Available online at www.sciencedirect.com

SciVerse ScienceDirect

Electronic Notes in Theoretical Computer Science 276 (2011) 191–213

Electronic Notes in Theoretical Computer Science

www.elsevier.com/locate/entcs

Capsules and Closures

Jean-Baptiste Jeannin[1]

Department of Computer Science
Cornell University
Ithaca, New York 14853-7501, USA

Abstract

Capsules are a clean representation of the state of a computation in higher-order programming languages with effects. Their intent is to simplify and replace the notion of closure. They naturally provide support for functional and imperative features, including recursion and mutable bindings, and ensure lexical scoping without the use of closures, heaps, stacks or combinators. We present a comparison of the use of closures and capsules in the semantics of higher-order programming languages with effects. In proving soundness of one to the other, we give a precise account of how capsule environments and closure environments relate to each other.

Keywords: Capsule, Closure, Functional Programming, Imperative Programming, State of Computation, Higher-Order Functions, Mutable Variables, Scoping, Programming Language Semantics.

1 Introduction

This paper compares *Capsules* and *Closures*. *Capsules* are a representation of the state of a computation for higher-order functional and imperative languages with effects, and were introduced in [1]. Many authors have studied the state of a computation, for example [2–14]. However, capsules are intended to be as simple as possible, and they correctly capture lexical scoping and handle variable assignment and recursion without any combinators, stacks or heaps, and while keeping everything typable with simple types.

Closures were first introduced by Peter J. Landin along with the SECD machine [13], and first implemented in the programming language Scheme [15]. The early versions of Lisp implemented *dynamic scoping*, which did not follow the semantics of the λ-calculus based on β-reduction. By keeping with each λ-abstraction the environment in which it was declared, thus forming a closure, closures were successful at implementing *static scoping* efficiently.

[1] Email: jeannin@cs.cornell.edu

doi:10.1016/j.entcs.2011.09.022

In [1], capsules are shown to be essentially finite coalgebraic representations of regular closed λ-coterms. Because of recursion and therefore of possible cycles in the environment, the state of computation should be able to represent all finite λ-terms and a subset of the infinite λ-terms, also called λ-coterms. Capsules represent all the regular λ-coterms, and that is enough to model every computation in the language. λ-coterms allow to represent recursive functions directly, without the need for the Y-combinator or recursive types.

The language we introduce is both functional and imperative: it has higher-order functions, but every variable is mutable. This leads to interesting interactions and allows to go further than just enforcing lexical scoping. In particular, what do we expect the result of an expression like ($\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ x := 2; f\ 0$) to be? Scheme (using set! for :=) and OCaml (using references) answer 2. Capsules give a rigorous mathematical definition that agrees and conservatively extends the scoping rules of the λ-calculus. Our semantics of closures also agrees with this definition, but this requires introducing a level of indirection, with both an environment and a store, à la ML. Finally, recursive definitions are often implemented using some sort of backpatching; this construction is known as “Landin’s knot”. We build this directly into the definition of the language by defining $\mathsf{let\,rec}\ x = d\ \mathsf{in}\ e$ as a syntactic sugar for $\mathsf{let}\ x = a\ \mathsf{in}\ x := d; e$, where a is any expression of the appropriate type.

There is much previous work on reasoning about references and local state; see [16–19]. State is typically modeled by some form of heap from which storage locations can be allocated and deallocated [9–12]. Others have used game semantics to reason about local state [20–22]. Mason and Talcott [2–4] and Felleisen and Hieb [5] present a semantics based on a heap and storage locations. A key difference is that Felleisen and Hieb’s semantics is based on continuations. Finally, Moggi [8] proposed monads, which can be used to model state and are implemented in Haskell.

This paper is organized as follows. In section 2, we formally introduce a programming language based on the λ-calculus containing both functional and imperative features. In section 3, we describe two semantics for this language, one based on capsules and the other on closures. In section 4, we show a very strong correspondence (Theorem 4.5) between the two semantics, showing that every computation in the semantics of capsules is bisimilar to a computation in the semantics of closures, and vice-versa. In section 5, we show (Propositions 5.1–5.4) that closure semantics retains some unnecessary information that capsule semantics omits, attesting of the simplicity of capsules. We finish with a discussion in section 6.

2 Syntax

2.1 *Expressions*

Expressions $\mathsf{Exp} = \{d, e, a, b, \ldots\}$ contain both functional and imperative features. There is an unlimited supply of *variables* $x, y, z, \ldots$ of all (simple) types, as well as *constants* $f, c, \ldots$ for primitive values. $()$ is the only constant of type unit, and true and false are the only two constants of type bool. In addition, there are functional features

- λ-abstraction $\quad \lambda x.e$
- application $\quad (d\ e)$,

imperative features

- assignment $\quad x := e$
- composition $\quad d; e$
- conditional $\quad \mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e$
- while loop $\quad \mathsf{while}\ b\ \mathsf{do}\ e$,

and syntactic sugars

- $\mathsf{let}\ x = d\ \mathsf{in}\ e \quad (\lambda x.e)\ d$
- $\mathsf{let\,rec}\ x = d\ \mathsf{in}\ e \quad \mathsf{let}\ x = a\ \mathsf{in}\ x := d; e$

where a is any expression of the appropriate type.

Let Var be the set of variables, Const the set of constants, and $\lambda\text{-}\mathsf{Abs}$ the set of λ-abstractions. Given an expression e, let $\mathsf{FV}(e)$ denote the set of free variables of e. Given a partial function $h : \mathsf{Var} \rightharpoonup \mathsf{Var}$ such that $\mathsf{FV}(e) \subseteq \mathsf{dom}\, h$, let $h(e)$ be the expression e where every instance of a free variable $x \in \mathsf{FV}(e)$ has been replaced by the variable $h(x)$. As usual, given two partial functions g and h, $g \circ h$ denotes their composition such that for all x, $g \circ h(x) = g(h(x))$. Given a function h, we write $h[x/v]$ the function such that $h[x/v](y) = h(y)$ for $y \neq x$ and $h[x/v](x) = v$. Given an expression e, we write $e[x/y]$ the expression e where all free occurrences of x have been replaced by y.

Throughout the paper, we focus on the features directly involving variables: variable calls x, λ-abstractions $\lambda x.e$, applications $(d\ e)$ where d reduces to a λ-abstraction, and assignment $x := e$. Most differences between capsules and closures arise using these features.

2.2 Types

Types $\alpha, \beta, \ldots$ are built inductively from an unspecified family of base types, including at least unit and bool, and a type constructor $\rightarrow$ such that functions with input type α and return type β have type $\alpha \rightarrow \beta$. All constants c of the language have a type $\mathsf{type}(c)$; by convention, we use c for a constant of a base type and f for a constant of a functional type. We follow [23] in assuming that each variable x is associated with a unique type $\mathsf{type}(x)$, that could for example be built into the variable name. Γ is a type environment, a partial function $\mathsf{Var} \rightharpoonup \mathsf{Type}$. As is standard, we write $\Gamma, x : \alpha$ for the typing environment Γ where x has been bound or rebound to α. The typing rules are standard:

$$\Gamma \vdash c : \alpha \text{ if } \mathsf{type}(c) = \alpha \qquad \Gamma, x : \alpha \vdash x : \alpha \qquad \frac{\mathsf{type}(x) = \alpha \quad \Gamma, x : \alpha \vdash e : \beta}{\Gamma \vdash \lambda x.e : \alpha \to \beta}$$

$$\frac{\Gamma \vdash d : \alpha \to \beta \quad \Gamma \vdash e : \alpha}{\Gamma \vdash (d\ e) : \beta} \qquad \frac{\Gamma \vdash x : \alpha \quad \Gamma \vdash e : \alpha}{\Gamma \vdash x := e : \mathsf{unit}} \qquad \frac{\Gamma \vdash d : \mathsf{unit} \quad \Gamma \vdash e : \alpha}{\Gamma \vdash d; e : \alpha}$$

$$\frac{\Gamma \vdash b : \mathsf{bool} \quad \Gamma \vdash d : \alpha \quad \Gamma \vdash e : \alpha}{\Gamma \vdash \mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e : \alpha} \qquad \frac{\Gamma \vdash b : \mathsf{bool} \quad \Gamma \vdash e : \mathsf{unit}}{\Gamma \vdash \mathsf{while}\ b\ \mathsf{do}\ e : \mathsf{unit}}$$

3 Semantics

We present two different semantics that have a strong correspondence:

- The semantics on *capsules* is a simplified version of the semantics on closure structures introduced in [24]. It has previously been described in [1];
- The semantics on *closures* is the semantics usually used and taught for functional languages. A level of indirection for variables has been added to support imperative features, *à la* ML.

