langle \{v_{-}\} | \tilde{\mu}\{x^{-}\}.c_{\leadsto\varepsilon} \rangle^{+} \rhd_{\Downarrow} \; c_{\leadsto\varepsilon}[v_{-}/x^{-}]$$

$$\langle (v_{+} \otimes w_{+}) | \tilde{\mu}(x^{+} \otimes y^{+}).c_{\leadsto\varepsilon} \rangle^{+} \rhd_{\otimes} \; c_{\leadsto\varepsilon}[v_{+}/x^{+}, w_{+}/y^{+}]$$

$$\langle \iota_i(v_{+}) | \tilde{\mu}[\iota_1(x^{+}_{1}).c^{1}_{\leadsto\varepsilon} | \iota_2(x^{+}_{2}).c^{2}_{\leadsto\varepsilon}] \rangle^{+} \rhd_{\oplus} \; c^{i}_{\leadsto\varepsilon}[v_{+}/x^{+}_{i}]$$

$$\langle () | \tilde{\mu}().c_{\leadsto\varepsilon} \rangle^{+} \rhd_{1} \; c_{\leadsto\varepsilon}$$

$$\rhd \stackrel{\text{def}}{=} \rhd_{\tilde{\mu}} \cup \rhd_{\tilde{\mu}} \cup \rhd_{\to} \cup \rhd_{\&} \cup \rhd_{\Uparrow} \cup \rhd_{\otimes} \cup \rhd_{\oplus} \cup \rhd_{\Downarrow} \cup \rhd_{1}$$

Figure V.2.3.c: Top-level η-expansion

$$t_\varepsilon \mathrel{\rfloor}_\mu \mu\star^\varepsilon.\langle t_\varepsilon|\star^\varepsilon\rangle^\varepsilon$$

$$e_{\varepsilon_1\rightsquigarrow\varepsilon_2} \mathrel{\rfloor}_{\widetilde{\mu}} \widetilde{\mu}x^{\varepsilon_1}.\langle x^{\varepsilon_1}|e_{\varepsilon_1\rightsquigarrow\varepsilon_2}\rangle^{\varepsilon_1} \qquad \text{if } x^{\varepsilon_1} \text{ fresh w.r.t. } e_{\varepsilon_1\rightsquigarrow\varepsilon_2}$$

$$\upsilon_- \mathrel{\rfloor}_{\tau_-^j} \mu\left\langle\begin{array}{c} \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{x_1},\star^{\varepsilon_{j,1}}).\left\langle \upsilon_- \middle| \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{x_1},\star^{\varepsilon_{j,1}})\right\rangle^- \\ \vdots \\ \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{x_l},\star^{\varepsilon_{j,l}}).\left\langle \upsilon_- \middle| \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{x_l},\star^{\varepsilon_{j,l}})\right\rangle^- \end{array}\right\rangle \qquad \text{if } \overrightarrow{x_1},\dots,\overrightarrow{x_l} \text{ fresh w.r.t. } \upsilon_-$$

$$s_{+\rightsquigarrow\varepsilon} \mathrel{\rfloor}_{\tau_+^j} \widetilde{\mu}\left[\begin{array}{c} \mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\left\langle \mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}) \middle| s_{+\rightsquigarrow\varepsilon}\right\rangle^+ \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}).\left\langle \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}) \middle| s_{+\rightsquigarrow\varepsilon}\right\rangle^+ \end{array}\right] \qquad \text{if } \overrightarrow{x_1},\dots,\overrightarrow{x_l} \text{ fresh w.r.t. } s_{+\rightsquigarrow\varepsilon}$$

$$\mathrel{\rfloor} \stackrel{\text{def}}{=} \mathrel{\rfloor}_\mu \cup \mathrel{\rfloor}_{\widetilde{\mu}} \cup \left(\bigcup_j \mathrel{\rfloor}_{\tau_-^j}\right) \cup \left(\bigcup_j \mathrel{\rfloor}_{\tau_+^j}\right)$$

Given a set of straightforwardly minimalistic type formers $\vec{\tau}$, the syntax of the $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus is given in Figure V.2.2a, where $\varepsilon_{j,k}$ denotes the (input) polarity of the stack argument of $\mathfrak{s}_{k}^{\tau_{-}^{j}}$. Note that although it is not explicit in the BNF grammar, the restriction to straightforwardly minimalistic types only allows strictly positive l_{j}^{+}. Instanciated with $\vec{\tau} = \rightarrow\Downarrow\Uparrow\otimes\oplus\&10\top$, this yields Figure V.2.3a.

Since the main difference between the syntax of $s_{\varepsilon\rightsquigarrow+}$ and $s_{\varepsilon\rightsquigarrow-}$ is only whether it contains $\star^{\varepsilon}$ or not (with $s_{+\rightsquigarrow+}$ containing $\star^{+}$, $s_{-\rightsquigarrow-}$ containing $\star^{-}$, and neither $s_{+\rightsquigarrow-}$ nor $s_{-\rightsquigarrow+}$ containing any $\star^{\varepsilon}$), we avoid duplications by having side conditions in the grammar, e.g.

$$s_{+\rightsquigarrow\varepsilon} ::= \star^{+}\}\varepsilon{=}+$$

means that $s_{+\rightsquigarrow+}$ can be $\star^{+}$ but $s_{+\rightsquigarrow-}$ can not. For example,

$$\begin{aligned} s_{+\rightsquigarrow\varepsilon} ::=\ & \star^{+}\}\varepsilon{=}+ \mid \tilde{\mu}x^{+}.c_{\rightsquigarrow\varepsilon} \\ & \mid \tilde{\mu}\left[\mathfrak{d}_{1}^{\tau_{+}^{1}}(\overrightarrow{x_{1}}).c_{\rightsquigarrow\varepsilon}^{1} | \ldots | \mathfrak{d}_{l_{1}^{+}}^{\tau_{+}^{1}}(\overrightarrow{x_{l_{1}^{+}}}).c_{\rightsquigarrow\varepsilon}^{l_{1}^{+}}\right] \\ & \mid \vdots \\ & \mid \tilde{\mu}\left[\mathfrak{d}_{1}^{\tau_{+}^{n}}(\overrightarrow{x_{1}}).c_{\rightsquigarrow\varepsilon}^{1} | \ldots | \mathfrak{d}_{l_{n}^{+}}^{\tau_{+}^{n}}(\overrightarrow{x_{l_{n}^{+}}}).c_{\rightsquigarrow\varepsilon}^{l_{n}^{+}}\right] \end{aligned}$$

stands for

$$\begin{aligned} s_{+\rightsquigarrow+} ::=\ & \star^{+} \mid \tilde{\mu}x^{+}.c_{\rightsquigarrow+} \\ & \mid \tilde{\mu}\left[\mathfrak{d}_{1}^{\tau_{+}^{1}}(\overrightarrow{x_{1}}).c_{\rightsquigarrow+}^{1} | \ldots | \mathfrak{d}_{l_{1}^{+}}^{\tau_{+}^{1}}(\overrightarrow{x_{l_{1}^{+}}}).c_{\rightsquigarrow+}^{l_{1}^{+}}\right] \\ & \mid \vdots \\ & \mid \tilde{\mu}\left[\mathfrak{d}_{1}^{\tau_{+}^{n}}(\overrightarrow{x_{1}}).c_{\rightsquigarrow+}^{1} | \ldots | \mathfrak{d}_{l_{n}^{+}}^{\tau_{+}^{n}}(\overrightarrow{x_{l_{n}^{+}}}).c_{\rightsquigarrow+}^{l_{n}^{+}}\right] \end{aligned}$$

and

$$\begin{aligned} s_{+\rightsquigarrow-} ::=\ & \tilde{\mu}x^{+}.c_{\rightsquigarrow-} \\ & \mid \tilde{\mu}\left[\mathfrak{d}_{1}^{\tau_{+}^{1}}(\overrightarrow{x_{1}}).c_{\rightsquigarrow-}^{1} | \ldots | \mathfrak{d}_{l_{1}^{+}}^{\tau_{+}^{1}}(\overrightarrow{x_{l_{1}^{+}}}).c_{\rightsquigarrow-}^{l_{1}^{+}}\right] \\ & \mid \vdots \\ & \mid \tilde{\mu}\left[\mathfrak{d}_{1}^{\tau_{+}^{n}}(\overrightarrow{x_{1}}).c_{\rightsquigarrow-}^{1} | \ldots | \mathfrak{d}_{l_{n}^{+}}^{\tau_{+}^{n}}(\overrightarrow{x_{l_{n}^{+}}}).c_{\rightsquigarrow-}^{l_{n}^{+}}\right] \end{aligned}$$

Fact V.2.16

The grammar given in Figure V.2.2a describes exactly all minimalistic terms.

Proof

By Proposition V.2.14 and induction on the term.

Remark V.2.17

For $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$, there are three additional kinds of judgements

$$c : (\text{?} \vdash), \quad \text{?} \mid \underline{e_+ : +} \vdash, \quad \text{and} \quad \text{?} \mid \underline{e_- : -} \vdash$$

Applying the same method as for $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, we could introduce extra non-terminal symbols $c_{\rightsquigarrow\emptyset}$, $e_{+\rightsquigarrow\emptyset}$, and $e_{-\rightsquigarrow\emptyset}$ for the judgements above. This would yield a grammar that tracks the uses of right weakening rules, which makes it ambiguous: there are two derivations

$$c_{\rightsquigarrow\varepsilon} \to \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}, \star^\varepsilon).c^1_{\rightsquigarrow\varepsilon} | \ldots | \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}, \star^\varepsilon).c^l_{\rightsquigarrow\varepsilon}\right] \to^* \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}, \star^\varepsilon).c^1_{\rightsquigarrow\emptyset} | \ldots | \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}, \star^\varepsilon).c^l_{\rightsquigarrow\emptyset}\right]$$

and

$$c_{\rightsquigarrow\varepsilon} \to c_{\rightsquigarrow\emptyset} \to \tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}, \star^\varepsilon).c^1_{\rightsquigarrow\emptyset} | \ldots | \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}, \star^\varepsilon).c^l_{\rightsquigarrow\emptyset}\right]$$

that correspond to applying $(\vdash \mathrm{w}t)$ before and after $\left(\tau_+^j \vdash\right)$ respectively. This grammar can be made non-ambiguous by explicitly tracking the free stack variables, but the resulting grammar would be fairly tedious to work with.

V.2.3. Properties

Disubstitutions

In $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, we are only interested in some disubstitutions:

Definition V.2.18

A disubstitution is said to be *minimalistic* if its image is contained in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, and it acts non-trivially on at most one stack variable $\star^\varepsilon$.

The $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus is closed under minimalistic disubstiutions:

Fact V.2.19

For any minimalistic term t and minimalistic disubstitution φ, $o[\varphi]$ is minimalistic (resp. intuitionistic).

Proof

By induction on t.

Since expressions have no free stacks variables, a minimalistic disubstitution can always be written as the composition of a value substitution and a stack substitution:

Fact V.2.20

For any minimalistic disubstitution $\varphi = \sigma, \star^{\varepsilon_1} \mapsto s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and minimalistic term t, we have

$$t[\varphi] = t[\sigma][s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}/\star^{\varepsilon_1}]$$

Proof

By induction on t. The base case $t = \star^{\varepsilon}$ is immediate, the base case $t = x^{\varepsilon}$ boils down to the fact that $x^{\varepsilon}[\sigma]$ is an expression and therefore has no stack variable, and in all the remaining cases, the induction hypothesis immediately allows to conclude.

Reductions

Descriptions of the restriction of $\triangleright$ and $\rceil$ to $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ are given in Figures V.2.2b and V.2.2c. Note in particular that these only involve minimalistic disubstitutions, and $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ is therefore closed under the operational reduction $\triangleright$, top-level η-expansion $\rceil$, and η-reduction $\lceil$, and their respective contextual closures $\mathrel{-\!\triangleright}$, $\mathrel{-\!\rceil}$, and $\mathrel{\lceil\!-}$:

Fact V.2.21: Closure of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ under $\mathrel{\lceil\!\rceil\!\triangleright}$

If $t \mathrel{\lceil\!\rceil\!\triangleright} t'$ and t is minimalistic then so is t'.

Proof

Closure under $\triangleright$ follows from closure under minimalistic disubstitution (Fact V.2.19). Closure under $\rceil$ and $\lceil$ is immediate. Closure under their contextual closures $\mathrel{-\!\triangleright}$, $\mathrel{-\!\rceil}$ and $\mathrel{\lceil\!-}$ follows by induction on the derivation.

Thanks to this closure property, disubstitutivity, confluence, postponement and factorization transfer from $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ to $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$:

Fact V.2.22

In $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, the reductions $\triangleright, \rceil, \mathrel{-\!\triangleright}, \mathrel{-\!\rceil}$ are disubstitutive.

Proof

Suppose that t is a minimalistic term such that that $t \twoheadrightarrow t'$ for some reduction $\twoheadrightarrow \in \{\triangleright, \rceil, \mathrel{-\!\triangleright}, \mathrel{-\!\rceil}\}$, and let φ be a minimalistic (resp. intuitionistic) disubstitution. By disubstitutivity in $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ of $\twoheadrightarrow$, we have $c[\varphi] \twoheadrightarrow c'[\varphi]$. By Fact V.2.19, $c[\varphi]$ is minimalistic, and by Fact V.2.21 so is $c'[\varphi]$.

Proposition V.2.23: Confluence of $\rightarrowtriangle$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

In $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, $\rightarrowtriangle$ is confluent .

Proof

By confluence in $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ (Proposition **??**) and closure of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ under $\rightarrowtriangle$ (Fact V.2.21).

Proposition V.2.24: Postponement of $\overset{\neg\circ}{\rightarrowtriangle}$ after $\triangleright$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

In $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, $\overset{\neg\circ}{\rightarrowtriangle}$ postpones after $\triangleright$: if $t \rightarrowtriangle^* t'$ then $t \triangleright^* \overset{\neg\circ}{\rightarrowtriangle}^* t'$.

Proof

By postponement in $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ (Proposition **??**) and closure of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ under $\rightarrowtriangle$ (Fact V.2.21).

Proposition V.2.25: Factorization of $\rightarrowtriangle^*$ as $\triangleright^* \overset{\neg\circ}{\rightarrowtriangle}^*$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

In $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, $\rightarrowtriangle^*$ factorizes as $\rightarrowtriangle^* = \triangleright^* \overset{\neg\circ}{\rightarrowtriangle}^*$.

Proof

By factorization in $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ (Proposition **??**) and closure of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ under $\rightarrowtriangle$ (Fact V.2.21).

V.3. A polarized λ-calculus with focus equivalent to $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$: $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$

In this section, we continue the process of turning the $\mathrm{L}_{\mathrm{p}}^{\vec{\tau}}$ calculus into a λ-calculus by introducing the $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ calculus, a polarized λ-calculus with focus that serves as an alternative syntax for $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ that looks like the λ-calculus with some underlinements $\underline{\cdot}$, and defining two inverse translations that relate them:

$$\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}} \underset{\underleftarrow{\cdot}}{\overset{\underrightarrow{\cdot}}{\rightleftarrows}} \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$$

The $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ calculus is obtained by first building an outside-out description of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, and then simply renaming a few things. All operations operations on $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ are then transferred to $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ by mapping their inputs through $\underrightarrow{\cdot}$ and their outputs through $\underleftarrow{\cdot}$.

V.3.1. The outside-out description of the $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus

The first step towards building the $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ calculus is to build the outside-out description of the $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus[8] described in Figure V.3.1. The basic idea is that every negative stack $s_{-\rightsquigarrow\varepsilon}$ is of the shape

$$s_{-\rightsquigarrow\varepsilon} = \mathfrak{s}^{\tau_-^{j_r}}_{k_r}\Big(\overrightarrow{v_r}, \mathfrak{s}^{\tau_-^{j_{r-1}}}_{k_{r-1}}\Big(\overrightarrow{v_{r-1}}, \dots\, \mathfrak{s}^{\tau_-^{j_2}}_{k_2}\Big(\overrightarrow{v_2}, \mathfrak{s}^{\tau_-^{j_1}}_{k_1}\big(\overrightarrow{v_1}, s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}\big)\Big)\Big)\Big)$$

where $s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}$ is a stack variable $\star^{\varepsilon}$ or a $\widetilde{\mu}$ binder, and can hence be decomposed as

$$s_{-\rightsquigarrow\varepsilon} = \mathfrak{s}^{\tau_-^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^-) \bigcirc \mathfrak{s}^{\tau_-^{j_{r-1}}}_{k_{r-1}}(\overrightarrow{v_{r-1}}, \star^-) \bigcirc \cdots \bigcirc \mathfrak{s}^{\tau_-^{j_2}}_{k_2}(\overrightarrow{v_2}, \star^-) \bigcirc \mathfrak{s}^{\tau_-^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}}) \bigcirc s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}$$

where $\bigcirc$ is the associative operation defined by

$$s^{\blacktriangledown}_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc s^{\blacktriangle}_{\varepsilon_2\rightsquigarrow\varepsilon_3} \stackrel{\text{def}}{\equiv} s^{\blacktriangledown}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\big[s^{\blacktriangle}_{\varepsilon_2\rightsquigarrow\varepsilon_3}/\star^{\varepsilon_2}\big]$$

The (inside-out) syntax of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ builds stacks from right-to-left as

$$s_{-\rightsquigarrow\varepsilon} = \mathfrak{s}^{\tau_-^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^-) \bigcirc \Big(\mathfrak{s}^{\tau_-^{j_{r-1}}}_{k_{r-1}}(\overrightarrow{v_{r-1}}, \star^-) \bigcirc \Big(\cdots \bigcirc \Big(\mathfrak{s}^{\tau_-^{j_2}}_{k_2}(\overrightarrow{v_2}, \star^-) \bigcirc \Big(\mathfrak{s}^{\tau_-^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}}) \bigcirc s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}\Big)\Big)\Big)\Big)$$

i.e. as

$$s_{-\rightsquigarrow\varepsilon} = \mathfrak{s}^{\tau_-^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^{\varepsilon_{j_r,k_r}})\Big[\mathfrak{s}^{\tau_-^{j_{r-1}}}_{k_{r-1}}(\overrightarrow{v_{r-1}}, \star^{\varepsilon_{j_{r-1},k_{r-1}}})\Big[\dots\, \mathfrak{s}^{\tau_-^{j_2}}_{k_2}(\overrightarrow{v_2}, \star^-)\Big[\mathfrak{s}^{\tau_-^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}})\big[s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}/\star^{\varepsilon_{j_1,k_1}}\big]\Big/\star^-\Big] \dots \Big/\star^-\Big]\Big/\star^-\Big]$$

The outside-out description of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ instead builds them from the left-to-right as

$$s_{-\rightsquigarrow\varepsilon} = \Big(\Big(\Big(\Big(\Big(\star^- \bigcirc \mathfrak{s}^{\tau_-^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^-)\Big) \bigcirc \mathfrak{s}^{\tau_-^{j_{r-1}}}_{k_{r-1}}(\overrightarrow{v_{r-1}}, \star^-)\Big) \bigcirc \dots\Big) \bigcirc \mathfrak{s}^{\tau_-^{j_2}}_{k_2}(\overrightarrow{v_2}, \star^-)\Big) \bigcirc \mathfrak{s}^{\tau_-^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}})\Big) \bigcirc s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}$$

i.e. as

$$s_{-\rightsquigarrow\varepsilon} = \star^-\Big[\mathfrak{s}^{\tau_-^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^-)\Big/\star^-\Big]\Big[\mathfrak{s}^{\tau_-^{j_{r-1}}}_{k_{r-1}}(\overrightarrow{v_{r-1}}, \star^-)\Big/\star^-\Big] \dots \Big[\mathfrak{s}^{\tau_-^{j_2}}_{k_2}(\overrightarrow{v_2}, \star^-)\Big/\star^-\Big]\Big[\mathfrak{s}^{\tau_-^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}})\Big/\star^-\Big]\big[s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}/\star^{\varepsilon_{j_1,k_1}}\big]$$

[8] In call-by-name (resp. call-by-value), the λ-calculus with focus $\underline{\lambda}_{\mathrm{n}}^{\rightarrow}$ (resp. $\underline{\lambda}_{\mathrm{v}}^{\rightarrow}$) could be build as an intermediate calculus between $\mathrm{Li}_{\mathrm{n}}^{\rightarrow}$ and $\boldsymbol{\lambda}_{\mathrm{N}}^{\rightarrow}$ (resp. $\mathrm{Li}_{\mathrm{v}}^{\rightarrow}$ and $\boldsymbol{\lambda}_{\mathrm{V}}^{\rightarrow}$). While we could we could similarly build $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ as an intermediate calculus between $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\boldsymbol{\lambda}_{\mathrm{p}}^{\vec{\tau}}$, this would involve a non-trivial amount of guesswork (because $\boldsymbol{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ has not been defined yet), which is why we prefer the more systematic way of doing it described below.

