Open call-by-push-value

M2 Internship

Xavier MONTILLET

under the supervision of

Guillaume Munch-Maccagnoni

13th June 2017

Call-By-Push-Value (CBPV) is a model of computation that allows to reconcile the Call-By-Name (CBN) and Call-By-Value (CBV) evaluation strategies on closed terms. Unfortunately, its extension to open terms faces the same problems as CBV: some redexes get stuck. Several solutions have been presented for CBV but not for CBPV. We show that a representative fragment of $\mathbf{LJ}_p^\eta$ (a polarised intuitionnistic abstract-machine-like calculi) is equivalent to CBPV on closed terms, and equip it with a notion of weak reduction on open terms.

Introduction

Call-By-Push-Value (CBPV) is the right way to reconcile the Call-By-Name (CBN) and Call-By-Value (CBV) evaluation strategies, and therefore to describe sums (that behave well in CBV but not in CBN) in the presence of higher-order functions (that behave well in CBN but not in CBV).

$\mathbf{LJ}_p^\eta$ is a polarised intuitionistic abstract-machine-like calculi with explicit evaluation order.

The relationship between CBPV and $\mathbf{LJ}_p^\eta$ in terms of models is given in [CFM16]. In this report, we show that CBPV can be encoded in $\mathbf{LJ}_p^\eta$ from a computational point of view. We also define a notion of weak reduction on $\mathbf{LJ}_p^\eta$.

More precisely, we give a macro-expressible translation from CBPV to $\mathbf{LJ}_p^\eta$, and prove that it is a simulation and preserves normal forms. We define a weak reduction on $\mathbf{LJ}_p^\eta$ (without sums) strictly stronger than head reduction and prove that it is confluent and that it has the same strongly-normalising closed terms as the head reduction.

$$\begin{array}{rll} t, u, r, f, g, a, b ::= & & \text{term} \\ & \mid x, y, z & \text{variable} \\ & \mid \lambda x.t & \text{abstraction} \\ & \mid tu & \text{application} \end{array}$$

(a) Syntax

$$\frac{}{(\lambda x.t)u \rhd_\beta t[u \mathbin{/} x]}\ \textbf{beta}^{\rhd_\beta}$$

(b) Head β-reduction

$$\frac{}{(\lambda x.t)u \Rightarrow_\beta t[u \mathbin{/} x]}\ \textbf{beta}^{\Rightarrow_\beta} \qquad \frac{t \Rightarrow_\beta t'}{\lambda x.t \Rightarrow_\beta \lambda x.t'}\ \textbf{lam}^{\Rightarrow_\beta} \qquad \frac{t \Rightarrow_\beta t'}{tu \Rightarrow_\beta t'u}\ \textbf{app}_\textbf{l}^{\Rightarrow_\beta} \qquad \frac{u \Rightarrow_\beta u'}{tu \Rightarrow_\beta tu'}\ \textbf{app}_\textbf{r}^{\Rightarrow_\beta}$$

(c) Strong β-reduction

$$\frac{}{(\lambda x.t)u \rightarrow_\beta t[u \mathbin{/} x]}\ \textbf{beta}^{\rightarrow_\beta} \qquad \frac{t \rightarrow_\beta t'}{tu \rightarrow_\beta t'u}\ \textbf{app}_\textbf{l}^{\rightarrow_\beta} \qquad \frac{u \rightarrow_\beta u'}{tu \rightarrow_\beta tu'}\ \textbf{app}_\textbf{r}^{\rightarrow_\beta}$$

(d) Weak β-reduction

Figure 1.1: λ-calculus

1. Background

1.1. λ-calculus

The λ-calculus is a well-known minimalist abstract system due to Alonzo Church that can be used to describe computations. Its syntax and reduction rules are recalled in figure 1.1. For example, the identity function is written as $\lambda x.x$. I will be denoted by I.

Remark 1.1. Whenever giving a (context-free) grammar, such as in figure 1.1, we will place several symbols instead of a single non-terminal one on the left of $::=$. Those will be the names given to the meta-variables for terms generated by the corresponding non-terminal symbol. The set of all such terms will be denoted by the first symbol in blackboard bold font. So that t and u are meta-variables ranging over the set $\mathbb{t}$ of λ-terms.

A unique set $\mathbb{x}$ of variables will be used for all languages considered. By $t[u \mathbin{/} x]$, we denote the result of the *capture-avoiding* substitution of x by u in t. If $\sigma : \mathbb{x} \to \mathbb{t}$ is a substitution, we write $t[\ \sigma\ ]$ for the result of the *capture-avoiding* substitution of each variable $x \in \mathbb{x}$ by $\sigma(x)$ in t. The symbol $\square$ is a distinguished variable that will denote a hole in a term. A term with a single occurrence of $\square$ will be called a context or a term with a hole. If t is a context, then $t[\ u\ ]$ is the result of the *non capture-avoiding* substitution of $\square$ by u in t. For example, $(\lambda x.y)[x \mathbin{/} y] = \lambda x.y \neq \lambda x.x = (\lambda x.\square)[x]$.

Several head reductions denoted by $\rhd$ (with some indices) will be considered. In each case, $\Rightarrow$ (with the same indices) will denote the compatible closure of $\rhd$.

If color is available, meta-variables, such as t, will be in blue while syntax, such as λ, will be in green. The rest (including parentheses to remove ambiguity, constants, substitution at the meta level, reductions) will be in black. The notations remain non-ambiguous in the absence colour.

In (most) popular programming languages, the body of a function is only used when the function is applied. To more precisely model the operational semantics of such languages, we define weak β-reduction $\rightharpoonup_\beta$ by removing the **lam** rule (see figure 1.1d). Unfortunately, weak reduction does not have uniqueness of normal forms (and is therefore not confluent), which means that a "computation" (represented by a term t) can have two *different* "results" (represented by two distinct normal forms $\not\leftharpoonup_\beta r_1 \leftharpoonup^*_\beta t \rightharpoonup^*_\beta r_2 \not\rightharpoonup_\beta$), as can be seen in figure 1.2.

$$\begin{tikzcd}
(\lambda x.\lambda y.x)((\lambda z.z)a) \arrow[r, "v"', "\rightharpoonup_\beta" phantom] \arrow[d, "u", "\beta"'] & (\lambda x.\lambda y.x)a \arrow[d, "v", "\beta"'] \\
\lambda y.(\lambda z.z)a \arrow[r, Rightarrow, "\beta"'] \arrow[d, "\beta"', "/" marking] & \lambda y.a \arrow[d, "\beta"', "/" marking] \\
{} & {}
\end{tikzcd}$$

Figure 1.2: Non-uniqueness of normal forms for weak β-reduction

This lack of uniqueness (of normal forms) leads to the study of two restrictions of weak reduction that have uniqueness of normal forms (because they are confluent): Call-By-Name (CBN) and Call-By-Value (CBV). CBN corresponds (loosely) to the evaluation strategy of lazy languages such as Haskell, while CBV corresponds to that of eager languages such as OCaml.

1.1.1. Call-By-Name

Call-By-Name is a restriction of weak β-reduction that ensures that the argument of an application is never reduced, i.e., that the $\mathbf{app_r}$ rule is removed (see figure 1.3). This corresponds to a lazy behaviour: an argument that is not used will never be computed.

$$\frac{t \rhd_\beta t'}{t \overset{n}{\rightharpoonup}_\beta t'}\ \mathbf{beta}^{\overset{n}{\rightharpoonup}_\beta} \qquad \frac{t \overset{n}{\rightharpoonup}_\beta t'}{tu \overset{n}{\rightharpoonup}_\beta t'u}\ \mathbf{app_l}^{\overset{n}{\rightharpoonup}_\beta}$$

Figure 1.3: Call-By-Name weak β-reduction

Another description of Call-By-Name, closer to an operational semantics, is given by an abstract machine: the Krivine machine [Kri07]. In an abstract machine, a term t is represented by a configuration $\langle u \parallel k\rangle$ where u is a term and k is a context so that $t = k[u]$. The several representation of a term t allow to specify which subterm u the machine is currently working on. A configuration can also be represented as a term with a box around u, i.e., $k[\boxed{u}]$ represents $\langle u \parallel k\rangle$. For example, the configuration $\langle f a_1 \parallel (\Box a_2)a_3\rangle$ represents $\boxed{fa_1}a_2a_3$ while $\langle f a_1 a_2 \parallel \Box a_3\rangle$ represents $\boxed{fa_1a_2}a_3$.

The abstract machine starts in $\langle t \parallel \Box\rangle$, meaning that it starts with the focus on the whole term, and then keeps applying its two reduction rules. The first one corresponds to going up the rule $\mathbf{app_l}$: If the focused term is an application $\boxed{fa}$, it moves the focus to the left part $\boxed{f}a$. The second rule corresponds to a β-reduction step, and therefore to the rule **beta**: $\boxed{\lambda x.t}u$ becomes $\boxed{t[u\ /\ x]}$.

While the representation of a term t as a term with a box $k[\ \boxed{u}\ ]$ works well for the first abstract machines we will present, it can

$k ::=$		context
	$\mid \star$	the empty context
	$\mid a \cdot k$	the context $k[\Box a]$

(a) Syntax

$$\begin{array}{lcll}
\langle f a \parallel k\rangle & \overset{n}{\blacktriangleright} & \langle f \parallel a \cdot k\rangle & (\mathbf{app_l}^{\overset{n}{\blacktriangleright}}) \\
\langle \lambda x.t \parallel a \cdot k\rangle & \overset{n}{\blacktriangleright} & \langle t[a\ /\ x] \parallel k\rangle & (\mathbf{beta}^{\overset{n}{\blacktriangleright}})
\end{array}$$

(b) Reductions

Figure 1.4: Krivine abstract machine (CBN)

not be (easily) extended to $\mathbf{LJ}^\eta_p$. We therefore henceforth use the representation of a term t as a configuration $\langle u \parallel k\rangle$. Under this representation, the two rules become $\langle fa \parallel k\rangle \overset{n}{\blacktriangleright} \langle f \parallel k[\Box a]\rangle$ and $\langle \lambda x.t \parallel k[\Box a]\rangle \overset{n}{\blacktriangleright} \langle t[a / x] \parallel k\rangle$. In both cases, the machine has to look deep in the context to move / find the argument a, as shown by the following example: $\langle \lambda x.t \parallel ((\Box a_1)a_2)a_3\rangle \overset{n}{\blacktriangleright} \langle t[a_1 / x] \parallel (\Box a_2)a_3\rangle$. This is because the machine looks at the first constructors inside and outside of the box, and while the first constructor of the term is easily accessible, the hole $\Box$ and the constructor just above it may be deep in the context. For this reason, contexts are represented *inside-out* using the notation $a \cdot k$ for $k[\Box a]$. In the case of the Krivine machine, the only contexts that are needed are of the form $((\Box a_1)\ldots)a_l$ and will be written as $a_1 \cdot (\ldots \cdot (a_l \cdot \star))$. Note that $\star$ represents the outside of the context. Munch-Maccagnoni and Scherer explain how this is related to continuation-passing style and defunctionalisation in [MS15].

1.1.2. Call-By-Value

Call-By-Value is a restriction of weak β-reduction that ensures that only "simple" terms named *values* can be substituted for variables, i.e., that restricts the rule **beta** to applications of the form fv where v is a value (see figure 1.5b). This corresponds to an eager behaviour: in an application fa, the argument a is computed before being given to the function f, even if the function does not use it.

$$\frac{}{(\lambda x.t)v \overset{v}{\rhd}_\beta t[v / x]}\ \mathbf{beta}^{\overset{v}{\rhd}_\beta}$$

(a) Head β_v-reduction

$$\frac{t \overset{v}{\rhd}_\beta t'}{t \overset{v}{\to}_\beta t'}\ \mathbf{beta}^{\overset{v}{\to}_\beta} \qquad \frac{t \overset{v}{\to}_\beta t'}{tu \overset{v}{\to}_\beta t'u}\ \mathbf{app_l}^{\overset{v}{\to}_\beta}$$

$$v ::= x \mid \lambda x.t \quad \text{value} \qquad \frac{u \overset{v}{\to}_\beta u'}{tu \overset{v}{\to}_\beta tu'}\ \mathbf{app_r}^{\overset{v}{\to}_\beta}$$

(b) CBV weak β-reduction

Figure 1.5: CBV β-reductions

Two deterministic variants of Call-By-Value are left-to-right where the function is evaluated before its argument (i.e. $\mathbf{app_r}$ is restricted to the case where t is a value), or right-to-left where the argument is evaluated before the function (i.e. $\mathbf{app_l}$ is restricted to the case where u is a value).