All the expressions we consider in this section are supposed well-typed with the rules of section 2.2.

3.1 Capsules

3.1.1 Definitions

An *irreducible term* is either a constant or a λ-abstraction. A *capsule environment* is a partial function from variables to irreducible terms.

Let $i, j, k, \ldots$ denote irreducible terms and $\gamma, \delta, \zeta, \eta, \ldots$ capsule environments. Let $\mathsf{Irred} = \mathsf{Const} + \lambda\text{-}\mathsf{Abs}$ be the set of irreducible terms. Thus we have:

$$\gamma : \mathsf{Var} \rightharpoonup \mathsf{Irred} \qquad\qquad \mathsf{Irred} = \mathsf{Const} + \lambda\text{-}\mathsf{Abs}$$

A capsule environment γ is *valid* if and only if

$$\forall x \in \mathsf{dom}\,\gamma,\ \mathsf{FV}(\gamma(x)) \subseteq \mathsf{dom}\,\gamma$$

3.1.2 Semantics

A *capsule* is a pair $\langle e, \gamma \rangle$. A capsule is *valid* if and only if $\mathsf{FV}(e) \subseteq \mathsf{dom}\,\gamma$ and γ is valid. We only consider valid capsule environments and valid capsules.

An *irreducible capsule* is a capsule $\langle i,\, \gamma\rangle$ where $i \in \mathsf{Irred}$. Let us define a big step semantics where the operator $\Downarrow_{\mathsf{ca}}$ relates capsules to irreducible capsules. The semantics of features directly involving variables is given by:

$$\langle x,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \gamma(x),\, \gamma\rangle \qquad \langle \lambda x.e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \lambda x.e,\, \gamma\rangle \qquad \frac{\langle e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \zeta\rangle}{\langle x := e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle (),\, \zeta[x/j]\rangle}$$

$$\frac{\langle d,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \lambda x.a,\, \zeta\rangle \qquad \langle e,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \eta\rangle \qquad \langle a[x/y],\, \eta[y/j]\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}{\langle d\ e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}\ (y \text{ fresh})$$

and the remaining semantics is:

$$\langle c,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle c,\, \gamma\rangle \qquad \frac{\langle d,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle f,\, \zeta\rangle \qquad \langle e,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle c,\, \delta\rangle}{\langle d\ e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle f(c),\, \delta\rangle}$$

$$\frac{\langle d,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle (),\, \zeta\rangle \qquad \langle e,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}{\langle d; e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}$$

$$\frac{\langle b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{true},\, \zeta\rangle \qquad \langle d,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}{\langle \mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle} \qquad \frac{\langle b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{false},\, \zeta\rangle \qquad \langle e,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}{\langle \mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle}$$

$$\frac{\begin{array}{c}\langle b,\, \gamma_i\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{true},\, \delta_i\rangle \qquad \langle e,\, \delta_i\rangle \Downarrow_{\mathsf{ca}} \langle (),\, \gamma_{i+1}\rangle,\ 0 \le i < n,\ n \ge 0 \\ \langle b,\, \gamma_n\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{false},\, \delta_n\rangle\end{array}}{\langle \mathsf{while}\ b\ \mathsf{do}\ e,\, \gamma_0\rangle \Downarrow_{\mathsf{ca}} \langle (),\, \delta_n\rangle}$$

3.1.3 Examples

The following examples show that lexical scoping and recursion are handled.

Example 3.1 $(\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f\ 0) \Downarrow_{\mathsf{ca}} 1$

Proof. For simplicity, we just show the different capsules of the computation.

$$\begin{array}{lr}
\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f\ 0 & [\,] \\
\mathsf{let}\ f = \lambda y.x'\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f\ 0 & [x' = 1] \\
\mathsf{let}\ x = 2\ \mathsf{in}\ f\ 0 & [x' = 1,\ f = \lambda y.x'] \\
f\ 0 & [x' = 1,\ f = \lambda y.x',\ x'' = 2] \\
(\lambda y.x')\ 0 & [x' = 1,\ f = \lambda y.x',\ x'' = 2] \\
x' & [x' = 1,\ f = \lambda y.x',\ x'' = 2,\ y' = 0] \\
1 & [x' = 1,\ f = \lambda y.x',\ x'' = 2,\ y' = 0]
\end{array}$$

□

Example 3.2 $(\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ x := 2; f\ 0) \Downarrow_{\mathsf{ca}} 2$

Proof.

$$\begin{array}{lr}
\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ x := 2; f\ 0 & [\,] \\
\mathsf{let}\ f = \lambda y.x'\ \mathsf{in}\ x' := 2; f\ 0 & [x' = 1] \\
x' := 2; f\ 0 & [x' = 1,\ f = \lambda y.x'] \\
f\ 0 & [x' = 2,\ f = \lambda y.x'] \\
(\lambda y.x')\ 0 & [x' = 2,\ f = \lambda y.x'] \\
x' & [x' = 2,\ f = \lambda y.x',\ y' = 0] \\
2 & [x' = 2,\ f = \lambda y.x',\ y' = 0]
\end{array}$$

□

Example 3.3 $(\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f := \lambda y.x; f\ 0) \Downarrow_{\mathsf{ca}} 2$

Proof.

$$\begin{array}{lr}
\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f := \lambda y.x; f\ 0 & [\,] \\
\mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f := \lambda y.x; f\ 0 & [x' = 1] \\
\mathsf{let}\ x = 2\ \mathsf{in}\ f := \lambda y.x; f\ 0 & [x' = 1,\ f = \lambda y.x'] \\
f := \lambda y.x''; f\ 0 & [x' = 1,\ f = \lambda y.x',\ x'' = 2] \\
f\ 0 & [x' = 1,\ f = \lambda y.x'',\ x'' = 2] \\
(\lambda y.x'')\ 0 & [x' = 1,\ f = \lambda y.x'',\ x'' = 2] \\
x'' & [x' = 1,\ f = \lambda y.x'',\ x'' = 2,\ y' = 0] \\
2 & [x' = 1,\ f = \lambda y.x'',\ x'' = 2,\ y' = 0]
\end{array}$$

□

Example 3.4 $(\mathsf{let\,rec}\ f = \lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ f(n-1) \times n\ \mathsf{in}\ f\ 3) \Downarrow_{\mathsf{ca}} 6$

Proof. In this example e stands for $\lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ f(n-1) \times n$.

$$
\begin{array}{lr}
\mathsf{let\,rec}\ f = \lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ f(n-1) \times n\ \mathsf{in}\ f\ 3 & [\,] \\
f\ 3 & [f = \lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ f(n-1) \times n] \\
\mathsf{if}\ n_1 = 0\ \mathsf{then}\ 1\ \mathsf{else}\ f(n_1 - 1) \times n_1 & [f = e,\ n_1 = 3] \\
(f\ 2) \times n_1 & [f = e,\ n_1 = 3] \\
(\mathsf{if}\ n_2 = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n_2 \times f(n_2 - 1)) \times n_1 & [f = e,\ n_1 = 3,\ n_2 = 2] \\
(f\ 1) \times n_2 \times n_1 & [f = e,\ n_1 = 3,\ n_2 = 2] \\
(\mathsf{if}\ n_3 = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n_3 \times f(n_3 - 1)) \times n_2 \times n_1 & \\
 & [f = e,\ n_1 = 3,\ n_2 = 2,\ n_3 = 1] \\
(f\ 0) \times n_3 \times n_2 \times n_1 & [f = e,\ n_1 = 3,\ n_2 = 2,\ n_3 = 3] \\
(\mathsf{if}\ n_4 = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n_4 \times f(n_4 - 1)) \times n_3 \times n_2 \times n_1 & \\
 & [f = e,\ n_1 = 3,\ n_2 = 2,\ n_3 = 1,\ n_4 = 0] \\
1 \times n_3 \times n_2 \times n_1 & [f = e,\ n_1 = 3,\ n_2 = 2,\ n_3 = 1,\ n_4 = 0] \\
6 & [f = e,\ n_1 = 3,\ n_2 = 2,\ n_3 = 1,\ n_4 = 0]
\end{array}
$$

□

3.2 Closures

3.2.1 Definitions

Closures were introduced in the language Scheme [15]. We present a version of them using a level of indirection, allowing us to handle mutable variables.

There is an unlimited number of locations $\ell, \ell_1, \ell_2 \ldots$; locations can be thought of as addresses in memory. An *environment* is a partial function from variables to locations. A *closure* is defined as a pair $\{\lambda x.e,\ \sigma\}$ such that $\mathsf{FV}(\lambda x.e) \subseteq \mathsf{dom}\,\sigma$, where $\lambda x.e$ is a λ-abstraction and σ is an environment that is used to interpret the free variables of $\lambda x.e$. A *value* is either a constant or a closure. Values for closures play the same role as irreducible terms for capsules. A *store* (or *memory*) is a partial function from locations to values.

