

Call-By-Name, Call-By-Value and Time

Benjamin Lepercqey

PPS, Université Denis Diderot, Paris

Abstract. We build denotational models for the call-by-name and call-by-value versions of a simply typed lambda calculus with recursion, and show that they are adequate: the denotation of a closed term of base type represents exactly the number of reduction steps. We achieve this in a very generic way, using monads to propagate the information about the number of steps. The denotations in the Kleisli category are then straightforward.

1 Introduction

Although the length of computation have been studied extensively by complexity analysis for *algorithms*, little work has been devoted to the analysis of actual *programs* by semantical means. Indeed, traditional denotational semantics is only about what a program computes, and programs computing the same “value” are deliberately equated by the models: we usually call “denotational model” a framework that keeps the denotation of the program constant throughout the reduction, so that, by translating the original, complex program, we get directly the translation of the value it computes.

We show here that denoting the time taken by reduction of a program is not incompatible with denoting its value, even at higher order. We show that this timing analysis can even be put “on top” of existing denotational models quite easily.

Thinking of a program as a computation taking some time to give a value leads naturally to considering monads. We first show that this fits easily in a call-by-value (CBV) setting, with the obvious monad $TA = (N \times A)_\perp$: a program either produces a value in A in a finite number of steps, or diverges.

We then consider the call-by-name (CBN) case. It is more difficult, because the time taken by the entire program is not the sum of the time taken by its parts. Consider the program $(\lambda x.x+x)M$, where M is a complicated, long way of computing 3. In CBV, the program computes M , getting the value 3, and then applies it to $\lambda x.x+x$, yielding 6. The length of the computation is therefore the length of the computation of M plus the length of the application and the addition. In CBN, our program calls M two times: although the result is 6 as in the CBV case, the length of the computation is *twice* the length of M plus the length of the application and the addition. As a consequence, in the call-by-name case, our construction will only work in models that record the number of times each argument is used: we give the example of the coherence spaces with the multiset exponential.

In [10], Sands presents operational techniques and cost models for a lazy declarative language. Although the translation we give can be viewed as syntactical transformations of the terms, this work focuses on denotational semantics. In this regard, it is closer to Escardo’s work on PCF [3]: although the denotations are very similar to his second model, we proved adequacy for more realistic notions of reduction, including call-by-value, and abstracting the time constructions in a monad allows the enrichment of various preexisting models.

In section 2, we define the language and the call-by-name and call-by-value reduction machines. In section 3, we present a naive model and the reason for using monads. In sections 4 and 5, we present categorical models for CBV and CBN, along with examples.

2 The Language

We consider a simply typed λ -calculus. Since we are mainly interested with termination and not with values, we only put one constant \star , with no associated operations. It would not be difficult to add integers, arithmetic operations and conditionals, but then there would be many reduction rules and the contexts would be more complex, which would make the proofs much longer, but not more difficult. In other words, we concentrated on application and variable fetching, because arithmetics and tests, being first order operations, are easy.

Since we consider a typed language, we put a recursion symbol rather than a simple λ -abstraction, to allow non-terminating programs. The terms are:

$$M ::= \star \mid x \mid \text{rec } fx.M \mid (M)N$$

Type algebra is $A ::= 1 \mid A \rightarrow A$, and the typing rules are

$$\frac{}{\Gamma \vdash \star : 1} \qquad \frac{}{x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i}$$

$$\frac{\Gamma, f : A \rightarrow B, x : A \vdash M : B}{\Gamma \vdash \text{rec } fx.M : A \rightarrow B} \quad \frac{\Gamma \vdash M : A \rightarrow B \quad \Gamma \vdash N : A}{\Gamma \vdash (M)N : B}$$

The usual definition of a language gives a equational theory. A denotational translation is then said to be a model when two terms equated by the theory are equated by the model. This approach is not well suited for describing the time of execution: the rules are often not atomic (for instance, β -reduction may perform many complex substitution and renaming operations), and there may be several “paths” between two terms, *i.e.* the rules are not deterministic.

We rather describe reduction machines: although they reduce λ -terms, they implement a precise reduction strategy (for any state, there is at most one applicable rule), and each step is reasonably complex. What we expect from them is that a term of base type either gives a value in finitely many steps, or that the reduction is infinite (in other words, there should not be “stuck” states for a typed term). As there is only one value at base type, namely \star , we are only interested in the length of reduction when it terminates: this is the time taken by the computation.

2.1 Call-By-Name Reduction

The weak head reduction of the λ -calculus is not realistic: many substitutions are performed in a single reduction step. We rather consider an environment machine, where we do not substitute but fetch the content of the variables when we need it. It is close to Krivine's machine [6], but we kept names and we reduce applications one at a time to keep definitions as simple as possible.

The machine reduces a closure $C = (M, E)$ (a term paired with an environment) in a continuation. As we implement call-by-name, continuations are only applicative contexts, and we represent them as stacks, *i.e.* finite lists of closures (ϵ is the empty list, \cdot is the cons operation):

$$C = (M, E) \quad S = C_0 \cdots C_n \cdot \epsilon$$

An environment $E = \{x_1 \rightarrow C_1, \dots, x_n \rightarrow C_n\}$ is a finite list of links from variable names to closures. The update operation, written $E[x \rightarrow C]$, puts the new links at the end. The lookup operation $E(x)$ fetches the *last* closure linked to x , so $E[x \rightarrow C](x) = C$.

A state of the machine is a closure C and a stack S , which we write C/S . The reduction rules are:

$$\begin{array}{ll} (x, E) / S \rightsquigarrow_N E(x)/S & \text{(fetch)} \\ ((M)N, E) / S \rightsquigarrow_N (M, E) / (N, E) \cdot S & \text{(push)} \\ (\text{rec } fx.M, E) / C \cdot S \rightsquigarrow_N (M, E[f \rightarrow (\text{rec } fx.M, E), x \rightarrow C]) / S & \text{(pop)} \end{array}$$

We say that the machine in a state C/S terminates in n steps (we write $C/S \Downarrow_N n$) if $C/S \rightsquigarrow_N^n (\star, E)/\epsilon$ for some environment E , and that it diverges (we write $C/S \not\Downarrow_N$) if the reduction sequence starting from C/S is infinite.

As we want to reduce typed λ -terms, we only consider typed states:

$$\frac{C_1 : A_1 \quad \dots \quad C_n : A_n}{\{x_1 \rightarrow C_1 \dots x_n \rightarrow C_n\} : (x_1 : A_1 \dots x_n : A_n)} \quad \frac{\Gamma \vdash M : A \quad E : \Gamma}{(M, E) : A}$$

$$\frac{C_1 : A_1 \quad \dots \quad C_n : A_n}{C_1 \cdots C_n \cdot \epsilon : (A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1)^\perp} \quad \frac{C : A \quad S : A^\perp}{C/S : 1}$$

Lemma 1. *If $C/S : 1$ and $C/S \rightsquigarrow_N C'/S'$, then $C'/S' : 1$.*

Proposition 1. *If M is a closed term of type 1, then $(M, \emptyset) / \epsilon$ either terminates or diverges.*

Proof. $(M, \emptyset) / \epsilon$ is a typed state. Since typing is preserved through reduction, its reduced forms also have type 1. Therefore, if the reduction stops, the last term must be \star .

2.2 Call-By-Value Reduction

We use a variant of the SECD machine, which was introduced by Landin and used by Plotkin [9] to reason about call-by-value reduction. Just as in the CBN case, the state of the machine comes in three parts: a term, an environment, and a continuation. The correspondence with the SECD machine is as follows: the term is the head of the control string C , the environment is the environment E , and the continuation combines the stack S , the tail of the control string C and the dump D .

In call-by-value, the environment will hold values rather than closures, so we first define the values:

$$V = \star \mid \text{Clos } fx.(M, E)$$

where E is a finite map from names to values: $E = \{x_1 \rightarrow V_1, \dots, x_n \rightarrow V_n\}$. The closures $\text{Clos } fx.(M, E)$ are not allowed in a program, they will be built from abstractions $\text{rec } fx.M$ during the reduction. The terms that the CBV machine reduces are:

$$M = V \mid x \mid \text{rec } fx.M \mid (M)M$$

Note that, unlike in the monadic λ -calculus, the variables are *not* values, because variable fetching takes time. In CBV, the continuations are more complex than simple applicative contexts: in an application, we first reduce the argument, remembering the term that it is applied to, and then the function, remembering the value of the argument. Only then we can perform the substitution. So, the continuations are

$$K = \epsilon \mid (M, E)[\] \cdot K \mid [\]V \cdot K$$

and the reduction rules of the machine are

$$\begin{aligned} (x, E) / K &\rightsquigarrow_V (E(x), E) / K \\ (\text{rec } fx.M, E) / K &\rightsquigarrow_V (\text{Clos } fx.(M, E), E) / K \\ ((M)N, E) / K &\rightsquigarrow_V (N, E) / (M, E)[\] \cdot K \\ (V, E) / (M, E')[\] \cdot K &\rightsquigarrow_V (M, E') / [\]V \cdot K \\ (\text{Clos } fx.(M, E), E') / [\]V \cdot K &\rightsquigarrow_V (M, E[f \rightarrow \text{Clos } fx.(M, E), x \rightarrow V]) / K \end{aligned}$$

Again, we say that the machine in a state $(M, E) / K$ terminates in n steps (we write $(M, E) / K \Downarrow_V n$) if $(M, E) / K \rightsquigarrow_V^n (\star, E') / \epsilon$ for some environment E' , and that it diverges (we write $(M, E) / K \not\Downarrow_V$) if the reduction sequence starting from $(M, E) / K$ is infinite. As in the CBN case, we type the parts of the state:

$$\frac{\frac{\frac{V_1 : A_1 \quad \dots \quad V_n : A_n}{\{x_1 \rightarrow V_1, \dots, x_n \rightarrow V_n\} : x_1 : A_1, \dots, x_n : A_n} \quad \Gamma, f : A \rightarrow B, x : A \vdash M : B \quad E : \Gamma}{\text{Clos } fx.(M, E) : A \rightarrow B}}{\epsilon : 1^\perp \quad \frac{K : B^\perp \quad \Gamma \vdash M : A \rightarrow B \quad E : \Gamma}{(M, E)[\] \cdot K : A^\perp} \quad \frac{K : B^\perp \quad V : A}{[\]V \cdot K : (A \rightarrow B)^\perp}}{\frac{\Gamma \vdash M : A \quad E : \Gamma \quad K : A^\perp}{(M, E) / K : 1}}}$$

Lemma 2. *If $(M, E) / K : 1$ reduces to $(M', E') / K'$ then $(M', E') / K' : 1$.*

Proposition 2. *If M is a closed term of type 1, then $(M, \emptyset) / \epsilon$ either terminates or diverges.*

3 Modelling Time

In this explanatory section, we consider CBN reduction.