Both left-to-right and right-to-left Call-By-Value can be described by abstract machines. In both cases, the context $f \odot k$ representing $k[f\Box]$ has to be introduced. The rules in figure 1.6 describe the behaviour of the machine. Its behaviour when encountering an application is described in figure 1.7.

$$\begin{array}{lll} k ::= & & \text{context} \\ & \mid \star & \text{the empty context} \\ & \mid a \cdot k & \text{the context } k[\Box a] \\ & \mid f \odot k & \text{the context } k[f\Box] \end{array}$$

(a) Syntax

$$\begin{array}{lcll} \langle fa \parallel k\rangle & \overset{v}{\blacktriangleright} & \langle a \parallel f \odot k\rangle & (\mathbf{app_r}^{\overset{v}{\blacktriangleright}}) \\ \langle v \parallel f \odot k\rangle & \overset{v}{\blacktriangleright} & \langle f \parallel v \cdot k\rangle & (\mathbf{app_l}^{\overset{v}{\blacktriangleright}}) \\ \langle \lambda x.t \parallel a \cdot k\rangle & \overset{v}{\blacktriangleright} & \langle t[a / x] \parallel k\rangle & (\mathbf{beta}^{\overset{v}{\blacktriangleright}}) \end{array}$$

(b) Reductions

Figure 1.6: Abstract machine (CBV, left-to-right)

An additional difficulty arises when looking at CBV on open terms: some normal forms,

$$\begin{array}{rlll}
\langle f\,a \parallel k\rangle & \overset{v}{\blacktriangleright} & \langle a \parallel f \odot k\rangle & \text{Push the function in the context}\\
 & \overset{v\,*}{\blacktriangleright} & \langle v \parallel f \odot k\rangle & \text{Reduce the argument until it becomes a value}\\
 & \overset{v}{\blacktriangleright} & \langle f \parallel v \cdot k\rangle & \text{Pop the function from the context and push the argument}\\
 & \overset{v\,*}{\blacktriangleright} & \langle \lambda x.t \parallel v \cdot k\rangle & \text{Reduce the function until it becomes an abstraction}\\
 & \overset{v}{\blacktriangleright} & \langle t[v \,/\, x] \parallel k\rangle & \beta\text{-reduction}
\end{array}$$

Figure 1.7: Abstract machine (CBV, left-to-right) - Example

called inert[1] $i ::= x t_1 \ldots t_n$, are not values. This makes some redex (those of the form $f\,i$) stuck, which in turn cause unexpected behaviours: If Ω is a closed diverging term, $(\lambda x.\Omega) i$ is a normal form (and therefore converges) but for any substitution σ sending the open variables of $(\lambda x.\Omega) i$ on closed terms, $(\lambda x.\Omega) i[\,\sigma\,]$ diverges. These problems spread to strong Call-By-Value because when reducing under an abstraction, some variables are (locally) free. Several (equivalent) solutions are known [AG16].

1.2. Effects

In the presence of effects, functions are well-behaved in Call-By-Name but not in Call-By-Value: the η-conversion for functions, $f \approx_\eta \lambda x.f x$ whenever f is of functional type, is not satisfied. The problem is that it allows to replace a term f that may not be a value by a value $\lambda x.f x$, which, in CBV, affects the order of evaluation. For example, given two functions f_1 and f_2, in the application $f_1 f_2$, we can force f_1 (resp. f_2) to be evaluated to a value by using η-conversion for f_2 (resp. f_1): $f_1 f_2 \approx_\eta f_1(\lambda y.f_2 y) \overset{v\,*}{\rightarrow}_\beta v_1(\lambda y.f_2 y)$ (resp. $f_1 f_2 \approx_\eta (\lambda y.f_1 y) f_2 \overset{v\,*}{\rightarrow}_\beta (\lambda y.f_2 y) v_2$). In the case where those two functions print different things before becoming values, this gives two different behaviours of two η-convertible terms. An example where $f_i := \mathsf{print}\ \text{“}i\text{”}\,; \lambda x_i.x_i$ prints “i” and then become the identity is given in figure 1.8. Note that the two terms may not even print the same things in different order: If f_1 drops its argument, the “2” will never get printed in the left branch. In Call-By-Name, the argument a of an application $f\,a$ is never reduced and the effect triggered by the evaluation of f will therefore always happen before everything else, including effects triggered by evaluating a.

In the presence of effects, the η-conversion for booleans, if x is free of type bool in t then $t[\,u \,/\, x\,] \approx_\eta \mathsf{if}\ u\ \mathsf{then}\ t[\mathsf{true} \,/\, x]\ \mathsf{else}\ t[\mathsf{false} \,/\, x]$, holds neither in CBN nor in CBV. This is expected because it says that a boolean can be evaluated at any time, which can change the observable behaviour if the boolean triggers effects. In CBV, this can be fixed by restricting the rule to the cases where u is normal (because then, evaluating it does nothing), which includes the cases where u is a variable y: $t[\,y \,/\, x\,] \approx_\eta \mathsf{if}\ y\ \mathsf{then}\ t[\mathsf{true} \,/\, x]\ \mathsf{else}\ t[\mathsf{false} \,/\, x]$. Indeed, y

[1]Following the terminology of [AG16]

$$\begin{tikzcd}[column sep=tiny]
 & (\mathsf{print}\ \text{“1”};\, f_1)(\mathsf{print}\ \text{“2”};\, f_2) \arrow[dl, phantom, sloped, "\approx_\eta"] \arrow[dr, phantom, sloped, "\approx_\eta"] & \\
(\mathsf{print}\ \text{“1”};\, f_1)(\lambda x.(\mathsf{print}\ \text{“2”};\, f_2)x) \arrow[d, phantom, "\triangledown", "\text{“1”}"'] & & (\lambda x.(\mathsf{print}\ \text{“1”};\, f_1)x)(\mathsf{print}\ \text{“2”};\, f_2) \arrow[d, phantom, "\triangledown", "\text{“2”}"'] \\
f_1(\lambda x.(\mathsf{print}\ \text{“2”};\, f_2)x) & & (\lambda x.(\mathsf{print}\ \text{“1”};\, f_1)x)f_2
\end{tikzcd}$$

Figure 1.8: η-conversion does not hold for functions in CBV

$$\begin{tikzcd}[column sep=small]
k\big[(\mathsf{print}\ \text{“1”};\, x)\big[y \,/\, x\big]\big] \arrow[r, phantom, "\approx_\eta"] \arrow[d, phantom, "\overset{\mathrm{def}}{=}"] & k\big[\mathsf{if}\ y\ \mathsf{then}\ ((\mathsf{print}\ \text{“1”};\, x)\big[\mathsf{true} \,/\, x\big])\ \mathsf{else}\ ((\mathsf{print}\ \text{“1”};\, x)\big[\mathsf{false} \,/\, x\big])\big] \arrow[d, phantom, "\overset{\mathrm{def}}{=}"] \\
k\big[(\mathsf{print}\ \text{“1”};\, y)\big] \arrow[d, phantom, "\triangledown"] & k\big[\mathsf{if}\ y\ \mathsf{then}\ (\mathsf{print}\ \text{“1”};\, \mathsf{true})\ \mathsf{else}\ (\mathsf{print}\ \text{“1”};\, \mathsf{false})\big] \arrow[d, phantom, "\triangledown"] \\
\mathsf{print}\ \text{“1”};\, \mathsf{print}\ \text{“2”};\, b \arrow[d, phantom, "\triangledown", "\text{“1”}"'] & \mathsf{if}\ (\mathsf{print}\ \text{“2”};\, b)\ \mathsf{then}\ (\mathsf{print}\ \text{“1”};\, \mathsf{true})\ \mathsf{else}\ (\mathsf{print}\ \text{“1”};\, \mathsf{false}) \arrow[d, phantom, "\triangledown", "\text{“2”}"'] \\
\mathsf{print}\ \text{“2”};\, b & \mathsf{if}\ b\ \mathsf{then}\ (\mathsf{print}\ \text{“1”};\, \mathsf{true})\ \mathsf{else}\ (\mathsf{print}\ \text{“1”};\, \mathsf{false})
\end{tikzcd}$$

$$\text{where } k := \mathsf{let}\ x\ \mathsf{be}\ (\mathsf{print}\ \text{“2”};\, b).\square$$

Figure 1.9: η-conversion does not hold for booleans in CBN

will always be replaced by a value which can not trigger an effect. This still does not work in CBN because the variable y could be replaced by a boolean triggering an effect, for example by placing the term in the context $k := \mathsf{let}\ x\ \mathsf{be}\ (\mathsf{print}\ \text{“2”};\, b).\square$ as in figure 1.9.

The drive to reconcile the behaviour with respect to η-conversion of positive types such as booleans and that of negative types such as functions led to the study of *polarised* systems [DJS97] mixing aspects of CBN and CBV, which in the intuitionistic case [LM07] was shown [CFM16] to be closely related to Call-By-Push-Value as introduced next.

1.3. Call-by-push-value

Call-by-push-value, introduced by Levy [Lev04], subsumes both Call-By-Value and Call-By-Name. Terms are split into values (denoted by W) and computations (denoted by M), according to the slogan *“A value is, a computation does”* (Levy).

Values W include variables x and constructors of some pattern-matchable type applied to values (such as the unique element of unit $()$, or a pair (W_1, W_2)). Computation include all the remaining usual constructions, including let expressions, pattern matches, and constructors and destructors of non-pattern-matchable types. Note that application is written with the argument first, i.e. $W`M$ should be understood as MW.

In addition to those, there is a constructor for values $\mathsf{thunk}\ M$ that freezes a computation and a constructor for computations $\mathsf{force}\ W$ that forces a frozen computation to execute. Freezing the computation can be though of as the suspending it by placing it under a λ-abstraction $\lambda\_.M$ while forcing a frozen computation can be though of as getting it out of the lambda abstraction by applying it to something $W()$. Under this view, the reduction $\mathsf{force}\ \mathsf{thunk}\ M \rhd M$ then becomes

$(\lambda\_.M)() \rhd M$. If M is of type $\underline{B}$, then $\mathsf{thunk}\, M$ is of type $\mathsf{U}\, \underline{B}$.

There is also a constructor that turns a value W into a computation $\mathsf{return}\, W$ that simply returns its result: W. The result of a computation M can be extracted and used (as x in N) using $M\, \mathsf{to}\, x.N$. Note that the let construction $\mathsf{let}\, W\, \mathsf{be}\, x.N$ could be seen as syntactic sugar for $\mathsf{return}\, W\, \mathsf{to}\, x.N$, and then reduction $\langle \mathsf{let}\, W\, \mathsf{be}\, x.N \parallel K\rangle \rhd \big\langle N\big[W \,/\, x\big] \parallel K\big\rangle$ would become $\langle \mathsf{return}\, W\, \mathsf{to}\, x.N \parallel K\rangle \rhd \langle \mathsf{return}\, W \parallel \square\, \mathsf{to}\, x.N :: K\rangle \rhd \big\langle N\big[W \,/\, x\big] \parallel K\big\rangle$. If W if of type A, then $\mathsf{return}\, W$ is of type $\mathsf{F}\, A$.

Value types are written A and computation types are written $\underline{B}$. Because the framework does not include dependent types, the usual dependent sums $\sum\limits_{t\in A} B_t$ are specialised into products $A_1 \times A_2 := \sum\limits_{i\in A_1} A_2$ and a restricted sigma where the sum is not indexed over a type but over a set of tags $\{1,2\}$: $\sum\limits_{i\in\{1,2\}} B_i$. Similarly, dependent functions $\prod\limits_{t\in A} \underline{B_t}$ are specialised into non-dependent functions $A \to \underline{B} := \prod\limits_{t\in A} \underline{B}$ and restricted pi: $\prod\limits_{i\in\{1,2\}} \underline{B_i}$.