Figure V.3.1: Outside-out description of the $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ calculus

Figure V.3.1.a: Commands, values, and expressions

Negative values / expressions:

$$
\begin{aligned}
v_-, w_-, t_-, u_- ::= {} & x^- \mid \mu\star^-.c_{\rightsquigarrow -} \\
& \mid \mu\left\langle \mathfrak{s}_1^{\tau_-^1}(\overrightarrow{x_1}, \star^{\varepsilon_{1,1}}).c^1_{\rightsquigarrow \varepsilon_{1,1}} \middle| \ldots \middle| \mathfrak{s}_{l_1^-}^{\tau_-^1}(\overrightarrow{x_{l_1^-}}, \star^{\varepsilon_{1,l_1^-}}).c^{l_1^-}_{\rightsquigarrow \varepsilon_{1,l_1^-}} \right\rangle \\
& \mid \vdots \\
& \mid \mu\left\langle \mathfrak{s}_1^{\tau_-^m}(\overrightarrow{x_1}, \star^{\varepsilon_{m,1}}).c^1_{\rightsquigarrow \varepsilon_{m,1}} \middle| \ldots \middle| \mathfrak{s}_{l_m^-}^{\tau_-^m}(\overrightarrow{x_{l_m^-}}, \star^{\varepsilon_{m,l_m^-}}).c^{l_m^-}_{\rightsquigarrow \varepsilon_{m,l_m^-}} \right\rangle
\end{aligned}
$$

Incomplete simple commands:

$$
\begin{aligned}
\breve{c}_{\rightsquigarrow \varepsilon} ::= {} & t_\varepsilon \mid \mathsf{stk}^\varepsilon(\star^\varepsilon) \\
& \mid \underbrace{\breve{c}_{\rightsquigarrow -}\left[\mathfrak{s}_1^{\tau_-^1}(\overrightarrow{v}, \star^{\varepsilon_{1,1}})/\star^-\right]}_{\varepsilon=\varepsilon_{1,1}} \mid \ldots \mid \underbrace{\breve{c}_{\rightsquigarrow -}\left[\mathfrak{s}_{l_1^-}^{\tau_-^1}(\overrightarrow{v}, \star^{\varepsilon_{1,l_1^-}})/\star^-\right]}_{\varepsilon=\varepsilon_{1,l_1^-}} \\
& \mid \vdots \qquad\qquad \mid \ddots \mid \vdots \\
& \mid \underbrace{\breve{c}_{\rightsquigarrow -}\left[\mathfrak{s}_1^{\tau_-^m}(\overrightarrow{v}, \star^{\varepsilon_{m,1}})/\star^-\right]}_{\varepsilon=\varepsilon_{m,1}} \mid \ldots \mid \underbrace{\breve{c}_{\rightsquigarrow -}\left[\mathfrak{s}_{l_m^-}^{\tau_-^m}(\overrightarrow{v}, \star^{\varepsilon_{m,l_m^-}})/\star^-\right]}_{\varepsilon=\varepsilon_{m,l_m^-}}
\end{aligned}
$$

Simple commands:

$$
\mathring{c}_{\rightsquigarrow \varepsilon} ::= \langle \breve{c}_{\rightsquigarrow \varepsilon} \rangle^{\pm}
$$

Positive values:

$$
\begin{aligned}
v_+, w_+ ::= {} & x^+ \\
& \mid \mathfrak{v}_1^{\tau_+^1}(\overrightarrow{v}) \mid \ldots \mid \mathfrak{v}_{l_1^+}^{\tau_+^1}(\overrightarrow{v}) \\
& \mid \vdots \qquad \mid \ddots \mid \vdots \\
& \mid \mathfrak{v}_1^{\tau_+^n}(\overrightarrow{v}) \mid \ldots \mid \mathfrak{v}_{l_n^+}^{\tau_+^n}(\overrightarrow{v})
\end{aligned}
$$

Commands:

$$
\begin{aligned}
c_{\rightsquigarrow \varepsilon} ::= {} & \langle \breve{c}_{\rightsquigarrow \varepsilon} \rangle^{\pm} \mid \langle \breve{c}_{\rightsquigarrow +}[\tilde{\mu}x^+.c_{\rightsquigarrow \varepsilon}/\star^+] \rangle^{\pm} \mid \langle t_- \mid \tilde{\mu}x^-.c_{\rightsquigarrow \varepsilon} \rangle^- \\
& \mid \left\langle \breve{c}_{\rightsquigarrow +}\left[\tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^1}(\overrightarrow{x_1}).c^1_{\rightsquigarrow \varepsilon} \middle| \ldots \middle| \mathfrak{v}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\rightsquigarrow \varepsilon}\right] / \star^+\right] \right\rangle^{\pm} \\
& \mid \vdots \\
& \mid \left\langle \breve{c}_{\rightsquigarrow +}\left[\tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^n}(\overrightarrow{x_1}).c^1_{\rightsquigarrow \varepsilon} \middle| \ldots \middle| \mathfrak{v}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\rightsquigarrow \varepsilon}\right] / \star^+\right] \right\rangle^{\pm}
\end{aligned}
$$

Positive expressions:

$$
t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mu\star^+.c_{\rightsquigarrow +}
$$

Figure V.3.1.b: Stacks and evaluation contexts

Negative simple stacks:

$$
\begin{array}{rcl}
\mathring{s}_{-\leadsto\varepsilon} & ::= & \star^-\}_{\varepsilon=-} \\
& \mid & \underbrace{\mathring{s}_{-\leadsto-}\left[\mathfrak{s}_1^{\tau_-^1}(\vec{v},\star^{\varepsilon_{1,1}})/\star^-\right]}_{\varepsilon=\varepsilon_{1,1}} \mid \dots \mid \underbrace{\mathring{s}_{-\leadsto-}\left[\mathfrak{s}_{l_1^-}^{\tau_-^1}(\vec{v},\star^{\varepsilon_{1,l_1^-}})/\star^-\right]}_{\varepsilon=\varepsilon_{1,l_1^-}} \\
& \mid & \vdots \qquad \mid \ddots \mid \vdots \\
& \mid & \underbrace{\mathring{s}_{-\leadsto-}\left[\mathfrak{s}_1^{\tau_-^m}(\vec{v},\star^{\varepsilon_{m,1}})/\star^-\right]}_{\varepsilon=\varepsilon_{m,1}} \mid \dots \mid \underbrace{\mathring{s}_{-\leadsto-}\left[\mathfrak{s}_{l_m^-}^{\tau_-^m}(\vec{v},\star^{\varepsilon_{m,l_m^-}})/\star^-\right]}_{\varepsilon=\varepsilon_{m,l_m^-}}
\end{array}
$$

Positive simple stacks:

$$
\mathring{s}_{+\leadsto\varepsilon} ::= \star^+\}_{\varepsilon=+}
$$

Negative stacks:

$$
\begin{array}{rcl}
s_{-\leadsto\varepsilon} & ::= & \mathring{s}_{-\leadsto\varepsilon} \mid \mathring{s}_{-\leadsto+}\left[\tilde{\mu}x^+.c_{\leadsto\varepsilon}/\star^+\right] \\
& \mid & \mathring{s}_{-\leadsto+}\left[\tilde{\mu}\left[\mathfrak{d}_1^{\tau_+^1}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon}\mid\dots\mid\mathfrak{d}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\leadsto\varepsilon}\right]/\star^+\right] \\
& \mid & \vdots \\
& \mid & \mathring{s}_{-\leadsto+}\left[\tilde{\mu}\left[\mathfrak{d}_1^{\tau_+^n}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon}\mid\dots\mid\mathfrak{d}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\leadsto\varepsilon}\right]/\star^+\right]
\end{array}
$$

Positive stacks / evaluation contexts:

$$
\begin{array}{rcl}
s_{+\leadsto\varepsilon}, e_{+\leadsto\varepsilon} & ::= & \star^+\}_{\varepsilon=+} \mid \tilde{\mu}x^+.c_{\leadsto\varepsilon} \\
& \mid & \tilde{\mu}\left[\mathfrak{d}_1^{\tau_+^1}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon}\mid\dots\mid\mathfrak{d}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\leadsto\varepsilon}\right] \\
& \mid & \vdots \\
& \mid & \tilde{\mu}\left[\mathfrak{d}_1^{\tau_+^n}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon}\mid\dots\mid\mathfrak{d}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\leadsto\varepsilon}\right]
\end{array}
$$

Negative evaluation contexts:

$$
e_{-\leadsto\varepsilon} ::= \mathsf{stk}^-(s_{-\leadsto\varepsilon}) \mid \tilde{\mu}x^-.c_{\leadsto\varepsilon}
$$

Figure V.3.2: Outside-out description of the $\mathrm{Lm}_{\mathrm{p}}^{\rightarrow\Downarrow\Uparrow\otimes\oplus\&10\top}$ calculus

Figure V.3.2.a: Commands, values, and expressions

Negative values / expressions:

$$\begin{aligned} v_-, w_-, t_-, u_- ::= {} & x^- \mid \mu\star^-.c_{\rightsquigarrow-} \\ & \mid \mu(x^+\cdot\star^-).c_{\rightsquigarrow-} \\ & \mid \mu\langle(\pi_1\cdot\star^-).c^1_{\rightsquigarrow-} | (\pi_2\cdot\star^-).c^2_{\rightsquigarrow-}\rangle \\ & \mid \mu\langle\rangle \\ & \mid \\ & \mid \mu\{\star^+\}.c \end{aligned}$$

Positive values:

$$\begin{aligned} v_+, w_+ ::= {} & x^+ \\ & \mid (v_+ \otimes w_+) \\ & \mid \iota_1(v_+) \mid \iota_2(v_+) \\ & \mid \{v_-\} \\ & \mid () \end{aligned}$$

Positive expressions:

$$t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mu\star^+.c_{\rightsquigarrow+}$$

Incomplete simple commands:

$$\begin{aligned} \breve{c}_{\rightsquigarrow-} ::= {} & t_- | \mathsf{stk}^-(\star^-) \\ & \mid \breve{c}_{\rightsquigarrow-}[v_+\cdot\star^-/\star^-] \\ & \mid \breve{c}_{\rightsquigarrow-}[\pi_1\cdot\star^-/\star^-] \mid \breve{c}_{\rightsquigarrow-}[\pi_2\cdot\star^-/\star^-] \end{aligned}$$

$$\begin{aligned} \breve{c}_{\rightsquigarrow+} ::= {} & t_+ | \mathsf{stk}^+(\star^+) \\ & \mid \breve{c}_{\rightsquigarrow-}[\{\star^+\}/\star^-] \end{aligned}$$

Simple commands:

$$\mathring{c}_{\rightsquigarrow\varepsilon} ::= \langle\breve{c}_{\rightsquigarrow\varepsilon}\rangle^{\pm}$$

Commands:

$$\begin{aligned} c_{\rightsquigarrow\varepsilon} ::= {} & \langle\breve{c}_{\rightsquigarrow\varepsilon}\rangle^{\pm} \mid \langle\breve{c}_{\rightsquigarrow+}[\tilde{\mu}x^+.c_{\rightsquigarrow\varepsilon}/\star^+]\rangle^{\pm} \mid \langle t_- | \tilde{\mu}x^-.c_{\rightsquigarrow\varepsilon}\rangle^{-} \\ & \mid \langle\breve{c}_{\rightsquigarrow+}[\tilde{\mu}(x^+\otimes y^+).c_{\rightsquigarrow\varepsilon}/\star^+]\rangle^{\pm} \\ & \mid \langle\breve{c}_{\rightsquigarrow+}[\tilde{\mu}[\iota_1(x_1^+).c^1_{\rightsquigarrow\varepsilon} | \iota_2(x_2^+).c^2_{\rightsquigarrow\varepsilon}]/\star^+]\rangle^{\pm} \\ & \mid \langle\breve{c}_{\rightsquigarrow+}[\tilde{\mu}\{x^-\}.c_{\rightsquigarrow\varepsilon}/\star^+]\rangle^{\pm} \\ & \mid \langle\breve{c}_{\rightsquigarrow+}[\tilde{\mu}().c_{\rightsquigarrow\varepsilon}/\star^+]\rangle^{\pm} \end{aligned}$$

Figure V.3.2.b: Stacks and evaluation contexts

Negative simple stacks:

$$\mathring{s}_{-\rightsquigarrow-} ::= \star^- \;\mid\; \mathring{s}_{-\rightsquigarrow-}[\upsilon_+ \cdot \star^- / \star^-] \;\mid\; \mathring{s}_{-\rightsquigarrow-}[\pi_1 \cdot \star^- / \star^-] \;\mid\; \mathring{s}_{-\rightsquigarrow-}[\pi_2 \cdot \star^- / \star^-]$$

$$\mathring{s}_{-\rightsquigarrow+} ::= \;\mid\; \mathring{s}_{-\rightsquigarrow-}[\{\star^+\}/\star^-]$$

Positive simple stacks:

$$\mathring{s}_{+\rightsquigarrow+} ::= \star^+$$

$$\mathring{s}_{+\rightsquigarrow-} ::=$$

Negative stacks:

$$s_{-\rightsquigarrow\varepsilon} ::= \mathring{s}_{-\rightsquigarrow\varepsilon} \;\mid\; \mathring{s}_{-\rightsquigarrow+}[\tilde{\mu}x^+.c_{\rightsquigarrow\varepsilon}/\star^+] \;\mid\; \mathring{s}_{-\rightsquigarrow+}[\tilde{\mu}(x^+ \otimes y^+).c_{\rightsquigarrow\varepsilon}/\star^+] \;\mid\; \mathring{s}_{-\rightsquigarrow+}[\tilde{\mu}[\iota_1(x_1^+).c^1_{\rightsquigarrow\varepsilon} | \iota_2(x_2^+).c^2_{\rightsquigarrow\varepsilon}]/\star^+] \;\mid\; \mathring{s}_{-\rightsquigarrow+}[\tilde{\mu}\{x^-\}.c_{\rightsquigarrow\varepsilon}/\star^+] \;\mid\; \mathring{s}_{-\rightsquigarrow+}[\tilde{\mu}().c_{\rightsquigarrow\varepsilon}/\star^+]$$

Positive stacks / evaluation contexts:

$$s_{+\rightsquigarrow\varepsilon}, e_{+\rightsquigarrow\varepsilon} ::= \star^+\}\varepsilon{=}+ \;\mid\; \tilde{\mu}x^+.c_{\rightsquigarrow\varepsilon} \;\mid\; \tilde{\mu}(x^+ \otimes y^+).c_{\rightsquigarrow\varepsilon} \;\mid\; \tilde{\mu}[\iota_1(x_1^+).c^1_{\rightsquigarrow\varepsilon} | \iota_2(x_2^+).c^2_{\rightsquigarrow\varepsilon}] \;\mid\; \tilde{\mu}\{x^-\}.c_{\rightsquigarrow\varepsilon} \;\mid\; \tilde{\mu}().c_{\rightsquigarrow\varepsilon}$$

Negative evaluation contexts:

$$e_{-\rightsquigarrow\varepsilon} ::= \mathsf{stk}^-(s_{-\rightsquigarrow\varepsilon}) \;\mid\; \tilde{\mu}x^-.c_{\rightsquigarrow\varepsilon}$$

To turn this into a BNF-like description, we introduce simple stacks

$$\mathring{s}_{\_\rightsquigarrow\varepsilon} = \star^{-}\left[\mathfrak{s}^{\tau_{-}^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^{-})\middle/\star^{-}\right]\left[\mathfrak{s}^{\tau_{-}^{j_{r-1}}}_{k_{r-1}}(\overrightarrow{v_{r-1}}, \star^{-})\middle/\star^{-}\right] \dots \left[\mathfrak{s}^{\tau_{-}^{j_2}}_{k_2}(\overrightarrow{v_2}, \star^{-})\middle/\star^{-}\right]\left[\mathfrak{s}^{\tau_{-}^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}})\middle/\star^{-}\right]$$

which are stacks $s_{\_\rightsquigarrow\varepsilon}$ for which $s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}$ is not a $\tilde{\mu}$ binder, i.e. those whose outside-out description can be extended by yet another disubsitution. The same decompositions are used for commands, with incomplete simple commands $\check{c}_{\rightsquigarrow\varepsilon_2} = t_{\varepsilon_1}|\mathring{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ (i.e. commands $\langle t_{\varepsilon_1}|\mathring{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\rangle^{\varepsilon_1}$ formed with a simple stack $\mathring{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ minus the surrounding $\langle\cdot\rangle^{\varepsilon_1}$) playing the same role as simple stacks for stacks.

Remark V.3.1

Since negative incomplete simple commands are built as

$$\check{c}_{\rightsquigarrow -} = (t_{\varepsilon}|\mathring{s}_{\varepsilon\rightsquigarrow -})\left[\mathfrak{s}^{\tau_{-}^{j_r}}_{k_r}(\overrightarrow{v_r}, \star^{-})\middle/\star^{-}\right] \dots \left[\mathfrak{s}^{\tau_{-}^{j_1}}_{k_1}(\overrightarrow{v_1}, \star^{\varepsilon_{j_1,k_1}})\middle/\star^{-}\right]\left[s^0_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}/\star^{\varepsilon_{j_1,k_1}}\right]$$

the polarity ε may only be available deep in $\check{c}_{\rightsquigarrow -}$ when we eventually for the command $\langle\check{c}_{\rightsquigarrow -}\rangle^{\varepsilon}$. While we could modify the description to keep track of ε, this would make the grammar more tedious to work with, so we instead write $\underline{\pm}$ for the to-be-inferred polarity ε. This has no practical consequences because the polarity annotations on commands were superfluous in the first place: the syntax of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ remains unambiguous without them.

Note that the grammar of commands, values, and expressions no longer refer to stacks and evaluation contexts. Since stacks are required to define the reduction (and describing evaluation contexts does not take much space), we nevertheless keep them in the syntax.

While not necessary, defining simple positive stacks sometimes allows to treat both polarities at once, and may prove useful for some applications[9].

V.3.2. The $\lambda_{\mathrm{p}}^{\vec{\tau}}$ calculus

The $\lambda_{\mathrm{p}}^{\vec{\tau}}$ calculus (resp. $\lambda_{\mathrm{p}}^{\to\Downarrow\Uparrow\otimes\oplus\&1\top}$) described in Figure V.3.3a (resp. Figure V.3.4a) is obtained from the outside-out description of $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ (resp. $\mathrm{Lm}_{\mathrm{p}}^{\to\Downarrow\Uparrow\otimes\oplus\&1\top}$) by simply renaming things (see ⚠ for details): $t_{\varepsilon}|\star^{\varepsilon}$ is replaced by $\underline{t_{\varepsilon}}$, $\langle\cdot\rangle^{\varepsilon}$ by $\mathsf{com}^{\varepsilon}(\cdot)$, $\mu\star^{\varepsilon}.c_{\rightsquigarrow\varepsilon}$ by $\mathsf{ctot}^{\varepsilon}(c_{\rightsquigarrow\varepsilon})$ (which stands for "command to term"), $\star^{\varepsilon}$ by $\square^{\varepsilon}$, $\tilde{\mu}x^{\varepsilon}.c_{\rightsquigarrow\varepsilon}$ by $\mathsf{let}\ x^{\varepsilon} := \square^{\varepsilon}\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon}$, and $\tilde{\mu}[\dots]$ by $\mathsf{match}\ \square^{+}\ \mathsf{with}[\dots]$. Uses of disubstitutions are replaced by plugging: $\check{c}_{\rightsquigarrow\varepsilon_1}[s_{\varepsilon_1\rightsquigarrow\varepsilon_2}/\star^{\varepsilon_1}]$ becomes $\mathbb{S}_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{\check{c}_{\rightsquigarrow\varepsilon_1}}$ and $\mathring{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}[s_{\varepsilon_2\rightsquigarrow\varepsilon_3}/\star^{\varepsilon_2}]$ becomes $\mathbb{S}_{\varepsilon_2\rightsquigarrow\varepsilon_3}\boxed{\mathring{\mathbb{S}}_{\varepsilon_1\rightsquigarrow\varepsilon_2}}$.

[9] Very loosely, in a simple command, the stack only acts as data for the expression, and the expression can choose to keep the control for the remainder of the computation, while in non-simple commands the expression is forced to eventually (diverge or) give back the control to the stack or evaluation context.

Notation V.3.2

While adding $\mathsf{ctot}^{\varepsilon}$ and $\mathsf{com}^{\varepsilon}$ makes the translations between $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ (and a few other things) easier to study, these are often superfluous when writing concrete expressions of $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ so we often leave them implicit, i.e. write $c_{\rightsquigarrow\varepsilon}$ for $\mathsf{ctot}^{\varepsilon}(c_{\rightsquigarrow\varepsilon})$ and $@_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{\underline{t_{\varepsilon_1}}}$ for $\mathsf{com}^{\varepsilon_1}\left(@_{\varepsilon_1\rightsquigarrow\varepsilon_2}\boxed{\underline{t_{\varepsilon_1}}}\right)$. However, the underlinement $\underline{\ }$ is essential and will always be explicit.