3.1 A Syntactical Approach

We want the model to give the result and the number of reduction states needed to get this result. Variable fetching takes one step, application takes two (push, then pop). We can imagine a syntactical transformation of the terms, adding a new construction $\text{tick}; t$ meaning “ask for a time slice and go on with t ” (this is similar to “chattering PCF” of Escardo [3]). We could define a translation into this enriched language:

- $\{\star\} = \star$,
- $\{x\} = \text{tick}; x$
- $\{(M)N\} = \text{tick}; \text{tick}; (\{M\})\{N\}$,
- $\{\text{rec } fx.M\} = \text{rec } fx.\{M\}$.

The equational theory would be defined by the following rules:

$$(\text{rec } fx.M)N \equiv M[x \rightarrow N, f \rightarrow \text{rec } fx.M] \quad (\text{tick}; M)N \equiv \text{tick}; (M)N$$

The normal form of the translation would then give the number of reduction steps, through the number of tick at the head position. For instancee:

$$\begin{aligned} \{(\text{rec } fx.x)\star\} &= \text{tick}; \text{tick}; (\text{rec } fx.\text{tick}; x)\star \equiv \text{tick}; \text{tick}; \text{tick}; \star \\ ((\text{rec } fx.x)\star, \epsilon) / \epsilon &\rightsquigarrow (\text{rec } fx.x, \epsilon) / (\star, \epsilon) \cdot \epsilon \\ &\rightsquigarrow (x, \{x = (\star, \epsilon), f = (\text{rec } fx.x, \epsilon)\}) / \epsilon \\ &\rightsquigarrow (\star, \epsilon) / \epsilon \end{aligned}$$

We could reason syntactically like Sands does in [10], but the presence of a recursion operator makes the proof of the adequacy theorem quite involved (one would need some kind of unwinding theorem, like in [8]). We will rather build a denotational model, where recursion is denoted by a fixpoint, which will make proofs much easier.

3.2 Naive Denotational Semantics

The natural idea is to model the base type by $\llbracket 1 \rrbracket = N_{\perp}$, meaning that either the program diverges (\perp), or it terminates in $n \in N$ steps. We write \underline{n} the injection of $n \in N$ in N_{\perp} . Arrow types are interpreted by $\llbracket A \rightarrow B \rrbracket = \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket$. and $\Gamma = x_1 : A_1 \dots x_n : A_n$ by $\llbracket \Gamma \rrbracket = \llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket$.

We define for each type A an operation $\text{tick}_A : \llbracket A \rrbracket \Rightarrow \llbracket A \rrbracket$. This will be used to interpret the tick operation of the syntactical translation we defined above.

$$\text{tick}_1 = \lambda x. \begin{cases} \underline{n+1} & \text{if } x = \underline{n} \\ \perp & \text{if } x = \perp \end{cases} \quad \text{tick}_{A \rightarrow B} = \lambda f. \lambda a. \text{tick}_B(fa)$$

Note that $(\text{tick}_{A \rightarrow B} f)x = \text{tick}_B(fx)$. It is easy to check that tick_A is continuous and mono. The denotation of a typing derivation ending with $\Gamma \vdash M : A$ is a map from $\llbracket \Gamma \rrbracket$ to $\llbracket A \rrbracket$:

- $\llbracket \Gamma \vdash \star : 1 \rrbracket \rho = \underline{0}$,
- $\llbracket x_1 : A_1 \dots x_n : A_n \vdash x_i : A_i \rrbracket \rho = \text{tick}_A(\pi_i \rho)$,
- $\llbracket \Gamma \vdash (M)N : B \rrbracket \rho = \text{tick}_B(\text{tick}_B(\llbracket \Gamma \vdash M : A \rightarrow B \rrbracket \rho)(\llbracket \Gamma \vdash N : A \rrbracket \rho))$,
- $\llbracket \Gamma \vdash \text{rec } f x.M : A \rightarrow B \rrbracket = \bigsqcup \varphi_n$ where

$$\begin{cases} \varphi_0 = \perp \\ \varphi_{n+1} = \lambda \rho. \lambda \xi. \llbracket \Gamma, f : A \rightarrow B, x : A \vdash M : B \rrbracket \langle \langle \rho, \varphi_n \rho \rangle, \xi \rangle \end{cases}$$

And finally, the continuation are denoted by the list of denotations of their elements and the denotation of a state is the plain application:

$$\llbracket C_1 \dots C_n \cdot 1 \rrbracket = (\llbracket C_1 \rrbracket, \dots, \llbracket C_n \rrbracket) \quad \llbracket C \rrbracket S = \llbracket C \rrbracket \llbracket S \rrbracket$$

This gives a sound and adequate model: a state of type 1 terminates in n steps if and only if its denotation is \underline{n} . This is nevertheless not a good model, because many elements are not definable: “cheating” is allowed, in the sense that an element can ask for something that takes a long time to compute, and pretend that the computation was short. For instance, define

$$\varphi = \lambda x. \text{if } x \neq \perp \text{ then } \underline{0} \text{ else } \perp$$

φ is a continuous function from $\llbracket 1 \rrbracket$ to $\llbracket 1 \rrbracket$, and $\varphi \mathfrak{3} = \underline{0}$. φ cannot be definable, because a term has to let the machine evaluate its argument before it can use it, and the reduction steps taken by this evaluation cannot be removed from the global reduction sequence, as φ does.

To get rid of this kind of behavior, we could constrain the morphisms, like Escardo does in [3], but this relies heavily on the inside structure of the category (ultrametric spaces). We rather hide the length of the computation of the arguments from the denotation of the function: this way, the function can only give the number of steps needed by its own evaluation. Adding the time taken by the arguments to the total time is then “hardwired” in the category.

We achieve this using a monad: $T\llbracket A \rrbracket$ is the object for computation of type A , that is a value in $\llbracket A \rrbracket$ together with a length of computation. The denotation of $\Gamma \vdash M : A$ is a morphism from $\llbracket \Gamma \rrbracket$ to $T\llbracket A \rrbracket$: given the *values* of the variables in the environment, it gives a *computation* of type A . We compose them in the Kleisli category:

$$\begin{cases} A \xrightarrow{f} TB \\ B \xrightarrow{g} TC \end{cases} \text{ gives } A \xrightarrow{f} TB \xrightarrow{Tg} T(TC) \xrightarrow{\mu_C} TC$$

Adding the time taken by f and g is done by μ , which collapses the timing information.

We will see that the only difficult part of building a model is actually defining the monad: the definitions of the denotations in the Kleisli category are pretty straightforward then. The next two sections are dedicated to the cases of call-by-value and call-by-name reduction.

3.3 Notations

First, \mathcal{C} will always be a cartesian closed category. Identities are written id_A , composition \circ , product \times , pairing $\langle \cdot, \cdot \rangle$, arrow \Rightarrow , curry transformation C and uncurry U . We write $\text{Ev}_{A,B} = U(\text{id}_{A \Rightarrow B}) : (A \Rightarrow B) \times A \rightarrow B$, the evaluation morphism. $!_A$ is the only morphism from A to 1 .

For definitions about monads, we take those of Moggi [7]. We will consider computational cartesian models over sub cartesian closed categories of CPO, *i.e.* a monad T, η, μ over \mathcal{C} with a tensorial strength $t_{X,Y} : X \times TY \rightarrow T(X \times Y)$. The Kleisli star is defined by $f^* = \mu \circ Tf$, so the composition in the Kleisli category is $f \bullet g = f^* \circ g$. We also define $\bar{t} : TX \times Y \rightarrow T(X \times Y)$ to be the morphism dual to t .

4 Semantics For Call-By-Value

In this section, we will not use the Kleisli category to avoid confusion: everything will be written in the original cartesian closed category.

4.1 Categorical Presentation

We require that \mathcal{C} is a sub category of CPO, with a computational model T, η, μ, t . We also need an object U such that $\mathcal{C}[1, TU]$ is the flat CPO N_\perp . To implement the tick operation, we require a natural transformation $\text{tick} : \text{id}_{\mathcal{C}} \rightarrow T$ such that:

$$\begin{aligned} \text{tick}_U^* \circ \underline{n} &= \underline{n+1} \\ \text{tick}_U^* \circ \perp &= \perp \end{aligned} \tag{1}$$

Naturality of tick means that the exact location of tick in the term does not matter: it is added to the global number of steps all the same.

These elements allow to define denotations of CBV machine parts. First, we define the objects for types:

- $\llbracket 1 \rrbracket = 0$ in N_\perp ,
- $\llbracket A \rightarrow B \rrbracket = \llbracket A \rrbracket \Rightarrow T\llbracket B \rrbracket$,
- $\llbracket \Gamma \rrbracket = (\dots (1 \times \llbracket A_1 \rrbracket) \times \dots) \times \llbracket A_n \rrbracket$ if $\Gamma = x_1 : A_1, \dots, x_n : A_n$.