The context $\square\, \mathsf{to}\, x.N :: K$ can be though of as $K[\square\, \mathsf{to}\, x.N]$, and $W :: K$ can be though of as $K[\square W]$ (and similarly with W replaced by $\hat{\imath}$).

1.4. $\mathbf{LJ}_p^\eta$

We now introduce the $\mathbf{LJ}_p^\eta$ calculus [MS15, CFM16] which is a term assignment for intuitionistic logic with explicit evaluation order which brings together CBPV and Curien-Herbelin's calculus.

In Curien-Herbelin's calculus [CH00], terms are presented by commands $\langle t \parallel e\rangle$ where t is a term and e is an evaluation context. A crucial difference with the previous abstract machines is that subterms are also represented by commands.

The usual rules of pattern matches are given, with the notation $\tilde{\mu}\ldots.c$ in place of $\mathsf{let}\,\square\,\mathsf{be}\ldots.c$ or $\mathsf{pm}\,\square\,\mathsf{as}\ldots.c$. For example, $\tilde{\mu}x.c$ represents $\mathsf{let}\,\square\,\mathsf{be}\, x.c$ and the reduction $\langle V \parallel \tilde{\mu}x.c\rangle \rhd_{\tilde{\mu}} c\big[\ V \,/\, x\ \big]$ can therefore be read as $\mathsf{let}\,\boxed{V}\,\mathsf{be}\, x.c \rhd c\big[\ V \,/\, x\ \big]$. Similarly, $\tilde{\mu}().c$ represents $\mathsf{pm}\,\square\,\mathsf{as}\,().c$ and the reduction $\big\langle () \parallel \tilde{\mu}().c\big\rangle \rhd_{\mathbf{1}} c$ can therefore be read as $\mathsf{pm}\,\boxed{()}\,\mathsf{as}\,().c \rhd c$.

The symbol $\star$ representing the outside of the context is now called a covariable[2] and an associated binder is introduced. The rule $\langle \mu\star.c \parallel S\rangle \rhd_{\mu} c\big[\ S \,/\, \star\big]$ is symmetrical to the one of let / $\tilde{\mu}$. It can be seen as moving focus: $S\big[\ \boxed{c}\ \big] \rhd S\big[\ c\ \big]$, although where the focus is moved depends on where the focus was in the subcommand c. Another way to see $\mu\star$ is to say that it allows to define terms by their behaviour in a given context. For example, in the CBN abstract machine, $\langle f a \parallel S\rangle \rhd \langle f \parallel a \cdot S\rangle$ can be seen as stating that $f a : S \mapsto \langle f \parallel a \cdot S\rangle$. This can then be encoded in $\bar{\lambda}\mu\tilde{\mu}$ by defining $f a := \mu\star.\langle f \parallel a \cdot \star\rangle$ and the reduction becomes an instance of $\rhd_{\mu}$. Using μ and $\tilde{\mu}$, we can limit the number of primitive constructs per type to two: one constructor and one destructor. For example, for the function type $\multimap$, the constructor is $V \cdot S$ and the destructor is $\mu(x \cdot \star).c$. The reduction $\big\langle \mu(x \cdot \star).c \parallel V \cdot S\big\rangle \rhd_{\multimap} c\big[V \,/\, x, S \,/\, \star\big]$ can be seen as $S\big[\boxed{\lambda x.c}V\big] \rhd S\big[c\big[V \,/\, x\big]\big]$.

[2]In $\bar{\lambda}\mu\tilde{\mu}$, there are more than one covariable. Only allowing to use one covariable amounts to restricting to an intuitionistic setting as suggested by Herbelin [Her05].

$$
\begin{array}{lcll}
\text{value} & W & ::= & x \mid \mathsf{thunk}\, M \mid () \mid (W, W) \mid (\hat{\imath}, W) \\
\text{computation} & M & ::= & \mathsf{let}\, W\, \mathsf{be}\, x.N \mid \mathsf{return}\, W \mid M\, \mathsf{to}\, x.N \mid \mathsf{force}\, W \\
 & & & \mid \mathsf{pm}\, W\, \mathsf{as}\, ().N \mid \mathsf{pm}\, W\, \mathsf{as}\, (x, y).N \mid \mathsf{pm}\, W\, \mathsf{as}\, \{(1, x_1).N_1, (2, x_2).N_2\} \\
 & & & \mid \lambda x.M \mid W`M \mid \lambda\{1.M_1, 2.M_2\} \mid \hat{\imath}`M \\
\text{stack} & K & ::= & \mathsf{nil} \mid \square\, \mathsf{to}\, x.M :: K \mid W :: K \mid \hat{\imath} :: K \\
\text{tag} & \hat{\imath} & ::= & 1 \mid 2 \\
\text{configuration} & E & ::= & \langle M \parallel K \rangle
\end{array}
$$

(a) Syntax

$$
\begin{array}{lcll}
\text{value} & A & ::= & \mathsf{U}\, \underline{B} \mid \mathbf{1} \mid A_1 \times A_2 \mid \sum_{i \in \{1,2\}} A_i \\
\text{computation} & \underline{B} & ::= & \mathsf{F}\, A \mid A \to \underline{B} \mid \prod_{i \in \{1,2\}} \underline{B_i}
\end{array}
$$

(b) Types

$$
\begin{array}{llrclr}
\langle \mathsf{let}\, W\, \mathsf{be}\, x.N & \parallel & K \rangle & \rhd & \langle N[W \,/\, x] & \parallel \quad K \rangle \\
\langle M\, \mathsf{to}\, x.N & \parallel & K \rangle & \rhd & \langle M & \parallel \square\, \mathsf{to}\, x.N :: K \rangle \\
\langle \mathsf{return}\, W & \parallel & \square\, \mathsf{to}\, x.N :: K \rangle & \rhd & \langle N[W \,/\, x] & \parallel \quad K \rangle \\
\langle \mathsf{force}\, \mathsf{thunk}\, M & \parallel & K \rangle & \rhd & \langle M & \parallel \quad K \rangle \\
\langle \mathsf{pm}\, ()\, \mathsf{as}\, ().N & \parallel & K \rangle & \rhd & \langle N & \parallel \quad K \rangle \\
\langle \mathsf{pm}\, (W_1, W_2)\, \mathsf{as}\, (x_1, x_2).N & \parallel & K \rangle & \rhd & \langle M[W_1 \,/\, x_1, W_2 \,/\, x_2] & \parallel \quad K \rangle \\
\langle \mathsf{pm}\, (\hat{\imath}, W)\, \mathsf{as}\, \{(1, x).N_1, (2, x).N_2\} & \parallel & K \rangle & \rhd & \langle N_{\hat{\imath}}[W \,/\, x] & \parallel \quad K \rangle \\
\langle W`M & \parallel & K \rangle & \rhd & \langle M & \parallel \quad W :: K \rangle \\
\langle \lambda x.N & \parallel & W :: K \rangle & \rhd & \langle N[W \,/\, x] & \parallel \quad K \rangle \\
\langle \hat{\imath}`M & \parallel & K \rangle & \rhd & \langle M & \parallel \quad \hat{\imath} :: K \rangle \\
\langle \lambda\{1.N_1, 2.N_2\} & \parallel & \hat{\imath} :: K \rangle & \rhd & \langle N_{\hat{\imath}} & \parallel \quad K \rangle
\end{array}
$$

(c) Reductions

Figure 1.10: Call-by-push-value

A disadvantage of Curien-Herbelin's calculus to represent computations is the presence of the Lafont critical pair shown in figure 1.11 which prevents uniqueness of normal forms (and therefore confluence). Curien and Herbelin show that the two ways to remove the critical pair (either favouring the left or favouring the right) correspond to a CBV and CBN restrictions of Curien-Herbelin's calculus. Inspired by Danos, Joinet and Schellinx [DJS97], Munch-Maccagnoni introduces a variant of Curien-Herbelin's calculus where the side to which the critical pair reduces is according to a polarity determined by the type (MM 2009). Polarities $+$ and $-$ are added to commands, only allowing $\tilde{\mu}$ to reduce in positive commands, and μ only in negative commands. This is done by adding two new syntactic categories: the values are terms that can be substituted for variables, and the stacks are contexts that can be substituted for covariables. Types are also assigned a polarity. Those whose constructor builds values and destructor binds variables via a $\tilde{\mu}$ are positive and those whose constructor build stacks and destructor binds covariables via μ are negative. The result is (a fragment of) the system $\mathbf{LJ}_p^\eta$ in the Curry-style variant in [Mun] which is described in figure 1.12.

$$c_1\left[\tilde{\mu}x.c_2 \;/\; \star\right] \lhd_\mu \langle \mu\star.c_1 \,\|\, \tilde{\mu}x.c_2\rangle \rhd_{\tilde{\mu}}$$
$$c_2\left[\mu\star.c_1 \;/\; x\right]$$

Figure 1.11: Lafont critical pair in $\overline{\lambda}\mu\tilde{\mu}$

Both Call-By-Name and Call-By-Value can be encoded in $\mathbf{LJ}_p^\eta$. Note that the presence of μ and $\tilde{\mu}$ in $\mathbf{LJ}_p^\eta$ allows to express all constructs related to functions with just two primitive constructs specific to functions: $\mu(x \cdot \star).c$ and $a \cdot S$.

2. Simulation of CBPV by $\mathbf{LJ}_p^\eta$

In this section, we describe a way to express CBPV in $\mathbf{LJ}_p^\eta$ via macros, in the sense of macro-expressibility in Felleisen [Fel91]. We also give a second slightly modified translation that behaves more nicely with respect to simulation.

There exists a translation in the other direction induced by the interpretation of $\mathbf{LJ}_p^\eta$ into CBPV models [CFM16] not studied here.

Types are translated by two mutually recursive functions: $\boxed{\cdot}_+^{\mathbb{A}} : \mathbb{A} \to \mathbb{P}$ sending value types of CBPV to positive types of $\mathbf{LJ}_p^\eta$ and $\boxed{\cdot}_-^{\mathbb{B}} : \underline{\mathbb{B}} \to \mathbb{N}$ sending computation types of CBPV to negative types of $\mathbf{LJ}_p^\eta$. Values and computations are also translated by two mutually recursive functions: $\boxed{\cdot}_+^{\mathbb{W}} : \mathbb{W} \to \mathbb{V}_+$ sending values of CBPV to positive values of $\mathbf{LJ}_p^\eta$ and $\boxed{\cdot}_-^{\mathbb{M}} : \mathbb{M} \to \mathbb{V}_-$ sending computations of CBPV to negative values of $\mathbf{LJ}_p^\eta$. Stacks are translated by $\boxed{\cdot}_-^{\mathbb{K}} : \mathbb{K} \to \mathbb{S}$ sending stacks of CBPV to negative stacks of $\mathbf{LJ}_p^\eta$. The translation of configurations, $\boxed{\cdot}^{\mathbb{E}} : \mathbb{E} \to \mathbb{c}$, sending configurations of CBPV to negative commands of $\mathbf{LJ}_p^\eta$, is defined by $\boxed{\langle M \,\|\, K\rangle}^{\mathbb{E}} := \langle \boxed{M}_-^{\mathbb{M}} \,\|\, \boxed{K}_-^{\mathbb{K}} \rangle^-$. Note that while the translations preserve typing, they are not restricted to typed values / computations / stacks / configurations. The translations are described in figure 2.1. The upper indices are left implicit. The translations are described in figure 2.1.