For negative stacks, things are slightly more more complex. In the instances of the calculus that we have seen earlier, the stack constructors of $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ and $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ were not the same. For example, for $\rightarrow$, we had the stack constructor $\mathfrak{s}_1^{\rightarrow}(v_+, s_{-\rightsquigarrow\varepsilon}) = v_+ \cdot s_{-\rightsquigarrow\varepsilon}$ which corresponded to the application $\mathbb{S}_{-\rightsquigarrow\varepsilon}\boxed{\square^- v_+}$; for $\Uparrow$, we had the stack constructor $\mathfrak{s}_1^{\Uparrow}(s_{-\rightsquigarrow\varepsilon}) = \{s_{+\rightsquigarrow\varepsilon}\}$ which corresponded to $\mathbb{S}_{+\rightsquigarrow\varepsilon}\boxed{\mathsf{unfreeze}(\square^-)}$; and for $\&$, we had two stack constructors $\mathfrak{s}_1^{\&}(s_{-\rightsquigarrow\varepsilon}) = \pi_1 \cdot s_{-\rightsquigarrow\varepsilon}$ and $\mathfrak{s}_2^{\&}(s_{-\rightsquigarrow\varepsilon}) = \pi_2 \cdot s_{-\rightsquigarrow\varepsilon}$ that corresponded to $\mathbb{S}_{-\rightsquigarrow\varepsilon}\boxed{\pi_1(\square^-)}$ and $\mathbb{S}_{-\rightsquigarrow\varepsilon}\boxed{\pi_2(\square^-)}$ respectively. Note that for each one of these examples, the negative stack constructor $\mathfrak{s}_k^{\tau_-^j}(\vec{v}, s_{\varepsilon_{j,k}\rightsquigarrow\varepsilon})$ becomes $\mathbb{S}_{\varepsilon_{j,k}\rightsquigarrow\varepsilon}\boxed{\mathfrak{s}_k^{\tau_-^j}(\vec{v}, \square^-)}$, where $\mathfrak{s}_1^{\rightarrow}(v_+, \square^-) = \square^- v_+$, $\mathfrak{s}_1^{\Uparrow}(\square^-) = \mathsf{unfreeze}(\square^-)$, $\mathfrak{s}_1^{\&}(\square^-) = \pi_1(\square^-)$ and $\mathfrak{s}_2^{\&}(\square^-) = \pi_2(\square^-)$. In a way, the stack constructor $\mathfrak{s}_k^{\tau_-^j}$ is connected to some values $\vec{v}$, the context $\mathbb{K}$ that surrounds it, and a stack $s_{\varepsilon_{j,k}\rightsquigarrow\varepsilon}$, and $\mathfrak{s}_k^{\tau_-^j}$ is connected to the same things with the positions of $\mathbb{K}$ and $s_{\varepsilon_{j,k}\rightsquigarrow\varepsilon}$ swapped. Very roughly,

$$\mathbb{K}\boxed{\mathfrak{s}_k^{\tau_-^j}(\vec{v}, s_{\varepsilon_{j,k}\rightsquigarrow\varepsilon})} \text{ becomes } \mathbb{S}_{\varepsilon_{j,k}\rightsquigarrow\varepsilon}\boxed{\mathfrak{s}_k^{\tau_-^j}(\vec{v}, \mathbb{K})}.$$

This does not hold for all $\mathbb{K}$, but does hold for those of the shape $\mathbb{K} = \mathfrak{s}_{k_r}^{\tau_-^{j_r}}\left(\vec{v_r}, \dots \mathfrak{s}_{k_1}^{\tau_-^{j_1}}(\vec{v_1}, \square^{\varepsilon_{k_1,l_{j_1}}})\right)$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and the corresponding $\mathfrak{s}_{k_1}^{\tau_-^{j_1}}\left(\vec{v_1}, \dots \mathfrak{s}_{k_r}^{\tau_-^{j_r}}(\vec{v_r}, \square^-)\right)$ in $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$:

$$\mathfrak{s}_{k_r}^{\tau_-^{j_r}}\left(\vec{v_r}, \dots \mathfrak{s}_{k_1}^{\tau_-^{j_1}}(\vec{v_1}, s_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon})\right) \text{ becomes } \mathbb{S}_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}\boxed{\mathfrak{s}_{k_1}^{\tau_-^{j_1}}\left(\vec{v_1}, \dots \mathfrak{s}_{k_r}^{\tau_-^{j_r}}(\vec{v_r}, \square^-)\right)}.$$

Combining these with expressions to form commands, we get that

$$\left\langle t_- \middle| \mathfrak{s}_{k_r}^{\tau_-^{j_r}}\left(\vec{v_r}, \dots \mathfrak{s}_{k_1}^{\tau_-^{j_1}}(\vec{v_1}, s_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon})\right)\right\rangle^- \text{ becomes } \mathsf{com}^-\left(\mathbb{S}_{\varepsilon_{j_1,k_1}\rightsquigarrow\varepsilon}\boxed{\mathfrak{s}_{k_1}^{\tau_-^{j_1}}\left(\vec{v_1}, \dots \mathfrak{s}_{k_r}^{\tau_-^{j_r}}(\vec{v_r}, \underline{t_-})\right)}\right)$$

Since they interact with negative stacks $\mathfrak{s}_k^{\tau_-^j}(\vec{x_k}, \star^{\varepsilon_{j,k}})$ that become $\mathfrak{s}_k^{\tau_-^j}(\vec{x_k}, \square^-)$, negative expressions also need to be modified:

$$\mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\vec{x_1}, \star^{\varepsilon_{j,1}}).c_{\rightsquigarrow\varepsilon_{j,1}} \middle| \dots \middle| \mathfrak{s}_l^{\tau_-^j}(\vec{x_l}, \star^{\varepsilon_{j,l}}).c_{\rightsquigarrow\varepsilon_{j,l}} \right\rangle$$

becomes

$$\lambda\left\langle \mathfrak{s}_1^{\tau_-^j}(\vec{x_1}, \blacksquare^-).c_{\rightsquigarrow\varepsilon_{j,1}} \middle| \dots \middle| \mathfrak{s}_l^{\tau_-^j}(\vec{x_l}, \blacksquare^-).c_{\rightsquigarrow\varepsilon_{j,l}} \right\rangle$$

where we write $\blacksquare^-$ instead of $\square^-$ to to emphasize that it is part of the syntax of expressions, and not a hole. The $\blacksquare^-$ can be though of as marking the spot where the $\lambda\langle\dots\rangle$ is in the surrounding stack. In $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, the reduction

$$\langle \mu(x^+ \cdot \star^-).\langle t_- | w_+ \cdot \star^- \rangle^- | v_+^1 \cdot v_+^2 \cdot \star^- \rangle^- \rhd_{\rightarrow} \langle t_- | w_+ \cdot \star^- \rangle^- [\varphi] = \langle t_-[v_+^1/x^+] | (w_+[v_+^1/x^+]) \cdot v_+^2 \cdot \star^- \rangle^-$$

builds the disubstitution

$$\varphi = x^+ \mapsto v^1_+, \star^+ \mapsto v^2_+ \cdot \star^-$$

by matching the stack pattern $\mathfrak{s}^{\to}_1(x^+, \star^-) = x^+ \cdot \star^-$ against the stack $\mathfrak{s}^{\to}_1(v_+, \mathfrak{s}^{\to}_1(w_+, \star^-)) = v^1_+ \cdot v^2_+ \cdot \star^-$. In $\underline{\lambda}^{\vec{\tau}}_{\mathrm{p}}$, this corresponds to the reduction

$$\underbrace{\lambda\langle(\blacksquare^- x^+).\underbrace{t_-}w_+\rangle} v^1_+ v^2_+ \triangleright_{\to} (\underbrace{t_-}w_+)[\varphi] = (\square^- v^2_+)\boxed{(\underbrace{t_-}w_+)[v^1_+/x^+]} = \underbrace{t_-[v^1_+/x^+]}(w_+[v^1_+/x^+])v^2_+$$

(where $\lambda\langle(\blacksquare^- x^-).c_{\sim-}\rangle$ is a heavy way of writing $\lambda x^+.c_{\sim-}$) building the disubstitution

$$\varphi = (x^+ \mapsto v^1_+, \square^- v^2_+)$$

by "matching" the stack "pattern" $\mathfrak{s}^{\to}_k(x^+, \blacksquare^-) = \blacksquare^- x^+$ with the stack $\mathfrak{s}^{\to}_k(v^2_-, \mathfrak{s}^{\to}_k(v^1_-, \square^-)) = \square^- v^1_- v^2_-$ by trying to superpose $\blacksquare^-$ and $\square^-$. This leads to $\mathfrak{s}^{\to}_k(x^+, \blacksquare^-) = \blacksquare^- x^+$ being matched with the inner part $\mathfrak{s}^{\to}_k(v^1_-, \square^-) = \square^- v^1_-$ of the stack, which yields $x^+ \mapsto v^1_-$ and a remaining outer part of the stack $\mathfrak{s}^{\to}_k(v^2_-, \square^-) = \square^- v^2_+$ in which the result is plugged.

Figure V.3.3: The $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ calculus

Figure V.3.3.a: Commands, values, and expressions

Negative values / expressions:

$$\begin{aligned} v_-, w_-, t_-, u_- ::=\ & x^- \mid \mathsf{ctot}^-(c_{\rightsquigarrow -}) \\ & \mid \lambda\left\langle \mathfrak{d}_1^{\tau_-^1}(\overrightarrow{x_1}, \blacksquare^-).c^1_{\rightsquigarrow \varepsilon_{1,1}} \mid \dots \mid \mathfrak{d}_{l_1^-}^{\tau_-^1}(\overrightarrow{x_{l_1^-}}, \blacksquare^-).c^{l_1^-}_{\rightsquigarrow \varepsilon_{1,l_1^-}} \right\rangle \\ & \mid \vdots \\ & \mid \lambda\left\langle \mathfrak{d}_1^{\tau_-^m}(\overrightarrow{x_1}, \blacksquare^-).c^1_{\rightsquigarrow \varepsilon_{m,1}} \mid \dots \mid \mathfrak{d}_{l_m^-}^{\tau_-^m}(\overrightarrow{x_{l_m^-}}, \blacksquare^-).c^{l_m^-}_{\rightsquigarrow \varepsilon_{m,l_m^-}} \right\rangle \end{aligned}$$

Incomplete simple commands:

$$\begin{aligned} \breve{c}_{\rightsquigarrow \varepsilon} ::=\ & \underbrace{t_+}\}_{\varepsilon=+} \mid \mathsf{instk}^-(\underbrace{t_-})\}_{\varepsilon=-} \\ & \mid \underbrace{\mathfrak{d}_1^{\tau_-^1}(\vec{v}, \breve{c}_{\rightsquigarrow -})}_{\varepsilon=\varepsilon_{1,1}} \mid \dots \mid \underbrace{\mathfrak{d}_{l_1^-}^{\tau_-^1}(\vec{v}, \breve{c}_{\rightsquigarrow -})}_{\varepsilon=\varepsilon_{1,l_1^-}} \\ & \mid \vdots \qquad\qquad \mid \ddots \mid \vdots \\ & \mid \underbrace{\mathfrak{d}_1^{\tau_-^m}(\vec{v}, \breve{c}_{\rightsquigarrow -})}_{\varepsilon=\varepsilon_{m,1}} \mid \dots \mid \underbrace{\mathfrak{d}_{l_m^-}^{\tau_-^m}(\vec{v}, \breve{c}_{\rightsquigarrow -})}_{\varepsilon=\varepsilon_{m,l_m^-}} \end{aligned}$$

Simple commands:

$$\mathring{c}_{\rightsquigarrow \varepsilon} ::= \mathsf{com}^{\pm}(\breve{c}_{\rightsquigarrow \varepsilon})$$

Positive values:

$$\begin{aligned} v_+, w_+ ::=\ & x^+ \\ & \mid \mathfrak{d}_1^{\tau_+^1}(\vec{v}) \mid \dots \mid \mathfrak{d}_{l_1^+}^{\tau_+^1}(\vec{v}) \\ & \mid \vdots \qquad \mid \ddots \mid \vdots \\ & \mid \mathfrak{d}_1^{\tau_+^n}(\vec{v}) \mid \dots \mid \mathfrak{d}_{l_n^+}^{\tau_+^n}(\vec{v}) \end{aligned}$$

Commands:

$$\begin{aligned} c_{\rightsquigarrow \varepsilon} ::=\ & \mathsf{com}^{\pm}(\breve{c}_{\rightsquigarrow \varepsilon}) \mid \mathsf{com}^{\pm}(\mathsf{let}\ x^+ := \breve{c}_{\rightsquigarrow +}\ \mathsf{in}\ c_{\rightsquigarrow \varepsilon}) \mid \mathsf{com}^-(\mathsf{let}\ x^- := \underbrace{t_-}\ \mathsf{in}\ c_{\rightsquigarrow \varepsilon}) \\ & \mid \mathsf{com}^{\pm}\left(\mathsf{match}\ \breve{c}_{\rightsquigarrow +}\ \mathsf{with}\left[\mathfrak{d}_1^{\tau_+^1}(\overrightarrow{x_1}).c^1_{\rightsquigarrow \varepsilon} \mid \dots \mid \mathfrak{d}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\rightsquigarrow \varepsilon}\right]\right) \\ & \mid \vdots \\ & \mid \mathsf{com}^{\pm}\left(\mathsf{match}\ \breve{c}_{\rightsquigarrow +}\ \mathsf{with}\left[\mathfrak{d}_1^{\tau_+^n}(\overrightarrow{x_1}).c^1_{\rightsquigarrow \varepsilon} \mid \dots \mid \mathfrak{d}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\rightsquigarrow \varepsilon}\right]\right) \end{aligned}$$

Positive expressions:

$$t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mathsf{ctot}^+(c_{\rightsquigarrow +})$$

Figure V.3.3.b: Stacks and evaluation contexts

Negative simple stacks:

$$\mathring{\mathbb{S}}_{-\rightsquigarrow\varepsilon} ::= \square^{-}\}\,\varepsilon{=}-$$

$$\begin{array}{l} \mid \underbrace{\mathfrak{d}_{1}^{\tau_{-}^{1}}(\overrightarrow{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{1,1}} \mid \dots \mid \underbrace{\mathfrak{d}_{l_{1}^{-}}^{\tau_{-}^{1}}(\overrightarrow{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{1,l_{1}^{-}}} \\ \mid \vdots \qquad \mid \ddots \mid \vdots \\ \mid \underbrace{\mathfrak{d}_{1}^{\tau_{-}^{m}}(\overrightarrow{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{m,1}} \mid \dots \mid \underbrace{\mathfrak{d}_{l_{m}^{-}}^{\tau_{-}^{m}}(\overrightarrow{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{m,l_{m}^{-}}} \end{array}$$

Positive simple stacks:

$$\mathring{\mathbb{S}}_{+\rightsquigarrow\varepsilon} ::= \square^{+}\}\,\varepsilon{=}+$$

Negative stacks:

$$\begin{array}{l} \mathbb{S}_{-\rightsquigarrow\varepsilon} ::= \mathring{\mathbb{S}}_{-\rightsquigarrow\varepsilon} \mid \mathsf{let}\ x^{+} := \mathring{\mathbb{S}}_{-\rightsquigarrow +}\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon} \\ \quad \mid \mathsf{match}\ \mathring{\mathbb{S}}_{-\rightsquigarrow +}\ \mathsf{with}\left[\mathfrak{v}_{1}^{\tau_{+}^{1}}(\overrightarrow{x_{1}}).c^{1}_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_{1}^{+}}^{\tau_{+}^{1}}(\overrightarrow{x_{l_{1}^{+}}}).c^{l_{1}^{+}}_{\rightsquigarrow\varepsilon}\right] \\ \quad \mid \vdots \\ \quad \mid \mathsf{match}\ \mathring{\mathbb{S}}_{-\rightsquigarrow +}\ \mathsf{with}\left[\mathfrak{v}_{1}^{\tau_{+}^{n}}(\overrightarrow{x_{1}}).c^{1}_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_{n}^{+}}^{\tau_{+}^{n}}(\overrightarrow{x_{l_{n}^{+}}}).c^{l_{n}^{+}}_{\rightsquigarrow\varepsilon}\right] \end{array}$$

Positive stacks / evaluation contexts:

$$\begin{array}{l} \mathbb{S}_{+\rightsquigarrow\varepsilon}, \mathbb{e}_{+\rightsquigarrow\varepsilon} ::= \square^{+}\}\,\varepsilon{=}+ \mid \mathsf{let}\ x^{+} := \square^{+}\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon} \\ \quad \mid \mathsf{match}\ \square^{+}\ \mathsf{with}\left[\mathfrak{v}_{1}^{\tau_{+}^{1}}(\overrightarrow{x_{1}}).c^{1}_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_{1}^{+}}^{\tau_{+}^{1}}(\overrightarrow{x_{l_{1}^{+}}}).c^{l_{1}^{+}}_{\rightsquigarrow\varepsilon}\right] \\ \quad \mid \vdots \\ \quad \mid \mathsf{match}\ \square^{+}\ \mathsf{with}\left[\mathfrak{v}_{1}^{\tau_{+}^{n}}(\overrightarrow{x_{1}}).c^{1}_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_{n}^{+}}^{\tau_{+}^{n}}(\overrightarrow{x_{l_{n}^{+}}}).c^{l_{n}^{+}}_{\rightsquigarrow\varepsilon}\right] \end{array}$$

Negative evaluation contexts:

$$\mathbb{e}_{-\rightsquigarrow\varepsilon} ::= \mathbb{S}_{-\rightsquigarrow\varepsilon}\,\boxed{\mathsf{instk}^{-}(\square^{-})} \mid \mathsf{let}\ x^{-} := \square^{-}\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon}$$

Figure V.3.3.c: Operational reduction

$$\mathsf{com}^{\varepsilon_1}\left(\mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \,\middle|\, \boxed{\underline{c_{\rightsquigarrow \varepsilon_1}}}\right) \rhd_{\mu} \mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2})$$

$$\mathsf{com}^{\varepsilon_1}(\mathsf{let}\ x^{\varepsilon_1} := \underline{v_{\varepsilon_1}}\ \mathsf{in}\ c_{\rightsquigarrow \varepsilon_2}) \rhd_{\tilde{\mu}} c_{\rightsquigarrow \varepsilon_2}[v_{\varepsilon_1}/x^{\varepsilon_1}]$$

$$\mathsf{com}^{-}\left(\mathbb{S}_{\varepsilon_{j,k} \rightsquigarrow \varepsilon} \,\middle|\, \boxed{\mathfrak{v}_k^{\tau_-^j}\left(\vec{v}, \underline{\lambda\left\langle \mathfrak{v}_1^{\tau_-^j}(\vec{x_1}, \blacksquare^-).c^1_{\rightsquigarrow \varepsilon_{j,1}} \mid \ldots \mid \mathfrak{v}_l^{\tau_-^j}(\vec{x_l}, \blacksquare^-).c^l_{\rightsquigarrow \varepsilon_{j,l}} \right\rangle}\right)}\right) \rhd_{\tau_-^j} \mathsf{defer}(c^k_{\rightsquigarrow \varepsilon_{j,k}}[\vec{v}/\vec{x}], \mathbb{S}_{\varepsilon_{j,k} \rightsquigarrow \varepsilon})$$

$$\mathsf{com}^{+}\left(\mathsf{match}\ \underline{\mathfrak{v}_k^{\tau_+^j}(\vec{v})}\ \mathsf{with}\left[\mathfrak{v}_1^{\tau_+^j}(\vec{x_1}).c^1_{\rightsquigarrow \varepsilon} \mid \ldots \mid \mathfrak{v}_l^{\tau_+^j}(x_l).c^l_{\rightsquigarrow \varepsilon}\right]\right) \rhd_{\tau_+^j} c^k_{\rightsquigarrow \varepsilon}[\vec{v}/\vec{x_k}]$$

$$\rhd \stackrel{\text{def}}{\equiv} \rhd_{\tilde{\mu}} \cup \rhd_{\tilde{\mu}} \cup \left(\bigcup_j \rhd_{\tau_-^j}\right) \cup \left(\bigcup_j \rhd_{\tau_+^j}\right) \qquad \rhd_{\mathsf{let}} \stackrel{\text{def}}{\equiv} \rhd_{\tilde{\mu}}$$