Values $V : A$ are denoted by morphisms $\llbracket V : A \rrbracket^V : 1 \rightarrow \llbracket A \rrbracket$, terms $\Gamma \vdash M : A$ by $\llbracket \Gamma \vdash M : A \rrbracket : \llbracket \Gamma \rrbracket \rightarrow T\llbracket A \rrbracket$ and continuations $K : A^\perp$ by $\llbracket K \rrbracket : \llbracket A \rrbracket \rightarrow TU$. Computations of type 1 are arrows from 1 to TU : \perp means that the computation

diverges, \underline{n} means that the computation terminates in n steps. The complete denotation of a state is the composition of the three parts:

$$\llbracket (M, E) / K : 1 \rrbracket = \llbracket K : A^\perp \rrbracket^* \circ \llbracket \Gamma \vdash M : A \rrbracket \circ \llbracket E : \Gamma \rrbracket$$

We define denotations for values, environments and terms inductively on the syntax (the base case being the empty environment). For values:

- $\llbracket \star : 1 \rrbracket^V = \star$,
- $\llbracket \text{Clos } fx.(M, E) : A \Rightarrow B \rrbracket^V = \bigsqcup \varphi_n \circ \llbracket E : \Gamma \rrbracket$ where

$$\begin{cases} \varphi_0 = \perp \\ \varphi_{n+1} = C[\llbracket \Gamma, f : A \Rightarrow B, x : A \vdash M : B \rrbracket \circ \langle \text{id}_{\llbracket \Gamma \rrbracket}, \varphi_n \rangle] \end{cases}$$

For environments, if $\Gamma = x_1 : A_1, \dots, x_n : A_n$:

$$\llbracket \{x_1 \rightarrow V_1 \dots x_n \rightarrow V_n\} : \Gamma \rrbracket = \langle \dots \langle 1, \llbracket V_1 : A_1 \rrbracket^V \rangle, \dots \rangle, \llbracket V_n : A_n \rrbracket^V \rangle$$

For terms, we add tick morphisms where they are needed:

$$\begin{aligned} \llbracket \Gamma \vdash \text{Val } V \rrbracket &= \eta_{\llbracket A \rrbracket} \circ \llbracket V \rrbracket^V \circ ! \\ \llbracket x_1 : A_1, \dots, x_n : A_n \vdash x_i : A_i \rrbracket &= \text{tick}_{\llbracket A_i \rrbracket} \circ \pi_i \\ \llbracket \Gamma \vdash \text{rec } fx.M : A \rightarrow B \rrbracket &= \text{tick}_{\llbracket A \Rightarrow B \rrbracket} \circ \bigsqcup \varphi_n \quad \text{where } \varphi_n \text{ is defined as above} \\ \llbracket \Gamma \vdash (M)N : B \rrbracket &= \text{tick}^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{Ev}^* \circ \bar{t}^* \circ t \circ \langle \llbracket \Gamma \vdash M : A \rightarrow B \rrbracket, \llbracket \Gamma \vdash N : A \rrbracket \rangle \end{aligned}$$

We complete the definition of the semantics with the denotations for continuations, again putting ticks to match the reduction:

- $\llbracket \epsilon : 1^\perp \rrbracket = \underline{0} \circ !$,
- $\llbracket [\] V \cdot K : (A \Rightarrow B)^\perp \rrbracket = \llbracket K \rrbracket^* \circ \text{tick}^* \circ \text{Ev} \circ \langle \text{id}_{\llbracket A \Rightarrow B \rrbracket}, \llbracket V : A \rrbracket \circ ! \rangle$
- $\llbracket (M, E) [\] \cdot K : A^\perp \rrbracket = \llbracket K \rrbracket^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{Ev}^* \circ \bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ !, \text{id}_{\llbracket A \rrbracket} \rangle$

The proofs require some care because the Kleisli category is not cartesian. We will only use the following result, which says that we can take a computation out of a pair if it is used only by one side:

Lemma 3. *For all $f : 1 \rightarrow A$, $g : A \Rightarrow TB$, $t \circ \langle f \circ !, g \rangle = T \langle f \circ !, \text{id}_B \rangle \circ g$.*

Proof. By naturality of t , $t \circ \langle f \circ !, g \rangle = T(f \times \text{id}_B) \circ t \circ r^{-1} \circ g$, where r is the isomorphism $r_X : 1 \times X \rightarrow X$. By definition of a tensorial strength, $Tr_B \circ t = r_{TB}$, so $t \circ r_{TB}^{-1} = T(r_B^{-1})$, which gives the desired equality.

Lemma 4. *For all $(M, E) / K : 1$, if $(M, E) / K \rightsquigarrow_V (M', E') / K'$, then $\llbracket (M, E) / K \rrbracket = \text{tick} \bullet \llbracket (M', E') / K' \rrbracket$.*

Proof. We only give the proof for the rule $((M)N, E) / K \rightsquigarrow_V (N, E) / (M, E) [\] \cdot K$, the other cases are similar. By definition:

$$\llbracket ((M)N, E) / K \rrbracket = \llbracket K \rrbracket^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{Ev}^* \circ \bar{t}^* \circ t \circ \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \circ \llbracket E \rrbracket$$

Axiom 1 gives that $\llbracket K \rrbracket^* \circ \text{tick}^* = (\llbracket K \rrbracket^* \circ \text{tick})^* = (\text{tick}^* \circ \llbracket K \rrbracket)^* = \text{tick}^* \circ \llbracket K \rrbracket^*$.
By lemma 3:

$$t \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket, \llbracket N \rrbracket \circ \llbracket E \rrbracket \rangle = T \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ !, \text{id}_{\llbracket A \rrbracket} \rangle \circ \llbracket N \rrbracket \circ \llbracket E \rrbracket$$

Thanks to the properties of the monad:

$$\bar{t}^* \circ T \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ !, \text{id} \rangle = (\bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ !, \text{id} \rangle)^*$$

Putting the pieces together, we get

$$\llbracket ((M)N, E) / K \rrbracket = \text{tick}^* \circ (\llbracket K \rrbracket^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{Ev}^* \circ \bar{t} \circ \langle \llbracket M \rrbracket \llbracket E \rrbracket \circ !, \text{id} \rangle)^* \circ \llbracket N \rrbracket \circ \llbracket E \rrbracket$$

as expected.

Proposition 3 (Soundness). *If $(M, E) / K \Downarrow_V n$, then $\llbracket (M, E) / K \rrbracket = \underline{n}$.*

Proof. Easy corollary of the previous lemma, using the properties of tick.

Proposition 4 (Adequacy). *If $\llbracket (M, E) / K \rrbracket = \underline{n}$, then $(M, E) / K \Downarrow_V n$.*

Proof. We define a heterogeneous relation, relating states $(M, E) / K$ and morphisms $f : 1 \rightarrow TU$ such that $f = \underline{n}$ entails $(M, E) / K \Downarrow_V n$. We extend them to relations between terms and their denotations, and prove a fundamental lemma. Adequacy is then an easy corollary. The details are given in appendix.

4.2 The CPO Model

We apply this directly to the category CPO. The monad is $TX = (N \times X)_\perp$. We write $[\cdot]$ the injection of $N \times X$ in $(N \times X)_\perp$. The natural transformations are:

$$\begin{aligned} \eta_X &= \lambda x. [(0, x)] \\ \mu_X &= \lambda d. \begin{cases} [(n_1 + n_2, x)] & \text{if } d = [(n_1, [(n_2, x)])] \\ \perp & \text{otherwise} \end{cases} \\ t_{X,Y} &= \lambda(x, d). \begin{cases} [(n, (x, y))] & \text{if } d = [(n, y)] \\ \perp & \text{otherwise} \end{cases} \end{aligned}$$

We define $U = \{*\}$, $\underline{n} = [(n, *)]$, and $\text{tick}_X = \lambda v. [(1, v)] : X \rightarrow TX$. As $TU = N_\perp$, $\mathcal{C}[1, TU] \approx N_\perp$. It is easy to check that tick is natural and satisfy the properties required in 1.

We can see that modelling call-by-value reduction is quite easy, with minimal categorical requirements: a computational model over a cartesian closed category and a natural transformation $\text{tick} : 1 \rightarrow T$. The definitions themselves require some care, because they mix composition in the Kleisli category (between the term and the continuation) with traditional composition (between the environment and the term), but the concrete CPO model is very straightforward.

We show in the next section that the call-by-name reduction, can eventually be described in this monadic setting in a more elegant way, thanks to the stronger requirements, which in counterpart make the models much harder to build.

5 Semantics of Call-By-Name

In call-by-name, the formula for monoidal structure ϕ , which computes the time taken by the environment from the time taken by each part, is not so simple: if A takes n_A steps and is used α times, and B takes n_B steps and is used β times, then the total time is $\alpha n_A + \beta n_B$. We need to switch to a category where the number of times each argument is used is relevant: we give the example of coherence spaces with the multiset exponential.

In CBN, all the denotations are morphisms of the Kleisli category, therefore we will write them using the Kleisli category operators. We write $\bar{t}_{X,Y} = T_{c_{Y,X}} \circ t \circ c_{TX,Y} : TX \times Y \rightarrow T(X \times Y)$, where $c_{A,B} : A \times B \rightarrow B \times A$ is the natural isomorphism of the cartesian structure of \mathcal{C} . We use it to define a monoidal structure over the Kleisli category: $\phi_{X,Y} = \bar{t}_{X,Y}^* \circ t_{TX,Y} : TX \times TY \rightarrow T(X \times Y)$

5.1 Categorical Presentation

We start with the same basic requirements as those for CBV: we need a sub cartesian closed category of CPO \mathcal{C} and a computational monad T, η, μ, ϕ over \mathcal{C} . We need an object U such that that $1 \rightarrow TU$ is the flat CPO $\{\underline{n} \mid n \in \mathbb{N}\}_\perp$ for a suitable family \underline{n} , and a morphism $\text{tick}_1 : U \rightarrow TU$, such that:

$$\begin{cases} \text{tick}_1 \bullet \underline{n} = \underline{n+1} \\ \text{tick}_1 \bullet \perp = \perp \end{cases} \quad (2)$$

This time, we do not require a morphism tick_X for every object X : the intended property was that tick commutes with continuations. In CBN, contexts are only applicative, so tick only need to commute with application: it is much easier to *define* it for higher types and prove the intended property than in CBV. Moreover, the concrete model we give as an example is much easier to define this way.