While type preservation and simulation (of E by $\boxed{E}$) work for the translation $\boxed{\cdot}$, there is no hope to obtain any (strong) simulation property in the other direction (i.e. of $\boxed{E}$ by E) because

$$
\begin{array}{llcl}
\text{value} & V & ::= & x \mid () \mid (V_1, V_2) \mid \iota_1(V) \mid \iota_2(V) \mid \{V\} \\
 & & & \mid \mu\star^{-}.c \mid \mu(x \cdot \star).x \mid \mu{<}\star.c_1\,;\star.c_2{>} \mid \mu\{\star\}.c \\
\text{term} & t & ::= & \mu\star^{+}.c \mid V \\
\text{stack} & S & ::= & \star \mid V \cdot S \mid \pi_1 \cdot S \mid \pi_2 \cdot S \mid \{S\} \\
 & & & \mid \tilde{\mu}x^{+}.c \mid \tilde{\mu}().c \mid \tilde{\mu}(x_1, x_2).c \mid \tilde{\mu}[x_1.c_2 \mid x_2.c_2] \mid \tilde{\mu}\{x\}.c \\
\text{context} & e & ::= & \tilde{\mu}x^{-}.c \mid S \\
\text{command} & c & ::= & \langle V \parallel e\rangle^{-} \mid \langle t \parallel S\rangle^{+}
\end{array}
$$

(a) Syntax

$$
\text{types } A, B \begin{cases} \text{positive} & A_{+}, B_{+} \quad ::= \quad \mid A \otimes B \mid A \oplus B \mid \Downarrow A \\ \text{negative} & A_{-}, B_{-} \quad ::= \quad A \rightarrow B \mid A \& B \mid \Uparrow A \end{cases}
$$

(b) Types

$$
\begin{array}{lllll}
\langle V & \parallel \tilde{\mu}x^{\varepsilon}.c & \rangle^{\varepsilon} & \rhd_{\tilde{\mu}} & c[V \,/\, x] \\
\langle \mu\star^{\varepsilon}.c & \parallel S & \rangle^{\varepsilon} & \rhd_{\mu} & c[S \,/\, \star] \\
\langle () & \parallel \tilde{\mu}().c & \rangle^{+} & \rhd_{\mathbf{1}} & c \\
\langle (V_1, V_2) & \parallel \tilde{\mu}(x_1, x_2).c & \rangle^{+} & \rhd_{\otimes} & c[V_1 \,/\, x_1, V_2 \,/\, x_2] \\
\langle \iota_{\hat{\imath}}(V) & \parallel \tilde{\mu}[x.c_1 \mid x.c_2] & \rangle^{+} & \rhd_{\oplus} & c_{\hat{\imath}}[V \,/\, x] \\
\langle \mu(x \cdot \star).c & \parallel V \cdot S & \rangle^{-} & \rhd_{\rightarrow} & c[V \,/\, x, S \,/\, \star] \\
\langle \mu{<}\star.c_1\,;\star.c_2{>} & \parallel \pi_{\hat{\imath}} \cdot S & \rangle^{-} & \rhd_{\&} & c_{\hat{\imath}}[S \,/\, \star] \\
\langle \mu\{\star\}.c & \parallel \{S\} & \rangle^{+} & \rhd_{\mu\{\}} & c[S \,/\, \star] \\
\langle \{V\} & \parallel \tilde{\mu}\{x\}.c & \rangle^{-} & \rhd_{\tilde{\mu}\{\}} & c[V \,/\, x]
\end{array}
$$

(c) Reductions

Figure 1.12: $\mathbf{LJ}_p^{\eta}$

t	$\boxed{t}^{n}_{-}$
x	x
$\lambda x.t$	$\mu(x \cdot \star).\langle \boxed{t}^{n}_{-} \parallel \star\rangle^{-}$
$f\,a$	$\mu\star^{-}.\langle \boxed{f}^{n}_{-} \parallel \boxed{a \cdot \star}^{n}_{-}\rangle^{-}$

(a) Translation of terms

k	$\boxed{k}^{n}_{-}$
$\star$	$\star$
$a \cdot k$	$\boxed{a}^{n}_{-} \cdot \boxed{k}^{n}_{-}$

(b) Translation of contexts

$$\boxed{\langle t \parallel k\rangle}^{n}_{-} := \langle \boxed{t}^{n}_{-} \parallel \boxed{k}^{n}_{-}\rangle^{-}$$

(c) Translation of configurations

Figure 1.13: Translation from CBN to $\mathbf{LJ}_p^{\eta}$

	t	$\boxed{t}^v_+$
	x	x
	$\lambda x.t$	$\{\mu(x\cdot\star).\langle \boxed{t}^v_+ \Vert \star\rangle^-\}$
left-to-right	$f\,a$	$\mu\star^+.\langle \boxed{f}^v_+ \Vert \boxed{a\cdot\star}^v_+\rangle^+ \quad = \quad \mu\star^+.\left\langle \boxed{f}^v_+ \,\middle\Vert\, \tilde\mu\{x_f\}.\langle \boxed{a}^v_+ \Vert \tilde\mu {x_a}^+.\langle x_f \Vert x_a\cdot\star\rangle^-\rangle^+ \right\rangle^+$
right-to-left	$f\,a$	$\mu\star^+.\langle \boxed{a}^v_+ \Vert \boxed{f\odot\star}^v_+\rangle^+ \quad = \quad \mu\star^+.\left\langle \boxed{a}^v_+ \,\middle\Vert\, \tilde\mu {x_a}^+.\langle \boxed{f}^v_+ \Vert \tilde\mu\{x_f\}.\langle x_f \Vert x_a\cdot\star\rangle^-\rangle^+ \right\rangle^+$

(a) Translation of terms

k	$\boxed{k}^v_+$ left-to-right	$\boxed{k}^v_+$ right-to-left
$\star$	$\star$	$\star$
$a\cdot k$	$\tilde\mu\{x_f\}.\langle \boxed{a}^v_+ \Vert \boxed{x_f\odot k}^v_+\rangle^-$	$\tilde\mu\{x_f\}.\langle x_f \Vert \boxed{a}^v_+\cdot\boxed{k}^v_+\rangle^-$
$f\odot k$	$\tilde\mu {x_a}^+.\langle \boxed{f}^v_+ \Vert x_a\cdot\boxed{k}^v_+\rangle^+$	$\tilde\mu {x_a}^+.\langle \boxed{f}^v_+ \Vert \boxed{x_a\cdot k}^v_+\rangle^+$

(b) Translation of contexts

$$\boxed{\langle t \Vert k\rangle}^v_+ := \langle \boxed{t}^v_+ \Vert \boxed{k}^v_+\rangle^+$$

(c) Translation of configurations

Figure 1.14: Translation from CBV to $\mathbf{LJ}^\eta_p$

a normal form (for example $\langle \mathsf{pm}\,W\,\mathsf{as}\,().N \Vert \mathsf{nil}\rangle$ where W can be anything except $()$, including a variable or a pair) is not always sent to a normal form:

$$\boxed{\langle \mathsf{pm}\,W\,\mathsf{as}\,().N \Vert \mathsf{nil}\rangle} = \left\langle \mu\star^-.\langle \boxed{W}_+ \Vert \tilde\mu().\langle \boxed{N}_- \Vert \star\rangle^-\rangle^+ \,\middle\Vert\, \star\right\rangle^- \rhd_\mu \langle \boxed{W}_+ \Vert \tilde\mu().\langle \boxed{N}_- \Vert \star\rangle^-\rangle^+$$

Translating E by the $\rhd_\mu$-normal form of (or the result of applying one $\rhd_\mu$ reduction to) $\boxed{E}$ does not work either because some reductions steps in CBPV are sent to $\rhd_\mu$ reduction steps in $\mathbf{LJ}^\eta_p$. For example, $\langle M\,\mathsf{to}\,x.N \Vert K\rangle \rhd \langle M \Vert \square\,\mathsf{to}\,x.N :: K\rangle$ becomes the following:

$$\left\langle \mu\star^-.\langle \boxed{M}_- \Vert \{\tilde\mu x^+.\langle \boxed{N}_- \Vert \star\rangle^-\}\rangle^- \,\middle\Vert\, \boxed{K}_-\right\rangle^- \rhd_\mu \langle \boxed{M}_- \Vert \{\tilde\mu x^+.\langle \boxed{N}_- \Vert \boxed{K}_-\rangle^-\}\rangle^-$$

This problem is fixed by introducing a second translation $\bbox[border:1px dashed]{\cdot} : \mathbb{E} \to \mathbb{c}$ defined from $\boxed{\cdot}$ by sometimes applying a $\rhd_\mu$ reduction. Intuitively, it will apply a $\rhd_\mu$ reduction if and only if it does not correspond to a reduction in CBPV.

More formally, if $\boxed{E} \not\rhd_\mu$, then $\bbox[border:1px dashed]{E}$ is defined to be $\boxed{E}$. Otherwise, there is some c' so that $\boxed{E} \rhd_\mu c'$. If this reduction corresponds to a reduction in CBPV, i.e. if $E \rhd E'$ for some configuration E' and $\boxed{E'} = c'$, then $\bbox[border:1px dashed]{E}$ is defined to be $\boxed{E}$. Otherwise, $\bbox[border:1px dashed]{E}$ is defined to be c'. This is summarised in figure 2.1d. All the cases where $\bbox[border:1px dashed]{E} \neq \boxed{E}$ are enumerated in the proof of proposition 2.1 on page 11.

2.1. Simulation

Both translations are simulations of CBPV by $\mathbf{LJ}^\eta_p$, and there is a “weak” simulation in the other direction for $\bbox[border:1px dashed]{\cdot}$.

Proposition 2.1 (Simulation)**.** *For any configurations E_1 and E_2, if $E_1 \rhd E_2$ then $\bbox[border:1px dashed]{E_1} \rhd^+ \bbox[border:1px dashed]{E_2}$.*

W	$: A$	$\boxed{W}_+$	$: \boxed{A}_+$
x	$: A$	x	$: \boxed{A}_+$
$\mathsf{thunk}\ M$	$: \mathsf{U}\,\underline{B}$	$\{\ \boxed{M}_-\ \}$	$: \Downarrow\boxed{\underline{B}}_-$
$()$	$: \mathbf{1}$	$()$	$: \mathbf{1}$
(W_1, W_2)	$: A_1 \times A_2$	$\left(\boxed{W_1}_+ , \boxed{W_2}_+\right)$	$: \boxed{A_1}_+ \otimes \boxed{A_2}_+$
$(\hat{\imath}, W)$	$: \sum_{i\in\{1,2\}} A_i$	$\iota_{\hat{\imath}}(\boxed{W}_+)$	$: \boxed{A_1}_+ \oplus \boxed{A_2}_+$

(a) Translation of values

K	$: \underline{B}$	$\mapsto \underline{C}$	$\boxed{K}_-$	$: \boxed{\underline{B}}_-$	$\mapsto \boxed{\underline{C}}_-$
nil	$: \underline{B}$	$\mapsto \underline{B}$	$\star$	$: \boxed{\underline{B}}_-$	$\mapsto \boxed{\underline{B}}_-$
$\square\ \mathsf{to}\ x.N :: K$	$: \mathsf{F}\,A$	$\mapsto \underline{C}$	$\{\tilde{\mu}x^+.\langle \boxed{N}_- \parallel \boxed{K}_- \rangle^-\}$	$: \Uparrow \boxed{A}_+$	$\mapsto \boxed{\underline{C}}_-$
$W :: K$	$: (A \to \underline{B})$	$\mapsto \underline{C}$	$\boxed{W}_+ \cdot \boxed{K}_-$	$: (\boxed{A}_+ \multimap \boxed{\underline{B}}_-)$	$\mapsto \boxed{\underline{C}}_-$
$\hat{\imath} :: K$	$: \prod_{i\in\{1,2\}} \underline{B_i}$	$\mapsto \underline{C}$	$\pi_{\hat{\imath}} \cdot \boxed{K}_-$	$: \boxed{\underline{B_1}}_- \,\&\, \boxed{\underline{B_2}}_-$	$\mapsto \boxed{\underline{C}}_-$