Let $u, v, w, \ldots$ denote values, $\sigma, \tau, \ldots$ environments and $\mu, \nu, \xi, \chi, \ldots$ stores. Let Val be the set of values, Loc the set of locations and Cl the set of closures. Thus we have:

$$\sigma : \mathsf{Var} \rightharpoonup \mathsf{Loc} \qquad \mu : \mathsf{Loc} \rightharpoonup \mathsf{Val} \qquad \mathsf{Val} = \mathsf{Const} + \mathsf{Cl}$$

3.2.2 Semantics

A *state* is a triple $\langle e,\, \sigma,\, \mu\rangle$. A state is *valid* if and only if

$$\mathsf{FV}(e) \subseteq \mathsf{dom}\,\sigma \qquad \mathsf{codom}\,\sigma \subseteq \mathsf{dom}\,\mu$$
$$\forall\{\lambda x.a,\, \tau\} \in \mathsf{codom}\,\mu,\ \mathsf{FV}(\lambda x.a) \subseteq \mathsf{dom}\,\tau \wedge \mathsf{codom}\,\tau \subseteq \mathsf{dom}\,\mu$$

A *result* is a pair $(v,\, \mu)$. A result is *valid* if and only if either $v \in \mathsf{Const}$, or $v = \{\lambda x.a, \tau\} \in \mathsf{Cl}$ and the triple $\langle \lambda x.a,\, \tau,\, \mu\rangle$ is valid. We only consider valid states and results. Let us define a big step semantics where the operator $\Downarrow_{\mathsf{cl}}$ relates valid states to valid results. The semantics of features directly involving variables is given by:

$$\langle x,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mu(\sigma(x)),\, \mu) \qquad \langle \lambda x.e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\{\lambda x.e,\, \sigma\},\, \mu)$$

$$\frac{\langle e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (v,\, \xi)}{\langle x := e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} ((),\, \xi[\sigma(x)/v])}$$

$$\frac{\langle d,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\{\lambda x.a,\, \tau\},\, \xi) \qquad \langle e,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (v,\, \chi) \qquad \langle a,\, \tau[x/\ell],\, \chi[\ell/v]\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}{\langle d\ e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}\ (\ell \text{ fresh})$$

and the remaining semantics is:

$$\langle c,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (c,\, \mu) \qquad \frac{\langle d,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (f,\, \xi) \qquad \langle e,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (c,\, \nu)}{\langle d\ e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (f(c),\, \nu)}$$

$$\frac{\langle d,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} ((),\, \xi) \qquad \langle e,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}{\langle d; e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}$$

$$\frac{\langle b,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mathsf{true},\, \xi) \qquad \langle d,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}{\langle \mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}$$

$$\frac{\langle b,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mathsf{false},\, \xi) \qquad \langle e,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}{\langle \mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)}$$

$$\frac{\langle b,\, \sigma,\, \mu_i\rangle \Downarrow_{\mathsf{cl}} (\mathsf{true},\, \nu_i) \qquad \langle e,\, \sigma,\, \nu_i\rangle \Downarrow_{\mathsf{cl}} ((),\, \mu_{i+1}),\ 0 \le i < n,\ n \ge 0 \qquad \langle b,\, \sigma,\, \mu_n\rangle \Downarrow_{\mathsf{cl}} (\mathsf{false},\, \nu_n)}{\langle \mathsf{while}\ b\ \mathsf{do}\ e,\, \sigma,\, \mu_0\rangle \Downarrow_{\mathsf{cl}} ((),\, \nu_n)}$$

3.2.3 Examples

Example 3.5 $(\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f\ 0) \Downarrow_{\mathsf{cl}} 1$

Example 3.6 $(\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ x := 2; f\ 0) \Downarrow_{\mathsf{cl}} 2$

Example 3.7 $(\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ f = \lambda y.x\ \mathsf{in}\ \mathsf{let}\ x = 2\ \mathsf{in}\ f := \lambda y.x; f\ 0) \Downarrow_{\mathsf{cl}} 2$

Example 3.8 $(\mathsf{let\,rec}\ f = \lambda n.\mathsf{if}\ n = 0\ \mathsf{then}\ 1\ \mathsf{else}\ n \times f(n-1)\ \mathsf{in}\ f\ 3) \Downarrow_{\mathsf{cl}} 6$

4 Equivalence of the semantics

4.1 Definitions

There is a very strong correspondence between the semantics of closures and capsules. To give a precise account of this correspondence, we introduce an injective partial function $h : \mathsf{Loc} \rightharpoonup \mathsf{Var}$ with which we define four relations. Each relation is between an element of the semantics of closures and an element of the semantics of capsules that play similar roles:

- $v \overset{h}{\dashrightarrow} i$ between values and irreducible terms;
- $\mu \overset{h}{\dashrightarrow} \gamma$ between stores and capsule environments;
- $\langle d,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e,\, \gamma\rangle$ between states and capsules;
- $(v,\, \mu) \overset{h}{\sim} \langle i,\, \gamma\rangle$ between results and irreducible capsules.

One thing to notice is that nothing in the semantics of capsules plays the same role as the environment σ in the semantics of closures: capsule environments γ relate to memories μ, and environments σ have been simplified. Let us now give precise definitions of those relations.

Definition 4.1 Given a value v and an irreducible term i, we say that h *transforms v into i*, where h is an injective map $h : \mathsf{Loc} \rightharpoonup \mathsf{Var}$, and we write $v \overset{h}{\dashrightarrow} i$, if and only if:

- $v = i$ when $v \in \mathsf{Const}$, or
- $\mathsf{codom}\,\tau \subseteq \mathsf{dom}\,h$ and $(h \circ \tau)(\lambda x.a) = i$ when $v = \{\lambda x.a,\, \tau\} \in \mathsf{Cl}$

Definition 4.2 Given a store μ and a capsule environment γ, we say that h *transforms μ into γ*, where h is an injective map $h : \mathsf{Loc} \rightharpoonup \mathsf{Var}$, and we write $\mu \overset{h}{\dashrightarrow} \gamma$, if and only if:

$$\mathsf{dom}\,h = \mathsf{dom}\,\mu \qquad h(\mathsf{dom}\,\mu) = \mathsf{dom}\,\gamma$$
$$\forall \ell \in \mathsf{dom}\,\mu,\ \mu(\ell) \overset{h}{\dashrightarrow} \gamma(h(\ell))$$

Definition 4.3 Given a state $\langle d,\, \sigma,\, \mu\rangle$ and a capsule $\langle e,\, \gamma\rangle$, both valid, we say that they are *bisimilar under h*, where h is an injective map $h : \mathsf{Loc} \rightharpoonup \mathsf{Var}$, and we write $\langle d,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e,\, \gamma\rangle$, if and only if

$$(h \circ \sigma)(d) = e \qquad\qquad \mu \overset{h}{\dashrightarrow} \gamma$$

Definition 4.4 Given a result $(v,\, \mu)$ and an irreducible capsule $\langle i,\, \gamma\rangle$, both valid, we say that they are *bisimilar under h*, where h is an injective map $h : \mathsf{Loc} \rightharpoonup \mathsf{Var}$, and we write $(v,\, \mu) \overset{h}{\sim} \langle i,\, \gamma\rangle$ if and only if:

$$v \xrightarrow{h} i \qquad\qquad \mu \xrightarrow{h} \gamma$$

4.2 Soundness of Capsules with respect to Closures

Now that we know how to relate each element of both semantics, theorem 4.5 shows that any derivation using capsules mirrors a derivation using closures, and vice-versa:

Theorem 4.5 *If $\langle d,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e,\, \gamma\rangle$ then $\langle d,\, \sigma,\, \mu\rangle {\Downarrow_{\mathsf{cl}}} (u,\, \nu)$ for some u, ν if and only if $\langle e,\, \gamma\rangle {\Downarrow_{\mathsf{ca}}} \langle i,\, \delta\rangle$ for some i, δ, and in that case we have*

$$(u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

where g is an extension of h, i.e., $\mathsf{dom}\, h \subseteq \mathsf{dom}\, g$ and h and g agree on $\mathsf{dom}\, h$.

Proof. We show the direct implication by induction on the big-step derivation of $\langle d,\, \sigma,\, \mu\rangle {\Downarrow_{\mathsf{cl}}} (u,\, \nu)$ and the converse by induction on the big-step derivation of $\langle e,\, \gamma\rangle {\Downarrow_{\mathsf{ca}}} \langle i,\, \delta\rangle$.