Figure V.3.3.d: η-expansion

$$t_{\varepsilon} \fatsemi_{\mu} \mathsf{ctot}^{\varepsilon}(\underline{t_{\varepsilon}})$$

$$\mathcal{e}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \fatsemi_{\tilde{\mu}} \mathsf{let}\ x^{\varepsilon_1} := \square^{\varepsilon_1}\ \mathsf{in}\ \mathcal{e}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \boxed{x^{\varepsilon_1}} \qquad \text{if } x^{\varepsilon_1} \text{ fresh w.r.t. } \mathcal{e}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$$

$$v_{-} \fatsemi_{\tau_-^j} \lambda \left\langle \begin{array}{c} \mathfrak{v}_1^{\tau_-^j}(\vec{x_1}, \blacksquare^-).\mathfrak{v}_1^{\tau_+^j}(\vec{x_1}, v_-) \\ \vdots \\ \mathfrak{v}_l^{\tau_-^j}(\vec{x_l}, \blacksquare^-).\mathfrak{v}_l^{\tau_+^j}(\vec{x_l}, v_-) \end{array} \right\rangle \qquad \text{if } \vec{x_1}, \ldots, \vec{x_l} \text{ fresh w.r.t. } v_-$$

$$\mathbb{S}_{+ \rightsquigarrow \varepsilon} \fatsemi_{\tau_+^j} \mathsf{match}\ \square^{+}\ \mathsf{with} \left[\begin{array}{c} \mathfrak{v}_1^{\tau_+^j}(\vec{x_1}).\,\mathbb{S}_{+ \rightsquigarrow \varepsilon} \boxed{\mathfrak{v}_1^{\tau_+^j}(\vec{x_1})} \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\vec{x_l}).\,\mathbb{S}_{+ \rightsquigarrow \varepsilon} \boxed{\mathfrak{v}_l^{\tau_+^j}(\vec{x_l})} \end{array} \right] \qquad \text{if } \vec{x_1}, \ldots, \vec{x_l} \text{ fresh w.r.t. } \mathbb{S}_{+ \rightsquigarrow \varepsilon}$$

$$\fatsemi \stackrel{\text{def}}{\equiv} \fatsemi_{\tilde{\mu}} \cup \fatsemi_{\tilde{\mu}} \cup \left(\bigcup_j \fatsemi_{\tau_-^j}\right) \cup \left(\bigcup_j \fatsemi_{\tau_+^j}\right)$$

Figure V.3.4: The $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ calculus

Figure V.3.4.a: Commands, values, and expressions

Negative values / expressions:

$$v_-, w_-, t_-, u_- ::= x^- \mid \mathsf{ctot}^-(c_{\leadsto -})$$
$$\mid \lambda\left\langle \mathfrak{d}_1^{\tau_-^1}(\overrightarrow{x_1}, \blacksquare^-).c^1_{\leadsto\varepsilon_{1,1}} \,|\dots|\, \mathfrak{d}_{l_1^-}^{\tau_-^1}(\overrightarrow{x_{l_1^-}}, \blacksquare^-).c^{l_1^-}_{\leadsto\varepsilon_{1,l_1^-}} \right\rangle$$
$$\mid \vdots$$
$$\mid \lambda\left\langle \mathfrak{d}_1^{\tau_-^m}(\overrightarrow{x_1}, \blacksquare^-).c^1_{\leadsto\varepsilon_{m,1}} \,|\dots|\, \mathfrak{d}_{l_m^-}^{\tau_-^m}(\overrightarrow{x_{l_m^-}}, \blacksquare^-).c^{l_m^-}_{\leadsto\varepsilon_{m,l_m^-}} \right\rangle$$

Positive values:

$$v_+, w_+ ::= x^+$$
$$\mid \mathfrak{d}_1^{\tau_+^1}(\vec{v}) \mid \dots \mid \mathfrak{d}_{l_1^+}^{\tau_+^1}(\vec{v})$$
$$\mid \vdots \qquad \mid \ddots \mid \vdots$$
$$\mid \mathfrak{d}_1^{\tau_+^n}(\vec{v}) \mid \dots \mid \mathfrak{d}_{l_n^+}^{\tau_+^n}(\vec{v})$$

Positive expressions:

$$t_+, u_+ ::= \mathsf{val}^+(v_+) \mid \mathsf{ctot}^+(c_{\leadsto +})$$

Incomplete simple commands:

$$\check{c}_{\leadsto\varepsilon} ::= \underline{t_+}\}_{\varepsilon=+} \mid \mathsf{instk}^-(\underline{t_-})\}_{\varepsilon=-}$$
$$\mid \underbrace{\mathfrak{d}_1^{\tau_-^1}(\vec{v}, \check{c}_{\leadsto -})}_{\varepsilon=\varepsilon_{1,1}} \mid \dots \mid \underbrace{\mathfrak{d}_{l_1^-}^{\tau_-^1}(\vec{v}, \check{c}_{\leadsto -})}_{\varepsilon=\varepsilon_{1,l_1^-}}$$
$$\mid \vdots \qquad \mid \ddots \mid \vdots$$
$$\mid \underbrace{\mathfrak{d}_1^{\tau_-^m}(\vec{v}, \check{c}_{\leadsto -})}_{\varepsilon=\varepsilon_{m,1}} \mid \dots \mid \underbrace{\mathfrak{d}_{l_m^-}^{\tau_-^m}(\vec{v}, \check{c}_{\leadsto -})}_{\varepsilon=\varepsilon_{m,l_m^-}}$$

Simple commands:

$$\mathring{c}_{\leadsto\varepsilon} ::= \mathsf{com}^\pm(\check{c}_{\leadsto\varepsilon})$$

Commands:

$$c_{\leadsto\varepsilon} ::= \mathsf{com}^\pm(\check{c}_{\leadsto\varepsilon}) \mid \mathsf{com}^\pm(\mathsf{let}\ x^+ := \check{c}_{\leadsto +}\ \mathsf{in}\ c_{\leadsto\varepsilon}) \mid \mathsf{com}^-(\mathsf{let}\ x^- := \underline{t_-}\ \mathsf{in}\ c_{\leadsto\varepsilon})$$
$$\mid \mathsf{com}^\pm\left(\mathsf{match}\ \check{c}_{\leadsto +}\ \mathsf{with}\left[\mathfrak{d}_1^{\tau_+^1}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon} \mid \dots \mid \mathfrak{d}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\leadsto\varepsilon}\right]\right)$$
$$\mid \vdots$$
$$\mid \mathsf{com}^\pm\left(\mathsf{match}\ \check{c}_{\leadsto +}\ \mathsf{with}\left[\mathfrak{d}_1^{\tau_+^n}(\overrightarrow{x_1}).c^1_{\leadsto\varepsilon} \mid \dots \mid \mathfrak{d}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\leadsto\varepsilon}\right]\right)$$

Figure V.3.4.b: Stacks and evaluation contexts

Negative simple stacks:

$$\mathring{\mathbb{S}}_{-\rightsquigarrow\varepsilon} ::= \square^{-}\}\varepsilon{=}-$$
$$\mid \underbrace{\mathfrak{v}_1^{\tau_-^1}(\vec{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{1,1}} \mid \dots \mid \underbrace{\mathfrak{v}_{l_1^-}^{\tau_-^1}(\vec{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{1,l_1^-}}$$
$$\mid \vdots \quad \mid \ddots \mid \vdots$$
$$\mid \underbrace{\mathfrak{v}_1^{\tau_-^m}(\vec{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{m,1}} \mid \dots \mid \underbrace{\mathfrak{v}_{l_m^-}^{\tau_-^m}(\vec{v}, \mathring{\mathbb{S}}_{-\rightsquigarrow -})}_{\varepsilon=\varepsilon_{m,l_m^-}}$$

Positive simple stacks:

$$\mathring{\mathbb{S}}_{+\rightsquigarrow\varepsilon} ::= \square^{+}\}\varepsilon{=}+$$

Negative stacks:

$$\mathbb{S}_{-\rightsquigarrow\varepsilon} ::= \mathring{\mathbb{S}}_{-\rightsquigarrow\varepsilon} \mid \text{let } x^{+} := \mathring{\mathbb{S}}_{-\rightsquigarrow +} \text{ in } c_{\rightsquigarrow\varepsilon}$$
$$\mid \text{match } \mathring{\mathbb{S}}_{-\rightsquigarrow +} \text{ with}\left[\mathfrak{v}_1^{\tau_+^1}(\vec{x_1}).c^1_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\rightsquigarrow\varepsilon}\right]$$
$$\mid \vdots$$
$$\mid \text{match } \mathring{\mathbb{S}}_{-\rightsquigarrow +} \text{ with}\left[\mathfrak{v}_1^{\tau_+^n}(\vec{x_1}).c^1_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\rightsquigarrow\varepsilon}\right]$$

Positive stacks / evaluation contexts:

$$\mathbb{S}_{+\rightsquigarrow\varepsilon}, \mathbb{e}_{+\rightsquigarrow\varepsilon} ::= \square^{+}\}\varepsilon{=}+ \mid \text{let } x^{+} := \square^{+} \text{ in } c_{\rightsquigarrow\varepsilon}$$
$$\mid \text{match } \square^{+} \text{ with}\left[\mathfrak{v}_1^{\tau_+^1}(\vec{x_1}).c^1_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).c^{l_1^+}_{\rightsquigarrow\varepsilon}\right]$$
$$\mid \vdots$$
$$\mid \text{match } \square^{+} \text{ with}\left[\mathfrak{v}_1^{\tau_+^n}(\vec{x_1}).c^1_{\rightsquigarrow\varepsilon} \mid \dots \mid \mathfrak{v}_{l_n^+}^{\tau_+^n}(\overrightarrow{x_{l_n^+}}).c^{l_n^+}_{\rightsquigarrow\varepsilon}\right]$$

Negative evaluation contexts:

$$\mathbb{e}_{-\rightsquigarrow\varepsilon} ::= \mathbb{S}_{-\rightsquigarrow\varepsilon} \boxed{\text{instk}^{-}(\square^{-})} \mid \text{let } x^{-} := \square^{-} \text{ in } c_{\rightsquigarrow\varepsilon}$$

Figure V.3.4.c: Operational reduction

$$\mathrm{com}^{\varepsilon_1}\left(\mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\,\boxed{\underline{c_{\rightsquigarrow \varepsilon_1}}}\right) \triangleright_{\mu} \mathrm{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2})$$

$$\mathrm{com}^{\varepsilon_1}\left(\mathrm{let}\; x^{\varepsilon_1} := \underline{v_{\varepsilon_1}}\; \mathrm{in}\; c_{\rightsquigarrow \varepsilon_2}\right) \triangleright_{\tilde{\mu}} c_{\rightsquigarrow \varepsilon_2}[v_{\varepsilon_1}/x^{\varepsilon_1}]$$

$$\mathrm{com}^{-}\left(\mathbb{S}_{- \rightsquigarrow \varepsilon}\,\boxed{\underline{(\lambda x^{+}.c_{\rightsquigarrow -})}\, v_{+}}\right) \triangleright_{\rightarrow} \mathrm{defer}(c_{\rightsquigarrow -}[v_{+}/x^{+}], \mathbb{S}_{- \rightsquigarrow \varepsilon})$$

$$\mathrm{com}^{-}\left(\mathbb{S}_{+ \rightsquigarrow \varepsilon}\,\boxed{\mathrm{unfreeze}(\underline{\mathrm{freeze}(c_{\rightsquigarrow +})})}\right) \triangleright_{\Uparrow} \mathrm{defer}(c_{\rightsquigarrow +}, \mathbb{S}_{+ \rightsquigarrow \varepsilon})$$

$$\mathrm{com}^{-}\left(\mathbb{S}_{- \rightsquigarrow \varepsilon}\,\boxed{\pi_i(\underline{(c^1_{\rightsquigarrow -} \,\&\, c^2_{\rightsquigarrow -})})}\right) \triangleright_{\&} \mathrm{defer}(c^i_{\rightsquigarrow -}, \mathbb{S}_{- \rightsquigarrow \varepsilon})$$

($\triangleright_{\top}$ is trivial)

$$\mathrm{com}^{+}\left(\mathrm{match}\; \underline{\mathrm{box}(v_{-})}\; \mathrm{with}[\mathrm{box}(x^{-}).c_{\rightsquigarrow \varepsilon}]\right) \triangleright_{\Downarrow} c_{\rightsquigarrow \varepsilon}[v_{-}/x^{-}]$$

$$\mathrm{com}^{+}\left(\mathrm{match}\; \underline{(v_{+} \otimes w_{+})}\; \mathrm{with}[(x^{+} \otimes y^{+}).c_{\rightsquigarrow \varepsilon}]\right) \triangleright_{\otimes} c_{\rightsquigarrow \varepsilon}[v_{+}/x^{+}, w_{+}/y^{+}]$$

$$\mathrm{com}^{+}\left(\mathrm{match}\; \underline{\iota_i(v_{+})}\; \mathrm{with}[\iota_1(x_1^{+}).c^1_{\rightsquigarrow \varepsilon} \mid \iota_2(x_2^{+}).c^2_{\rightsquigarrow \varepsilon}]\right) \triangleright_{\oplus} c^i_{\rightsquigarrow \varepsilon}[v_{+}/x_i^{+}]$$

$$\mathrm{match}()\; \mathrm{with}[().c_{\rightsquigarrow \varepsilon}] \triangleright_{1} c_{\rightsquigarrow \varepsilon}$$

$$\triangleright \stackrel{\mathrm{def}}{\equiv} \triangleright_{\tilde{\mu}} \cup \triangleright_{\tilde{\mu}} \cup \triangleright_{\rightarrow} \cup \triangleright_{\&} \cup \triangleright_{\Uparrow} \cup \triangleright_{\otimes} \cup \triangleright_{\oplus} \cup \triangleright_{\Downarrow} \cup \triangleright_{1} \qquad \triangleright_{\mathrm{let}} \stackrel{\mathrm{def}}{\equiv} \triangleright_{\tilde{\mu}}$$

Figure V.3.4.d: η-expansion

$$
\begin{array}{rcll}
t_\varepsilon & \rightsquigarrow_{\mu} & \mathsf{ctot}^{\varepsilon}(t_\varepsilon) & \\
e_{\varepsilon_1 \rightsquigarrow \varepsilon_2} & \rightsquigarrow_{\tilde\mu} & \mathsf{let}\ x^{\varepsilon_1} := \square^{\varepsilon_1}\ \mathsf{in}\ e_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\ \boxed{x^{\varepsilon_1}} & \text{if } x^{\varepsilon_1} \text{ fresh w.r.t. } e_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \\
v_- & \rightsquigarrow_{\to} & \lambda x^+ . v_-\, x^+ & \text{if } x^+ \text{ fresh w.r.t. } v_- \\
v_- & \rightsquigarrow_{\Uparrow} & \mathsf{freeze}(\mathsf{unfreeze}(v_-)) & \\
v_- & \rightsquigarrow_{\&} & (\pi_1(v_-) \& \pi_2(v_-)) & \\
v_- & \rightsquigarrow_{\top} & \lambda <> & \\
\mathbb{S}_{+\rightsquigarrow\varepsilon} & \rightsquigarrow_{\Downarrow} & \mathsf{match}\ \square^+\ \mathsf{with}\big[\mathsf{box}(x^-).\mathbb{S}_{+\rightsquigarrow\varepsilon}\ \boxed{x^+}\big] & \text{if } x^- \text{ fresh w.r.t. } \mathbb{S}_{+\rightsquigarrow\varepsilon} \\
\mathbb{S}_{+\rightsquigarrow\varepsilon} & \rightsquigarrow_{\otimes} & \mathsf{match}\ \square^+\ \mathsf{with}\big[(x^+ \otimes y^+).\mathbb{S}_{+\rightsquigarrow\varepsilon}\ \boxed{(x^+ \otimes y^+)}\big] & \text{if } x^+ \text{ and } y^+ \text{ fresh w.r.t. } \mathbb{S}_{+\rightsquigarrow\varepsilon} \\
\mathbb{S}_{+\rightsquigarrow\varepsilon} & \rightsquigarrow_{\oplus} & \mathsf{match}\ \square^+\ \mathsf{with}\big[\iota_1(x_1^+).\mathbb{S}_{+\rightsquigarrow\varepsilon}\ \boxed{x_1^+} \mid \iota_2(x_2^+).\mathbb{S}_{+\rightsquigarrow\varepsilon}\ \boxed{x_2^+}\big] & \text{if } x_1^+ \text{ and } x_2^+ \text{ fresh w.r.t. } \mathbb{S}_{+\rightsquigarrow\varepsilon} \\
\mathbb{S}_{+\rightsquigarrow\varepsilon} & \rightsquigarrow_{1} & \mathsf{match}\ \square^+\ \mathsf{with}\big[().\mathbb{S}_{+\rightsquigarrow\varepsilon}\ \boxed{()}\big] &
\end{array}
$$

$$
\rightsquigarrow \stackrel{\text{def}}{=} \rightsquigarrow_{\tilde\mu} \cup \rightsquigarrow_{\tilde\mu} \cup \rightsquigarrow_{\to} \cup \rightsquigarrow_{\Uparrow} \cup \rightsquigarrow_{\&} \cup \rightsquigarrow_{\top} \cup \rightsquigarrow_{\Downarrow} \cup \rightsquigarrow_{\otimes} \cup \rightsquigarrow_{\oplus} \cup \rightsquigarrow_{1}
$$

V.4. Equivalence between $\lambda_{\mathrm{p}}^{\vec{\tau}}$ and $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$

In this section, we show that $\lambda_{\mathrm{p}}^{\vec{\tau}}$ and $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ are two equivalent descriptions of the same objets by defining two inverse translations $\underset{\rightarrow}{\cdot} : \lambda_{\mathrm{p}}^{\vec{\tau}} \to \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\underset{\leftarrow}{\cdot} : \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \to \lambda_{\mathrm{p}}^{\vec{\tau}}$. Since $\lambda_{\mathrm{p}}^{\vec{\tau}}$ is outside-out and $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ is inside-out, the definitions and properties of these translations will rely on the properties of stack composition:

Fact V.4.1

In $\lambda_{\mathrm{p}}^{\vec{\tau}}$ (resp. $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$), stacks form a category $\mathcal{C}$ with $\mathrm{Obj}(\mathcal{C}) = \{+, -\}$ and

$$\mathrm{Hom}_{\mathcal{C}}(\varepsilon_1, \varepsilon_2) = \{(\sigma, \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) | \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \in \mathbf{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\} \quad (\text{resp. } \{(\sigma, \star^{\varepsilon_1} \mapsto s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) | s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \in \mathbf{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\})$$

whose composition and identities are given by

$$\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \circ_{\mathrm{d}} \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} \stackrel{\text{def}}{=} \mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}) \quad (\text{resp. } s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \circ_{\star} s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} \stackrel{\text{def}}{=} s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\mathrm{N}}])$$

and

$$\mathrm{Id}_{\varepsilon} = \square^{\varepsilon} \quad (\text{resp. } \mathrm{Id}_{\varepsilon} = \star^{\varepsilon})$$

respectively, and this category acts on commands (on the right) via

$$c_{\rightsquigarrow \varepsilon_1} \bullet_{\mathrm{d}} \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \stackrel{\text{def}}{=} \mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) \quad (\text{resp. } c_{\rightsquigarrow \varepsilon_1} \bullet_{\star} s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \stackrel{\text{def}}{=} c_{\rightsquigarrow \varepsilon_1}[s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}])$$

In other words:

- [cat-closure] for any stacks $\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$), $\mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3})$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}]$) is a stack $\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_3}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_3}$).
- [cat-id] for any stack $\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$), we have
$$\mathsf{defer}(\square^{\varepsilon_1}, \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) = \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} = \mathsf{defer}(\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \square^{\varepsilon_2})$$
$$(\text{resp. } \star^{\varepsilon_1}[s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}] = s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} = s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[\star^{\varepsilon_2} / \star^{\varepsilon_2}])$$
- [cat-accoc] for any stacks $\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$, $\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$, and $\mathbb{s}^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$, $s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$, and $s^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}$), we have
$$\mathsf{defer}(\mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}), \mathbb{s}^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}) = \mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathsf{defer}(\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}, \mathbb{s}^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}))$$
$$(\text{resp. } s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}][s^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4} / \star^{\varepsilon_3}] = s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}[s^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4} / \star^{\varepsilon_3}] / \star^{\varepsilon_2}])$$
- [act-closure] for any command $c_{\rightsquigarrow \varepsilon_1}$ and stack $\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$), $\mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2})$ (resp. $c_{\rightsquigarrow \varepsilon_1}[s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}]$) is a command $c^2_{\rightsquigarrow \varepsilon_2}$.
- [act-id] for any command $c_{\rightsquigarrow \varepsilon}$, we have
$$\mathsf{defer}(c_{\rightsquigarrow \varepsilon}, \square^{\varepsilon}) = c_{\rightsquigarrow \varepsilon} \quad (\text{resp. } c_{\rightsquigarrow \varepsilon}[\star^{\varepsilon} / \star^{\varepsilon}] = c_{\rightsquigarrow \varepsilon})$$
- [act-assoc] for any command $c_{\rightsquigarrow \varepsilon_1}$ and stacks $\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$), we have
$$\mathsf{defer}(\mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}), \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}) = \mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}))$$
$$(\text{resp. } c_{\rightsquigarrow \varepsilon_1}[s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}][s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}] = c_{\rightsquigarrow \varepsilon_1}[s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}] / \star^{\varepsilon_1}])$$

Proof sketch (See page 191 for details)

By a few inductions. For (cat-assoc) and (act-assoc), the induction is on the size of the middle stack, and in the inductive cases, this middle stack is the composition of two strictly smaller stacks, which allows to conclude by applying the induction hypothesis four times.