(b) Translation of stacks

M	$: \underline{B}$	$\boxed{M}_-$	$: \boxed{\underline{B}}_-$
$\mathsf{let}\ W\ \mathsf{be}\ x.N$	$: \underline{B}$	$\mu\star^-.\langle \boxed{W}_+ \parallel \tilde{\mu}x^+.\langle \boxed{N}_- \parallel \star\rangle^- \rangle^+$	$: \boxed{\underline{B}}_-$
$\mathsf{return}\ W$	$: \mathsf{F}\,A$	$\mu\{\star\}.\langle \boxed{W}_+ \parallel \star\rangle^+$	$: \Uparrow \boxed{A}_+$
$M\ \mathsf{to}\ x.N$	$: \underline{B}$	$\mu\star^-.\langle \boxed{M}_- \parallel \{\tilde{\mu}x^+.\langle \boxed{N}_- \parallel \star\rangle^-\}\rangle^-$	$: \boxed{\underline{B}}_-$
$\mathsf{force}\ W$	$: \underline{B}$	$\mu\star^-.\langle \boxed{W}_+ \parallel \tilde{\mu}\{x\}.\langle x \parallel \star\rangle^- \rangle^+$	$: \boxed{\underline{B}}_-$
$\mathsf{pm}\ W\ \mathsf{as}\ ().N$	$: \underline{B}$	$\mu\star^-.\langle \boxed{W}_+ \parallel \tilde{\mu}().\langle \boxed{N}_- \parallel \star\rangle^- \rangle^+$	$: \boxed{\underline{B}}_-$
$\mathsf{pm}\ W\ \mathsf{as}\ (x_1, x_2).N$	$: \underline{B}$	$\mu\star^-.\langle \boxed{W}_+ \parallel \tilde{\mu}(x_1, x_2).\langle \boxed{N}_- \parallel \star\rangle^- \rangle^+$	$: \boxed{\underline{B}}_-$
$\mathsf{pm}\ W\ \mathsf{as}\ \{(1, x_1).N_1, (2, x_2).N_2\}$	$: \underline{B}$	$\mu\star^-.\langle \boxed{W}_+ \parallel \tilde{\mu}[x_1.\boxed{N_1}_- \mid x_2.\boxed{N_2}_-]\rangle^+$	$: \boxed{\underline{B}}_-$
$\lambda x.N$	$: A \to \underline{B}$	$\mu(x \cdot \star).\langle \boxed{N}_- \parallel \star\rangle^-$	$: \boxed{A}_+ \multimap \boxed{\underline{B}}_-$
$W`M$	$: \underline{B}$	$\mu\star^-.\langle \boxed{M}_- \parallel \boxed{W}_+ \cdot \star\rangle^-$	$: \boxed{\underline{B}}_-$
$\lambda\{1.N_1, 2.N_2\}$	$: \prod_{i\in\{1,2\}} \underline{B_i}$	$\mu{<}\star.\langle \boxed{N_1}_- \parallel \star\rangle^-;\star.\langle \boxed{N_2}_- \parallel \star\rangle^-{>}$	$: \boxed{\underline{B_1}}_- \,\&\, \boxed{\underline{B_2}}_-$
$\hat{\imath}`M$	$: \underline{B}$	$\mu\star^-.\langle \boxed{M}_- \parallel \pi_{\hat{\imath}} \cdot \star\rangle^-$	$: \boxed{\underline{B}}_-$

(c) Translation of computations

$$\boxed{\langle M \parallel K\rangle} \quad := \quad \langle \boxed{M}_- \parallel \boxed{K}_- \rangle^-$$

$$\boxed{E} \quad := \quad \begin{cases} \boxed{E} & \text{if } \boxed{E} \not\rhd_\mu \\ c' & \text{if } \boxed{E} \rhd_\mu c' \text{ and whenever } E \rhd E', c' \neq \boxed{E'} \end{cases}$$

(d) Translation of configurations

Figure 2.1: Translation from CBPV to $\mathbf{LJ}_p^\eta$

Proposition 2.2. *A configuration E is in normal form if and only if its translation $\ulcorner E \lrcorner$ is.*

Corollary 2.3.
- *For any configuration E_1 and command c_2, if $\ulcorner E_1 \lrcorner \rhd c_2$, then there exist some E_3 so that $E_1 \rhd E_3$ and $c_2 \rhd^* \ulcorner E_3 \lrcorner$.*
- *For any configurations $E_1$and E_2, $E_1 \rhd^+ E_2$ if and only if $\ulcorner E_1 \lrcorner \rhd^+ \ulcorner E_2 \lrcorner$.*
- *For any configurations $E_1$and E_2, E_2 is a normal form of E_1 if and only if $\ulcorner E_2 \lrcorner$ is a $\rhd$-normal form of $\ulcorner E_1 \lrcorner$.*
- *For any configuration E, E is (strongly) $\rhd$-normalising if and only if $\ulcorner E \lrcorner$ is.*

Remark 2.4. We would also like "good" normal forms to be sent on "good" normal forms, where by "good" we mean "locally well-typed". For example, $\mathsf{pm}\ (v_1, v_2)\ \mathsf{as}\ ().t$ is not "good" because we're pattern matching a pair against $()$. While we could define such forms by adding reductions from "bad" states to some error state $\mathscr{E}$, in both CBPV and $\mathbf{LJ}^\eta_p$, there is a simple characterisation of "good" normal forms.

Definition 2.5. A configuration E (resp. command c) is potentially reducible if there is a substitution $\sigma : \mathbb{x} \to \mathbb{W}$ (resp. $\sigma : \mathbb{x} \to \mathbb{V}$) and a stack K (resp. S) so that $E\big[\sigma, K \ /\ \mathsf{nil}\big]$ (resp. $c[\sigma, S \ /\ \star])$ is not normal.

Remark 2.6. The intuition is that "bad" normal forms have incompatible head constructors / destructors, and substitutions can not change head constructors.

Proposition 2.7. *If E is a potentially reducible normal form if and only if $\ulcorner E \lrcorner$ is.*

2.2. Type preservation

Both translations preserve types.

Proposition 2.8.
- *For any context Γ, value W and value type A, if $\Gamma \vdash W : A$, then $\boxed{\Gamma}_+ \vdash \boxed{W}_+ : \boxed{A}_+$.*
- *For any context Γ, computation M and computation type $\underline{B}$, if $\Gamma \vdash M : \underline{B}$, then $\boxed{\Gamma}_+ \vdash \boxed{M}_- : \boxed{\underline{B}}_-$.*
- *For any context Γ, stack K and computation types $\underline{B}$ and $\underline{C}$, if $\Gamma \mid \underline{B} \vdash K : \underline{C}$, then $\boxed{\Gamma}_+ \vdash \boxed{K}_- : \boxed{\underline{B}}_- \mid \star : \underline{C}$.*
- *For any context Γ, computation M, stack K, and computations types $\underline{B}$ and $\underline{C}$, if $\Gamma \vdash M : \underline{B}$ and $\Gamma \mid \underline{B} \vdash K : \underline{C}$, then $\boxed{\langle M \parallel K \rangle} : (\boxed{\Gamma}_+ \vdash \star : \boxed{\underline{C}}_-)$.*

Proof. This is proved by a straightforward induction on the derivation. We have proved it in Coq, using the formalisation of $\mathbf{LJ}^\eta_p$ by Simon Boulier. □

Proposition 2.9.
- *For any context Γ, computation M, stack K, and computations types $\underline{B}$ and $\underline{C}$, if $\Gamma \vdash M : \underline{B}$ and $\Gamma \mid \underline{B} \vdash K : \underline{C}$, then $\ulcorner \langle M \parallel K \rangle \lrcorner : (\boxed{\Gamma}_+ \vdash \star : \boxed{\underline{C}}_-)$.*

Proof. By the fact that $\boxed{E} \rhd^* \ulcorner E \lrcorner$ and subject reduction in $\mathbf{LJ}^\eta_p$, $\boxed{E}$ and $\ulcorner E \lrcorner$ have the same type. We conclude by using the last item of proposition 2.8. □

3. Weak reduction in $\mathbf{LJ}_p^\eta$

3.1. A general notion of weak reduction

While weak reduction is a common name for reductions that do not reduce under λ-abstractions, there is, to our knowledge, no general definition of weak reduction. The idea behind weak reduction is that to give an operational semantics, head reduction is not sufficient because it gets blocked (for example on $(II)(II)$) but strong reduction is not satisfactory either because it may cause some terms to diverge even though they should not (for example $(\lambda x.I)\Omega$ in CBN where Ω is a term that diverges). Weak reduction is therefore introduced as a reduction that "unblocks" head reduction without "breaking termination".

These two notions are formalised (in the setting where $\triangleright$ is a head reduction and $\Rightarrow$ is the corresponding strong reduction) as follows:

Definition 3.1. A relation $\rightarrow_1 \subseteq \Rightarrow$ unblocks[3] another relation $\rightarrow_2 \subseteq \Rightarrow$ if and only if $\Rightarrow^* \rightarrow_2 \subseteq \rightarrow_1^* \rightarrow_2$. It is written $U(\rightarrow_1, \rightarrow_2)$.

Remark 3.2. The intuition is that $\rightarrow_2$ is "strong enough" to allow $\rightarrow_1$ to progress.

Example 3.3. For any relation $\rightarrow_2 \subseteq \Rightarrow$, the strong reduction unblocks it: $U(\Rightarrow, \rightarrow_2)$.

Example 3.4. The head reduction $\triangleright_\beta$ in λ-calculus does not unblock itself because $(II)(II) \Rightarrow_\beta II \triangleright_\beta I$ but $(II)(II) \not\triangleright_\beta$. But weak reduction $\rightarrow_\beta$ does unblock head reduction $\triangleright_\beta$ (because the only redex that the weak reduction can not reduce are those under λ-abstractions and reducing those can not create a head redex).

Definition 3.5. Two relations are termination equivalent if they have the same strongly normalising closed terms.

Remark 3.6. We only consider closed terms because if Ω does not terminate, then in CBV λ-calculus, $x\Omega$ is strongly $\triangleright$-normalising but not strongly $\overset{v}{\rightarrow}_\beta$-normalising.

Remark 3.7. For this notion to work, we need to add a new term $\mathscr{E}$ that represents an error state and transitions from "bad" (i.e. not locally well-typed) terms to that error state, such as $\mathsf{pm}\ (v_1, v_2)\ \mathsf{as}\ ().u \triangleright \mathscr{E}$ and $()v \triangleright \mathscr{E}$.

Definition 3.8. A relation $\rightarrow$ so that $\triangleright \subseteq \rightarrow \subseteq \Rightarrow$ is said to be a weak reduction with respect to $\triangleright$ if:

- It unblocks head reduction ($U(\rightarrow, \triangleright)$) and itself ($U(\rightarrow, \rightarrow)$).
- It is termination equivalent to the head reduction.

A weak reduction with respect to $\triangleright$ is called an $\triangleright$-weak reduction.

Remark 3.9. This definition is meant for abstract-machine-like calculi. For arbitrary calculi, the second condition could be replaced by "Any weakly normalizing term for $\Rightarrow$ is strongly normalizing for $\rightarrow$.".

Proposition 3.10. *$\overset{n}{\rightarrow}_\beta$ is a $\blacktriangleright_\beta$-weak reductions.*

[3] While we have not found this notion in the literature, it seems very likely that it has already been used.

Proposition 3.11. *$\overset{v}{\to}_\beta$ is a $\overset{v}{\blacktriangleright}_\beta$-weak reduction.*

Remark 3.12. In both cases, we use the representation with boxes to allow both reductions to operate. Note that replacing $\blacktriangleright_\beta$ by $\rhd_\beta$ would make the propositions false because $(II)(I\Omega)$ (where Ω diverges) is normal for $\rhd_\beta$ but diverges for $\to_\beta$. It would however work using the alternative definition from 3.9.