In the interest of space, we only show the most interesting cases of the induction in the main text: variable call x, λ-abstraction $\lambda x.e$, function application of a λ-abstraction $d\ e$ where d reduces to a λ-abstraction, and variable assignment $x := e$. In all these cases, both implications are very similar proofs, therefore we only show the direct implication ($\Rightarrow$). The other cases, constant c, function application of a constant function $d\ e$ where d reduces to a constant f, composition $d; e$, if conditional $\mathsf{if}\ b\ \mathsf{then}\ d\ \mathsf{else}\ e$ and while loop $\mathsf{while}\ b\ \mathsf{do}\ e$, are detailed in the appendix.

Variable call

If $d = x$ for some variable x then $e = (h \circ \sigma)(d) = y$ with y the variable such that $y = (h \circ \sigma)(x)$.

($\Rightarrow$) By definition of $\Downarrow_{\mathsf{cl}}$, $(u,\, \nu) = (\mu(\sigma(x)),\, \mu)$, and by definition of $\Downarrow_{\mathsf{ca}}$, $\langle e,\, \gamma\rangle = \langle y,\, \gamma\rangle {\Downarrow_{\mathsf{ca}}} \langle \gamma(y),\, \gamma\rangle$. Moreover $\mu \xrightarrow{h} \gamma$, therefore by definition of $\xrightarrow{h}$, $\mu(\sigma(x)) \xrightarrow{h} \gamma(h(\sigma(x))) = \gamma(y)$. Therefore, with $g = h$, $(u,\, \nu) = (\mu(\sigma(x)),\, \mu) \overset{g}{\sim} \langle \gamma(y),\, \gamma\rangle$ which completes this case.

λ-Abstraction

If $d = \lambda x.a$, then $e = (h \circ \sigma)(\lambda x.a)$ which is a term α-equivalent to d, so $e = \lambda x.b$ for some b. Indeed, the variable x does not change from d to e since only the free variables of d are affected by $h \circ \sigma$.

($\Rightarrow$) By definition of $\Downarrow_{\mathsf{cl}}$, $(u,\, \nu) = (\{\lambda x.a,\, \sigma\},\, \mu)$, and by definition of $\Downarrow_{\mathsf{ca}}$, $\langle e,\, \gamma\rangle = \langle \lambda x.b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \lambda x.b,\, \gamma\rangle$. But $\mathsf{codom}\,\sigma \subseteq \mathsf{dom}\,h$ and $\lambda x.b = (h \circ \sigma)(\lambda x.a)$, therefore $\{\lambda x.a,\, \sigma\} \overset{h}{\to} \lambda x.b$. Moreover we know $\mu \overset{h}{\to} \gamma$ and with $g = h$, we get $(\{\lambda x.a,\, \sigma\},\, \mu) \overset{g}{\sim} \langle \lambda x.b,\, \gamma\rangle$ which completes this case.

Function application of a λ-abstraction

If $d = d_1\ d_2$, then let $e_1 = (h\circ\sigma)(d_1)$ and $e_2 = (h\circ\sigma)(d_2)$. Since $e = (h\circ\sigma)(d)$ means that e is α-equivalent to d, $e = e_1\ e_2$, and we can easily check that $\langle d_1,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e_1,\, \gamma\rangle$ and $\langle d_2,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e_2,\, \gamma\rangle$.

($\Rightarrow$) If $\langle d_1\ d_2,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)$ because

$$\langle d_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\{\lambda x.a,\, \tau\},\, \xi) \qquad \langle d_2,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (v,\, \chi) \qquad \langle a,\, \tau[x/\ell],\, \chi[\ell/v]\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)$$

with ℓ fresh, then by induction hypothesis on the derivation of d_1, there exist k, ζ and h_1 an extension of h such that

$$\langle e_1,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle k,\, \zeta\rangle \qquad\qquad (\{\lambda x.a,\, \tau\},\, \xi) \overset{h_1}{\sim} \langle k,\, \zeta\rangle$$

The second condition implies that $k = \lambda x.b = (h_1 \circ \tau)(\lambda x.a)$ for some expression b, and that $\xi \overset{h_1}{\to} \zeta$. Moreover $d_2 \overset{h_1}{\to} e_2$ since $d_2 \overset{h}{\to} e_2$, therefore $\langle d_2,\, \sigma,\, \xi\rangle \overset{h_1}{\sim} \langle e_2,\, \zeta\rangle$. By induction hypothesis on the derivation of d_2, there exist j, η and h_2 an extension of h_1 such that

$$\langle e_2,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \eta\rangle \qquad\qquad (v,\, \chi) \overset{h_2}{\sim} \langle j,\, \eta\rangle$$

As ℓ is the fresh location chosen in the derivation of $\Downarrow_{\mathsf{cl}}$ for d, let y be a fresh variable for the derivation of $\Downarrow_{\mathsf{ca}}$ for e. Let $h_3 : \mathsf{Loc} \rightharpoonup \mathsf{Var}$ such that:

$$\begin{aligned} h_3 : \mathsf{dom}\,h_2 \cup \{\ell\} &\to \mathsf{codom}\,h_2 \cup \{y\} \\ \ell_2 \in \mathsf{dom}\,h_2 &\mapsto h_2(\ell_2) \\ \ell &\mapsto y \end{aligned}$$

Lemma 4.6 $\langle a,\, \tau[x/\ell],\, \chi[\ell/v]\rangle \overset{h_3}{\sim} (b[x/y],\, \eta[y/j])$

Proof. First of all, $\lambda x.b = (h_1 \circ \tau)(\lambda x.a)$, h_3 is an extension of h_1 and $\mathsf{FV}(\lambda x.a) \subseteq \mathsf{dom}\,h_1$, therefore $\lambda x.b = (h_3 \circ \tau)(\lambda x.a)$. Now $b[x/y] = ((h_3 \circ \tau)[x/y])(\lambda x.a) = (h_3 \circ \tau[x/\ell])(\lambda x.a)$ since $h_3(\ell) = y$.

We further need to argue that $\chi[\ell/v] \overset{h_3}{\to} \eta[y/j]$. We already know that $\mathsf{dom}\,h_3 = \mathsf{dom}\,h_2 \cup \{\ell\} = \mathsf{dom}\,\chi \cup \{\ell\} = \mathsf{dom}\,\chi[\ell/v]$, and $h_3(\mathsf{dom}\,\chi[\ell/v]) = \mathsf{codom}\,h_2 \cup \{y\} = \mathsf{dom}\,\eta[y/j]$. Let $\ell_3 \in \mathsf{dom}\,\chi[\ell/v]$. If $\ell_3 \in \mathsf{dom}\,\chi$, then $\chi[\ell/v](\ell_3) = \chi(\ell_3) \overset{h_2}{\to} \eta(h_3(\ell_3)) = \eta[y/j](h_3(\ell_3))$ by injectivity of h_3, therefore $\chi[\ell/v](\ell_3) \overset{h_3}{\to} \eta[y/j](h_3(\ell_3))$.

Otherwise, $\ell_3 = \ell$ and then $\chi[\ell/v](\ell) = v \xrightarrow{h_2} j = \eta[y/j](y) = \eta[y/j](h_3(\ell))$, therefore since h_3 is an extension of h_2, $\chi[\ell/v](\ell) \xrightarrow{h_3} \eta[y/j](h_3(\ell))$. This completes the proof of the lemma. □

Using lemma 4.6 and by induction hypothesis on the derivation of a, there exist i, δ and g an extension of h_3 such that

$$\langle b[x/y],\, \eta[y/j]\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle \qquad\qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

Therefore, by definition of $\Downarrow_{\mathsf{cl}}$, $\langle e_1\ e_2,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle$ and $(u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$, which completes this case.

Variable assignment

If $d = (x := d_1)$ for some variable x and expression d_1, then $e = (h \circ \sigma)(x := d_1) = (y := e_1)$ with y a variable such that $y = (h \circ \sigma)(x)$ and $e_1 = (h \circ \sigma)(d_1)$. Therefore $\langle d_1,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e_1,\, \gamma\rangle$.