In CBN, the notion of value is not relevant, so every denotation is a morphism in the Kleisli category ($\llbracket E : \Gamma \rrbracket : 1 \rightarrow T[\Gamma]$ and $\llbracket \Gamma \vdash M : A \rrbracket : [\Gamma] \rightarrow T[A]$). In order to have everything work smoothly, we need that the cartesian structure of \mathcal{C} induces a cartesian structure on the Kleisli category:

$$\langle f, g \rangle = \bar{t}^* \circ t \circ \langle f, g \rangle \text{ defines a cartesian structure in the Kleisli category} \quad (3)$$

In our model of call-by-value, this only induced a monoidal structure: copying (resp. deleting) meant evaluating several times (resp. nothing instead of something). As in CBN, the environment is computed only when it is needed, copying and deleting should be harmless.

Unlike in the CBV case, we cannot use the curry/uncurry transformations of \mathcal{C} : they swap $\Gamma \times A \rightarrow TB$ and $\Gamma \rightarrow A \Rightarrow TB$, whereas we want to swap between $\Gamma \times A \rightarrow TB$ and $\Gamma \rightarrow T(A \Rightarrow B)$. So, we require that:

$$\text{The Kleisli category must be cartesian closed.} \quad (4)$$

This might seem a very strong requirement, but we give in the next subsection a generic monad that makes the Kleisli category cartesian closed. The difficulty of building a model then resides in finding a category in which there is a suitable tick morphism.

Starting from these axioms, we can build a sound and adequate model for the call-by-name λ -calculus. In the remaining of the section, we work in the Kleisli category. The operator are written in boldface:

- $\mathbf{id}_A = \eta_A$,
- $f \bullet g = f^* \circ g = \mu \circ Tf \circ g$,
- $\langle f, g \rangle = \bar{t}^* \circ t \circ \langle f, g \rangle$,
- \mathbf{C} and \mathbf{U} are the curry and uncurry transformations,
- $\mathbf{Ev} = \mathbf{U}\mathbf{id}_{A \Rightarrow B}$ is the application morphism,

We first define the domains for types:

- $\llbracket 1 \rrbracket = U$,
- $\llbracket A \rightarrow B \rrbracket = \llbracket A \rrbracket \Rightarrow \llbracket B \rrbracket$.
- $\llbracket \Gamma \rrbracket = (\dots(1 \times A_1) \times A_2 \dots A_{n-1}) \times A_n$ if $\Gamma = x_1 : A_1, \dots, x_n : A_n$.

and we extend tick to all types so that $\mathbf{Ev} \bullet \langle \text{tick}_{A \Rightarrow B} \bullet f, x \rangle = \text{tick}_B \bullet \mathbf{Ev} \bullet \langle f, x \rangle$:

$$\text{tick}_{A \Rightarrow B} = \llbracket A \rrbracket \Rightarrow \text{tick}_B = \mathbf{C}(\text{tick}_B \bullet \mathbf{Ev})$$

The denotation of $\Gamma \vdash M : A$ is a morphism from $\llbracket \Gamma \rrbracket$ to $\llbracket A \rrbracket$ in the Kleisli category. Since, it is cartesian closed the denotations look very familiar:

- $\llbracket \Gamma \vdash \star : 1 \rrbracket = \mathbf{Q} \circ !$,
- $\llbracket x_1 : A_1 \dots x_n : A_n \vdash x_i : A_i \rrbracket = \text{tick}_A \circ \pi_i$,
- $\llbracket \Gamma \vdash (M)N : B \rrbracket = \text{tick}_B \bullet \mathbf{Ev} \bullet \langle \llbracket \Gamma \vdash M : A \rightarrow B \rrbracket, \llbracket \Gamma \vdash N : A \rrbracket \rangle$,
- $\llbracket \Gamma \vdash \text{rec } f.x.M : A \rightarrow B \rrbracket = \bigsqcup \varphi_n$ where

$$\begin{cases} \varphi_0 = \perp \circ ! \\ \varphi_{n+1} = \mathbf{C}(\llbracket \Gamma, f : A \rightarrow B, x : A \vdash M : B \rrbracket) \bullet \langle \mathbf{id}_\Gamma, \varphi_n \rangle \end{cases}$$

Denotations for closure and environments are defined inductively (the base case being the empty environment, denoted by \mathbf{id}_1). If $\Gamma = x_1 : A_1 \dots x_n : A_n$,

$$\begin{aligned} \llbracket \{x_1 \rightarrow C_1, \dots, x_n \rightarrow C_n\} : \Gamma \rrbracket &= \langle \langle \dots \langle \mathbf{id}_1, \llbracket C_1 : A_1 \rrbracket \rangle, \dots \rangle, \llbracket C_n : A_n \rrbracket \rangle \\ \llbracket (M, E) : A \rrbracket &= \llbracket \Gamma \vdash M : A \rrbracket \bullet \llbracket E : \Gamma \rrbracket \end{aligned}$$

In call-by-name, continuations are simple stacks:

$$\llbracket (A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1)^\perp \rrbracket = \llbracket A_1 \rrbracket \times \dots \times \llbracket A_n \rrbracket$$

$$\llbracket (C_1, \dots, C_n) : (A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1)^\perp \rrbracket = (\llbracket C_n : A_n \rrbracket, \dots, \llbracket C_1 : A_1 \rrbracket)$$

We conclude the definitions with the denotation of a complete state by $\llbracket C/S \rrbracket = \llbracket C \rrbracket / \llbracket S \rrbracket$ where

$$\begin{cases} c/\epsilon = c \\ c/c' \cdot \sigma = \text{tick} \bullet ((\mathbf{Ev} \bullet \langle c, c' \rangle) / \sigma) \end{cases}$$

Since we work in the Kleisli category, which is a CCC, the proofs are much more concise than those for the CBV interpretation.

Lemma 5. *If $C/S \rightsquigarrow C'/S'$, then $\llbracket C/S \rrbracket = \text{tick}_1 \llbracket C'/S' \rrbracket$.*

Proof. We only give the proof for the (pop) rule, see the appendix the complete proof. Assume $C = (\text{rec } fx.M, E)$, $S = C_x \cdot S'$, and $C' = (M, E[f \rightarrow C, x \rightarrow C_x])$. By definition, $\llbracket C \rrbracket = \bigsqcup \varphi_n \bullet \llbracket E \rrbracket$ where $\varphi_{n+1} = \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \text{id}, \varphi_n \rangle$, and $\varphi_0 = \perp$, so

$$\begin{aligned} \llbracket C \rrbracket &= \bigsqcup \varphi_n \bullet \llbracket E \rrbracket \\ &= \bigsqcup \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \text{id}, \varphi_n \rangle \bullet \llbracket E \rrbracket \\ &= \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \bigsqcup \varphi_n \bullet \llbracket E \rrbracket \rangle \\ &= \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle \end{aligned}$$

Since the Kleisli category is cartesian closed, we get:

$$\begin{aligned} \llbracket C/S \rrbracket &= (\mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle) / \llbracket C_x \rrbracket \cdot \llbracket S' \rrbracket \\ &= \text{tick} \bullet (\mathbf{E}v \bullet \langle \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle, \llbracket C_x \rrbracket \rangle / \llbracket S' \rrbracket) \\ &= \text{tick} \bullet (\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle, \llbracket C_x \rrbracket \rangle / \llbracket S' \rrbracket) \\ &= \text{tick} \bullet (\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \llbracket E[f \rightarrow C, x \rightarrow C_x] \rrbracket / \llbracket S' \rrbracket) \end{aligned}$$

Just as in the CBV case, this lemma gives:

Proposition 5 (Soundness). *For all $C/S : 1$, if $C/S \Downarrow_N n$, then $\llbracket C/S \rrbracket = \underline{n}$.*

Proposition 6 (Adequacy). *If $\llbracket C/S \rrbracket = \underline{n}$, then $C/S \Downarrow_n$.*

Proof. Again, we define heterogeneous relations: see the appendix.

5.2 Time as a resource

Rather than trying to fit the monad $(N \times _)\perp$ in this setting, we give another, more appropriate monad: $T \Rightarrow _$. The idea is that denotation take an extra argument T , and that each tick will use its argument in T . More formally, we start with a cartesian closed category \mathcal{C} with a distinguished object T , and we define:

$$\begin{aligned} T &= \lambda f : A \Rightarrow B. \lambda \varphi : T \Rightarrow A. \lambda t : T. f(\varphi t) \\ \eta_X &= \lambda x : X. \lambda t : T. a \\ \mu_X &= \lambda f : T \Rightarrow T \Rightarrow X. \lambda t : T. ftt \\ \phi_{X,Y} &= \lambda(f, g). \lambda t. (ft, gt) \\ \mathbf{C} &= \lambda f : X \times Y \Rightarrow T \Rightarrow Z. \lambda x : X. \lambda t : T. \lambda y : Y. f(x, y)t \\ \mathbf{U} &= \lambda f : X \Rightarrow T \Rightarrow Y \Rightarrow Z. \lambda(x, y) : X \times Y. \lambda t : T. fxyt \end{aligned}$$

The proof of the following proposition is straightforward, though a bit long:

Proposition 7. *$T \Rightarrow _$ is a monad, and the Kleisli category is cartesian closed.*

The only problem is now to find a category with a tick morphism. We need a cartesian closed category where the number of times each argument is used is recorded: we can then count in the end how many ticks were consumed by the computation.