Definition and basic properties

The translations $\underrightarrow{\cdot} : \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}} \to \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\underleftarrow{\cdot} : \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \to \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ are defined in Figure V.4.1.

Fact V.4.2

The translations $\underrightarrow{\cdot} : \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}} \to \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\underleftarrow{\cdot} : \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \to \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ are well-defined and map commands (resp. values, terms, stacks, evaluation contexts) to commands (resp. values, terms, stacks, evaluation contexts) with the same polarities.

Proof

By induction on the translated term, using (cat-closure) and (act-closure) of Fact V.4.1.

Fact V.4.3

The translations $\underrightarrow{\cdot} : \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}} \to \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\underleftarrow{\cdot} : \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \to \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ induce functors between the categories defined in Fact V.4.1 and preserve the action on commands. In other words:

- functor-id for each polarity ε,

$$\underrightarrow{\square^{\varepsilon}} = \star^{\varepsilon} \quad \left(\text{resp. } \underleftarrow{\star^{\varepsilon}} = \square^{\varepsilon}\right)$$

- functor-compose for any stacks $\mathbb{S}^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $\mathbb{S}^{2}_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$ (resp. $s^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $s^{2}_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$), we have

$$\underrightarrow{\mathrm{defer}(\mathbb{S}^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{S}^{2}_{\varepsilon_2 \rightsquigarrow \varepsilon_3})} = \underrightarrow{\mathbb{S}^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}}\left[\underrightarrow{\mathbb{S}^{2}_{\varepsilon_2 \rightsquigarrow \varepsilon_3}} / \star^{\varepsilon_2}\right] \quad \left(\text{resp. } \underleftarrow{s^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^{2}_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}]} = \mathrm{defer}\left(\underleftarrow{s^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}}, \underleftarrow{s^{2}_{\varepsilon_2 \rightsquigarrow \varepsilon_3}}\right)\right)$$

- functor-act for any command $c_{\rightsquigarrow \varepsilon_1}$ and stack $\mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$), we have

$$\underrightarrow{\mathrm{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2})} = \underrightarrow{c_{\rightsquigarrow \varepsilon_1}}\left[\underrightarrow{\mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}} / \star^{\varepsilon_2}\right] \quad \left(\text{resp. } \underleftarrow{c_{\rightsquigarrow \varepsilon_1}[s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_2}]} = \mathrm{defer}\left(\underleftarrow{c_{\rightsquigarrow \varepsilon_1}}, \underleftarrow{s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}}\right)\right)$$

Proof

- functor-id Immediate.
- functor-compose By induction on $\mathbb{S}^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ (resp. $s^{1}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$), using Fact V.4.1.

Figure V.4.1: Translations from the inside-out description of $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ to $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ ($\underrightarrow{\cdot}$) and back ($\underleftarrow{\cdot}$)

Figure V.4.1.a: Expressions

Positive values:	Positive values:
$\underrightarrow{x^+} \stackrel{\text{def}}{=} x^+$	$\underleftarrow{x^+} = x^+$
$\underrightarrow{\mathfrak{v}_k^{\tau_+^j}(v^1, \dots, v^q)} \stackrel{\text{def}}{=} \mathfrak{v}_k^{\tau_+^j}(\underrightarrow{v^1}, \dots, \underrightarrow{v^q})$	$\underleftarrow{\mathfrak{v}_k^{\tau_+^j}(v^1, \dots, v^q)} = \mathfrak{v}_k^{\tau_+^j}(\underleftarrow{v^1}, \dots, \underleftarrow{v^q})$
Positive expressions:	Positive expressions:
$\underrightarrow{\mathsf{val}^+(v_+)} \stackrel{\text{def}}{=} \mathsf{val}^+(\underrightarrow{v_+})$	$\underleftarrow{\mathsf{val}^+(v_+)} = \mathsf{val}^+(\underleftarrow{v_+})$
$\underrightarrow{\mathsf{ctot}^+(c_{\rightsquigarrow +})} \stackrel{\text{def}}{=} \mu\star^+.\underrightarrow{c_{\rightsquigarrow +}}$	$\underleftarrow{\mu\star^+.c_{\rightsquigarrow +}} = \mathsf{ctot}^+(\underleftarrow{c_{\rightsquigarrow +}})$
Negative values / expressions:	Negative values / expressions:
$\underrightarrow{x^-} \stackrel{\text{def}}{=} x^-$	$\underleftarrow{x^-} = x^-$
$\underrightarrow{\mathsf{ctot}^-(c_{\rightsquigarrow -})} \stackrel{\text{def}}{=} \mu\star^-.\underrightarrow{c_{\rightsquigarrow -}}$	$\underleftarrow{\mu\star^-.c_{\rightsquigarrow -}} = \mathsf{ctot}^-(\underleftarrow{c_{\rightsquigarrow -}})$

$$\underrightarrow{\lambda\left\langle \begin{matrix} \mathfrak{d}_1^{\tau_-^j}(\overrightarrow{x_1}, \blacksquare^-).\, c^1_{\rightsquigarrow\varepsilon_{j,1}} \\ \vdots \\ \mathfrak{d}_l^{\tau_-^j}(\overrightarrow{x_l}, \blacksquare^-).\, c^l_{\rightsquigarrow\varepsilon_{j,l}} \end{matrix} \right\rangle} \stackrel{\text{def}}{=} \mu \left\langle \begin{matrix} \mathfrak{d}_1^{\tau_-^j}(\overrightarrow{x_1}, \star^{\varepsilon_{j,1}}).\, \underrightarrow{c^1_{\rightsquigarrow\varepsilon_{j,1}}} \\ \vdots \\ \mathfrak{d}_l^{\tau_-^j}(\overrightarrow{x_l}, \star^{\varepsilon_{j,l_{\overline{j}}}}).\, \underrightarrow{c^l_{\rightsquigarrow\varepsilon_{j,l}}} \end{matrix} \right\rangle$$

$$\underleftarrow{\mu\left\langle \begin{matrix} \mathfrak{d}_1^{\tau_-^j}(\overrightarrow{x_1}, \star^{\varepsilon_{j,1}}).\, c^1_{\rightsquigarrow\varepsilon_{j,1}} \\ \vdots \\ \mathfrak{d}_l^{\tau_-^j}(\overrightarrow{x_l}, \star^{\varepsilon_{j,l_{\overline{j}}}}).\, c^l_{\rightsquigarrow\varepsilon_{j,l}} \end{matrix} \right\rangle} = \lambda \left\langle \begin{matrix} \mathfrak{d}_1^{\tau_-^j}(\overrightarrow{x_1}, \blacksquare^-).\, \underleftarrow{c^1_{\rightsquigarrow\varepsilon_{j,1}}} \\ \vdots \\ \mathfrak{d}_l^{\tau_-^j}(\overrightarrow{x_l}, \blacksquare^-).\, \underleftarrow{c^l_{\rightsquigarrow\varepsilon_{j,l}}} \end{matrix} \right\rangle$$

Figure V.4.1.b: Negative stacks and evaluation contexts

Positive stacks / evaluation contexts:

$$\underrightarrow{\star^+} \stackrel{\text{def}}{\equiv} \square^+$$

$$\underrightarrow{\mathsf{let}\ x^+ := \square^+\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon}} \stackrel{\text{def}}{\equiv} \tilde{\mu}x^+.\underrightarrow{c_{\rightsquigarrow\varepsilon}}$$

$$\underrightarrow{\mathsf{match}\ \square^+\ \mathsf{with}\begin{bmatrix}\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\,c^1_{\rightsquigarrow\varepsilon}\\ \vdots\\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}).\,c^l_{\rightsquigarrow\varepsilon}\end{bmatrix}} \stackrel{\text{def}}{\equiv} \tilde{\mu}\begin{bmatrix}\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\,\underrightarrow{c^1_{\rightsquigarrow\varepsilon}}\\ \vdots\\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}).\,\underrightarrow{c^l_{\rightsquigarrow\varepsilon}}\end{bmatrix}$$

Negative stacks:

$$\underrightarrow{\square^-} \stackrel{\text{def}}{\equiv} \star^-$$

$$\underrightarrow{\mathfrak{s}_k^{\tau_-^j}\left(v^1_{\varepsilon_1},\dots,v^q_{\varepsilon_q},\mathring{\mathbb{S}}_{-\rightsquigarrow -}\right)} \stackrel{\text{def}}{\equiv} \underrightarrow{\mathring{\mathbb{S}}_{-\rightsquigarrow -}}\left[\mathfrak{s}_k^{\tau_-^j}\left(\underrightarrow{v^1_{\varepsilon_1}},\dots,\underrightarrow{v^q_{\varepsilon_q}},\star^{\varepsilon_{j,k}}\right)\Big/\star^-\right]$$

$$\underrightarrow{\mathsf{let}\ x^+ := \mathring{\mathbb{S}}_{-\rightsquigarrow +}\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon}} \stackrel{\text{def}}{\equiv} \underrightarrow{\mathring{\mathbb{S}}_{-\rightsquigarrow +}}\left[\tilde{\mu}x^+.\underrightarrow{c_{\rightsquigarrow\varepsilon}}/\star^+\right]$$

$$\underrightarrow{\mathsf{match}\ \mathring{\mathbb{S}}_{-\rightsquigarrow +}\ \mathsf{with}\begin{bmatrix}\mathfrak{v}_1^{\tau_+^1}(\overrightarrow{x_1}).\,c^1_{\rightsquigarrow\varepsilon}\\ \vdots\\ \mathfrak{v}_{l_1^+}^{\tau_+^1}(\overrightarrow{x_{l_1^+}}).\,c^l_{\rightsquigarrow\varepsilon}\end{bmatrix}} \stackrel{\text{def}}{\equiv} \underrightarrow{\mathring{\mathbb{S}}_{-\rightsquigarrow +}}\left[\tilde{\mu}\begin{bmatrix}\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\,\underrightarrow{c^1_{\rightsquigarrow\varepsilon}}\\ \vdots\\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}).\,\underrightarrow{c^l_{\rightsquigarrow\varepsilon}}\end{bmatrix}\Bigg/\star^+\right]$$

Negative evaluation contexts:

$$\underrightarrow{\mathbb{S}_{-\rightsquigarrow\varepsilon}\boxed{\mathsf{instk}^-(\square^-)}} \stackrel{\text{def}}{\equiv} \mathsf{stk}^-\left(\underrightarrow{\mathbb{S}_{-\rightsquigarrow\varepsilon}}\right)$$

$$\underrightarrow{\mathsf{let}\ x^- := \square^-\ \mathsf{in}\ c_{\rightsquigarrow\varepsilon}} \stackrel{\text{def}}{\equiv} \tilde{\mu}x^-.\underrightarrow{c_{\rightsquigarrow\varepsilon}}$$

Positive stacks / evaluation contexts:

$$\underleftarrow{\square^+} = \star^+$$

$$\underleftarrow{\tilde{\mu}x^+.c_{\rightsquigarrow\varepsilon}} = \mathsf{let}\ x^+ := \square^+\ \mathsf{in}\ \underleftarrow{c_{\rightsquigarrow\varepsilon}}$$

$$\underleftarrow{\tilde{\mu}\begin{bmatrix}\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\,c^1_{\rightsquigarrow\varepsilon}\\ \vdots\\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_{l_j}}).\,c^l_{\rightsquigarrow\varepsilon}\end{bmatrix}} = \mathsf{match}\ \square^+\ \mathsf{with}\begin{bmatrix}\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}).\,\underleftarrow{c^1_{\rightsquigarrow\varepsilon}}\\ \vdots\\ \mathfrak{v}_{l_j}^{\tau_+^j}(\overrightarrow{x_{l_j}}).\,\underleftarrow{c^l_{\rightsquigarrow\varepsilon}}\end{bmatrix}$$

Negative stacks:

$$\underleftarrow{\star^-} \stackrel{\text{def}}{\equiv} \square^-$$

$$\underleftarrow{\mathfrak{s}_k^{\tau_-^j}\left(v^1_{\varepsilon_1},\dots,v^q_{\varepsilon_q},s_{\varepsilon_{j,k}\rightsquigarrow\varepsilon}\right)} \stackrel{\text{def}}{\equiv} \underleftarrow{s_{\varepsilon_{j,k}\rightsquigarrow\varepsilon}}\boxed{\mathfrak{s}_k^{\tau_-^j}\left(\underleftarrow{v^1_{\varepsilon_1}},\dots,\underleftarrow{v^q_{\varepsilon_q}},\square^-\right)}$$

Negative evaluation contexts:

$$\underleftarrow{\mathsf{stk}^-(s_{-\rightsquigarrow\varepsilon})} \stackrel{\text{def}}{\equiv} \underleftarrow{s_{-\rightsquigarrow\varepsilon}}\boxed{\mathsf{instk}^-(\square^-)}$$

$$\underleftarrow{\tilde{\mu}x^-.c_{\rightsquigarrow\varepsilon}} \stackrel{\text{def}}{\equiv} \mathsf{let}\ x^- := \square^-\ \mathsf{in}\ \underleftarrow{c_{\rightsquigarrow\varepsilon}}$$

Figure V.4.1.c: Commands

Incomplete simple commands:

$$\underrightarrow{\underline{t_+}} \stackrel{\text{def}}{=} \underrightarrow{t_+}|\star^+$$

$$\underrightarrow{\mathsf{instk}^-(\underline{t_-})} \stackrel{\text{def}}{=} \underrightarrow{t_-}|\mathsf{stk}^-(\star^-)$$

$$\underrightarrow{\mathfrak{s}_k^{\tau_-^j}(v_{\varepsilon_1}^1, \dots, v_{\varepsilon_q}^q, \breve{c}_{\rightsquigarrow -})} \stackrel{\text{def}}{=} \underrightarrow{\breve{c}_{\rightsquigarrow -}}\left[\mathfrak{s}_k^{\tau_-^j}(\underrightarrow{v_{\varepsilon_1}^1}, \dots, \underrightarrow{v_{\varepsilon_q}^q}, \star^{\varepsilon_{j,k}}) / \star^-\right]$$

Commands:

$$\underrightarrow{\mathsf{com}^{\varepsilon_1}(\breve{c}_{\rightsquigarrow\varepsilon_2})} \stackrel{\text{def}}{=} \langle \underrightarrow{\breve{c}_{\rightsquigarrow\varepsilon_2}} \rangle^{\varepsilon_1}$$

$$\underrightarrow{\mathsf{com}^{\varepsilon_1}(\mathsf{let}\, x^+ := \breve{c}_{\rightsquigarrow +}\, \mathsf{in}\, c_{\rightsquigarrow\varepsilon_2})} \stackrel{\text{def}}{=} \langle \underrightarrow{\breve{c}_{\rightsquigarrow\varepsilon_2}}[\tilde{\mu} x^+ . \underrightarrow{c_{\rightsquigarrow\varepsilon_2}} / \star^+] \rangle^{\varepsilon_1}$$

$$\underrightarrow{\mathsf{com}^-(\mathsf{let}\, x^- := \underline{t_-}\, \mathsf{in}\, c_{\rightsquigarrow\varepsilon})} \stackrel{\text{def}}{=} \langle \underrightarrow{t_-} | \tilde{\mu} x^- . \underrightarrow{c_{\rightsquigarrow\varepsilon}} \rangle^-$$

$$\underrightarrow{\mathsf{com}^{\varepsilon_1}\left(\mathsf{match}\, \breve{c}_{\rightsquigarrow +}\, \mathsf{with} \begin{bmatrix} \mathfrak{d}_1^{\tau_+^j}(\overrightarrow{x_1}).\, c_{\rightsquigarrow\varepsilon_2}^1 \\ \vdots \\ \mathfrak{d}_l^{\tau_+^j}(\overrightarrow{x_l}).\, c_{\rightsquigarrow\varepsilon_2}^l \end{bmatrix}\right)} \stackrel{\text{def}}{=} \left\langle \underrightarrow{\breve{c}_{\rightsquigarrow +}} \left[\tilde{\mu} \begin{bmatrix} \mathfrak{d}_1^{\tau_+^j}(\overrightarrow{x_1}).\, \underrightarrow{c_{\rightsquigarrow\varepsilon_2}^1} \\ \vdots \\ \mathfrak{d}_l^{\tau_+^j}(\overrightarrow{x_l}).\, \underrightarrow{c_{\rightsquigarrow\varepsilon_2}^l} \end{bmatrix} \Big/ \star^+ \right] \right\rangle^{\varepsilon_1}$$

Incomplete simple commands:

$$\underleftarrow{t_{\varepsilon_1} | e_{\varepsilon_1 \rightsquigarrow \varepsilon_2}} \stackrel{\text{def}}{=} \underleftarrow{e_{\varepsilon_1 \rightsquigarrow \varepsilon_2}} \boxed{\underleftarrow{\underline{t_{\varepsilon_1}}}}$$

Commands:

$$\underleftarrow{\langle t_{\varepsilon_1} | e_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \rangle^{\varepsilon_1}} \stackrel{\text{def}}{=} \mathsf{com}^{\varepsilon_1}\left(\underleftarrow{e_{\varepsilon_1 \rightsquigarrow \varepsilon_2}} \boxed{\underleftarrow{\underline{t_{\varepsilon_1}}}}\right)$$

Figure V.4.1.d: Disubstitutions

Substitutions:

$$\underrightarrow{x_1^{\varepsilon_1} \mapsto v_{\varepsilon_1}^1, \dots, x_q^{\varepsilon_q} \mapsto v_{\varepsilon_q}^q} \stackrel{\text{def}}{=} x_1^{\varepsilon_1} \mapsto \underrightarrow{v_{\varepsilon_1}^1}, \dots, x_q^{\varepsilon_q} \mapsto \underrightarrow{v_{\varepsilon_q}^q}$$

Disubstitutions:

$$\underrightarrow{(\sigma, \mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2})} \stackrel{\text{def}}{=} \underrightarrow{\sigma}, \star^{\varepsilon_1} \mapsto \underrightarrow{\mathbb{S}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}}$$

Substitutions:

$$\underleftarrow{x_1^{\varepsilon_1} \mapsto v_{\varepsilon_1}^1, \dots, x_q^{\varepsilon_q} \mapsto v_{\varepsilon_q}^q} \stackrel{\text{def}}{=} x_1^{\varepsilon_1} \mapsto \underleftarrow{v_{\varepsilon_1}^1}, \dots, x_q^{\varepsilon_q} \mapsto \underleftarrow{v_{\varepsilon_q}^q}$$

Disubstitutions:

$$\underleftarrow{\sigma, \star^{\varepsilon_1} \mapsto s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}} \stackrel{\text{def}}{=} (\underleftarrow{\sigma}, \underleftarrow{s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}})$$

- [functor-act] By induction on $c_{\rightsquigarrow \varepsilon_1}$, using Fact V.4.1.

Fact V.4.4

The translations $\underrightarrow{\cdot} : \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}} \to \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$ and $\underleftarrow{\cdot} : \mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \to \underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ are inverses.

Proof

By case analysis on the translated term, using Fact V.4.3.

Equality through translations

We define $\rightleftharpoons$ as equality through those translations:

Definition V.4.5

Given a term $t_<$ in $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ and another term $t_>$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, we write $t_< \rightleftharpoons t_>$ when $\underrightarrow{t_<} = t_>$, or equivalently when $t_< = \underleftarrow{t_>}$. Similarly, given a disubstitution $\varphi_<$ in $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ and another disubstitution $\varphi_>$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, we write $\varphi_< \rightleftharpoons \varphi_>$ when $\underrightarrow{\varphi_<} = \varphi_>$, or equivalently when $\varphi_< = \underleftarrow{\varphi_>}$.

Remark V.4.6

Note that $\rightleftharpoons$ is *not* symmetric. Whenever we write $t_< \rightleftharpoons t_>$, we implicitly assume that $t_<$ lives in $\underline{\lambda}_{\mathrm{p}}^{\vec{\tau}}$ and $t_>$ in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$.

This equality preserves everything we are interested in:

Fact V.4.7

If $t_< \rightleftharpoons t_>$ and $\varphi_< \rightleftharpoons \varphi_>$ then $t_<[\varphi_<] \rightleftharpoons t_>[\varphi_>]$.