($\Rightarrow$) The derivation of $\Downarrow_{\mathsf{cl}}$ for d shows that $(u,\, \nu) = ((),\, \xi[\sigma(x)/v])$ for some v, ξ such that

$$\langle e_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (v,\, \xi)$$

By induction hypothesis on the derivation of $\Downarrow_{\mathsf{cl}}$ for d_1, there exist j, ζ and g an extension of h such that

$$\langle e_1,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \zeta\rangle \qquad\qquad (v,\, \xi) \overset{g}{\sim} \langle j,\, \zeta\rangle$$

Lemma 4.7 $((),\, \xi[\sigma(x)/v]) \overset{g}{\sim} \langle (),\, \zeta[y/j]\rangle$

Proof. The domain conditions are fulfilled since $(v,\, \xi) \overset{g}{\sim} \langle j,\, \zeta\rangle$, $\mathsf{dom}\,\xi = \mathsf{dom}\,\xi[\sigma(x)/v]$ and $\mathsf{dom}\,\zeta = \mathsf{dom}\,\zeta[y/j]$. Let $\ell \in \mathsf{dom}\,\xi[\sigma(x)/v] = \mathsf{dom}\,\xi$. If $\ell = \sigma(x)$ then $\xi[\sigma(x)/v](\ell) = v \overset{g}{\sim} j = \zeta[y/j](y) = \zeta[y/j](g(\ell))$ since $g(\ell) = (g \circ \sigma)(x) = (h \circ \sigma)(x) = y$. Otherwise $\xi[\sigma(x)/v](\ell) = \xi(\ell) \overset{g}{\sim} \zeta(h(\ell)) = \zeta[y/j](g(\ell))$ using that h is injective and g is an extension of h. Finally $() \xrightarrow{g} ()$, which completes the proof of the lemma. □

Using lemma 4.7 and by definition of $\Downarrow_{\mathsf{ca}}$, $\langle x := e_1,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle (),\, \zeta[y/j]\rangle$ and $\langle u,\, \nu\rangle = ((),\, \xi[\sigma(x)/v]) \overset{g}{\sim} \langle (),\, \zeta[y/j]\rangle$, which completes this case.

The other cases are proved in the appendix.

□

5 Capsules encode less information

When evaluating an expression using capsules, less information is kept than when evaluating the same expression using closures. Intuitively, when using closures, the state of the computation keeps track of exactly what variables of a λ-abstraction are in scope, even if those variables do not appear in the λ-abstraction itself and will therefore never be used. When using capsules however, the capsule only keeps track of the variables that are both in scope and appear in the λ-abstraction.

For example, let us evaluate the expressions $d = (\mathsf{let}\ x = 1\ \mathsf{in}\ \mathsf{let}\ y = \lambda y.0\ \mathsf{in}\ y)$ and $e = (\mathsf{let}\ y = \lambda y.0\ \mathsf{in}\ \mathsf{let}\ x = 1\ \mathsf{in}\ y)$. Using the definitions of $\Downarrow_{\mathsf{cl}}$ and $\Downarrow_{\mathsf{ca}}$, we can prove that:

$$\begin{aligned}
&d \Downarrow_{\mathsf{cl}} (\{\lambda y.0,\ [x = \ell_1]\},\ [\ell_1 = 1, \ell_2 = \{\lambda y.0,\ [x = 1]\}])\\
&e \Downarrow_{\mathsf{cl}} (\{\lambda y.0,\ [\,]\},\ [\ell_1 = 1, \ell_2 = \{\lambda y.0,\ [\,]\}])\\
&d \Downarrow_{\mathsf{ca}} \langle \lambda y.0,\ [x' = 1, y' = \lambda y.0]\rangle\\
&e \Downarrow_{\mathsf{ca}} \langle \lambda y.0,\ [x' = 1, y' = \lambda y.0]\rangle
\end{aligned}$$

On this example, the result of evaluating d and e with $\Downarrow_{\mathsf{cl}}$ keeps track of whether x is in scope or not, but evaluating d and e with $\Downarrow_{\mathsf{ca}}$ does not. This information is completely superfluous for the rest of the computation and suppressing it with capsules avoids some overhead. Propositions 5.1 to 5.4 give a more precise account of what is happening.

Proposition 5.1 *If $v \xrightarrow{h} i$ then given h, i can be uniquely determined from v; the converse is not true.*

Proof. If $v \xrightarrow{h} i_1$ and $v \xrightarrow{h} i_2$ then either:

- $v \in \mathsf{Const}$ and then $v = i_1$ and $v = i_2$ thus $i_1 = i_2$;
- $v = \{\lambda x.a,\ \tau\} \in \mathsf{Cl}$ and then $i_1 = (h \circ \tau)(\lambda x.a)$ and $i_2 = (h \circ \tau)(\lambda x.a)$ thus $i_1 = i_2$.

However, $\{\lambda y.0,\ [\,]\} \xrightarrow{h} (\lambda y.0)$ and $\{\lambda y.0,\ [x = \ell]\} \xrightarrow{h} (\lambda y.0)$. □

Proposition 5.2 *If $\mu \xrightarrow{h} \gamma$ then given h, γ can be uniquely determined from μ; the converse is not true.*

Proof. If $\mu \xrightarrow{h} \gamma_1$ and $\mu \xrightarrow{h} \gamma_2$ then $\mathsf{dom}\,\gamma_1 = h(\mathsf{dom}\,\mu) = \mathsf{dom}\,\gamma_2$. Moreover, for all $\ell \in \mathsf{dom}\, mu$, $\mu(\ell) \xrightarrow{h} \gamma_1(h(\ell))$ and $\mu(\ell) \xrightarrow{h} \gamma_2(h(\ell))$ therefore using proposition 5.1, $\gamma_1(h(\ell)) = \gamma_2(h(\ell))$. This covers all the domain of γ_1 and γ_2 since $\mathsf{dom}\,\gamma_1 = \mathsf{dom}\,\gamma_2 = h(\mathsf{dom}\,\mu)$.

However, with h transforming ℓ in z, $[\ell = \{\lambda y.0,\ [\,]\}] \xrightarrow{h} [z = \lambda y.0]$ and $[\ell = \{\lambda y.0,\ [x = \ell]\}] \xrightarrow{h} [z = \lambda y.0]$ □

Proposition 5.3 *If $\langle d, \sigma, \mu\rangle \overset{h}{\sim} \langle e, \gamma\rangle$ then given h, $\langle e, \gamma\rangle$ can be uniquely determined from $\langle d, \sigma, \mu\rangle$; the converse is not true.*

Proof. If $\langle d, \sigma, \mu\rangle \overset{h}{\sim} \langle e_1, \gamma_1\rangle$ and $\langle d, \sigma, \mu\rangle \overset{h}{\sim} \langle e_2, \gamma_2\rangle$, then $(h \circ \sigma(d)) = e_1$ and $(h \circ \sigma(d)) = e_2$ therefore $e_1 = e_2$. Moreover $\mu \overset{h}{\to} \gamma_1$ and $\mu \overset{h}{\to} \gamma_2$ therefore using proposition 5.2, $\gamma_1 = \gamma_2$.

However, with h transforming ℓ in z,

$$\langle x, [x = \ell], [\ell = \{\lambda y.0, [\,]\}]\rangle \overset{h}{\sim} \langle z, [z = \lambda y.0]\rangle$$
$$\langle x, [x = \ell], [\ell = \{\lambda y.0, [x = \ell]\}]\rangle \overset{h}{\sim} \langle z, [z = \lambda y.0]\rangle$$

□

Proposition 5.4 *If $(v, \mu) \overset{h}{\sim} \langle i, \gamma\rangle$ then given h, $\langle i, \gamma\rangle$ can be uniquely determined from (v, μ); the converse is not true.*

Proof. The unicity of $\langle i, \gamma\rangle$ is a direct consequence of propositions 5.1 and 5.2. However,

$$(\{\lambda y.0, [\,]\}, [\,]) \overset{h}{\sim} \langle \lambda y.0, [\,]\rangle$$
$$(\{\lambda y.0, [x = \ell]\}, [\ell = 1]) \overset{h}{\sim} \langle \lambda y.0, [\,]\rangle$$

□

The idea behind those propositions is that for every capsule, there are several bisimilar states corresponding to different computations, and each keeping track of a different set of superfluous information. Similarly, for every irreducible capsules, there are several bisimilar results keeping track of superfluous information. Capsules thus offer a much cleaner representation of the state of computation.

6 Discussion

6.1 Capsules and Closures: a strong correspondence

Theorem 4.5 shows that capsules and closures are very strongly related. Not only is there a derivation based on capsules for every derivation based on closures, but these two derivations mirror each other. This is because each rule of the definition of $\Downarrow_{\mathsf{ca}}$ mirrors a rule of the definition of $\Downarrow_{\mathsf{cl}}$, and because the proof of the theorem is a direct structural induction on the definitions of $\Downarrow_{\mathsf{cl}}$ and $\Downarrow_{\mathsf{ca}}$. Thus the computations are completely bisimilar, even though definining computations for capsules is simpler.