5.3 A Coherence Model

We apply this monad to the category of coherence spaces with the multiset exponential. We first recall the definitions about coherence (see [4], [2] for more details). A coherence space $X = (|X|, \circ_X)$ is a set X endowed with a symmetric, reflexive relation \circ_X (the coherence). The strict coherence \wedge_X is defined by $x \wedge_X x' \iff x \circ_X x' \wedge x \neq x'$.

A clique of a coherence space X is a subset s of $|X|$, such that $\forall x, x' \in s, x \circ_X x'$. A multiclique is a multiset m of X , such that the set of its elements, written $|m|$ is a clique of X . We write \uplus the union of multisets.

The product coherence space $X \times Y$ is defined by $|X \times Y| = |X| + |Y|$ (the disjoint union) and two elements are coherent either if one is in $|X|$ and one in $|Y|$, or both are in $|X|$ (resp. $|Y|$) and they are coherent in X (resp. in Y). The intended property is that *a multiclique of $X \times Y$ is a pair of a multiclique of X and a multiclique of Y* . The linear arrow coherence space $X \multimap Y$ is defined by $|X \multimap Y| = |X| \times |Y|$ and $(x, y) \circ_{X \multimap Y} (x', y')$ if

$$\begin{cases} x \circ_X x' \implies y \circ_Y y' \\ x \wedge_X x' \implies y \wedge_Y y' \end{cases}$$

The multiset exponential coherence space $!X$ is defined as the set of *finite* multisets of X , two multisets being coherent iff their union is a multiclique. The co-Kleisli category of the comonad $!$ is a cartesian closed category: the intuitionistic arrow $X \Rightarrow Y$ is defined to be $!X \multimap Y$.

So, the coherence space $X \Rightarrow Y$ is: $|X \Rightarrow Y|$ is the set of pairs of a multiclique of X and an element of Y , and $(m, y) \circ_{X \Rightarrow Y} (m', y')$ if whenever $m \cup m'$ is a multiclique, then $y \circ_Y y'$, and whenever $m \cup m'$ is a multiclique and $m \neq m'$, then $y \wedge_Y y'$.

We recall the composition a clique $f : X \Rightarrow Y$ with a clique $g : Y \Rightarrow Z$: $g \circ f$ is the set of $(m_X, z) \in |X \Rightarrow Z|$ such that:

- there exists $m_Y = \{y_1, \dots, y_n\}$ such that $(m_Y, z) \in g$,
- there exist $m_X^1 \dots m_X^n$ such that $\forall i, (m_X^i, y_i) \in f$ and $m_X = m_X^1 \uplus \dots \uplus m_X^n$.

To interpret the monad, we choose $|T| = \{*\}$: each tick will “consume” one $*$ in T . The finite multisets of T are the multisets $n = \{*, \dots, *\}$ with n occurrences of $*$, for $n \in \mathbb{N}$. The monadic transformations are, translating the above definitions:

$$\begin{aligned} \eta_A &= \{(\{a\}, (\emptyset, a)) \mid a \in |A|\} \\ \mu_A &= \{(\{(m_1, (m_2, a))\}, (m_1 \uplus m_2, a)) \mid m_1, m_2 \in |!T|, a \in |A|\} \\ \mathbf{C}f &= \{(g, (t, (a, b))) \mid ((g, a), (t, b)) \in f\} \\ \mathbf{U}f &= \{((g, a), (t, b)) \mid (g, (t, (a, b))) \in f\} \end{aligned}$$

We can see that μ sums up the numbers in each T . The morphism which gives the cartesian structure is the clique of these two kinds of elements:

$$\begin{array}{ccc} (T \Rightarrow X) \times (T \Rightarrow Y) & \xrightarrow{\phi_{X,Y}} & T \Rightarrow (X \times Y) \\ \{m \quad , \quad x\} & & m \quad x \\ & & \{m \quad , \quad y\} \quad m \quad y \end{array}$$

We show how to compute the Kleisli star. The element $(\{b_1 \dots b_k\}, (n, c))$ in f is transformed by:

$$\begin{array}{ccc}
B & \xrightarrow{f} & T \Rightarrow C \\
\{b_1 & & \\
\vdots & n & c \\
b_k\} & & \\
\end{array}
\qquad
\begin{array}{ccc}
T & \Rightarrow & B \xrightarrow{f^*} T \Rightarrow C \\
\{(n_1 & b_1) \\
\vdots & \\
(n_k & b_k)\} \\
\end{array}$$

where $n_1 \dots n_k$ are all possible multicliques of T such that $\{(n_1, b_1), \dots, (n_k, b_k)\}$ is a multiclique of $T \Rightarrow B$. We see that each b_i requires some time n_i , and that the total time is the sum of n (the time to compute c from the b_i) and all the n_i . It is easy to show that $\phi \circ \langle f^*, g^* \rangle = (\phi \circ \langle f, g \rangle)^*$, so $\langle f, g \rangle \bullet x = \langle f \bullet x, g \bullet x \rangle$.

We now give the remaining details of the model. The coherence space U is also the singleton $\{*\}$ (it is the same as T , but we give them different names to avoid confusion). $\underline{n} = \{(n, *)\} : T \Rightarrow U$: it needs n times its argument in T to produce the result in U . $\text{tick}_1 = \{(\{*\}, (1, *))\}$: it takes 1 tick in T to forward the value $*$ in U . Note that this only works at base type, and this is why we *defined* tick for arrow types, rather than try to find a natural transformation from id_C to T .

For instance, let x be the computation taking 3 steps to return $*$, and f be a map from 1 to $T(U \Rightarrow U) = T \Rightarrow U \Rightarrow U$ using its argument twice before returning in 4 steps:

$$\begin{array}{ccc}
1 & \xrightarrow{x} & T \Rightarrow U \\
\emptyset & \underline{3} & * \\
\end{array}
\qquad
\begin{array}{ccc}
1 & \xrightarrow{f} & T \Rightarrow U \Rightarrow U \\
\emptyset & \underline{4} & \{*, *\} \quad * \\
\end{array}$$

When we compose them, we see, that this application computes $*$ in $10=4+3+3$ steps, as expected:

$$\begin{array}{ccc}
1 & \xrightarrow{x} & T \Rightarrow U \xrightarrow{(Uf)^*} T \Rightarrow U \\
\emptyset \uplus \emptyset & \{(\underline{3} & *) \\
& (\underline{3} & *)\} \\
& & \underline{4} \uplus \underline{3} \uplus \underline{3} \quad * \\
\end{array}$$

6 Conclusions and Further Work

We have shown that the length of the computation can be denoted in a concise and compositional way through the use of monads, both for quite realistic implementations of call-by-name and call-by-value. The set of requirements on the cartesian closed category is rather small. We also applied these constructions to concrete categories, namely complete partial orders with the straightforward $(N \times _)\perp$ monad and coherence spaces with the $T \Rightarrow _$ monad.

Comparing the two cases is surprising: although the call-by-value case seemed much easier at first, the definitions and the proofs are trickier than those for call-by-name, because the Kleisli category is not cartesian in that case. The closure property required for the interpretation of call-by-name seemed quite hard, but the monad $T \Rightarrow _$ tackles this problem easily. The models, on the other hand, have the expected complexity: the call-by-value CPO is pleasingly short, and the model for call-by-name reduction is more involved, because we do not work in a well-pointed category.

This work could be extended in many ways. First, we could apply these constructions to other models: game semantics, with their dynamic feeling, should be particularly well-suited for that purpose. Another track would be to model more realistic execution than the simple environment machines presented here, such as “assembly-like” languages rather than λ -calculus. One could also investigate the call-by-need case: the set exponential coherence model (where we record if a parameter is used, but not the number of times it is used) should be relevant there. A third possibility is the study of languages that integrate the notion of time such as chattering PCF of [3].

The links with linear logic should also be investigated, both for the case of call-by-value with its monoidal structure, and call-by-name, because the number of times a term uses its variable is relevant. This leads naturally to the question: what about complexity? Polytime computation (for instance [5], [1]) must have links with this work, that still wait to be unveiled.

References

1. Patrick Baillot and Virgile Mogbil. Soft lambda-calculus: A language for polynomial time computation. In *FOSSACS*. LNCS, 2004.
2. Nuno Barreiro and Thomas Ehrhard. Anatomy of an extensional collapse. 1997.
3. Martin H. Escardó. A metric model of PCF. Laboratory for Foundations of Computer Science, University of Edinburgh. Unpublished research note, 1998.
4. Jean-Yves Girard, Yves Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science 7. Cambridge University Press, 1988.
5. M. Hofmann. Linear types and non-size-increasing polynomial time computation. In *LICS'99*, pages 464–473. IEEE, 1999.
6. Jean-Louis Krivine. A call-by-name lambda calculus machine. To appear in *Higher Order and Symbolic Computation*.
7. Eugenio Moggi. Notions of computation and monads. *Information and Computation*, 93:55–92, 1991.
8. Andrew Pitts and Ian Stark. Operational reasoning for functions with local state. In *Higher-Order Operational Techniques in Semantics*. Cambridge University Press, 1998.
9. G. D. Plotkin. Call-by-name, call-by-value and the lambda calculus. *Theoretical Computer Science*, 1:125–159, 1975.
10. David Sands. A naïve time analysis and its theory of cost equivalence. *Journal of Logic and Computation*, 5(4), 1995.

7 Appendix: Proofs

7.1 Soundness for Call By Value

Lemma 6. For all $(M, E) / K : 1$, if $(M, E) / K \rightsquigarrow_V (M', E') / K'$, then $\llbracket (M, E) / K \rrbracket = \text{tick} \bullet \llbracket (M', E') / K' \rrbracket$.

Proof. The proof relies on the naturality of tick and

$$t \circ \langle f, g \rangle = T \langle f \circ!, \text{id}_A \rangle \circ g \quad \text{and} \quad \bar{t} \circ \langle f, g \rangle = T \langle \text{id}_A, g \circ! \rangle \circ f$$

which are proved thanks to the naturality of t and axiom 3. We consider each reduction rule.