Proof

V.5. A polarized λ-calculus: $\boldsymbol{\lambda}_{\mathrm{P}}^{\vec{\tau}}$

VI. Dynamically typed polarized calculi

VI.1. Clashes and dynamically typed calculi

VI.2. A dynamically typed polarized λ-calculus: $\boldsymbol{\lambda}_{\mathrm{P}}^{\mathcal{PN}}$

VI.3. A dynamically typed polarized λ-calculus with focus: $\underline{\lambda}_{\mathrm{p}}^{\mathcal{PN}}$

VI.4. A dynamically typed polarized intuitionistic L calculus: $\mathbf{Li}_{\mathrm{p}}^{\mathcal{PN}}$

VI.5. A dynamically typed polarized classical L calculus: $\mathbf{L}_{\mathrm{p}}^{\mathcal{PN}}$

Part C.

Solvability in polarized calculi

Part C is about two well-known and very useful properties of λ-terms: operational relevance and solvability.

Most common definitions of solvability are optimized to make proofs easier, which has the unfortunate consequence of making it look like a fairly arbitrary notion that just happens to have some use cases. This is of course not the case, and in this introduction we aim at explaining why solvability is a very natural and useful notion when looking at semantics of programming languages. In the λ-calculus, it is well known that solvable expressions are exactly the operationally relevant one and, as will be explained in the next section, this can be understood as saying (somewhat indirectly) that the output of programs can be used internally, i.e. as an intermediate result in a larger program.

Content

Contribution

Introduction to solvability and operational completeness

VII. Call-by-name solvability

VIII. Call-by-value solvability

IX. Polarized solvability

Bibliography

[Abr90] S. Abramsky, “The lazy lambda calculus,” 1990 (cit. on pp. 5, 9).

[AbrOng93] S. Abramsky and C.-H. L. Ong, “Full abstraction in the lazy lambda calculus,” *Inf. Comput.*, vol. 105, no. 2, pp. 159–267, Aug. 1993, ISSN: 0890-5401. DOI: 10.1006/inco.1993.1044. [Online]. Available: https://doi.org/10.1006/inco.1993.1044 (cit. on p. 8).

[AccGue16] B. Accattoli and G. Guerrieri, “Open call-by-value,” in *Programming Languages and Systems*, A. Igarashi, Ed., Cham: Springer International Publishing, 2016, pp. 206–226, ISBN: 978-3-319-47958-3 (cit. on pp. 5, 23).

[AccPao12] B. Accattoli and L. Paolini, “Call-by-value solvability, revisited,” in *Functional and Logic Programming*, T. Schrijvers and P. Thiemann, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 4–16, ISBN: 978-3-642-29822-6 (cit. on pp. 5, 7, 23).

[Bar84] H. Barendregt, *The lambda calculus: its syntax and semantics* (Studies in logic and the foundations of mathematics). North-Holland, 1984, ISBN: 9780444867483. [Online]. Available: https://books.google.fr/books?id=eMtTAAAAYAAJ (cit. on pp. 3, 5, 9, 18, 20, 112).

[BucKesRíoVis20] A. Bucciarelli, D. Kesner, A. Ríos, and A. Viso, “The bang calculus revisited,” in *Functional and Logic Programming*, K. Nakano and K. Sagonas, Eds., Cham: Springer International Publishing, 2020, pp. 13–32, ISBN: 978-3-030-59025-3 (cit. on p. 5).

[Chu85] A. Church, *The Calculi of Lambda Conversion. (AM-6) (Annals of Mathematics Studies)*. USA: Princeton University Press, 1985, ISBN: 0691083940 (cit. on p. 8).

[CurFioMun16] P.-L. Curien, M. Fiore, and G. Munch-Maccagnoni, “A Theory of Effects and Resources: Adjunction Models and Polarised Calculi,” in *Proc. POPL*, 2016. DOI: 10.1145/2837614.2837652 (cit. on pp. 6, 85, 118).

[CurHer00] P.-L. Curien and H. Herbelin, “The duality of computation,” in *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP ’00), Montreal, Canada, September 18-21, 2000*, ser. SIGPLAN Notices 35(9), ACM, 2000, pp. 233–243, ISBN: 1-58113-202-6. DOI: http://doi.acm.org/10.1145/351240.351262 (cit. on pp. 4–6, 16, 18, 62).

[CurMun10] P.-L. Curien and G. Munch-Maccagnoni, "The duality of computation under focus," in *IFIP TCS*, C. S. Calude and V. Sassone, Eds., ser. IFIP Advances in Information and Communication Technology, vol. 323, Springer, 2010, pp. 165–181 (cit. on p. 6).

[DanNie04] O. Danvy and L. R. Nielsen, "Refocusing in reduction semantics," *BRICS Report Series*, vol. 11, no. 26, Nov. 2004. DOI: 10.7146/brics.v11i26.21851. [Online]. Available: https://tidsskrift.dk/brics/article/view/21851 (cit. on pp. 18, 29, 49).

[dVri16] F.-J. de Vries, "On Undefined and Meaningless in Lambda Definability," in *1st International Conference on Formal Structures for Computation and Deduction (FSCD 2016)*, D. Kesner and B. Pientka, Eds., ser. Leibniz International Proceedings in Informatics (LIPIcs), vol. 52, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, 18:1–18:15, ISBN: 978-3-95977-010-1. DOI: 10.4230/LIPIcs.FSCD.2016.18. [Online]. Available: http://drops.dagstuhl.de/opus/volltexte/2016/5978 (cit. on p. 7).

[DezGio01] M. Dezani-Ciancaglini and E. Giovannetti, "From Böhm's Theorem to Observational Equivalences: an Informal Account," in *BOTH'01*, ser. Electronic Notes in Theoretical Computer Science (http://www.elsevier.nl/locate/entcs/volume
vol. 50, Elsevier, 2001, pp. 83–116. [Online]. Available: http://www.di.unito.it/~dezani/papers/both01.ps (cit. on p. 9).

[Dij68] E. W. Dijkstra, "Letters to the editor: Go to statement considered harmful," *Commun. ACM*, vol. 11, no. 3, pp. 147–148, Mar. 1968, ISSN: 0001-0782. DOI: 10.1145/362929.362947. [Online]. Available: https://doi.org/10.1145/362929.362947 (cit. on p. 3).

[DowAri18] P. Downen and Z. M. Ariola, "A tutorial on computational classical logic and the sequent calculus," *Journal of Functional Programming*, vol. 28, e3, 2018. DOI: 10.1017/S0956796818000023 (cit. on p. 6).

[EhrGue16] T. Ehrhard and G. Guerrieri, "The bang calculus: An untyped lambda-calculus generalizing call-by-name and call-by-value," in *Proceedings of the 18th International Symposium on Principles and Practice of Declarative Programming*, ser. PPDP '16, Edinburgh, United Kingdom: Association for Computing Machinery, 2016, pp. 174–187, ISBN: 9781450341486. DOI: 10.1145/2967973.2968608. [Online]. Available: https://doi.org/10.1145/2967973.2968608 (cit. on p. 5).

[GarNog16] Á. García-Pérez and P. Nogueira, "No solvable lambda-value term left behind," *Logical Methods in Computer Science*, vol. Volume 12, Issue 2, Jun. 2016. DOI: 10.2168/LMCS-12(2:12)2016. [Online]. Available: https://lmcs.episciences.org/1644 (cit. on p. 5).

[Gir11] J.-Y. Girard, "The blind spot: Lectures on logic," 2011 (cit. on p. 129).

[Hue97] G. P. Huet, "The zipper," *J. Funct. Program.*, vol. 7, no. 5, pp. 549–554, 1997. DOI: 10.1017/s0956796897002864. [Online]. Available: https://doi.org/10.1017/s0956796897002864 (cit. on p. 30).

[IntManPol17] B. Intrigila, G. Manzonetto, and A. Polonsky, "Refutation of sallé's long-standing conjecture," in *2nd International Conference on Formal Structures for Computation and Deduction, FSCD 2017, September 3-9, 2017, Oxford, UK*, D. Miller, Ed., ser. LIPIcs, vol. 84, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2017, 20:1–20:18. DOI: 10.4230/LIPIcs.FSCD.2017.20. [Online]. Available: https://doi.org/10.4230/LIPIcs.FSCD.2017.20 (cit. on p. 9).

[Kri07] J.-L. Krivine, "A call-by-name lambda-calculus machine," *Higher Order Symbolic Computation*, vol. 20, pp. 199–207, 2007. [Online]. Available: https://hal.archives-ouvertes.fr/hal-00154508 (cit. on pp. 6, 16, 18, 30).

[Lev01] P. B. Levy, "Call-by-push-value," Ph.D. dissertation, Queen Mary University of London, UK, 2001. [Online]. Available: http://ethos.bl.uk/OrderDetails.do?uin=uk.bl.ethos.369233 (cit. on p. 85).

[Lev04] P. B. Levy, *Call-By-Push-Value: A Functional/Imperative Synthesis* (Semantics Structures in Computation). Springer, 2004, vol. 2, ISBN: 1-4020-1730-8 (cit. on pp. 4, 5, 85).

[Lev06] P. B. Levy, "Call-by-push-value: Decomposing call-by-value and call-by-name," *High. Order Symb. Comput.*, vol. 19, no. 4, pp. 377–414, 2006. DOI: 10.1007/s10990-006-0480-6. [Online]. Available: https://doi.org/10.1007/s10990-006-0480-6 (cit. on pp. 4, 5, 85).

[Mog89] E. Moggi, "Computational lambda-calculus and monads," in *Proceedings of the Fourth Annual Symposium on Logic in Computer Science (LICS '89), Pacific Grove, California, USA, June 5-8, 1989*, IEEE Computer Society, 1989, pp. 14–23. DOI: 10.1109/LICS.1989.39155. [Online]. Available: https://doi.org/10.1109/LICS.1989.39155 (cit. on pp. 5, 85).

[Mog91] E. Moggi, "Notions of computation and monads," *Inf. Comput.*, vol. 93, no. 1, pp. 55–92, 1991. DOI: 10.1016/0890-5401(91)90052-4. [Online]. Available: https://doi.org/10.1016/0890-5401(91)90052-4 (cit. on p. 5).

[Mor69] J. H. Morris, "Lambda calculus models of programming languages," Ph.D. dissertation, Massachusets Institute of Technology, 1969 (cit. on p. 9).

[MunSch15] G. Munch-Maccagnoni and G. Scherer, "Polarised Intermediate Representation of Lambda Calculus with Sums," in *Proceedings of the Thirtieth Annual ACM/IEEE Symposium on Logic In Computer Science (LICS 2015)*, 2015. DOI: 10.1109/LICS.2015.22 (cit. on p. 6).

[Ong88] C. L. Ong, "Fully abstract models of the lazy lambda calculus," in *29th Annual Symposium on Foundations of Computer Science, White Plains, New York, USA, 24-26 October 1988*, IEEE Computer Society, 1988, pp. 368–376. DOI: 10.1109/SFCS.1988.21953. [Online]. Available: https://doi.org/10.1109/SFCS.1988.21953 (cit. on p. 5).

[PaoRon99] L. Paolini and S. Ronchi Della Rocca, "Call-by-value solvability," *RAIRO Theor. Informatics Appl.*, vol. 33, no. 6, pp. 507–534, 1999. DOI: 10.1051/ita:1999130. [Online]. Available: https://doi.org/10.1051/ita:1999130 (cit. on pp. 5, 23).

[Par92] M. Parigot, "$\lambda\mu$-calculus: An algorithmic interpretation of classical natural deduction," in *Logic Programming and Automated Reasoning*, A. Voronkov, Ed., Berlin, Heidelberg: Springer Berlin Heidelberg, 1992, pp. 190–201, ISBN: 978-3-540-47279-7 (cit. on p. 5).

[Reg94] L. Regnier, "Une équivalence sur les lambda-termes," *Theor. Comput. Sci.*, vol. 126, no. 2, pp. 281–292, 1994. DOI: 10.1016/0304-3975(94)90012-4. [Online]. Available: https://doi.org/10.1016/0304-3975(94)90012-4 (cit. on pp. 16, 18, 23).

[Tak95] M. Takahashi, "Parallel reductions in λ-calculus," *Inf. Comput.*, vol. 118, no. 1, pp. 120–127, Apr. 1995, ISSN: 0890-5401. DOI: 10.1006/inco.1995.1057. [Online]. Available: https://doi.org/10.1006/inco.1995.1057 (cit. on p. 112).

[Wad76] C. P. Wadsworth, "The relation between computational and denotational properties for scott's d_{infty}-models of the lambda-calculus," *SIAM J. Comput.*, vol. 5, no. 3, pp. 488–521, 1976. DOI: 10.1137/0205036. [Online]. Available: https://doi.org/10.1137/0205036 (cit. on pp. 7, 9).

Appendix

.1. Properties of disubstitutions

Recall the definitions of disubstitutions in the different calculi:

Summary .1.1

- In λ-calculi, a disubstitution φ is a pair $\varphi = (\sigma, \mathbb{S})$ that consists of a substitution σ and a stack $\mathbb{S}$, with
$$T[\varphi] = \mathbb{S}\,\boxed{T[\sigma]}$$
- In λ-calculi with focus, a disubstitution φ is a pair $\varphi = (\sigma, \mathbb{s})$ that consists of a substitution σ and a stack $\mathbb{s}$, with
$$t[\varphi] \stackrel{\text{def}}{=} t[\sigma]$$
$$c[\varphi] \stackrel{\text{def}}{=} \mathsf{defer}(c[\sigma], \mathbb{s})$$
$$\mathbb{e}[\varphi] \stackrel{\text{def}}{=} \mathsf{defer}(\mathbb{e}[\sigma], \mathbb{s})$$
- In L-calculi, a disubstitution φ is a substitution whose domain may contain stack variables α in addition to the usual value variable x.

We now define:

Definition .1.2

In each calculus, we define
$$\varphi_2 \bigcirc \varphi_1 \stackrel{\text{def}}{=} \varphi_1[\varphi_2] \quad \text{and} \quad \varphi \bullet t \stackrel{\text{def}}{=} t[\varphi]$$
We also define
$$1_{\bigcirc} \stackrel{\text{def}}{=} (\mathrm{Id}_{\mathcal{V}}, \square) \quad (\text{resp. } 1_{\bigcirc} \stackrel{\text{def}}{=} \mathrm{Id}_{\mathcal{V} \cup \mathcal{S}})$$

Fact .1.3

In each calculus, the set of disubstitutions φ has a monoid structure $(\varphi, \bigcirc, 1_{\bigcirc})$ and this monoid acts on commands, expressions, AND VALUES AND STACKSand evaluation contexts via $\bullet$.

Proof

While substitutions that act on both value variables x^{n} and the stack variable $\star^{\mathrm{n}}$ really are substitutions, we call them disubstitutions to avoid any confusion:

Definition .1.4

We call *disubstitutions*, and denote by φ, that act on both value variables and stack variables.

Since we only have one stack variable in $\text{Li}_{\text{n}}^{\rightarrow}$, those are of the shape $\sigma, \star^{\text{n}} \mapsto s_{\text{n}}$. The action of disubstitutions on terms, and their compositions are defined in the expected way. A full description of their action can be found in the right column of Figure **??**.

Since terms are either variable x^{n}, or bind $\star^{\text{n}}$, only having one stack variable $\star^{\text{n}}$ enforces the following property:

Fact .1.5

Term t_{n} have no free stack variables, i.e.

$$\mathsf{FV}_{\mathcal{S}}(t_{\text{n}}) = \emptyset$$

Command c_{n} and evaluation contexts e_{n} have exactly one free stack variable $\star^{\text{n}}$, i.e.

$$\mathsf{FV}_{\mathcal{S}}(c_{\text{n}}) = \mathsf{FV}_{\mathcal{S}}(e_{\text{n}}) = \{\star^{\text{n}}\}$$

Proof

By induction.

Terms having no free stack variables implies disubstitutions can be decomposed as a substitution and a disubstitution of the shape $\star^{\text{n}} \mapsto s_{\text{n}}$:

Fact .1.6

Given a disubstitution $\varphi = \sigma, \star^{\text{n}} \mapsto s_{\text{n}}$:

- for any expression, evaluation context or command t,

$$t[\sigma, s_{\text{n}}/\star^{\text{n}}] = t[\sigma][s_{\text{n}}/\star^{\text{n}}]$$

  - for any expression t_{n},

$$t_{\text{n}}[\sigma, s_{\text{n}}/\star^{\text{n}}] = t_{\text{n}}[\sigma]$$

Proof

- $\boxed{t_{\text{n}}[\sigma, s_{\text{n}}/\star^{\text{n}}] = t_{\text{n}}[\sigma]}$ By induction on t_{n}.
- $\boxed{t[\sigma, s_{\text{n}}/\star^{\text{n}}] = t[\sigma][s_{\text{n}}/\star^{\text{n}}]}$ By induction on t, using the fact that $\sigma[s_{\text{n}}/\star^{\text{n}}] = \sigma$ by the previous bullet.

This also allows simplifying the composition of two disubstitutions:

Fact .1.7

For any disubstitutions $\varphi_1 = \sigma_1, \star^{n} \mapsto s^1_{n}$ and $\varphi_2 = \sigma_2, \star^{n} \mapsto s^2_{n}$, we have

$$\varphi_1[\varphi_2] = \sigma_1[\sigma_2], \star^{n} \mapsto s^1_{n}[s^2_{n}[\sigma_1]/\star^{n}]$$

Proof

By the previous fact,

$$\varphi_1[\varphi_2] = (\sigma_1[\varphi_2], \star^{n} \mapsto s^1_{n}[\varphi_2])\sigma_1[\sigma_2], \star^{n} \mapsto s^1_{n}[s^2_{n}[\sigma_1]/\star^{n}]$$

.2. Properties of reductions

.3. Detailed proofs

Fact A.1.8: Equivalence between $\triangleright^{\circledast}$ and $\bowtie^{\circledast}$

- The $\bowtie$-normal expressions are exactly the $\triangleright$-normal expressions:
$$T_{\mathrm{N}} \not\bowtie \Leftrightarrow T_{\mathrm{N}} \not\triangleright$$
- The $\sqsubset$ steps can be postponed at the cost of strengthening $\triangleright_{\mathsf{let}}$ to $\triangleright_{\mathsf{let}}$:
$$T_{\mathrm{N}} \bowtie^{*} T'_{\mathrm{N}} \Leftrightarrow T_{\mathrm{N}} \triangleright^{*}\sqsubset^{*} T'_{\mathrm{N}}$$
- Evaluating with $\bowtie$ or $\triangleright$ yields the same result:
$$T_{\mathrm{N}} \bowtie^{\circledast} T'_{\mathrm{N}} \Leftrightarrow T_{\mathrm{N}} \triangleright^{\circledast} T'_{\mathrm{N}}$$

Proof of Fact I.1.8 from page 25

Recall that
$$\triangleright = \triangleright_{\rightarrow} \cup \triangleright_{\mathsf{let}} \quad \text{and} \quad \bowtie = \triangleright_{\rightarrow} \cup \triangleright_{\mathsf{let}} \cup \sqsubset$$

- $\boxed{T_{\mathrm{N}} \not\triangleright \Leftrightarrow T_{\mathrm{N}} \not\bowtie}$ Take $\mathbb{S}_{\mathrm{N}}$ maximal such that $T_{\mathrm{N}} = \mathbb{S}_{\mathrm{N}}\boxed{U_{\mathrm{N}}}$. The result is immediate by case analysis on $\mathbb{S}_{\mathrm{N}}$ and U_{N}.
- $\boxed{T_{\mathrm{N}} \bowtie^{*} T'_{\mathrm{N}} \Leftarrow T_{\mathrm{N}} \triangleright^{*}\sqsubset^{*} T'_{\mathrm{N}}}$ Since we have $\sqsubset \cup \triangleright_{\rightarrow} \subseteq \bowtie$ by definition, it suffices to show that $\triangleright_{\mathsf{let}} \subseteq \bowtie^{*}$. This is immediate: any reduction
$$(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}})V^{1}_{\mathrm{N}} \ldots V^{q}_{\mathrm{N}} \triangleright_{\mathsf{let}} (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}])V^{1}_{\mathrm{N}} \ldots V^{q}_{\mathrm{N}}$$
can be simulated by
$$\begin{aligned}(\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}})V^{1}_{\mathrm{N}} \ldots V^{q}_{\mathrm{N}} &\sqsubset (\mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V^{1}_{\mathrm{N}})V^{2}_{\mathrm{N}} \ldots V^{q}_{\mathrm{N}} \\ &\sqsubset^{*} \mathsf{let}\ x^{\mathrm{N}} := T_{\mathrm{N}}\ \mathsf{in}\ U_{\mathrm{N}}V^{1}_{\mathrm{N}} \ldots V^{q}_{\mathrm{N}} \\ &\triangleright_{\mathsf{let}} (U_{\mathrm{N}}[T_{\mathrm{N}}/x^{\mathrm{N}}])V^{1}_{\mathrm{N}} \ldots V^{q}_{\mathrm{N}}\end{aligned}$$
- $\boxed{T_{\mathrm{N}} \sqsubset^{*}\triangleright T'_{\mathrm{N}} \Rightarrow T_{\mathrm{N}} \triangleright T'_{\mathrm{N}}}$ By induction on the number of $\sqsubset$ steps and case analysis on T_{N}.
- $\boxed{T_{\mathrm{N}} \bowtie^{*} T'_{\mathrm{N}} \Rightarrow T_{\mathrm{N}} \triangleright^{*}\sqsubset^{*} T'_{\mathrm{N}}}$ Suppose that $T_{\mathrm{N}} \bowtie^{*} T'_{\mathrm{N}}$. By definition of $\bowtie$ (and monotonicity of the reflexive transitive closure), we have $T_{\mathrm{N}} (\triangleright \cup \sqsubset)^{*} T'_{\mathrm{N}}$. By the previous bullet, this simplifies to $T_{\mathrm{N}} \triangleright^{*}\sqsubset^{*} T'_{\mathrm{N}}$.
- $\boxed{T_{\mathrm{N}} \bowtie^{\circledast} T'_{\mathrm{N}} \Leftrightarrow T_{\mathrm{N}} \triangleright^{\circledast} T'_{\mathrm{N}}}$ The $\Leftarrow$ implication follows from the previous bullets. Now suppose that $T_{\mathrm{N}} \bowtie^{\circledast} T'_{\mathrm{N}}$. By the previous bullets, we have $T_{\mathrm{N}} \triangleright^{*}\sqsubset^{l} T'_{\mathrm{N}} \not\triangleright$ for some l. Since any $\sqsubset$-reduct is $\triangleright$-reducible, we necessarily have $l = 0$, and we are done.