6.2 Capsules allow to suppress the environment σ

When using closures, a state is a triple $\langle d,\, \sigma,\, \mu\rangle$ whereas when using capsules, it is just a capsule $\langle e,\, \gamma\rangle$. It they are bisimilar under h, it means that $(h \circ \sigma)(d) = e$ and $\mu \xrightarrow{h} \gamma$. Really, capsules eliminate the need for the environment σ and thus suppress the indirection in closures that was needed to handle imperative features. Moreover, the initial idea between the capsule environment γ was that it would replace the (closure) environment σ. However, it is remarkable that γ is much closer to the store μ, while at the same time eliminates the need for the (closure) environment σ.

6.3 A simple small-step semantics for capsules

When establishing theorem 4.5, we tried to build a small-step semantics for closures and capsules. We only present here what happens on the rule for the application $(d\ e)$ when d has already been reduced to a λ-term and e to a value, as all the other rules are reasonably straightforward.

Using closures, we are trying to take the next small step in the state $\langle\{\lambda x.a,\, \tau\}\ v,\, \sigma,\, \mu\rangle$. We would like to write something like:

$$\langle\{\lambda x.a,\, \tau\}\ v,\, \sigma,\, \mu\rangle \to_{\mathsf{cl}} \langle a,\, \tau[x/\ell],\, \mu[\ell/v]\rangle \qquad (\ell \text{ fresh})$$

This rule is wrong: it drops the environment σ, but when this evaluation is in context, σ has to come back once we finish evaluating a. One solution is to write a rule involving several small steps, which is really a big step rule. Another solution is to keep track of the whole stack of environments to come back to the previous environment each time we get out of a scope (see [24]).

Using capsules however, the following rule comes very naturally:

$$\langle(\lambda x.a)\ i,\, \gamma\rangle \to_{\mathsf{ca}} \langle a[x/y],\, \gamma[y/i]\rangle \qquad (y \text{ fresh})$$

Along with the other small-step rules, this shows that the capsule semantics is fully relational and does not need any stack or auxiliary data structure.

References

[1] J.-B. Jeannin and D. Kozen, “Computing with capsules,” Computing and Information Science, Cornell University, Tech. Rep. http://hdl.handle.net/1813/22082, January 2011.

[2] I. Mason and C. Talcott, “Equivalence in functional languages with effects,” 1991.

[3] ——, “Programming, transforming, and proving with function abstractions and memories.”

[4] ——, “Axiomatizing operational equivalence in the presence of side effects,” in *Fourth Annual Symposium on Logic in Computer Science. IEEE.* IEEE Computer Society Press, 1989, pp. 284–293.

[5] M. Felleisen and R. Hieb, "The revised report on the syntactic theories of sequential control and state," *Theoretical Computer Science*, vol. 103, pp. 235–271, 1992.

[6] K. Aboul-Hosn, "Programming with private state," Honors Thesis, The Pennsylvania State University, December 2001. [Online]. Available: http://www.cs.cornell.edu/%7Ekamal/thesis.pdf

[7] K. Aboul-Hosn and D. Kozen, "Relational semantics of local variable scoping," Cornell University, Tech. Rep. 2005-2000, 2005. [Online]. Available: http://www.cs.cornell.edu/%7Ekamal/local.pdf

[8] E. Moggi, "Notions of computation and monads," *Information and Computation*, vol. 93, no. 1, 1991.

[9] R. Milne and C. Strachey, *A Theory of Programming Language Semantics.* New York, NY, USA: Halsted Press, 1977.

[10] D. Scott, "Mathematical concepts in programmng language semantics," in *Proc. 1972 Spring Joint Computer Conferences.* Montvale, NJ: AFIPS Press, 1972, pp. 225–34.

[11] J. E. Stoy, *Denotational Semantics: The Scott-Strachey Approach to Programming Language Theory.* Cambridge, MA, USA: MIT Press, 1981.

[12] J. Y. Halpern, A. R. Meyer, and B. A. Trakhtenbrot, "The semantics of local storage, or what makes the free-list free?" in *Proc. 11th ACM Symp. Principles of Programming Languages (POPL'84)*, New York, NY, USA, 1984, pp. 245–257.

[13] P. J. Landin, "The mechanical evaluation of expressions," *Computer Journal*, vol. 6, no. 5, pp. 308–320, 1964.

[14] ——, "The next 700 programming languages," *Commun. ACM*, vol. 9, pp. 157–166, March 1966. [Online]. Available: http://doi.acm.org/10.1145/365230.365257

[15] G. J. Sussman and G. L. Steele, "Scheme: A interpreter for extended lambda calculus," *Higher-Order and Symbolic Computation*, vol. 11, pp. 405–439, 1998, 10.1023/A:1010035624696. [Online]. Available: http://dx.doi.org/10.1023/A:1010035624696

[16] I. A. Mason and C. L. Talcott, "References, local variables and operational reasoning," in *Seventh Annual Symposium on Logic in Computer Science.* IEEE, 1992, pp. 186–197. [Online]. Available: http://www-formal.stanford.edu/MT/92lics.ps.Z

[17] A. M. Pitts and I. D. B. Stark, "Observable properties of higher order functions that dynamically create local names, or what's new?" in *MFCS*, ser. Lecture Notes in Computer Science, A. M. Borzyszkowski and S. Sokolowski, Eds., vol. 711. Springer, 1993, pp. 122–141.

[18] A. M. Pitts, "Operationally-based theories of program equivalence," in *Semantics and Logics of Computation*, ser. Publications of the Newton Institute, P. Dybjer and A. M. Pitts, Eds. Cambridge University Press, 1997, pp. 241–298. [Online]. Available: http://www.cs.tau.ac.il/~nachumd/formal/exam/pitts.pdf

[19] A. M. Pitts and I. D. B. Stark, "Operational reasoning in functions with local state," in *Higher Order Operational Techniques in Semantics*, A. D. Gordon and A. M. Pitts, Eds. Cambridge University Press, 1998, pp. 227–273. [Online]. Available: http://homepages.inf.ed.ac.uk/stark/operfl.pdf

[20] S. Abramsky, K. Honda, and G. McCusker, "A fully abstract game semantics for general references," in *LICS '98: Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science.* Washington, DC, USA: IEEE Computer Society, 1998, pp. 334–344.

[21] J. Laird, "A game semantics of local names and good variables." in *FoSSaCS*, ser. Lecture Notes in Computer Science, I. Walukiewicz, Ed., vol. 2987. Springer, 2004, pp. 289–303.

[22] S. Abramsky and G. McCusker, "Linearity, sharing and state: a fully abstract game semantics for idealized ALGOL with active expressions." *Electr. Notes Theor. Comput. Sci.*, vol. 3, 1996.

[23] G. Winskel, *The Formal Semantics of Programming Languages.* MIT Press, 1993.

[24] K. Aboul-Hosn and D. Kozen, "Relational semantics for higher-order programs," in *Proc. 8th Int. Conf. Mathematics of Program Construction (MPC'06)*, ser. Lecture Notes in Computer Science, T. Uustalu, Ed., vol. 4014. Springer, July 2006, pp. 29–48.

A Appendix: Proof of theorem 4.5

We include here the cases we have not included in the main text.

Variable call

($\Leftarrow$) The converse is similar. By definition of $\Downarrow_{\mathsf{ca}}$, $\langle i,\, \delta\rangle = \langle \gamma(y),\, \gamma\rangle$, and by definition of $\Downarrow_{\mathsf{cl}}$, $\langle d,\, \sigma,\, \mu\rangle = \langle x,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mu(\sigma(x)),\, \mu)$. Moreover $\mu \overset{h}{\to} \gamma$, therefore by definition of $\overset{h}{\to}$, $\mu(\sigma(x)) \overset{h}{\to} \gamma(h(\sigma(x))) = \gamma(y)$. Therefore, with $g = h$, $(\mu(\sigma(x)),\, \mu) \overset{g}{\sim} \langle \gamma(y),\, \gamma\rangle = \langle i,\, \delta\rangle$ which completes this case.

λ-Abstraction

($\Leftarrow$) The converse is similar. By definition of $\Downarrow_{\mathsf{ca}}$, $\langle i,\, \delta\rangle = \langle \lambda x.b,\, \gamma\rangle$, and by definition of $\Downarrow_{\mathsf{cl}}$, $\langle d,\, \sigma,\, \mu\rangle = \langle \lambda x.a,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\{\lambda x.a,\, \sigma\},\, \mu)$. But $\mathsf{codom}\,\sigma \subseteq \mathsf{dom}\,h$ and $\lambda x.b = (h \circ \sigma)(\lambda x.a)$, therefore $\{\lambda x.a,\, \sigma\} \overset{h}{\to} \lambda x.b$. Moreover we know $\mu \overset{h}{\to} \gamma$ and with $g = h$, we get $(\{\lambda x.a,\, \sigma\},\, \mu) \overset{g}{\sim} \langle \lambda x.b,\, \gamma\rangle$ which completes this case.