– $(x, E) / K \rightsquigarrow_V (E(x), E) / K$. By naturality of tick and definition of $\llbracket E \rrbracket$:

$$\begin{aligned} \llbracket (x, E) / K \rrbracket &= \llbracket K \rrbracket^* \circ \text{tick} \circ \pi_i \circ \llbracket E \rrbracket \\ &= \text{tick} \circ \llbracket K \rrbracket \circ \llbracket E(x) \rrbracket^V \\ &= \text{tick} \circ \llbracket K \rrbracket^* \circ \eta \circ \llbracket E(x) \rrbracket^V \circ! \circ \llbracket E \rrbracket \\ &= \text{tick} \circ \llbracket K \rrbracket^* \circ [\Gamma \vdash E(x) : A] \circ \llbracket E \rrbracket \end{aligned}$$

– $((M)N, E) / K \rightsquigarrow_V (N, E) / (M, E)[\] \cdot K$, By definition:

$$\llbracket ((M)N, E) / K \rrbracket = \llbracket K \rrbracket^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{Ev}^* \circ \bar{t}^* \circ t \circ \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \circ \llbracket E \rrbracket$$

Axiom 1 gives that $\llbracket K \rrbracket^* \circ \text{tick}^* \circ \text{tick}^* \circ \text{tick}^* = \text{tick}^* \circ \text{tick}^* \circ \text{tick}^* \circ \llbracket K \rrbracket^*$. By the previous result:

$$t \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket, \llbracket N \rrbracket \circ \llbracket E \rrbracket \rangle = T \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ!, \text{id}_{[A]} \rangle \circ \llbracket N \rrbracket \circ \llbracket E \rrbracket$$

Thanks to the properties of the monad:

$$\bar{t}^* \circ T \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ!, \text{id} \rangle = (\bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ!, \text{id} \rangle)^*$$

Putting the pieces together:

$$\llbracket ((M)N, E) / K \rrbracket = \text{tick}^* \circ (\text{tick}^* \circ \text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev}^* \circ \bar{t} \circ \langle \llbracket M \rrbracket \llbracket E \rrbracket \circ!, \text{id} \rangle)^* \circ \llbracket N \rrbracket \circ \llbracket E \rrbracket$$

as expected.

– $(V, E') / (M, E)[\] \cdot K \rightsquigarrow_V (M, E) / [\]V \cdot K$. By definition:

$$\begin{aligned} \llbracket (V, E) / (M, E')[\] \cdot K \rrbracket &= (\text{tick}^* \circ \text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev}^* \circ \bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ!, \text{id}_A \rangle)^* \circ \eta \circ \llbracket V \rrbracket^V \\ &= \text{tick}^* \circ \text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev}^* \circ \bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket \circ!, \text{id}_A \rangle \circ \llbracket V \rrbracket^V \\ &= \text{tick}^* \circ \text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev}^* \circ \bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket, \llbracket V \rrbracket^V \rangle \end{aligned}$$

By axiom 3:

$$\begin{aligned} \bar{t} \circ \langle \llbracket M \rrbracket \circ \llbracket E \rrbracket, \llbracket V \rrbracket^V \rangle &= \bar{t} \circ \langle \text{id}_{T[A \Rightarrow B]}, \llbracket V \rrbracket^V \circ! \rangle \circ \llbracket M \rrbracket \circ \llbracket E \rrbracket \\ &= T \langle \text{id}_{[A \Rightarrow B]}, \llbracket V \rrbracket^V \circ! \rangle \circ \llbracket M \rrbracket \circ \llbracket E \rrbracket \end{aligned}$$

Again, putting the pieces together:

$$\llbracket (V, E) / (M, E')[\] \cdot K \rrbracket = \text{tick}^* \circ (\text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev} \circ \langle \text{id}_{[A \Rightarrow B]}, \llbracket V \rrbracket^V \circ! \rangle)^* \circ \llbracket M \rrbracket \circ \llbracket E \rrbracket$$

as expected.

– $(\text{Clos } fx.(M, E), E') / []V \cdot K \rightsquigarrow_V (M, E[f \rightarrow \text{Clos } fx.(M, E), x \rightarrow V]) / K$ By definition:

$$\begin{aligned} & \llbracket (\text{Clos } fx.(M, E), E') / []V \cdot K \rrbracket \\ &= (\text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev} \circ \langle \text{id}, \llbracket V \rrbracket^V \circ ! \rangle)^* \circ \eta \circ \sqcup \varphi_n \circ \llbracket E \rrbracket \\ &= \text{tick}^* \circ \llbracket K \rrbracket^* \circ \text{Ev} \circ \langle \text{id}, \llbracket V \rrbracket^V \circ ! \rangle \circ \sqcup \varphi_n \circ \llbracket E \rrbracket \end{aligned}$$

As

$$\begin{aligned} \sqcup \varphi_n \circ \llbracket E \rrbracket &= \sqcup C[\Gamma, f, x \vdash M] \circ \langle \llbracket E \rrbracket, \varphi_n \circ \llbracket E \rrbracket \rangle \\ &= C[\Gamma, f, x \vdash M] \circ \langle \llbracket E \rrbracket, \sqcup \varphi_n \circ \llbracket E \rrbracket \rangle \\ &= C[\Gamma, f, x \vdash M] \circ \langle \llbracket E \rrbracket, \llbracket \text{Clos } fx.(M, E) \rrbracket^V \rangle \\ &= C[\Gamma, f, x \vdash M] \circ \llbracket E[f \rightarrow \text{Clos } fx.(M, E)] \rrbracket \end{aligned}$$

we get:

$$\begin{aligned} & \text{Ev} \circ \langle \text{id}, \llbracket V \rrbracket^V \circ ! \rangle \circ \sqcup \varphi_n \circ \llbracket E \rrbracket \\ &= \text{Ev} \circ \langle \text{id}, \llbracket V \rrbracket^V \circ ! \rangle \circ C[\Gamma, f, x \vdash M] \circ \llbracket E[f \rightarrow \text{Clos } fx.(M, E)] \rrbracket \\ &= \text{Ev} \circ \langle C[\Gamma, f, x \vdash M], \llbracket V \rrbracket^V \circ ! \rangle \circ \llbracket E[f \rightarrow \text{Clos } fx.(M, E)] \rrbracket \\ &= \llbracket \Gamma, f, x \vdash M \rrbracket \circ \langle \text{id}_{\llbracket \Gamma, f \rrbracket}, \llbracket V \rrbracket^V \circ ! \rangle \circ \llbracket E[f \rightarrow \text{Clos } fx.(M, E)] \rrbracket \\ &= \llbracket \Gamma, f, x \vdash M \rrbracket \circ \langle \llbracket E[f \rightarrow \text{Clos } fx.(M, E)] \rrbracket, \llbracket V \rrbracket^V \rangle \\ &= \llbracket \Gamma, f, x \vdash M \rrbracket \circ \llbracket E[f \rightarrow \text{Clos } fx.(M, E), x \rightarrow V] \rrbracket \end{aligned}$$

which complete the proof.

Proposition 8 (Soundness). *If $(M, E) / K \Downarrow_V n$, then $\llbracket (M, E) / K \rrbracket = \underline{n}$.*

Proof. By induction on n . If $n = 0$, then $M = \star$ and $K = \epsilon$, so $\llbracket (M, E) / K \rrbracket = \underline{0}^* \circ \eta \circ ! \circ \llbracket E \rrbracket = \underline{0}$.

If $n > 0$, then there exist M', E', K' such that $(M, E) / K \rightsquigarrow_V (M', E') / K'$ and $(M', E') / K' \Downarrow_V n - 1$. By inductive hypothesis, $\llbracket (M', E') / K' \rrbracket = \underline{n - 1}$. By lemma 6,

$$\llbracket (M, E) / K \rrbracket = \text{tick}_1^* \circ \llbracket (M', E') / K' \rrbracket = \text{tick}_1^* \circ \underline{n - 1} = \underline{n}$$

7.2 Adequacy for Call By Value

For the adequacy proof, we define heterogeneous relations, between syntactical entities and the denotations: if the denotation side “terminates” in n steps, then so does the syntactical side. We prove a fundamental lemma, *i.e.* that a term and its denotation are always related, which will give the adequacy theorem: if the denotation of a state is \underline{n} , then the state terminates in n steps.

We will define:

$$\begin{aligned} \mathcal{R}_A &\subseteq \{V \mid V : A\} \times \mathcal{C}[1, \llbracket A \rrbracket] \\ \mathcal{R}_{A^\perp} &\subseteq \{K \mid K : A^\perp\} \times \mathcal{C}[\llbracket A \rrbracket, TU] \\ \mathcal{R}_{TA} &\subseteq \{(M, E) \mid \Gamma \vdash M : A, E : \Gamma\} \times \mathcal{C}[1, T[\llbracket A \rrbracket]] \\ \mathcal{R}_\Gamma &\subseteq \{E \mid E : \Gamma\} \times \mathcal{C}[1, \llbracket \Gamma \rrbracket] \\ \mathcal{R}_{\Gamma \vdash A} &\subseteq \{M \mid \Gamma \vdash M : A\} \times \mathcal{C}[\llbracket \Gamma \rrbracket, T[\llbracket A \rrbracket]] \end{aligned}$$

First, we define the relation between the states and their denotations:

$$\Sigma = \{((M, E) / K, f) \mid f = \underline{n} \implies (M, E) / K \Downarrow_n\}$$

Note that we put an implication, so \perp is related to all states, and this will be the base case for the induction in the proof of the application case.