Proposition 3.13. *In the three abstract-machine-like calculi described in figures 1.4, 1.6, and 1.12, head reduction is a weak reduction with respect to itself.*

Proof. We only need to check that it unblocks itself. This comes from the fact that reductions only depend on the head constructor on each side of the command, and that those will not change unless a head reduction step happens. □

3.2. A stronger weak reduction

Up to some μ and $\tilde{\mu}$ (un)folding, the translations from CBV / CBN λ-calculus to $\mathbf{LJ}_p^\eta$ (described in figure 1.14) are satisfactory on closed terms. But on open terms, for example if f is a variable in $\left\langle \boxed{f}_+^v \,\middle\|\, \tilde{\mu}\{x_f\}.\langle \boxed{a}_+^v \,\|\, \tilde{\mu}{x_a}^+.\langle x_f \,\|\, x_a \cdot \star\rangle^- \rangle^+ \right\rangle^+$, the (un)folding can get stuck. Since this problem could be fixed by reducing under $\tilde{\mu}\{\}$, it motivates looking for a weak reduction strictly stronger that head reduction. Furthermore, head reduction does not account for other works [Ehr16, AG16].

In this section, we restrict ourselves to $\mathbf{LJ}_p^\eta$ without sums (i.e. without $\iota_i(V)$ and $\tilde{\mu}[x_1.c_1 \mid x_2.c_2]$). While we believe that the results could be adapted to all of $\mathbf{LJ}_p^\eta$, adding sums naively breaks both the uniform confluence of proposition 3.21 (because the fact that $\star$ appears twice in $\tilde{\mu}[x_1.c_1 \mid x_2.c_2]$ means that you may have to do two transitions) and the termination equivalence of proposition 3.23 (because a diverging subterm can be dropped by head reduction). Note that in [Ehr16], the weak reduction does not reduce in the branches of sum pattern-matches either.

Definition 3.14. The weak reduction $\to$ on $\mathbf{LJ}_p^\eta$ is defined as the closure of the head reduction $\rhd$ by all constructors except negative covariable binders ($\mu\star^-.c$, $\mu(x \cdot \star).c$, $\mu{<}\star.c_1\,;\star.c_2{>}$ and $\mu\{\star\}.c$).

Remark 3.15. The restriction ensures that values are normal forms which is needed for confluence.

3.3. Confluence

Lemma 3.16. *For any value V, $V \not\to$, i.e. V is a normal form with respect to $\to$.*

Proof. By induction on V. □

Remark 3.17. This comes from the fact that we disallowed reduction under negative covariable binders.

Remark 3.18. It is necessary for confluence because otherwise a value could be reduced and then placed out of reach of weak reduction (for example under a λ-abstraction) by a substitution as in figure 1.2.

Lemma 3.19. *For any command c and stacks S and S', if $S \to S'$, then $c[S / \star] \to c[S' / \star]$.*

Remark 3.20. This comes from the fact that only one covariable is used, which ensures that no matter where the free $\star$ is, it is not under a negative covariable binder.

Proof. The proof of lemma B.2 (which is a slight generalisation of lemma 3.19) is on page 21. □

Proposition 3.21. *The weak reduction $\to$ is uniformly confluent*[4].

Proof. The proof uses lemmas 3.16 and B.2. See page 22. □

Corollary 3.22. *The weak reduction $\to$ is confluent.*

Proof. By proposition 3.21 (and induction on the length of the derivation). □

3.4. Termination

Proposition 3.23. *For any closed command c, c is strongly $\to$-normalising if and only if it is strongly $\triangleright$-normalising.*

Proof. See page 23. □

References

[AG16] ACCATTOLI, Beniamino ; GUERRIERI, Giulio: Open Call-by-Value (Extended Version). In: *CoRR* abs/1609.00322 (2016). http://arxiv.org/abs/1609.00322

[CFM16] CURIEN, Pierre-Louis ; FIORE, Marcelo P. ; MUNCH-MACCAGNONI, Guillaume: A theory of effects and resources: adjunction models and polarised calculi. In: BODÍK, Rastislav (Hrsg.) ; MAJUMDAR, Rupak (Hrsg.): *Proceedings of the 43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 2016, St. Petersburg, FL, USA, January 20 - 22, 2016*, ACM, 2016. – ISBN 978–1–4503–3549–2, 44–56

[CH00] CURIEN, Pierre-Louis ; HERBELIN, Hugo: The duality of computation. In: ODERSKY, Martin (Hrsg.) ; WADLER, Philip (Hrsg.): *Proceedings of the Fifth ACM SIGPLAN International Conference on Functional Programming (ICFP '00), Montreal, Canada, September 18-21, 2000.*, ACM, 2000. – ISBN 1–58113–202–6, 233–243

[DJS97] DANOS, Vincent ; JOINET, Jean-Baptiste ; SCHELLINX, Harold: A New Deconstructive Logic: Linear Logic. In: *J. Symb. Log.* 62 (1997), Nr. 3, 755–807. http://dx.doi.org/10.2307/2275572. – DOI 10.2307/2275572

[4]If $c_l \leftarrow c \to c_r$ with $c_l \neq c_r$, then there is some c_{lr} so that $c_l \to c_{lr} \leftarrow c_r$

[Ehr16] EHRHARD, Thomas: Call-By-Push-Value from a Linear Logic Point of View. In: THIEMANN, Peter (Hrsg.): *Programming Languages and Systems - 25th European Symposium on Programming, ESOP 2016, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2016, Eindhoven, The Netherlands, April 2-8, 2016, Proceedings* Bd. 9632, Springer, 2016 (Lecture Notes in Computer Science). – ISBN 978–3–662–49497–4, 202–228

[Fel91] FELLEISEN, Matthias: On the Expressive Power of Programming Languages. In: *Sci. Comput. Program.* 17 (1991), Nr. 1-3, 35–75. http://dx.doi.org/10.1016/0167-6423(91)90036-W. – DOI 10.1016/0167–6423(91)90036–W

[Her05] HERBELIN, Hugo: *C'est maintenant qu'on calcule*. Version: 2005. http://pauillac.inria.fr/~herbelin/habilitation/memoire+errata.pdf

[Kri07] KRIVINE, Jean-Louis: A call-by-name lambda-calculus machine. In: *Higher-Order and Symbolic Computation* 20 (2007), Nr. 3, 199–207. http://dx.doi.org/10.1007/s10990-007-9018-9. – DOI 10.1007/s10990–007–9018–9

[Lev04] LEVY, Paul B.: *Semantics Structures in Computation*. Bd. 2: *Call-By-Push-Value: A Functional/Imperative Synthesis*. Springer, 2004. – ISBN 1–4020–1730–8

[LM07] LIANG, Chuck ; MILLER, Dale: Focusing and Polarization in Intuitionistic Logic. In: *CoRR* abs/0708.2252 (2007). http://arxiv.org/abs/0708.2252

[MS15] MUNCH-MACCAGNONI, Guillaume ; SCHERER, Gabriel: Polarised Intermediate Representation of Lambda Calculus with Sums. In: *30th Annual ACM/IEEE Symposium on Logic in Computer Science, LICS 2015, Kyoto, Japan, July 6-10, 2015*, IEEE Computer Society, 2015. – ISBN 978–1–4799–8875–4, 127–140

[Mun] MUNCH-MACCAGNONI, Guillaume: *Note on Curry's style for Linear Call-by-Push-Value*. http://guillaume.munch.name/files/curry.pdf

Contents

A. Proofs of section 2

Lemma A.1. *For any computation M (resp. value W, stack K), the free variables of $\boxed{M}_-$ (resp. $\boxed{W}_+$, $\boxed{K}_-$) are exactly those of M (resp. W, K).*

Proof. By mutual induction on M and W, and then by induction on K. □

Corollary A.2. *For any configuration E, $\boxed{E}$ is closed if and only if E is.*

Definition A.3. Given a substitution $\sigma : \mathbb{x} \to \mathbb{W}$, we define the translated substitution $\boxed{\sigma}_+ : \mathbb{x} \to \mathbb{V}$ by $\boxed{\sigma}_+(x) := \boxed{\sigma(x)}_+$ for all variable x.

Lemma A.4 (Substitution lemma)**.**

- *If $\sigma : \mathbb{x} \to \mathbb{W}$ is a substitution, then for any computation M (resp. value W), $\boxed{M[\sigma]}_- = \boxed{M}_-[\boxed{\sigma}_+]$ (resp. $\boxed{W[\sigma]}_+ = \boxed{W}_+[\boxed{\sigma}_+]$).*
- *If K is a stack, then for any stack K', $\boxed{K'[K \;/\; \mathsf{nil}]}_- = \boxed{K'}_-[\boxed{K}_- \;/\; \star]$.*

Proof.

- By mutual induction on M and W.
- By induction on K.

□

A.1. Proofs of section 2.1

Proof of proposition 2.1 (page 11). We prove it by case analysis.

- If $E = \langle \mathsf{let}\, x\, \mathsf{be}\, W.N \parallel K\rangle \rhd \langle N[W \;/\; x] \parallel K\rangle = E'$,

$$\begin{tikzcd}
E \arrow[r, phantom, "="] & \langle \mathsf{let}\, x\, \mathsf{be}\, W.N \parallel K\rangle \arrow[r, phantom, "\rightsquigarrow"] \arrow[ddd, phantom, "\triangledown"] & \left\langle \mu\star^-.\langle \boxed{W}_+ \parallel \tilde\mu x^+.\langle \boxed{N}_- \parallel K\rangle^-\rangle^+ \,\middle\|\, \boxed{K}_- \right\rangle^- \arrow[r, phantom, "="] \arrow[d, phantom, "\triangledown_{\mu}"] & \boxed{E} \\
& & \langle \boxed{W}_+ \parallel \tilde\mu x^+.\langle \boxed{N}_- \parallel \boxed{K}_-\rangle^-\rangle^+ \arrow[r, phantom, "="] \arrow[d, phantom, "\triangledown_{\tilde\mu}"] & \boxed{E} \\
& & \langle [\boxed{M}_- \;/\; \boxed{N}_+]x \parallel \boxed{K}_-\rangle^- \arrow[d, equal] & \\
E \arrow[r, phantom, "="] & \langle N[W \;/\; x] \parallel K\rangle \arrow[r, phantom, "\rightsquigarrow"] & \langle \boxed{N[W \;/\; x]}_- \parallel \boxed{K}_-\rangle^- \arrow[r, phantom, "="] & \boxed{E'}
\end{tikzcd}$$

- If $E = \langle M \text{ to } x.N \parallel K\rangle \rhd \langle M \parallel \square \text{ to } x.N :: K\rangle = E'$,

$$\begin{array}{rclcccccc}
E & = & \langle M \text{ to } x.N \parallel K\rangle & \rightsquigarrow & \Big\langle \mu\star^-.\langle \boxed{M}_- \parallel \{\tilde\mu x^+.\langle \boxed{N}_- \parallel \star\rangle^-\}\rangle^- \Big\| \boxed{K}_- \Big\rangle^- & = & \boxed{E} & = & \dbox{E} \\
 & & \triangledown & & \triangledown_{\mu\{\}} & & & & \\
E' & = & \langle M \parallel \square \text{ to } x.N :: K\rangle & \rightsquigarrow & \langle \boxed{M}_- \parallel \{\tilde\mu x^+.\langle N \parallel \boxed{K}_-\rangle^-\}\rangle^- & = & \boxed{E'} & &
\end{array}$$

- If $E = \langle \text{return } W \parallel \square \text{ to } x.N :: K\rangle \rhd \langle N[W \,/\, x] \parallel K\rangle = E'$,

$$\begin{array}{rclcccccc}
E & = & \langle \text{return } W \parallel \square \text{ to } x.N :: K\rangle & \rightsquigarrow & \langle \mu\{\star\}.\langle \boxed{W}_+ \parallel \star\rangle^+ \parallel \{\tilde\mu x^+.\langle \boxed{N}_- \parallel \boxed{K}_-\rangle^-\}\rangle^- & = & \boxed{E} & = & \dbox{E} \\
 & & & & \triangledown_{\tilde\mu\{\}} & & & & \\
 & & & & \langle \boxed{W}_+ \parallel \tilde\mu x^+.\langle \boxed{N}_- \parallel \boxed{K}_-\rangle^-\rangle^+ & & & & \\
 & & \triangledown & & \triangledown_{\tilde\mu} & & & & \\
 & & & & \langle \boxed{N}_- [\boxed{W}_+ \,/\, x] \parallel \boxed{K}_-\rangle^- & & & & \\
 & & & & \parallel & & & & \\
E' & = & \langle N[W \,/\, x] \parallel K\rangle & \rightsquigarrow & \langle \boxed{N[W \,/\, x]}_- \parallel \boxed{K}_-\rangle^- & = & \boxed{E'} & &
\end{array}$$