Function application of a λ-abstraction

($\Leftarrow$) The converse is similar. If $\langle e_1\ e_2,\, \gamma\rangle \Downarrow_{\mathsf{cl}} \langle i,\, \delta\rangle$ because

$$\langle e_1,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \lambda x.b,\, \zeta\rangle \qquad \langle e_2,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \eta\rangle \qquad \langle b[x/y],\, \eta[y/j]\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle$$

with y fresh, then by induction hypothesis on the derivation of e_1, there exist w, ξ and h_1 an extension of h such that

$$\langle d_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{ca}} (w,\, \xi) \qquad\qquad (w,\, \xi) \overset{h_1}{\sim} \langle \lambda x.b,\, \zeta\rangle$$

The second condition implies that $w = \{\lambda x.a,\, \tau\}$ for some a, τ such that $(h_1 \circ \tau)(\lambda x.a) = \lambda x.b$, and that $\xi \overset{h_1}{\to} \zeta$. Moreover $d_2 \overset{h_1}{\to} e_2$ since $d_2 \overset{h}{\to} e_2$, therefore $\langle d_2,\, \sigma,\, \xi\rangle \overset{h_1}{\sim} \langle e_2,\, \zeta\rangle$. By induction hypothesis on the derivation of e_2, there exist v, χ and h_2 an extension of h_1 such that

$$\langle d_2,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{ca}} (v,\, \chi) \qquad\qquad (j,\, \eta) \overset{h_2}{\sim} (v,\, \chi)$$

As y is the fresh variable chosen in the derivation of $\Downarrow_{\mathsf{ca}}$ for e, let ℓ be a fresh location for the derivation of $\Downarrow_{\mathsf{cl}}$ for d. Let $h_3 : \mathsf{Loc} \rightharpoonup \mathsf{Var}$ such that:

$$\begin{gathered} h_3 : \mathsf{dom}\,h_2 \cup \{\ell\} \to \mathsf{codom}\,h_2 \cup \{y\} \\ \ell_2 \in \mathsf{dom}\,h_2 \mapsto h_2(\ell_2) \\ \ell \mapsto y \end{gathered}$$

Lemma A.1 $\langle a,\, \tau[x/\ell],\, \chi[\ell/v]\rangle \overset{h_3}{\sim} (b[x/y],\, \eta[y/j])$

Proof. This is the same as lemma 4.6, and the same proof holds. □

Using lemma A.1 and by induction hypothesis on the derivation of $b[x/y]$, there exist u, ν and g an extension of h_3 such that

$$\langle a,\, \tau[x/\ell],\, \chi[\ell/v]\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu) \qquad\qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

Therefore, by definition of $\Downarrow_{\mathsf{cl}}$,

$$\langle d_1\ d_2,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu) \qquad\qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

which completes this case.

Variable assignment

($\Leftarrow$) The converse is similar. The derivation of $\Downarrow_{\mathsf{ca}}$ for e shows that $\langle i,\, \delta\rangle = \langle (),\, \zeta[x/j]\rangle$ for some j, ζ such that

$$\langle e_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (v,\, \xi)$$

By induction hypothesis on the derivation of $\Downarrow_{\mathsf{ca}}$ for e_1, there exists v, ξ and g an extension of h such that

$$\langle d_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{ca}} \langle v,\, \xi\rangle \qquad\qquad (v,\, \xi) \overset{g}{\sim} \langle j,\, \zeta\rangle$$

Lemma A.2 $((),\, \xi[\sigma(x)/v]) \overset{g}{\sim} \langle (),\, \zeta[y/j]\rangle$

Proof. This is the same as lemma 4.7, and the same proof holds. □

Using lemma A.2 and by definition of $\Downarrow_{\mathsf{ca}}$,

$$\langle x := d_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} ((),\, \xi[\sigma(x)/v]) \qquad ((),\, \xi[\sigma(x)/v]) \overset{g}{\sim} \langle (),\, \zeta[y/j]\rangle = \langle i,\, \delta\rangle$$

which completes this case.

Constant

If $d = c$ then $e = (h \circ \sigma)(d) = c$ as well.

($\Rightarrow$) The derivation of $\Downarrow_{\mathsf{cl}}$ shows that $(u,\, \nu) = (c,\, \mu)$, and the derivation of $\Downarrow_{\mathsf{ca}}$ shows that $\langle e,\, \gamma\rangle = \langle c,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle c,\, \gamma\rangle$. Moreover $\mu \xrightarrow{h} \gamma$, therefore with $g = h$, $(c,\, \mu) \overset{g}{\sim} \langle c,\, \gamma\rangle$ which completes this case.

($\Leftarrow$) The derivation of $\Downarrow_{\mathsf{ca}}$ shows that $\langle i,\, \delta\rangle = \langle c,\, \gamma\rangle$, and the derivation of $\Downarrow_{\mathsf{ca}}$ shows that $\langle d,\, \sigma,\, \mu\rangle = \langle c,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (c,\, \mu)$. Moreover $\mu \xrightarrow{h} \gamma$, therefore with $g = h$, $(c,\, \mu) \overset{g}{\sim} \langle c,\, \gamma\rangle$ which completes this case.

Function application of a constant function

($\Rightarrow$) If $\langle d_1\ d_2,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)$ because

$$\langle d_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (f,\, \xi) \qquad\qquad \langle d_2,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (c,\, \nu) \qquad\qquad u = f(c)$$

then, recalling that $\langle d_1,\, \sigma,\, \mu\rangle \overset{h}{\sim} (e_1,\, \gamma)$, by induction hypothesis on the derivation of d_1, there exist j, ζ and h_1 an extension of h such that

$$\langle e_1,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \zeta\rangle \qquad\qquad (f,\, \xi) \overset{h_1}{\sim} \langle j,\, \zeta\rangle$$

The second condition implies $j = f$ and $\xi \xrightarrow{h_1} \zeta$. Moreover $d_2 \xrightarrow{h_1} e_2$ since $d_2 \xrightarrow{h} e_2$, therefore $\langle d_2,\, \sigma,\, \xi\rangle \overset{h_1}{\sim} \langle e_2,\, \zeta\rangle$. By induction hypothesis on the derivation of d_2, there exist k, δ and g an extension of h_1 such that

$$\langle e_2,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle k,\, \delta\rangle \qquad\qquad (c,\, \nu) \overset{g}{\sim} \langle k,\, \delta\rangle$$

The second condition implies $k = c$ and $\nu \xrightarrow{g} \delta$. Therefore, by definition of $\Downarrow_{\mathsf{ca}}$,

$$\langle e_1\ e_2,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle f(c),\, \delta\rangle \qquad\qquad (f(c),\, \nu) \overset{g}{\sim} \langle f(c),\, \delta\rangle$$

which completes this case.

($\Leftarrow$) If $\langle e_1\ e_2,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle$ because

$$\langle e_1,\, \gamma\rangle \Downarrow_{\mathsf{cl}} \langle f,\, \zeta\rangle \qquad\qquad \langle e_2,\, \zeta\rangle \Downarrow_{\mathsf{cl}} \langle c,\, \delta\rangle \qquad\qquad u = f(c)$$

then, recalling that $\langle d_1,\, \sigma,\, \mu\rangle \overset{h}{\sim} (e_1,\, \gamma)$, by induction hypothesis on the derivation of e_1, there exist v, ξ and h_1 an extension of h such that

$$\langle d_1,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (v,\, \xi) \qquad\qquad (v,\, \xi) \overset{h_1}{\sim} \langle f,\, \zeta\rangle$$

The second condition implies $v = f$ and $\xi \xrightarrow{h_1} \zeta$. Moreover $d_2 \xrightarrow{h_1} e_2$ since $d_2 \xrightarrow{h} e_2$, therefore $\langle d_2,\, \sigma,\, \xi\rangle \overset{h_1}{\sim} \langle e_2,\, \zeta\rangle$. By induction hypothesis on the derivation of e_2, there exist w, ν and g an extension of h_1 such that

$$\langle d_2,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{ca}} (w,\, \nu) \qquad\qquad (w,\, \nu) \overset{g}{\sim} \langle c,\, \delta\rangle$$

The second condition implies $w = c$ and $\nu \xrightarrow{g} \delta$. Therefore, by definition of $\Downarrow_{\mathsf{ca}}$,

$$\langle d_1\ d_2,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{ca}} \langle f(c),\, \delta\rangle \qquad\qquad (f(c),\, \nu) \overset{g}{\sim} \langle f(c),\, \delta\rangle$$

which completes this case.