The relations themselves are defined inductively on the type by:

$$\begin{aligned} \mathcal{R}_1 &= \{\star, \ast\} \\ \mathcal{R}_{A \Rightarrow B} &= \{(\text{Clos } fx.(M, E), f) \mid \forall (V, v) \in \mathcal{R}_A, \\ &\quad ((M, E[f \rightarrow \text{Clos } fx.(M, E), x \rightarrow V]), \text{Ev} \circ \langle f, v \rangle) \in \mathcal{R}_{TB}\} \\ \mathcal{R}_{A^\perp} &= \{(K, k) \mid \forall (V, v) \in \mathcal{R}_A, ((\text{Val } V, \emptyset) / K, kv) \in \Sigma\} \\ \mathcal{R}_{TA} &= \{((M, E), c) \mid \forall (K, k) \in \mathcal{R}_{A^\perp}, ((M, E) / K, k^\ast \circ c) \in \Sigma\} \end{aligned}$$

We extend the relations to environments and open terms:

$$\begin{aligned} \mathcal{R}_\Gamma &= \{(E, e) \mid \forall (x : A) \in \Gamma, (E(x), \pi \circ e) \in \mathcal{R}_A\} \\ \mathcal{R}_{\Gamma \vdash A} &= \{(M, f) \mid \forall (E, e) \in \mathcal{R}_\Gamma, ((M, E), f \circ e) \in \mathcal{R}_A\} \end{aligned}$$

The proof of the fundamental lemma and the adequacy theorem are very similar to those of the call-by-name case, although much longer. We only give the intermediate steps of the proofs. The following two lemmas are easy:

Lemma 7. *Basic properties of the heterogeneous relations:*

- if $C/K \rightsquigarrow C'/K'$ and $(C'/K', f) \in \Sigma$, then $(C/K, \text{tick}^\ast \circ f) \in \Sigma$.
- if (x_n) is an ascending chain and $(X, x_n) \in \mathcal{R}_Y$ for all n (where Y is a A, A^\perp, TA, Γ or $\Gamma \vdash A$), then $(X, \bigsqcup x_n) \in \mathcal{R}_Y$ too.
- if $(V, v) \in \mathcal{R}_A$, then for all $(E, e) \in \mathcal{R}_\Gamma$, $((V, E), \eta \circ v \circ !) \in \mathcal{R}_{TA}$.

Lemma 8. *Extending the continuations: if $(K, k) \in \mathcal{R}_{B^\perp}$, then*

$$\begin{aligned} ((M, E), c) \in \mathcal{R}_{T(A \Rightarrow B)} &\implies ((M, E)[\] \cdot K, \text{tick}^\ast \circ \text{tick}^\ast \circ k^\ast \circ \text{Ev}^\ast \bar{t} \circ \langle c \circ !, \text{id}_{[\]} \rangle) \in \mathcal{R}_{A^\perp} \\ (V, v) \in \mathcal{R}_A &\implies ([\]V \cdot K, \text{tick}^\ast \circ k^\ast \circ \text{Ev} \circ \langle \text{id}_{[\]}, v \circ ! \rangle) \in \mathcal{R}_{(A \Rightarrow B)^\perp} \end{aligned}$$

We then get to the fundamental lemma, which is proved by a single induction:

Lemma 9. – If $K : A^\perp$, $(K, [\]K : A^\perp) \in \mathcal{R}_{A^\perp}$,
– if $V : A$, then $(V, [\]V : A) \in \mathcal{R}_A$,
– if $\Gamma \vdash M : A$, then $(M, [\]\Gamma \vdash M : A) \in \mathcal{R}_{\Gamma \vdash A}$,

The expected adequacy of the semantics is then an immediate corollary:

Proposition 9. *For all typed states $M/EK : 1$, if $[(M, E) / K] = \underline{n}$ then $M/EK \Downarrow_V n$.*

7.3 Soundness for Call By Name

Lemma 10. *If $d \in \llbracket A \rrbracket$ and $\sigma \in \llbracket A^\perp \rrbracket$, $(\text{tick}_A \bullet f)/\sigma = \text{tick}_1 \bullet (f/\sigma)$.*

Lemma 11. *If $C/S \rightsquigarrow C'/S'$, then $\llbracket C/S \rrbracket = \text{tick}_1 \llbracket C'/S' \rrbracket$.*

Proof. We consider the three possible cases. If $C = (x, E)$, then $C' = E(x)$ and $S' = S$.

$$\begin{aligned} \llbracket C/S \rrbracket &= \text{tick}_A \circ \pi_i \bullet \llbracket E \rrbracket / \llbracket S \rrbracket \\ &= \text{tick}_A \bullet \eta \circ \pi_i \bullet \llbracket E \rrbracket / \llbracket S \rrbracket \\ &= \text{tick}_1 \bullet (\eta \circ \pi_i \bullet \llbracket E \rrbracket / \llbracket S \rrbracket) \\ &= \text{tick}_1 \bullet (\llbracket E(x) \rrbracket / \llbracket S \rrbracket) \\ &= \text{tick}_1 \bullet (\llbracket C' \rrbracket / \llbracket S' \rrbracket) \end{aligned}$$

If $C = ((M)N, E)$ then $C' = (M, E)$ and $S' = (N, E) \cdot S$.

$$\begin{aligned} \llbracket C/S \rrbracket &= (\text{tick} \bullet \mathbf{Ev} \bullet \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \bullet \llbracket E \rrbracket) / \llbracket S \rrbracket \\ &= \text{tick} \bullet (\mathbf{Ev} \bullet \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \bullet \llbracket E \rrbracket / \llbracket S \rrbracket) \\ &= \text{tick} \bullet (\mathbf{Ev} \bullet \langle \llbracket M \rrbracket \bullet \llbracket E \rrbracket, \llbracket N \rrbracket \bullet \llbracket E \rrbracket \rangle / \llbracket S \rrbracket) \\ &= \text{tick} \bullet (\llbracket M \rrbracket \bullet \llbracket E \rrbracket / (\llbracket N \rrbracket \bullet \llbracket E \rrbracket) \cdot \llbracket S \rrbracket) \\ &= \text{tick} \bullet (\llbracket (M, E) \rrbracket / \llbracket (N, E) \cdot S \rrbracket) \\ &= \text{tick} \bullet (\llbracket C' \rrbracket / \llbracket S' \rrbracket) \end{aligned}$$

And finally, if $C = (\text{rec } fx.M, E)$ and $S = C_x \cdot S''$, then $C' = (M, E[f \rightarrow C, x \rightarrow C_x])$ and $S' = S''$. We have

$$\llbracket (\text{rec } fx.M, E) \rrbracket = \bigsqcup \varphi_n \bullet \llbracket E \rrbracket$$

where

$$\begin{cases} \varphi_0 = \perp \\ \varphi_{n+1} = \mathbf{C}[\Gamma, f : A \rightarrow B, x : A \vdash M : B] \bullet \langle \mathbf{id}_\Gamma, \varphi_n \rangle \end{cases}$$

We get:

$$\begin{aligned} \llbracket C/S \rrbracket &= \bigsqcup \varphi_n \bullet \llbracket E \rrbracket / \llbracket C_x \cdot S' \rrbracket \\ &= \bigsqcup \varphi_{n+1} \bullet \llbracket E \rrbracket / \llbracket C_x \rrbracket \cdot \llbracket S' \rrbracket \\ &= \bigsqcup \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \mathbf{id}_\Gamma, \varphi_n \rangle \bullet \llbracket E \rrbracket / \llbracket C_x \rrbracket \cdot \llbracket S' \rrbracket \\ &= \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \bigsqcup \varphi_n \bullet \llbracket E \rrbracket \rangle / \llbracket C_x \rrbracket \cdot \llbracket S' \rrbracket \\ &= \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle / \llbracket C_x \rrbracket \cdot \llbracket S' \rrbracket \\ &= \text{tick} \bullet (\mathbf{Ev} \bullet \langle \mathbf{C}[\Gamma, f, x \vdash M] \bullet \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle, \llbracket C_x \rrbracket \rangle / \llbracket S' \rrbracket) \\ &= \text{tick} \bullet (\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle \langle \llbracket E \rrbracket, \llbracket C \rrbracket \rangle, \llbracket C_x \rrbracket \rangle / \llbracket S' \rrbracket) \\ &= \text{tick} \bullet (\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \llbracket E[f \rightarrow C, x \rightarrow C_x] \rrbracket / \llbracket S' \rrbracket) \\ &= \text{tick} \bullet (\llbracket (M, E[f \rightarrow C, x \rightarrow C_x]) \rrbracket / \llbracket S' \rrbracket) \\ &= \text{tick} \bullet (\llbracket C' \rrbracket / \llbracket S' \rrbracket) \end{aligned}$$

Proposition 10 (Soundness). *If $C/S : 1$ terminates in n steps, then $\llbracket C/S \rrbracket = \underline{n}$.*

Proof. By induction on n . If $n = 0$, then $C = (\star, E)$ for some E and $S = \epsilon$, so $\llbracket C/S \rrbracket = \underline{0}$.