Fact A.4.3

In $\underline{\lambda}_{\mathrm{N}}^{\rightarrow}$ (resp. $\mathrm{M}_{\mathrm{N}}^{\rightarrow}$), the set of stacks $\mathbf{S}_{\mathrm{N}}$ has a monoid structure

$$(\mathbf{S}_{\mathrm{N}}, \circ_{\square}, \square) \quad (\text{resp. } (\mathbf{S}_{\mathrm{N}}, \circ_{\star}, \star^{\mathrm{N}}))$$

where

$$\mathbb{S}_{\mathrm{N}}^{2} \circ_{\square} \mathbb{S}_{\mathrm{N}}^{1} \stackrel{\text{def}}{=} \mathbb{S}_{\mathrm{N}}^{2}\boxed{\mathbb{S}_{\mathrm{N}}^{1}} \quad (\text{resp. } S_{\mathrm{N}}^{1} \circ_{\star} S_{\mathrm{N}}^{2} \stackrel{\text{def}}{=} S_{\mathrm{N}}^{1}[S_{\mathrm{N}}^{2}/\star^{\mathrm{N}}])$$

and this monoid acts on configurations on the left (resp. on the right) via

$$\mathbb{S}_{\mathrm{N}} \bullet_{\square} C_{\mathrm{N}} \stackrel{\text{def}}{=} \mathbb{S}_{\mathrm{N}}\boxed{C_{\mathrm{N}}} \quad (\text{resp. } C_{\mathrm{N}} \bullet_{\star} S_{\mathrm{N}} \stackrel{\text{def}}{=} C_{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}])$$

In other words:

- (mon-unit) for any stack $\mathbb{S}_{\mathrm{N}}$ (resp. S_{N}), we have
$$\square\boxed{\mathbb{S}_{\mathrm{N}}} = \mathbb{S}_{\mathrm{N}} = \mathbb{S}_{\mathrm{N}}\boxed{\square} \quad (\text{resp. } \star^{\mathrm{N}}[S_{\mathrm{N}}/\star^{\mathrm{N}}] = S_{\mathrm{N}} = S_{\mathrm{N}}[\star^{\mathrm{N}}/\star^{\mathrm{N}}])$$
- (mon-accoc) for any stacks $\mathbb{S}_{\mathrm{N}}^{1}$, $\mathbb{S}_{\mathrm{N}}^{2}$, and $\mathbb{S}_{\mathrm{N}}^{3}$ (resp. S_{N}^{1}, S_{N}^{2}, and S_{N}^{3}), we have
$$\mathbb{S}_{\mathrm{N}}^{3}\boxed{\mathbb{S}_{\mathrm{N}}^{2}\boxed{\mathbb{S}_{\mathrm{N}}^{1}}} = \left(\mathbb{S}_{\mathrm{N}}^{3}\boxed{\mathbb{S}_{\mathrm{N}}^{2}}\right)\boxed{\mathbb{S}_{\mathrm{N}}^{1}} \quad (\text{resp. } S_{\mathrm{N}}^{1}[S_{\mathrm{N}}^{2}/\star^{\mathrm{N}}][S_{\mathrm{N}}^{3}/\star^{\mathrm{N}}] = S_{\mathrm{N}}^{1}[S_{\mathrm{N}}^{2}[S_{\mathrm{N}}^{3}/\star^{\mathrm{N}}]/\star^{\mathrm{N}}])$$
- (act-unit) for any configuration C_{N}, we have
$$\square\boxed{C_{\mathrm{N}}} = C_{\mathrm{N}} \quad (\text{resp. } C_{\mathrm{N}} = C_{\mathrm{N}}[\star^{\mathrm{N}}/\star^{\mathrm{N}}])$$
- (act-assoc) for any configuration C_{N}^{1} and stacks $\mathbb{S}_{\mathrm{N}}^{2}$ and $\mathbb{S}_{\mathrm{N}}^{3}$ (resp. S_{N}^{2} and S_{N}^{3}), we have
$$\mathbb{S}_{\mathrm{N}}^{3}\boxed{\mathbb{S}_{\mathrm{N}}^{2}\boxed{C_{\mathrm{N}}^{1}}} = \left(\mathbb{S}_{\mathrm{N}}^{3}\boxed{\mathbb{S}_{\mathrm{N}}^{2}}\right)\boxed{C_{\mathrm{N}}^{1}} \quad (\text{resp. } C_{\mathrm{N}}^{1}[S_{\mathrm{N}}^{2}/\star^{\mathrm{N}}][S_{\mathrm{N}}^{3}/\star^{\mathrm{N}}] = C_{\mathrm{N}}^{1}[S_{\mathrm{N}}^{2}[S_{\mathrm{N}}^{3}/\star^{\mathrm{N}}]/\star^{\mathrm{N}}])$$

Proof of Fact I.4.3 from page 38

- (mon-unit) One equality is by definition and the other is by induction on $\mathbb{S}_{\mathrm{N}}$ (resp. S_{N}).
- (mon-accoc) By induction on the size of $\mathbb{S}_{\mathrm{N}}^{2}$ (resp. S_{N}^{2}). The base case $\mathbb{S}_{\mathrm{N}}^{2} = \square$ (resp. $S_{\mathrm{N}}^{2} = \star^{\mathrm{N}}$), follows from (i). The inductive case follows from several applications of the induction hypothesis.
- (act-unit) By definition (resp. by induction on C_{N}).
- (act-assoc) By induction on $\mathbb{S}_{\mathrm{N}}^{2}$ (resp. S_{N}^{2}). The base case $\mathbb{S}_{\mathrm{N}}^{2} = \square$ (resp. $S_{\mathrm{N}}^{2} = \star^{\mathrm{N}}$) follows from (i) and (iii). The inductive case follows from several applications of the induction hypothesis.

Fact E.2.8

The following are equivalent:

- (i) there exists a derivation of well-polarization which is valid in $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$ but not in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$;
- (ii) there exists a derivation of well-polarization which is valid in $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$ but not in $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}}$, and whose conclusion is of the shape
$$c : (\Gamma \vdash) \quad \text{or} \quad \Gamma \mid \underline{e_\varepsilon : \varepsilon} \vdash$$
i.e. has no succedent;
- (iii) there exists a stack s_ε in $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$ such that $\Gamma \mid \underline{s_\varepsilon : \varepsilon} \vdash$ is derivable for some Γ;
- (iv) at least one of the following holds:
  - (a) there exists a stack constructor $\mathfrak{s}_k^{\tau_-^j}$ with zero stack arguments (e.g. $\neg_-(v_+)$ or $\tilde{()}$); or
  - (b) there exists a postitive type former τ_+^j whose value constructors $\mathfrak{v}_k^{\tau_+^j}$ all have exactly one stack arguments (e.g. $\neg_+$ or 0).
- (v) there exists a stack s_ε in $\mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$ of the shape
$$s_\varepsilon = \mathfrak{s}_k^{\tau_-^j}(\vec{x}) \quad (\text{e.g. } \neg_-(x^+) \quad \text{or} \quad \tilde{()})$$
or
$$s_\varepsilon = \tilde{\mu}\begin{bmatrix} \mathfrak{v}_1^{\tau_+^j}(\vec{x_1}, \alpha_1^{\varepsilon_1}, \vec{y_1}).\langle z_1^{\varepsilon_1} | \alpha_1^{\varepsilon_1}\rangle^{\varepsilon_1} \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\vec{x_l}, \alpha_l^{\varepsilon_l}, \vec{y_l}).\langle z_l^{\varepsilon_l} | \alpha_l^{\varepsilon_l}\rangle^{\varepsilon_l} \end{bmatrix} \quad (\text{e.g. } \tilde{\mu}\neg_+(\alpha^-).\alpha\langle x^- | \alpha^-\rangle^- \quad \text{or} \quad \tilde{\mu}[\,])$$

Furthermore, if all positive type formers in $\vec{\tau}$ have at least one constructor (i.e. there are no copies of 0), then these are also equivalent to:

- (vi) $\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \subsetneq \mathrm{Li}_{\mathrm{p}}^{\vec{\tau}}$.

In particular, for $\vec{\tau} \subseteq \{\rightarrow \Downarrow \Uparrow \neg_- \neg_+ \otimes ⅋ \oplus \& 1 \bot \top\}$[a], we have
$$\mathrm{Lm}_{\mathrm{p}}^{\vec{\tau}} \subsetneq \mathrm{Li}_{\mathrm{p}}^{\vec{\tau}} \quad \Leftrightarrow \quad \vec{\tau} \cap \{\neg_- \neg_+ \bot\} \neq \emptyset$$

[a] Note the absence of 0.

Proof of Fact V.2.8 from page 128

- (i) ⇒ (ii) This derivation necessarily uses a sequent of the shape
$$c:(\Gamma\vdash)\quad\text{or}\quad \Gamma\mid \underline{e_\varepsilon:\varepsilon}\vdash$$
and there is therefore a subderivation whose conclusion is that sequent.

- (ii) ⇒ (iii) By induction on the derivation: if the derivation ends with
$$\frac{\Gamma_1\vdash \underline{t_\varepsilon:A_\varepsilon}\mid \qquad \Gamma_2\mid \underline{e_\varepsilon:A_\varepsilon}\vdash}{\langle t_\varepsilon|e_\varepsilon\rangle^\varepsilon:(\Gamma_1,\Gamma_2\vdash)}\ (\textsc{cut})\qquad \left(\text{resp. } \frac{c:(\Gamma,x^\varepsilon:A_\varepsilon\vdash)}{\Gamma\mid \underline{\tilde\mu x^\varepsilon.c:A_\varepsilon}\vdash}\ (\tilde\mu\vdash)\right)$$
then we apply the induction hypothesis to the derivation of $\Gamma_2\mid \underline{e_\varepsilon:A_\varepsilon}\vdash$ (resp. $c:(\Gamma,x^\varepsilon:A_\varepsilon\vdash)$).

- (iii) ⇒ (iv) By induction on the derivation. If the last rule of the derivation is
$$\frac{\begin{array}{ccc}\Gamma_1\vdash \underline{v^1_{\varepsilon_1}:A^1_{\varepsilon_1}}\mid & \dots & \Gamma_q\vdash \underline{v^q_{\varepsilon_q}:A^q_{\varepsilon_q}}\mid\\ \Gamma_{q+1}\mid \underline{s^1_{\varepsilon_{q+1}}:A^{q+1}_{\varepsilon_{q+1}}}\vdash & \dots & \Gamma_{q+r}\mid \underline{s^r_{\varepsilon_{q+r}}:A^{q+r}_{\varepsilon_{q+r}}}\vdash\end{array}}{\Gamma_1,\dots,\Gamma_{q+r}\mid \underline{\mathfrak{s}^{\tau^j_-}_k(v^1_{\varepsilon_1},\dots,v^q_{\varepsilon_q},s^1_{\varepsilon_{q+1}},\dots,s^r_{\varepsilon_{q+r}}):\tau^j_-(\vec B)}\vdash}\ (\mathfrak{s}^{\tau^j_-}_k\vdash)$$
then either $r>0$ and we apply the induction hypothesis to one of the derivations of $\Gamma_{q+k}\mid \underline{s^k_{\varepsilon_{q+k}}:A^{q+k}_{\varepsilon_{q+k}}}\vdash$, or $r=0$, and we can immediately conclude that (iv). If the last rule is
$$\frac{c_1:\left(\Gamma,\vec{x_1}:\vec{A^1}\vdash \alpha_1^{\varepsilon_r}:B^1_{\varepsilon_r}\right)\qquad \dots\qquad c_l:\left(\Gamma,\vec{x_l}:\vec{A^l}\vdash \alpha_l^{\varepsilon_r}:B^l_{\varepsilon_r}\right)}{\Gamma\mid \underline{\tilde\mu\left[\mathfrak{d}^{\tau^j_+}_1(\vec{x_1},\alpha_1).c_1|\dots|\mathfrak{d}^{\tau^j_+}_l(\vec{x_l},\alpha_l).c_l\right]:\tau^j_+(\vec C)}\vdash}\ (\tau^j_+\vdash)$$
we can immediately conclude that (iv).

- (iv) ⇒ (v) Immediate.

- (v) ⇒ (i) If $s_\varepsilon\neq\tilde\mu[]$, then we have $s_\varepsilon\notin \mathrm{Lm}^{\vec\tau}_{\mathrm{p}}$, and in particular, the derivation that shows that $s_\varepsilon\in \mathrm{Li}^{\vec\tau}_{\mathrm{p}}$ works. For $s_\varepsilon=\tilde\mu[]$, there are two possible shapes for the derivation of $s_\varepsilon\in \mathrm{Li}^{\vec\tau}_{\mathrm{p}}$
$$\frac{}{\Gamma\mid \underline{\tilde\mu[]:0}\vdash \alpha^\varepsilon:A_\varepsilon}\ (0\vdash)\qquad\text{and}\qquad \frac{}{\Gamma\mid \underline{\tilde\mu[]:0}\vdash}\ (0\vdash)$$
and while the former is also valid in $\mathrm{Lm}^{\vec\tau}_{\mathrm{p}}$, the latter is not.

- (v) ⇒ (iv) We have $s_\varepsilon\in \mathrm{Li}^{\vec\tau}_{\mathrm{p}}\setminus \mathrm{Lm}^{\vec\tau}_{\mathrm{p}}$. This is immediate for the case $s_\varepsilon=\mathfrak{s}^{\tau^j_-}_k(\vec x)$, and since all positive type formers have at least one constructor, the case

$$s_\varepsilon = \widetilde{\mu}\begin{bmatrix} \mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}, \alpha_1^{\varepsilon_1}, \overrightarrow{y_1}).\ \langle z_1^{\varepsilon_1} | \alpha_1^{\varepsilon_1} \rangle^{\varepsilon_1} \\ \vdots \\ \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}, \alpha_l^{\varepsilon_l}, \overrightarrow{y_l}).\ \langle z_l^{\varepsilon_l} | \alpha_l^{\varepsilon_l} \rangle^{\varepsilon_l} \end{bmatrix}$$

is restricted to $l > 0$, i.e. $s_\varepsilon = \widetilde{\mu}[]$ is ruled out, which ensures that $s_\varepsilon \notin \mathrm{Lm}_\mathrm{p}^{\vec{\tau}}$.

- (iv) ⇒ (i) Take any $t \in \mathrm{Li}_\mathrm{p}^{\vec{\tau}} \setminus \mathrm{Lm}_\mathrm{p}^{\vec{\tau}}$. The derivation that $t \in \mathrm{Li}_\mathrm{p}^{\vec{\tau}}$ works.

Fact E.4.1

In $\lambda_\mathrm{p}^{\vec{\tau}}$ (resp. $\mathrm{Lm}_\mathrm{p}^{\vec{\tau}}$), stacks form a category $\mathcal{C}$ with $\mathrm{Obj}(\mathcal{C}) = \{+, -\}$ and

$$\mathrm{Hom}_\mathcal{C}(\varepsilon_1, \varepsilon_2) = \{(\sigma, \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) | \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \in \mathbf{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\} \quad (\text{resp. } \{(\sigma, \star^{\varepsilon_1} \mapsto s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) | s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \in \mathbf{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}\})$$

whose composition and identities are given by

$$\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \circ_\mathrm{d} \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} \stackrel{\text{def}}{=} \mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}) \quad (\text{resp. } s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \circ_\star s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} \stackrel{\text{def}}{=} s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^\mathrm{N}])$$

and

$$\mathrm{Id}_\varepsilon = \square^\varepsilon \quad (\text{resp. } \mathrm{Id}_\varepsilon = \star^\varepsilon)$$

respectively, and this category acts on commands (on the right) via

$$c_{\rightsquigarrow \varepsilon_1} \bullet_\mathrm{d} \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \stackrel{\text{def}}{=} \mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) \quad (\text{resp. } c_{\rightsquigarrow \varepsilon_1} \bullet_\star s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} \stackrel{\text{def}}{=} c_{\rightsquigarrow \varepsilon_1}[s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}])$$

In other words:

- cat-closure for any stacks $\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ and $s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$), $\mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3})$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}]$) is a stack $\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_3}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_3}$).

- cat-id for any stack $\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$), we have

$$\mathsf{defer}(\square^{\varepsilon_1}, \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}) = \mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2} = \mathsf{defer}(\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \square^{\varepsilon_2})$$
$$(\text{resp. } \star^{\varepsilon_1}[s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}] = s_{\varepsilon_1 \rightsquigarrow \varepsilon_2} = s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[\star^{\varepsilon_2} / \star^{\varepsilon_2}])$$

- cat-accoc for any stacks $\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$, $\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$, and $\mathbb{s}^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$, $s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}$, and $s^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}$), we have

$$\mathsf{defer}(\mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}), \mathbb{s}^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}) = \mathsf{defer}(\mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}, \mathsf{defer}(\mathbb{s}^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}, \mathbb{s}^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4}))$$
$$(\text{resp. } s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3} / \star^{\varepsilon_2}][s^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4} / \star^{\varepsilon_3}] = s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2}[s^2_{\varepsilon_2 \rightsquigarrow \varepsilon_3}[s^3_{\varepsilon_3 \rightsquigarrow \varepsilon_4} / \star^{\varepsilon_3}] / \star^{\varepsilon_2}])$$

- act-closure for any command $c_{\rightsquigarrow \varepsilon_1}$ and stack $\mathbb{s}_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$ (resp. $s_{\varepsilon_1 \rightsquigarrow \varepsilon_2}$), $\mathsf{defer}(c_{\rightsquigarrow \varepsilon_1}, \mathbb{s}^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2})$ (resp. $c_{\rightsquigarrow \varepsilon_1}[s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} / \star^{\varepsilon_1}]$) is a command $c^2_{\rightsquigarrow \varepsilon_2}$.

- act-id for any command $c_{\rightsquigarrow \varepsilon}$, we have

$$\mathsf{defer}(c_{\rightsquigarrow \varepsilon}, \square^\varepsilon) = c_{\rightsquigarrow \varepsilon} \quad (\text{resp. } c_{\rightsquigarrow \varepsilon}[\star^\varepsilon / \star^\varepsilon] = c_{\rightsquigarrow \varepsilon})$$

- act-assoc for any command $c_{\rightsquigarrow\varepsilon_1}$ and stacks $\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ and $\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}$ (resp. $s^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ and $s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}$), we have

$$\mathsf{defer}\big(\mathsf{defer}(c_{\rightsquigarrow\varepsilon_1}, \mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}), \mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}\big) = \mathsf{defer}\big(c_{\rightsquigarrow\varepsilon_1}, \mathsf{defer}(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}, \mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3})\big)$$
$$(\text{resp. } c_{\rightsquigarrow\varepsilon_1}[s^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}/\star^{\varepsilon_1}][s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}/\star^{\varepsilon_2}] = c_{\rightsquigarrow\varepsilon_1}[s^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}[s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}/\star^{\varepsilon_2}]/\star^{\varepsilon_1}])$$

Proof of Fact V.4.1 from page 158

- cat-closure By induction on $\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ (resp. $s^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}$).