- If $E = \langle \text{force thunk } M \parallel K\rangle \rhd \langle M \parallel K\rangle = E'$,

$$\begin{array}{rclcccc}
E & = & \langle \text{force thunk } M \parallel K\rangle & \rightsquigarrow & \Big\langle \mu\star^-.\langle \{\boxed{M}_-\} \parallel \tilde\mu\{x\}.\langle x \parallel \star\rangle^-\rangle^+ \Big\| \boxed{K}_- \Big\rangle^- & = & \boxed{E} \\
 & & & & \triangledown_{\mu} & & \\
 & & \triangledown & & \langle \{\boxed{M}_-\} \parallel \tilde\mu\{x\}.\langle x \parallel \boxed{K}_-\rangle^-\rangle^+ & = & \dbox{E} \\
 & & & & \triangledown_{\tilde\mu\{\}} & & \\
E' & = & \langle M \parallel K\rangle & \rightsquigarrow & \langle \boxed{M}_- \parallel \boxed{K}_-\rangle^- & = & \boxed{E'}
\end{array}$$

- If $E = \big\langle \text{pm } () \text{ as } ().M \,\big\|\, K\big\rangle \rhd \langle M \parallel K\rangle = E'$,

$$\begin{array}{rclcccc}
E & = & \langle \text{pm } () \text{ as } ().M \parallel K\rangle & \rightsquigarrow & \Big\langle \mu\star^-.\langle () \parallel \tilde\mu().\langle \boxed{M}_- \parallel \star\rangle^-\rangle^+ \Big\| \boxed{K}_- \Big\rangle^- & = & \boxed{E} \\
 & & & & \triangledown_{\mu} & & \\
 & & \triangledown & & \langle () \parallel \tilde\mu().\langle \boxed{M}_- \parallel \boxed{K}_-\rangle^-\rangle^+ & = & \dbox{E} \\
 & & & & \triangledown_{\mathbf{1}} & & \\
E' & = & \langle M \parallel K\rangle & \rightsquigarrow & \langle \boxed{M}_- \parallel \boxed{K}_-\rangle^- & = & \boxed{E'}
\end{array}$$

- If $E = \big\langle \text{pm } (W_1, W_2) \text{ as } (x_1, x_2).N \,\big\|\, K\big\rangle \rhd \big\langle N\big[W_1 \,/\, x_1, W_2 \,/\, x_2\big] \,\big\|\, K\big\rangle = E'$,

$$\begin{array}{rclcccc}
E & = & \langle \text{pm } (W_1, W_2) \text{ as } (x_1, x_2).N \parallel K\rangle & \rightsquigarrow & \Big\langle \mu\star^-.\langle (\boxed{W_1}_+, \boxed{W_2}_+) \parallel \tilde\mu(x_1, x_2).\langle \boxed{N}_- \parallel \star\rangle^-\rangle^+ \Big\| \boxed{K}_- \Big\rangle^- & = & \boxed{E} \\
 & & & & \triangledown_{\mu} & & \\
 & & & & \Big\langle (\boxed{W_1}_+, \boxed{W_2}_+) \Big\| \langle \tilde\mu(x_1, x_2).\langle \boxed{N}_- \parallel \star\rangle^- \parallel \boxed{K}_-\rangle^+ \Big\rangle^+ & = & \dbox{E} \\
 & & \triangledown & & \triangledown_{\otimes} & & \\
 & & & & \langle \boxed{N}_- [\boxed{W_1}_+ \,/\, x, \boxed{W_2}_+ \,/\, x_2] \parallel \boxed{K}_-\rangle^- & & \\
 & & & & \parallel & & \\
E' & = & \langle M[W_1 \,/\, x_1, W_2 \,/\, x_2] \parallel K\rangle & \rightsquigarrow & \langle \boxed{M[W_1 \,/\, x_1, W_2 \,/\, x_2]}_- \parallel \boxed{K}_-\rangle^- & = & \boxed{E'}
\end{array}$$

- If $E = \big\langle \text{pm } (\hat\imath, W) \text{ as } \{(1, x).N_1, (2, x).N_2\} \,\big\|\, K\big\rangle \rhd \big\langle N_{\hat\imath}\big[W \,/\, x\big] \,\big\|\, K\big\rangle$,

$$\begin{array}{rclcccc}
E & = & \langle \text{pm } (\hat\imath, W) \text{ as } \{(1, x).N_1, (2, x).N_2\} \parallel K\rangle & \rightsquigarrow & \Big\langle \mu\star^-.\langle (\hat\imath, \boxed{W}_+) \parallel \tilde\mu[x.\langle \boxed{N_1}_- \parallel \star\rangle^- \mid x.\langle \boxed{N_2}_- \parallel \star\rangle^-]\rangle^+ \Big\| \boxed{K}_- \Big\rangle^- & = & \boxed{E} \\
 & & & & \triangledown_{\mu} & & \\
 & & & & \langle (\hat\imath, \boxed{W}_+) \parallel \tilde\mu[x.\langle \boxed{N_1}_- \parallel \boxed{K}_-\rangle^- \mid x.\langle \boxed{N_2}_- \parallel \boxed{K}_-\rangle^-]\rangle^+ & = & \dbox{E} \\
 & & \triangledown & & \triangledown_{\oplus} & & \\
 & & & & \langle N_{\hat\imath}[\boxed{W}_+ \,/\, x] \parallel \boxed{K}_-\rangle^- & & \\
 & & & & \parallel & & \\
E' & = & \langle N_{\hat\imath}[W \,/\, x] \parallel K\rangle & \rightsquigarrow & \langle \boxed{N_{\hat\imath}[W \,/\, x]}_- \parallel \boxed{K}_-\rangle^- & = & \boxed{E'}
\end{array}$$

- If $E = \langle W`M \parallel K \rangle \triangleright \langle M \parallel W :: K \rangle = E'$,

$$\begin{array}{ccccccccc}
E & = & \langle W`M \parallel K\rangle & \rightsquigarrow & \langle \mu\star^{-}.\langle \boxed{M}_{-} \parallel \boxed{W}_{+} \cdot \star\rangle^{-} \parallel \boxed{K}_{-}\rangle^{-} & = & \boxed{E} & = & \boxed{\bar{E}} \\
 & & \triangledown & & \triangledown_{\mu} & & & & \\
E' & = & \langle M \parallel W :: K\rangle & \rightsquigarrow & \langle \boxed{M}_{-} \parallel \boxed{W}_{+} \cdot \boxed{K}_{-}\rangle^{-} & = & \boxed{E'} & &
\end{array}$$

- If $E = \langle \lambda x.M \parallel W :: K \rangle \triangleright \left\langle N\big[W \,/\, x\big] \,\big\|\, K \right\rangle = E'$,

$$\begin{array}{ccccccccc}
E & = & \langle \lambda x.N \parallel W :: K\rangle & \rightsquigarrow & \langle \mu(x\cdot\star).\langle \boxed{N}_{-} \parallel \star\rangle^{-} \parallel \boxed{W}_{+} \cdot \boxed{K}_{-}\rangle^{-} & = & \boxed{E} & = & \boxed{\bar{E}} \\
 & & & & \triangledown \\
 & & & & \downarrow \\
 & & \triangledown & & \langle \boxed{N}_{-} \big[\boxed{W}_{+} \,/\, x\big] \parallel \boxed{K}_{-}\rangle^{-} & & & & \\
 & & & & \parallel & & & & \\
E' & = & \langle N\big[W \,/\, x\big] \parallel K\rangle & \rightsquigarrow & \langle \boxed{N\big[W \,/\, x\big]}_{-} \parallel \boxed{K}_{-}\rangle^{-} & = & \boxed{E'} & &
\end{array}$$

- If $E = \langle \hat{\imath}`M \parallel K \rangle \triangleright \langle M \parallel \hat{\imath} :: K \rangle = E'$,

$$\begin{array}{ccccccccc}
E & = & \langle \hat{\imath}`M \parallel K\rangle & \rightsquigarrow & \langle \mu\star^{-}.\langle \boxed{M}_{-} \parallel \hat{\imath} :: \star\rangle^{-} \parallel \boxed{K}_{-}\rangle^{-} & = & \boxed{E} & = & \boxed{\bar{E}} \\
 & & \triangledown & & \triangledown_{\mu} & & & & \\
E' & = & \langle M \parallel \hat{\imath} :: K\rangle & \rightsquigarrow & \langle \boxed{M}_{-} \parallel \hat{\imath} :: \boxed{K}_{-}\rangle^{-} & = & \boxed{E'} & &
\end{array}$$

- If $E = \big\langle \lambda\{1.N_1, 2.N_2\} \,\big\|\, \hat{\imath} :: K \big\rangle \triangleright \langle N_{\hat{\imath}} \parallel K \rangle = E'$,

$$\begin{array}{ccccccccc}
E & = & \big\langle \lambda\{1.N_1, 2.N_2\} \,\big\|\, \hat{\imath} :: K\big\rangle & \rightsquigarrow & \langle \mu{<}\star.\langle \boxed{N_1}_{-} \parallel \star\rangle^{-};\star.\langle \boxed{N_2}_{-} \parallel \star\rangle^{-}{>} \parallel \hat{\imath} :: \boxed{K}_{-}\rangle^{-} & = & \boxed{E} & = & \boxed{\bar{E}} \\
 & & \triangledown & & \triangledown_{\mu} & & & & \\
E' & = & \langle N_{\hat{\imath}} \parallel K\rangle & \rightsquigarrow & \langle \boxed{M_{\hat{\imath}}}_{-} \parallel \boxed{K}_{-}\rangle^{-} & = & \boxed{E'} & &
\end{array}$$

□

Proof of proposition 2.2 (page 13). We prove each direction separately:

$\boxed{\Rightarrow}$ There are 11 reductions rules for CBPV, among which 4 apply as soon as the head constructor of the computation matches (the 3 that push things to the context, and the let). We therefore only have 7 cases to handle, each corresponding to a different head constructor of the computation:

◇ If $E = \langle \mathsf{return}\ W \parallel K\rangle$ where K is not of the form $\square$ to $x.N :: K'$, $\boxed{\bar{E}} = \big\langle \mu\{\star\}.\langle \boxed{W}_{+} \parallel \star\rangle^{+} \,\big\|\, \boxed{K}_{-}\big\rangle^{-}$ where $\boxed{K}_{-}$ is not of the form $\{S\}$ which is indeed a normal form.

◇ If $E = \langle \mathsf{force}\ W \parallel K\rangle$ where W is not of the form $\mathsf{thunk}\ M$, $\boxed{\bar{E}} = \big\langle \boxed{W}_{+} \,\big\|\, \tilde{\mu}\{x\}.\langle x \parallel \boxed{K}_{-}\rangle^{-}\big\rangle^{+}$ where $\boxed{W}_{+}$ is not of the form $\{V\}$ which is indeed a normal form.

◇ If $E = \big\langle \mathsf{pm}\ W\ \mathsf{as}\ ().N \,\big\|\, K\big\rangle$ where W is not of the form $()$, $\boxed{\bar{E}} = \big\langle \boxed{W}_{+} \,\big\|\, \tilde{\mu}().\langle \boxed{N}_{-} \parallel \boxed{K}_{-}\rangle^{-}\big\rangle^{+}$ where $\boxed{W}_{+}$ is not of the form $()$ which is indeed a normal form.

◇ If $E = \big\langle \mathsf{pm}\ W\ \mathsf{as}\ (x_1, x_2).N \,\big\|\, K\big\rangle$ where W is not of the form (W_1, W_2), $\boxed{\bar{E}} = \big\langle \boxed{W}_{+} \,\big\|\, \tilde{\mu}(x_1, x_2).\langle \boxed{N}_{-} \parallel \boxed{K}_{-}\rangle^{-}\big\rangle^{+}$ where $\boxed{W}_{+}$ is not of the form (V_1, V_2) which is indeed a normal form.