If $n > 0$, then there exist C', S' such that $C/S \rightsquigarrow C'/S'$ and $C'/S' \Downarrow_{n-1}$. By inductive hypothesis, $\llbracket C'/S' \rrbracket = \underline{n-1}$. By lemma 11,

$$\llbracket C/S \rrbracket = \text{tick}_1 \bullet \llbracket C'/S' \rrbracket = \text{tick}_1 \bullet \underline{n-1} = \underline{n}$$

7.4 Adequacy for Call By Name

Again, we define a heterogenous relation between syntactical entities and their denotations, but this time there is no distinction between values and computations (everything is a computation):

$$\begin{aligned}\mathcal{R}_A &\subseteq \{C \mid C : A\} \times \mathcal{C}[1, \llbracket A \rrbracket] \\ \mathcal{R}_{A^\perp} &\subseteq \{S \mid S : A^\perp\} \times \mathcal{C}[1, \llbracket A^\perp \rrbracket] \\ \mathcal{R}_\Gamma &\subseteq \{E \mid E : \Gamma\} \times \mathcal{C}[1, \llbracket \Gamma \rrbracket] \\ \mathcal{R}_{\Gamma \vdash A} &\subseteq \{M \mid \Gamma \vdash M : A\} \times \mathcal{C}[\llbracket \Gamma \rrbracket, \llbracket A \rrbracket]\end{aligned}$$

The set Σ is the same: $\Sigma = \{(C/S, f) \mid \forall n, f = \underline{n} \implies C/S \Downarrow_n\}$. Since the continuations have a precise shape, we can make the definition of the relations much more straightforward:

$$\begin{aligned}\mathcal{R}_A &= \{(C, c) \mid \forall (S, \sigma) \in \mathcal{R}_{A^\perp}, (C/S, c/\sigma) \in \Sigma\} \\ \mathcal{R}_{(A_1 \rightarrow \dots \rightarrow A_n \rightarrow 1)^\perp} &= \{((C_1, \dots, C_n), \sigma) \mid \forall i, (C_i, \pi_i \bullet \sigma) \in \mathcal{R}_{A_i}\}\end{aligned}$$

This definition is well-founded, the base case being $\mathcal{R}_{1^\perp} = \{(\epsilon, \mathbf{id}_1)\}$. This is simpler than in the CBV case, because we do not have to worry about values here. Again, we extend the relations to environments and open terms:

$$\begin{aligned}\mathcal{R}_{x_1:A_1, \dots, x_n:A_n} &= \{(E, e) \mid \forall (x : A) \in \Gamma, (E(x_i), \pi_i \bullet e) \in \mathcal{R}_{A_i}\} \\ \mathcal{R}_{\Gamma \vdash A} &= \{(M, d) \mid \forall (E, e) \in \mathcal{R}_\Gamma, ((M, E), d \bullet e) \in \mathcal{R}_A\}\end{aligned}$$

Lemma 12. *If $C/S \rightsquigarrow C'/S'$ and $(C'/S', x) \in \Sigma$, then $(C/S, \text{tick}_1 \circ x) \in \Sigma$.*

Lemma 13. *The relations are closed under limits of chains: if (x_n) is an ascending chain and if $(X, x_n) \in \mathcal{R}_Y$ (where Y is a A, A^\perp, Γ or $\Gamma \vdash A$), for all n , then $(X, \bigsqcup x_n) \in \mathcal{R}_Y$ too.*

Lemma 14 (Fundamental lemma). *If $\Gamma \vdash M : A$, then $(M, \llbracket \Gamma \vdash M : A \rrbracket) \in \mathcal{R}_{\Gamma \vdash A}$.*

Proof. By induction on the derivation of $\Gamma \vdash t : A$. For $\Gamma \vdash \star : 1$, we have for all environments $(E, e) \in \mathcal{R}_\Gamma$, $\llbracket \Gamma \vdash \star : 1 \rrbracket \bullet e = \underline{0}$ and $(\star, E) / \epsilon \Downarrow_0$, so $((\star, E) / \epsilon, \llbracket \Gamma \vdash \star : 1 \rrbracket \circ e) \in \Sigma$.

For $\Gamma \vdash x : A$, when $(x : A) \in \Gamma$. We have, for all $(E, e) \in \mathcal{R}_\Gamma$ and $(S, \sigma) \in \mathcal{R}_{A^\perp}$

$$(x, E) / S \rightsquigarrow E(x)/S$$

$$\llbracket \Gamma \vdash x : A \rrbracket \bullet e/\sigma = (\text{tick}_A \circ \pi_i \bullet e)/\sigma = \text{tick}_1 \bullet (\pi_i \bullet e/\sigma)$$

By definition, $(E(x), \pi_i \bullet e) \in \mathcal{R}_A$, so $(E(x)/S, \pi_i \bullet e/\sigma) \in \Sigma$, and we conclude by lemma 12.

For $\Gamma \vdash (M)N : B$ when $\Gamma \vdash M : A \rightarrow B$ and $\Gamma \vdash N : A$. We have, for all $(E, e) \in \mathcal{R}_\Gamma$ and $(S, \sigma) \in \mathcal{R}_{A^\perp}$

$$((M)N, E) / S \rightsquigarrow (M, E) / (N, E) \cdot S$$

$$\begin{aligned}
\llbracket \Gamma \vdash (M)N : B \rrbracket e/\sigma &= (\text{tick}_B \bullet \mathbf{Ev} \bullet \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \bullet e)/\sigma \\
&= \text{tick}_1 \bullet (\mathbf{Ev} \bullet \langle \llbracket M \rrbracket, \llbracket N \rrbracket \rangle \bullet e/\sigma) \\
&= \text{tick}_1 \bullet (\mathbf{Ev} \bullet \langle \llbracket M \rrbracket \bullet e, \llbracket N \rrbracket \bullet e \rangle / \sigma) \\
&= \text{tick}_1 \bullet (\llbracket M \rrbracket \bullet e / (\llbracket N \rrbracket \bullet e) \cdot \sigma)
\end{aligned}$$

By inductive hypothesis, $((N, E), \llbracket N \rrbracket \bullet e) \in \mathcal{R}_A$, so $((N, E) \cdot S, (\llbracket N \rrbracket \bullet e) \cdot \sigma) \in \mathcal{R}_{(A \rightarrow B)^\perp}$. Again, by inductive hypothesis, $(M, \llbracket M \rrbracket) \in \mathcal{R}_{\Gamma \vdash A}$, so

$$((M, E) / (N, E) \cdot S, \llbracket M \rrbracket \bullet e / (\llbracket N \rrbracket \bullet e) \cdot \sigma) \in \Sigma$$

and we conclude again by lemma 12.

For $\Gamma \vdash \text{rec } fx.M : A \rightarrow B$ when $\Gamma, f : A \rightarrow B, x : A \vdash M : B$. We have $\llbracket \Gamma \vdash \text{rec } fx.M : A \rightarrow B \rrbracket = \bigsqcup \varphi_n$ where

$$\begin{aligned}
\varphi_0 &= \perp \\
\varphi_{n+1} &= \mathbf{C}[\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle \mathbf{id}_{\llbracket \Gamma \rrbracket}, \varphi_n \rangle]
\end{aligned}$$

Let us prove that $(\text{rec } fx.M, \varphi_n) \in \mathcal{R}_{\Gamma \vdash A \rightarrow B}$ for all n . We will then conclude with lemma 13.

For $n = 0$, for all $(E, e) \in \mathcal{R}_\Gamma$ and $(S, \sigma) \in \mathcal{R}_{(A \rightarrow B)^\perp}$, $\varphi_0 \bullet e/\sigma = \perp$, so $(\text{rec } fx.M/ES, \varphi_0 \bullet e/\sigma) \in \Sigma$.

Assume that $(\text{rec } fx.M, \varphi_n) \in \mathcal{R}_{\Gamma \vdash A \rightarrow B}$. Let $(E, e) \in \mathcal{R}_\Gamma$ and $(S, \sigma) \in \mathcal{R}_{(A \rightarrow B)^\perp}$. By definition, $S = C \cdot S'$ and $\sigma = \langle c, \sigma' \rangle$ with $(C, c) \in \mathcal{R}_A$ and $(S', \sigma') \in \mathcal{R}_{B^\perp}$. We have

$$\begin{aligned}
(\text{rec } fx.M, E) / S &\rightsquigarrow (M, E[f \rightarrow (\text{rec } fx.M, E), x \rightarrow C]) / S' \\
(\varphi_{n+1} \circ e)/\sigma &= (\text{tick} \bullet \mathbf{C}[\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle \mathbf{id}_{\llbracket \Gamma \rrbracket}, \varphi_n \rangle \bullet e] \bullet e)/\sigma \\
&= \text{tick} \bullet (\mathbf{C}[\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle e, \varphi_n \bullet e \rangle] \bullet e/\sigma) \\
&= \text{tick} \bullet (\mathbf{Ev} \bullet \langle \mathbf{C}[\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle e, \varphi_n \bullet e \rangle], c \rangle / \sigma') \\
&= \text{tick} \bullet (\llbracket \Gamma, f, x \vdash M \rrbracket \bullet \langle \langle e, \varphi_n \bullet e \rangle, c \rangle / \sigma')
\end{aligned}$$

Let $e' = \langle \langle e, \varphi_n \bullet e \rangle, c \rangle$, $E' = E[f \rightarrow (\text{rec } fx.M, E), x \rightarrow C]$. By hypothesis, $(\text{rec } fx.M, \varphi_n) \in \mathcal{R}_{\Gamma \vdash A \rightarrow B}$, so $((\text{rec } fx.M, E), \varphi_n \circ e) \in \mathcal{R}_{A \rightarrow B}$. Since $(C, c) \in \mathcal{R}_A$,

$$(E', e') \in \mathcal{R}_{\Gamma, f : A \rightarrow B, x : A}$$

By inductive hypothesis,

$$(M, \llbracket \Gamma, f : A \rightarrow B \vdash M : B, x : A \rrbracket) \in \mathcal{R}_{\Gamma, f : A \rightarrow B, x : A \vdash B}$$

As $(S', \sigma') \in \mathcal{R}_{B^\perp}$:

$$((M, E') / S', \llbracket \Gamma, f : A \rightarrow B, x : A \vdash M : B \rrbracket \circ e'/\sigma') \in \Sigma$$

Lemma 12 gives

$$((\text{rec } fx.M, E) / S, \varphi_{n+1}/\sigma) \in \Sigma$$

Proposition 11 (Adequacy). *For all typed states $C/S : 1$, $\llbracket C/S \rrbracket = \underline{n}$ entails $C/S \Downarrow_N n$.*

Proof. An easy induction and the fundamental lemma prove that for all closures $C : A$, $(C, \llbracket C \rrbracket) \in \mathcal{R}_A$. This in turns entails that for all stacks $S : A^\perp$, $(S, \llbracket S \rrbracket) \in \mathcal{R}_{A^\perp}$. We finally get that, for all typed states C/S , $(C/S, \llbracket C/S \rrbracket) \in \Sigma$.