- act-closure By induction on $c_{\rightsquigarrow\varepsilon_1}$.

- cat-id We have

$$\mathsf{defer}(\Box^{\varepsilon_1}, \mathbb{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}) = \mathbb{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2} \quad (\text{resp. } \star^{\varepsilon_1}[s_{\varepsilon_1\rightsquigarrow\varepsilon_2}/\star^{\varepsilon_1}] = s_{\varepsilon_1\rightsquigarrow\varepsilon_2})$$

  by definition, and

$$\mathbb{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2} = \mathsf{defer}(\mathbb{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}, \Box^{\varepsilon_2}) \quad (\text{resp. } s_{\varepsilon_1\rightsquigarrow\varepsilon_2} = s_{\varepsilon_1\rightsquigarrow\varepsilon_2}[\star^{\varepsilon_2}/\star^{\varepsilon_2}])$$

  by induction on $\mathbb{s}_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ (resp. $s_{\varepsilon_1\rightsquigarrow\varepsilon_2}$).

- act-id By induction on $c_{\rightsquigarrow\varepsilon}$.

- cat-accoc By induction on the size of $\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}$(resp. $s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}$). The base case $\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3} = \Box^{\varepsilon_2}$ (resp. $s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3} = \star^{\varepsilon_2}$) follows from (cat-id). All the inductive case decompose $\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}$(resp. $s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}$) as

$$\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3} = \mathsf{defer}\big(\mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon}, \mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}\big) \quad (\text{resp. } s^2_{\varepsilon_2\rightsquigarrow\varepsilon_3} = s^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon}[s^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}/\star^{\varepsilon}])$$

  where $\mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon}$ and $\mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}$ (resp. $s^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon}$ and $s^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}$) are strictly smaller, and we can immediately conclude by applying the induction hypothesis four times:

$$\begin{aligned}
\mathsf{defer}\big(\mathsf{defer}(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}, \mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}), \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4}\big) &= \big(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}\big) \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4} \\
&= \big(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \big(\mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon} \bigcirc_{\mathrm{d}} \mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}\big)\big) \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4} \\
&= \big(\big(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon}\big) \bigcirc_{\mathrm{d}} \mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}\big) \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4} \\
&= \big(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon}\big) \bigcirc_{\mathrm{d}} \big(\mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3} \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4}\big) \\
&= \mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \big(\mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon} \bigcirc_{\mathrm{d}} \big(\mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3} \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4}\big)\big) \\
&= \mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \big(\big(\mathbb{s}^{2a}_{\varepsilon_2\rightsquigarrow\varepsilon} \bigcirc_{\mathrm{d}} \mathbb{s}^{2b}_{\varepsilon\rightsquigarrow\varepsilon_3}\big) \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4}\big) \\
&= \mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} \bigcirc_{\mathrm{d}} \big(\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3} \bigcirc_{\mathrm{d}} \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4}\big) \\
&= \mathsf{defer}\big(\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}, \mathsf{defer}(\mathbb{s}^2_{\varepsilon_2\rightsquigarrow\varepsilon_3}, \mathbb{s}^3_{\varepsilon_3\rightsquigarrow\varepsilon_4})\big)
\end{aligned}$$

  (resp. the same proof with $\bigcirc_{\star}$).

- act-assoc By induction on the size of $\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}$ (resp. $s^1_{\varepsilon_1\rightsquigarrow\varepsilon_2}$). The base case $\mathbb{s}^1_{\varepsilon_1\rightsquigarrow\varepsilon_2} =$

$\square^{\varepsilon_1}$ (resp. $s^1_{\varepsilon_1 \rightsquigarrow \varepsilon_2} = \star^{\varepsilon_1}$) follows from (act-id). The inductive cases are handled just like those of (cat-assoc).

.4. Extra figures

Figure .4.1: Well polarized $L_p^{\vec{\tau}}$

Figure .4.1.a: Core rules

$$\frac{}{x^\varepsilon : \varepsilon \vdash \underline{x^\varepsilon : \varepsilon} \mid}\ (\vdash\text{AX}) \qquad \frac{}{\mid \underline{\alpha^\varepsilon : \varepsilon} \vdash \alpha^\varepsilon : \varepsilon}\ (\text{AX}\vdash)$$

$$\frac{c : (\Gamma \vdash \alpha^\varepsilon : \varepsilon, \Delta)}{\Gamma \vdash \underline{\mu\alpha^\varepsilon.c : \varepsilon} \mid \Delta}\ (\vdash\mu) \qquad \frac{c : (\Gamma, x^\varepsilon : \varepsilon \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}x^\varepsilon.c : \varepsilon} \vdash \Delta}\ (\tilde{\mu}\vdash)$$

$$\frac{\Gamma_1 \vdash \underline{t_\varepsilon : \varepsilon} \mid \Delta_1 \qquad \Gamma_2 \mid \underline{e_\varepsilon : \varepsilon} \vdash \Delta_2}{\langle t_\varepsilon | e_\varepsilon \rangle^\varepsilon : (\Gamma_1, \Gamma_2 \vdash \Delta_1, \Delta_2)}\ (\text{CUT})$$

Figure .4.1.b: Structural rules (commands)

$$\frac{c : (\Gamma \vdash \Delta)}{c : (\Gamma \vdash \alpha^\varepsilon : \varepsilon, \Delta)}\ (\vdash\text{W}c) \qquad \frac{c : (\Gamma \vdash \alpha_1^\varepsilon : \varepsilon, \alpha_2^\varepsilon : \varepsilon, \Delta)}{c[\beta^\varepsilon/\alpha_1^\varepsilon, \beta^\varepsilon/\alpha_2^\varepsilon] : (\Gamma \vdash \beta^\varepsilon : \varepsilon, \Delta)}\ (\vdash\text{C}c)$$

$$\frac{c : (\Gamma \vdash \Delta)}{c : (\Gamma, x^\varepsilon : \varepsilon \vdash \Delta)}\ (\text{W}c\vdash) \qquad \frac{c : (\Gamma, x_1^\varepsilon : \varepsilon, x_2^\varepsilon : \varepsilon \vdash \Delta)}{c[y^\varepsilon/x_1^\varepsilon, y^\varepsilon/x_2^\varepsilon] : (\Gamma, y^\varepsilon : \varepsilon \vdash \Delta)}\ (\text{C}c\vdash)$$

$$\frac{c : (\Gamma \vdash \Delta_1, \alpha_1^\varepsilon : \varepsilon, \alpha_2^\varepsilon : \varepsilon, \Delta_2)}{c : (\Gamma \vdash \Delta_1, \alpha_2^\varepsilon : \varepsilon, \alpha_1^\varepsilon : \varepsilon, \Delta_2)}\ (\vdash\text{P}c) \qquad \frac{c : (\Gamma_1, x_1^\varepsilon : \varepsilon, x_2^\varepsilon : \varepsilon, \Gamma_2 \vdash \Delta)}{c : (\Gamma_1, x_2^\varepsilon : \varepsilon, x_1^\varepsilon : \varepsilon, \Gamma_2 \vdash \Delta)}\ (\text{P}c\vdash)$$

Figure .4.1.c: Structural rules (expressions)

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta}{\Gamma \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \alpha^\varepsilon : \varepsilon, \Delta}\ (\vdash\text{W}t) \qquad \frac{\Gamma \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \alpha_1^\varepsilon : \varepsilon, \alpha_2^\varepsilon : \varepsilon, \Delta}{\Gamma \vdash \underline{t_{\varepsilon_0}[\beta^\varepsilon/\alpha_1^\varepsilon, \beta^\varepsilon/\alpha_2^\varepsilon] : \varepsilon_0} \mid \beta^\varepsilon : \varepsilon, \Delta}\ (\vdash\text{C}t)$$

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta}{\Gamma, x^\varepsilon : \varepsilon \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta}\ (\text{W}t\vdash) \qquad \frac{\Gamma, x_1^\varepsilon : \varepsilon, x_2^\varepsilon : \varepsilon \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta}{\Gamma, x^\varepsilon : \varepsilon \vdash \underline{t_{\varepsilon_0}[x^\varepsilon/x_1^\varepsilon, x^\varepsilon/x_2^\varepsilon] : \varepsilon_0} \mid \Delta}\ (\text{C}t\vdash)$$

$$\frac{\Gamma \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta_1, \alpha_1^\varepsilon : \varepsilon, \alpha_2^\varepsilon : \varepsilon, \Delta_2}{\Gamma \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta_1, \alpha_2^\varepsilon : \varepsilon, \alpha_1^\varepsilon : \varepsilon, \Delta_2}\ (\vdash\text{P}t) \qquad \frac{\Gamma_1, x_1^\varepsilon : \varepsilon, x_2^\varepsilon : \varepsilon, \Gamma_2 \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta}{\Gamma_1, x_2^\varepsilon : \varepsilon, x_1^\varepsilon : \varepsilon, \Gamma_2 \vdash \underline{t_{\varepsilon_0} : \varepsilon_0} \mid \Delta}\ (\text{P}t\vdash)$$

Figure .4.1.d: Structural rules (evaluation contexts)

$$\frac{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \alpha^{\varepsilon} : \varepsilon, \Delta}\ (\vdash\mathrm{we}) \qquad \frac{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \alpha_1^{\varepsilon} : \varepsilon, \alpha_2^{\varepsilon} : \varepsilon, \Delta}{\Gamma \mid \underline{e_{\varepsilon_0}[\beta^{\varepsilon}/\alpha_1^{\varepsilon}, \beta^{\varepsilon}/\alpha_2^{\varepsilon}] : \varepsilon_0} \vdash \beta^{\varepsilon} : \varepsilon, \Delta}\ (\vdash\mathrm{ce})$$

$$\frac{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}{\Gamma, x^{\varepsilon} : \varepsilon \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}\ (\mathrm{we}\vdash) \qquad \frac{\Gamma, x_1^{\varepsilon} : \varepsilon, x_2^{\varepsilon} : \varepsilon \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}{\Gamma, x^{\varepsilon} : \varepsilon \mid \underline{e_{\varepsilon_0}[x^{\varepsilon}/x_1^{\varepsilon}, x^{\varepsilon}/x_2^{\varepsilon}] : \varepsilon_0} \vdash \Delta}\ (\mathrm{ce}\vdash)$$

$$\frac{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta_1, \alpha_1^{\varepsilon} : \varepsilon, \alpha_2^{\varepsilon} : \varepsilon, \Delta_2}{\Gamma \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta_1, \alpha_2^{\varepsilon} : \varepsilon, \alpha_1^{\varepsilon} : \varepsilon, \Delta_2}\ (\vdash\mathrm{pe}) \qquad \frac{\Gamma_1, x_1^{\varepsilon} : \varepsilon, x_2^{\varepsilon} : \varepsilon, \Gamma_2 \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}{\Gamma_1, x_2^{\varepsilon} : \varepsilon, x_1^{\varepsilon} : \varepsilon, \Gamma_2 \mid \underline{e_{\varepsilon_0} : \varepsilon_0} \vdash \Delta}\ (\mathrm{pe}\vdash)$$

Figure .4.1.e: General shape of logic rules

$$\frac{\begin{array}{ccc} \Gamma_1 \vdash \underline{v_{\varepsilon_1}^1 : \varepsilon_1} \mid \Delta_1 & \dots & \Gamma_q \vdash \underline{v_{\varepsilon_q}^q : \varepsilon_q} \mid \Delta_q \\ \Gamma_{q+1} \mid \underline{s_{\varepsilon_{q+1}}^1 : \varepsilon_{q+1}} \vdash \Delta_{q+1} & \dots & \Gamma_{q+r} \mid \underline{s_{\varepsilon_{q+r}}^r : \varepsilon_{q+r}} \vdash \Delta_{q+r} \end{array}}{\Gamma_1, \dots, \Gamma_{q+r} \mid \underline{\mathfrak{s}_k^{\tau_-^j}(v_{\varepsilon_1}^1, \dots, v_{\varepsilon_q}^q, s_{\varepsilon_{q+1}}^1, \dots, s_{\varepsilon_{q+r}}^r) : -} \vdash \Delta_1, \dots, \Delta_{q+r}}\ \left(\mathfrak{s}_k^{\tau_-^j}\vdash\right)$$

$$\frac{c_1 : \left(\Gamma, \overrightarrow{x_1} : \overrightarrow{\varepsilon_1} \vdash \overrightarrow{\alpha_1} : \overrightarrow{\varepsilon'_1}, \Delta\right) \quad \dots \quad c_l : \left(\Gamma, \overrightarrow{x_l} : \overrightarrow{\varepsilon_l} \vdash \overrightarrow{\alpha_l} : \overrightarrow{\varepsilon'_l}, \Delta\right)}{\Gamma \vdash \underline{\mu\left\langle \mathfrak{s}_1^{\tau_-^j}(\overrightarrow{x_1}, \overrightarrow{\alpha_1}).c_1 | \dots | \mathfrak{s}_l^{\tau_-^j}(\overrightarrow{x_l}, \overrightarrow{\alpha_l}).c_l \right\rangle : -} \mid \Delta}\ (\vdash\tau_-^j)$$

$$\frac{\begin{array}{ccc} \Gamma_1 \vdash \underline{v_{\varepsilon_1}^1 : \varepsilon_1} \mid \Delta_1 & \dots & \Gamma_q \vdash \underline{v_{\varepsilon_q}^q : \varepsilon_q} \mid \Delta_q \\ \Gamma_{q+1} \mid \underline{s_{\varepsilon_{q+1}}^1 : \varepsilon_{q+1}} \vdash \Delta_{q+1} & \dots & \Gamma_{q+r} \mid \underline{s_{\varepsilon_{q+r}}^r : \varepsilon_{q+r}} \vdash \Delta_{q+r} \end{array}}{\Gamma_1, \dots, \Gamma_q \vdash \underline{\mathfrak{v}_k^{\tau_+^j}(v_{\varepsilon_1}^1, \dots, v_{\varepsilon_q}^q, s_{\varepsilon_{q+1}}^1, \dots, s_{\varepsilon_{q+r}}^r) : +} \mid \Delta_1, \dots, \Delta_q}\ \left(\vdash\mathfrak{v}_k^{\tau_+^j}\right)$$

$$\frac{c_1 : \left(\Gamma, \overrightarrow{x_1} : \overrightarrow{\varepsilon_1} \vdash \overrightarrow{\alpha_1} : \overrightarrow{\varepsilon'_1}, \Delta\right) \quad \dots \quad c_l : \left(\Gamma, \overrightarrow{x_l} : \overrightarrow{\varepsilon_l} \vdash \overrightarrow{\alpha_l} : \overrightarrow{\varepsilon'_l}, \Delta\right)}{\Gamma \mid \underline{\tilde{\mu}\left[\mathfrak{v}_1^{\tau_+^j}(\overrightarrow{x_1}, \overrightarrow{\alpha_1}).c_1 | \dots | \mathfrak{v}_l^{\tau_+^j}(\overrightarrow{x_l}, \overrightarrow{\alpha_l}).c_l \right] : +} \vdash \Delta}\ (\tau_+^j\vdash)$$

Figure .4.1.f: Logic rules for multiplicative types

$$\frac{c : (\Gamma, x^+ : + \vdash \alpha^- : -, \Delta)}{\Gamma \vdash \underline{\mu(x^+ \cdot \alpha^-).c : -} \mid \Delta}\ (\vdash\to) \qquad \frac{\Gamma_1 \vdash \underline{v_+ : +} \mid \Delta_1 \qquad \Gamma_2 \mid \underline{s_- : -} \vdash \Delta_2}{\Gamma_1, \Gamma_2 \mid \underline{v_+ \cdot s_- : -} \vdash \Delta_1, \Delta_2}\ (\to\vdash)$$

$$\frac{c : (\Gamma \vdash \alpha^- : -, \beta^- : -, \Delta)}{\Gamma \vdash \underline{\mu(\alpha^- \parr \beta^-).c : -} \mid \Delta}\ (\vdash\parr) \qquad \frac{\Gamma_1 \mid \underline{s_-^1 : -} \vdash \Delta_1 \qquad \Gamma_2 \mid \underline{s_-^2 : -} \vdash \Delta_2}{\Gamma_1, \Gamma_2 \mid \underline{(s_-^1 \parr s_-^2) : -} \vdash \Delta_1, \Delta_2}\ (\parr\vdash)$$

$$\frac{\Gamma_1 \vdash \underline{v_+^1 : +} \mid \Delta_1 \qquad \Gamma_2 \vdash \underline{v_+^2 : +} \mid \Delta_2}{\Gamma_1, \Gamma_2 \vdash \underline{(v_+^1 \otimes v_+^2) : +} \mid \Delta_1, \Delta_2}\ (\vdash\otimes) \qquad \frac{c : (\Gamma, x^+ : +, y^+ : + \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}(x^+ \otimes y^-).c : +} \vdash \Delta}\ (\otimes\vdash)$$

$$\frac{c : (\Gamma \vdash \Delta)}{\Gamma \vdash \underline{\mu\tilde{()}.c : -} \mid \Delta}\ (\vdash\bot) \qquad \frac{}{\mid \underline{\tilde{()} : -} \vdash}\ (\bot\vdash)$$

$$\frac{}{\vdash \underline{() : +} \mid}\ (1\vdash) \qquad \frac{c : (\Gamma \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}().c : +} \vdash \Delta}\ (\vdash 1)$$

Figure .4.1.g: Logic rules for additive types

$$\frac{c_1 : (\Gamma \vdash \alpha_1^- : -, \Delta) \qquad c_2 : (\Gamma \vdash \alpha_2^- : -, \Delta)}{\Gamma \vdash \underline{\mu\langle(\pi_1 \cdot \alpha_1^-).c_1 \mid (\pi_2 \cdot \alpha_2^-).c_2\rangle : -} \mid \Delta}\ (\vdash\&) \qquad \frac{\Gamma \mid \underline{s_- : -} \vdash \Delta}{\Gamma \mid \underline{\pi_i \cdot s_- : -} \vdash \Delta}\ (\&\vdash)$$

$$\frac{\Gamma \vdash \underline{v_+ : +} \mid \Delta}{\Gamma \vdash \underline{\iota_i(v_+) : +} \mid \Delta}\ (\vdash\oplus) \qquad \frac{c_1 : (\Gamma, x_1^+ : + \vdash \Delta) \qquad c_2 : (\Gamma, x_2^+ : + \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}[\iota_1(x_1^+).c_1 \mid \iota_2(x_2^+).c_2] : +} \vdash \Delta}\ (\oplus\vdash)$$

$$\frac{}{\Gamma \vdash \underline{\mu\langle\rangle : -} \mid \Delta}\ (\vdash\top)$$

(No ($\top\vdash$) rule)

(No ($\vdash 0$) rule)

$$\frac{}{\Gamma \mid \underline{\tilde{\mu}[] : +} \vdash \Delta}\ (0\vdash)$$

Figure .4.1.h: Logic rules for shifts

$$\frac{c : (\Gamma \vdash \alpha^+ : +, \Delta)}{\Gamma \vdash \underline{\mu\{\alpha^+\}.c : -} \mid \Delta}\ (\vdash\Uparrow) \qquad \frac{\Gamma \mid \underline{s_+ : +} \vdash \Delta}{\Gamma \mid \underline{\{s_+\} : -} \vdash \Delta}\ (\Uparrow\vdash)$$

$$\frac{\Gamma \vdash \underline{v_- : -} \mid \Delta}{\Gamma \vdash \underline{\{v_-\} : +} \mid \Delta}\ (\vdash\Downarrow) \qquad \frac{c : (\Gamma, x^- : - \vdash \Delta)}{\Gamma \mid \underline{\tilde{\mu}\{x^-\}.c : +} \vdash \Delta}\ (\Downarrow\vdash)$$

Figure .4.1.i: Logic rules for negations

$$\frac{c : (\Gamma, x^+ : + \vdash \Delta)}{\Gamma \vdash \underline{\mu\neg_-(x^+).xc : -} \mid \Delta}\ (\vdash\neg_-) \qquad \frac{\Gamma \vdash \underline{v_+ : +} \mid \Delta}{\Gamma \mid \underline{\neg_-(v_+) : -} \vdash \Delta}\ (\neg_-\vdash)$$

$$\frac{\Gamma \mid \underline{s_- : -} \vdash \Delta}{\Gamma \vdash \underline{\neg_+(s_-) : +} \mid \Delta}\ (\vdash\neg_+) \qquad \frac{c : (\Gamma \vdash \alpha^- : -, \Delta)}{\Gamma \mid \underline{\tilde{\mu}\neg_+(\alpha^-).\alpha c : +} \vdash \Delta}\ (\neg_+\vdash)$$

Contents