◇ If $E = \big\langle \mathsf{pm}\ W\ \mathsf{as}\ \{(1,x).N_1, (2,x).N_2\} \,\big\|\, K\big\rangle$ where W is not of the form $(\hat{\imath}, W')$, $\boxed{\bar{E}} = \big\langle \boxed{W}_{+} \,\big\|\, \tilde{\mu}\big[x.\langle \boxed{N_1}_{-} \parallel \boxed{K}_{-}\rangle^{-} \mid x.\langle \boxed{N_2}_{-} \parallel \boxed{K}_{-}\rangle^{-}\big]\big\rangle^{+}$ where $\boxed{W}_{+}$ is not of the form $\iota_{\hat{\imath}}(V')$ which is indeed a normal form.

◇ If $E = \langle \lambda x.N \parallel K\rangle$ where K is not of the form $W :: K'$, $\boxed{\bar{E}} = \big\langle \mu(x\cdot\star).\langle \boxed{N}_{-} \parallel \star\rangle^{-} \,\big\|\, \boxed{K}_{-}\big\rangle^{-}$ where $\boxed{K}_{-}$ is not of the form $V \cdot S$ which is indeed a normal form.

◇ If $E = \langle \lambda\{1.N_1, 2.N_2\} \,\|\, K \rangle$ where K is not of the form $\hat{\imath} :: K'$,
$\dbox{E} = \langle \mu{<}\star.\langle \boxed{N_1}_{\_} \,\|\, \star\rangle^{-} ; \star.\langle \boxed{N_2}_{\_} \,\|\, \star\rangle^{-}{>} \,\|\, \boxed{K}_{\_} \rangle^{-}$ where $\boxed{K}_{\_}$ is not of the form $\pi_{\hat{\imath}} \cdot S$.

$\boxed{\Leftarrow}$ We prove the contrapositive. Suppose that the command E is not a normal form. There exist a command E' so that $E \rhd E'$. By proposition 2.1, $\dbox{E} \rhd \boxed{E'}$ so that $\dbox{E}$ isn't a normal form either.

$\square$

Proof of proposition 2.3 (page 13).

- By proposition 2.2, E_1 is not a normal form so that E_3 exists. Since the head reduction is deterministic, we can conclude that $c_2 \rhd^* E_3$.
- By strong induction on the length of the reduction, proposition 2.1 and the previous bullet.
- By the previous bullet and proposition 2.2.
- By the previous bullet.

$\square$

Proof of proposition 2.7 (page 13). $\square$

B. Proofs of section 3

Proposition B.1. *If $c_l \lhd c(\to \setminus \rhd)c_r$, then there exists some c_{lr} so that $c_l \to c_{lr} \lhd c_r$.*

Proof of B.1. We start by proving the following generalisation of lemma 3.19:

Lemma B.2. *For any command c, substitution $\sigma : \mathbb{x} \to \mathbb{V}$ and stacks S and S', if $S \to S'$, then $c[\sigma, S \,/\, \star] \to c[\sigma, S' \,/\, \star]$.*

Proof. We prove by mutual induction on the command c and the context e that for any command c (resp. context e), substitution $\sigma : \mathbb{x} \to \mathbb{V}$ and stacks S and S', if $S \to S'$, then $c[\sigma, S \,/\, \star] \to c[\sigma, S \,/\, \star]$ (resp. $e[\sigma, S \,/\, \star] \to e[\sigma, S \,/\, \star]$). $\square$

Lemma B.3. *For any commands c and c', substitution $\sigma : \mathbb{x} \to \mathbb{V}$ and stack S, if $c \to c'$, then $c[\sigma, S \,/\, \star] \rhd c'[\sigma, S \,/\, \star]$.*

Proof. By case analysis on the reduction. $\square$

Lemma B.4. *For any commands c and c', substitution $\sigma : \mathbb{x} \to \mathbb{V}$ and stack S, if $c \to c'$, then $c[\sigma, S \,/\, \star] \to c'[\sigma, S \,/\, \star]$.*

Proof. We have a command with a hole c_0 and two commands c_1 and c_1' so that $c = c_{\square}[\,c_1\,]$, $c' = c_{\square}[\,c_1'\,]$ and $c_1 \rhd c_1'$. By lemma B.3, we have $c_1[\,\sigma, S \,/\, \star\,] \rhd c_1'[\,\sigma, S \,/\, \star\,]$. The hole $\square$ is still at a position where weak reduction can be done in $c_{\square}[\,\sigma, S \,/\, \star\,]$ (because the substitution only replaces leaves of the syntax tree, and therefore replaces nothing above the hole $\square$). We can therefore conclude that $c[\,\sigma, S \,/\, \star\,] = (\,c_{\square}[\sigma, S \,/\, \star]\,)[\,c_1[\sigma, S \,/\, \star]\,] \to (c_{\square}[\sigma, S \,/\, \star])[c_1'[\sigma, S \,/\, \star]] = c'[\sigma, S \,/\, \star]$. $\square$

Let c, c_l, c_r be commands such that $c_l \lhd c(\to \setminus \rhd)c_r$. Note that this is equivalent to $c_l \lhd c \to c_r$ and $c_l \neq c_r$. We show that there is some c_{lr} such that $c_l \to c_{lr} \lhd c_r$.

- If $c = \langle V \,\|\, \tilde{\mu} x^{\varepsilon}.c_1 \rangle^{\varepsilon} \triangleright c_1\big[V \,/\, x\big] = c_l$, then by lemma 3.16 the other reduction is of the form $c = \langle V \,\|\, \tilde{\mu} x^{\varepsilon}.c_1 \rangle^{\varepsilon} \to \langle V \,\|\, \tilde{\mu} x^{\varepsilon}.c_1' \rangle^{\varepsilon} = c_r$ where $c_1 \to c_1'$. And then:

$$\begin{tikzcd} \langle V \,\|\, \tilde{\mu} x^{\varepsilon}.c_1 \rangle^{\varepsilon} \arrow[r, phantom, "\triangleright"] \arrow[d] & c_1\big[V \,/\, x\big] \arrow[d, dashed, "\text{by lemma B.4}"] \\ \langle V \,\|\, \tilde{\mu} x^{\varepsilon}.c_1' \rangle^{\varepsilon} \arrow[r, phantom, "\triangleright"] & c_1'\big[V \,/\, x\big] \end{tikzcd}$$

  A similar argument works for all commands of the form $c = \langle V \,\|\, \tilde{\mu} \ldots \rangle^{\varepsilon}$.
- If $c = \langle \mu\star^{\varepsilon}.c_1 \,\|\, S \rangle^{\varepsilon} \triangleright c_1\big[\, S \,/\, \star \,\big] = c_l$, then there are two possibilities for the other reduction:
  - ◇ Either $c = \langle \mu\star^{\varepsilon}.c_1 \,\|\, S \rangle^{\varepsilon} \to \langle \mu\star^{\varepsilon}.c_1' \,\|\, S \rangle^{\varepsilon} = c_r$ and then:

$$\begin{tikzcd} \langle \mu\star^{\varepsilon}.c_1 \,\|\, S \rangle^{\varepsilon} \arrow[r, phantom, "\triangleright"] \arrow[d] & c_1\big[S \,/\, \star\big] \arrow[d, dashed, "\text{by lemma B.4}"] \\ \langle \mu\star^{\varepsilon}.c_1' \,\|\, S \rangle^{\varepsilon} \arrow[r, phantom, "\triangleright"] & c_1'\big[S \,/\, \star\big] \end{tikzcd}$$

  - ◇ Or $c = \langle \mu\star^{\varepsilon}.c_1 \,\|\, S \rangle^{\varepsilon} \to \langle \mu\star^{\varepsilon}.c_1 \,\|\, S' \rangle^{\varepsilon} = c_r$ and then:

$$\begin{tikzcd} \langle \mu\star^{\varepsilon}.c_1 \,\|\, S \rangle^{\varepsilon} \arrow[r, phantom, "\triangleright"] \arrow[d] & c_1\big[S \,/\, \star\big] \arrow[d, dashed, "\text{by lemma B.2}"] \\ \langle \mu\star^{\varepsilon}.c_1 \,\|\, S' \rangle^{\varepsilon} \arrow[r, phantom, "\triangleright"] & c_1\big[S' \,/\, \star\big] \end{tikzcd}$$

  A similar argument works for all commands of the form $c = \langle \mu \ldots \,\|\, S \rangle^{\varepsilon}$. We describe two more cases.
  - ◇ In the case where $c = \big\langle \mu(x \cdot \star).c_1 \,\big\|\, V \cdot S \big\rangle^{-}$, if the stack gets reduced, then the reduction is of the form $V \cdot S \to V \cdot S'$ by lemma 3.19, and then:

$$\begin{tikzcd} \big\langle \mu(x \cdot \star).c_1 \,\big\|\, V \cdot S \big\rangle^{-} \arrow[r, phantom, "\triangleright"] \arrow[d] & c_1\big[V \,/\, x, S \,/\, \star\big] \arrow[d, dashed, "\text{by lemma B.2}"] \\ \big\langle \mu(x \cdot \star).c_1 \,\big\|\, V \cdot S' \big\rangle^{-} \arrow[r, phantom, "\triangleright"] & c_1\big[V \,/\, x, S' \,/\, \star\big] \end{tikzcd}$$

  - ◇ In the case where $c = \langle \mu{<}\star.c_1 \,;\, \star.c_2{>} \,\|\, \pi_i \cdot S \rangle^{-}$, we have:

$$\begin{tikzcd} \langle \mu{<}\star.c_1 \,;\, \star.c_2{>} \,\|\, \pi_i \cdot S \rangle^{-} \arrow[r, phantom, "\triangleright"] \arrow[d] & c_i\big[S \,/\, \star\big] \arrow[d, dashed, "\text{by lemma B.2}"] \\ \langle \mu{<}\star.c_1 \,;\, \star.c_2{>} \,\|\, \pi_i \cdot S' \rangle^{-} \arrow[r, phantom, "\triangleright"] & c_i\big[S' \,/\, \star\big] \end{tikzcd}$$

□

Proof of proposition 3.21 on page 16. Let c, c_l, c_r be commands such that $c_l \leftarrow c \to c_r$ and $c_l \neq c_r$ We want to find some c_{lr} so that $c_l \to c_{lr} \leftarrow c_r$.

If the two contracted redexes are in disjoint subterms, then contracting the remaining redex (among the two that are being considered) in both c_l and c_r yields the same term, which we pick as c_{lr}.

Otherwise, one of the two redexes is (strictly) above the other one (because since the head reduction $\triangleright$ is deterministic, they can not be at the same level). Suppose, without loss of generality, that the contraction of the highest of the two redex is $c_l \leftarrow c$. There is some command with a hole $c_\square$ and commands c^*, c_l^* and c_r^* so that $c = c_\square[\, c^*\,], c_l = c_\square[\, c_l^*\,]$, $c_r = c_\square[\, c_r^*\,]$, $c_l^* \lhd c^*(\to \setminus \rhd)c_r^*$. By proposition B.1, we have some $c^{*\prime}$ so that $c_l^*(\to \setminus \rhd)c^* \lhd c_r^*$. We choose $c' = c_\square[c^{*\prime}]$. □

Proof of proposition 3.23 (page 16). Let c be a closed strongly $\rhd$-normalising command and consider a sequence of $\to$-reductions starting from c.

proposition 3.21 induces a standardisation theorem: we can "move" all the head reductions in a sequence of reductions to the beginning of the sequence without changing the number of steps. Since c is $\rhd$-normalising, there can be only finitely many of those. We can therefore consider c', the $\rhd$-normal form of c. Showing that c' is $\to$-normalising will imply that c is.

Since we assume that "bad" normal forms reduce to $\mathscr{E}$ in an extended system, c' is either of the form $\langle V \parallel \star\rangle^\varepsilon$ or of the form $\langle x \parallel S\rangle^\varepsilon$.

In the first case, lemma 3.16 allows to conclude.

In the second case, since c is closed, so are c' and the normal form of c'. So this case is not possible. □