Composition

If $d = (d_1; d_2)$, then $e = (e_1; e_2)$ for $e_1 = (h \circ \sigma)(d_1)$ and $e_2 = (h \circ \sigma)(d_2)$, therefore $\langle d_1, \sigma, \mu\rangle \overset{h}{\sim} \langle e_1, \gamma\rangle$ and $\langle d_2, \sigma, \mu\rangle \overset{h}{\sim} \langle e_2, \gamma\rangle$.

($\Leftarrow$) The derivation of $\Downarrow_{\mathsf{cl}}$ for d shows that

$$\langle d_1, \sigma, \mu\rangle \Downarrow_{\mathsf{cl}} ((), \xi) \qquad \langle d_2, \sigma, \xi\rangle \Downarrow_{\mathsf{cl}} (u, \nu)$$

for some ξ. By induction hypothesis on the derivation of d_1, there exist j, ζ and h_1 an extension of h such that

$$\langle e_1, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle j, \zeta\rangle \qquad ((), \xi) \overset{h_1}{\sim} \langle j, \zeta\rangle$$

The second condition implies $j = ()$ and $\xi \xrightarrow{h_1} \zeta$. Moreover $d_2 \xrightarrow{h_1} e_2$ since $d_2 \xrightarrow{h} e_2$, therefore $\langle d_2, \sigma, \xi\rangle \overset{h_1}{\sim} \langle e_2, \zeta\rangle$. By induction hypothesis on the derivation of d_2, there exist i, δ and g an extension of h_1 such that

$$\langle e_2, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i, \delta\rangle \qquad (u, \nu) \overset{g}{\sim} \langle i, \delta\rangle$$

Therefore, by definition of $\Downarrow_{\mathsf{ca}}$,

$$\langle e_1; e_2, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i, \delta\rangle \qquad (u, \nu) \overset{g}{\sim} \langle i, \delta\rangle$$

which completes this case.

($\Rightarrow$) The derivation of $\Downarrow_{\mathsf{ca}}$ for e shows that

$$\langle e_1, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle (), \zeta\rangle \qquad \langle e_2, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i, \delta\rangle$$

for some ζ. By induction hypothesis on the derivation of e_1, there exist v, ξ and h_1 an extension of h such that

$$\langle d_1, \sigma, \mu\rangle \Downarrow_{\mathsf{cl}} (v, \xi) \qquad (v, \xi) \overset{h_1}{\sim} \langle j, \zeta\rangle$$

The second condition implies $v = ()$ and $\xi \xrightarrow{h_1} \zeta$. Moreover $d_2 \xrightarrow{h_1} e_2$ since $d_2 \xrightarrow{h} e_2$, therefore $\langle d_2, \sigma, \xi\rangle \overset{h_1}{\sim} \langle e_2, \zeta\rangle$. By induction hypothesis on the derivation of e_2, there exist u, ν and g an extension of h_1 such that

$$\langle d_2, \sigma, \xi\rangle \Downarrow_{\mathsf{cl}} (u, \nu) \qquad (u, \nu) \overset{g}{\sim} \langle i, \delta\rangle$$

Therefore, by definition of $\Downarrow_{\mathsf{cl}}$,

$$\langle d_1; d_2, \sigma\rangle \mu \Downarrow_{\mathsf{ca}} (u, \nu) \qquad (u, \nu) \overset{g}{\sim} \langle i, \delta\rangle$$

which completes this case.

if conditional

If $d = (\mathsf{if}\ a\ \mathsf{then}\ d_1\ \mathsf{else}\ d_2)$, then $e = (\mathsf{if}\ b\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2)$ for $b = (h \circ \sigma)(a)$, $e_1 = (h\circ\sigma)(d_1)$ and $e_2 = (h\circ\sigma)(d_2)$, therefore $\langle a,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle b,\, \gamma\rangle$, $\langle d_1,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e_1,\, \gamma\rangle$ and $\langle d_2,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e_2,\, \gamma\rangle$.

($\Leftarrow$) The derivation of $\Downarrow_{\mathsf{cl}}$ for d shows that either

$$\langle a,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mathsf{true},\, \xi) \qquad \langle d_1,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)$$

or

$$\langle a,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mathsf{false},\, \xi) \qquad \langle d_2,\, \sigma,\, \xi\rangle \Downarrow_{\mathsf{cl}} (u,\, \nu)$$

For some ξ. Let us consider the case where $\langle a,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (\mathsf{true},\, \xi)$; the other case has a very similar proof. By induction hypothesis on the derivation of a, there exist j, ζ and h_1 an extension of h such that

$$\langle b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle j,\, \zeta\rangle \qquad (\mathsf{true},\, \xi) \overset{h_1}{\sim} \langle j,\, \zeta\rangle$$

The second condition implies $j = \mathsf{true}$ and $\xi \xrightarrow{h_1} \zeta$. Moreover $d_1 \xrightarrow{h_1} e_1$ since $d_1 \xrightarrow{h} e_1$, therefore $\langle d_1,\, \sigma,\, \xi\rangle \overset{h_1}{\sim} \langle e_1,\, \zeta\rangle$. By induction hypothesis on the derivation of d_1, there exist i, δ and g an extension of h_1 such that

$$\langle e_1,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle \qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

Therefore, by definition of $\Downarrow_{\mathsf{ca}}$,

$$\langle \mathsf{if}\ b\ \mathsf{then}\ e_1\ \mathsf{else}\ e_2,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle \qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

which completes this case.

($\Rightarrow$) The derivation of $\Downarrow_{\mathsf{ca}}$ for e shows that either

$$\langle b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{true},\, \zeta\rangle \qquad \langle e_1,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle$$

or

$$\langle b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{false},\, \zeta\rangle \qquad \langle e_2,\, \zeta\rangle \Downarrow_{\mathsf{ca}} \langle i,\, \delta\rangle$$

For some ζ. Let us consider the case where $\langle b,\, \gamma\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{true},\, \zeta\rangle$; the other case has a very similar proof. By induction hypothesis on the derivation of b, there exist v, ξ and h_1 an extension of h such that

$$\langle a,\, \sigma,\, \mu\rangle \Downarrow_{\mathsf{cl}} (v,\, \xi) \qquad (v,\, \xi) \overset{h_1}{\sim} \langle j,\, \zeta\rangle$$

The second condition implies $v = \mathsf{true}$ and $\xi \overset{h_1}{\to} \zeta$. Moreover $d_1 \overset{h_1}{\to} e_1$ since $d_1 \overset{h}{\to} e_1$, therefore $\langle d_1,\, \sigma,\, \xi\rangle \overset{h_1}{\sim} \langle e_1,\, \zeta\rangle$. By induction hypothesis on the derivation of e_1, there exist u, ν and g an extension of h_1 such that

$$\langle d_1,\, \sigma,\, \xi\rangle {\Downarrow_{\mathsf{cl}}} (u,\, \nu) \qquad\qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

Therefore, by definition of $\Downarrow_{\mathsf{cl}}$,

$$\langle \mathsf{if}\ a\ \mathsf{then}\ d_1\ \mathsf{else}\ d_2,\, \sigma,\, \mu\rangle {\Downarrow_{\mathsf{cl}}} (u,\, \nu) \qquad\qquad (u,\, \nu) \overset{g}{\sim} \langle i,\, \delta\rangle$$

which completes this case.

while loop

If $d = (\mathsf{while}\ a\ \mathsf{do}\ d_1)$, then $e = (\mathsf{while}\ b\ \mathsf{do}\ e_1)$ for $b = (h \circ \sigma)(a)$ and $e_1 = (h \circ \sigma)(d_1)$, therefore $\langle a,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle b,\, \gamma\rangle$ and $\langle d_1,\, \sigma,\, \mu\rangle \overset{h}{\sim} \langle e_1,\, \gamma\rangle$. Let $\mu_0 = \mu$, $\gamma_0 = \gamma$ and $h_0 = h$.

($\Rightarrow$) Let $\nu_n = \nu$. The derivation of $\Downarrow_{\mathsf{cl}}$ for d shows that

$$\begin{array}{ll} \langle a,\, \sigma,\, \mu_i\rangle {\Downarrow_{\mathsf{cl}}} (\mathsf{true},\, \nu_i) & \langle d_1,\, \sigma,\, \nu_i\rangle {\Downarrow_{\mathsf{cl}}} ((),\, \mu_{i+1}),\ 0 \le i < n \\ \langle a,\, \sigma,\, \mu_n\rangle {\Downarrow_{\mathsf{cl}}} (\mathsf{false},\, \nu_n) & u = () \end{array}$$

for some $n \ge 0, \mu_1, \ldots, \mu_n, \nu_0, \ldots, \nu_{n-1}$. Let us prove by recurrence on $0 \le i < n$ that there exists h_i, γ_i such that $\langle a,\, \sigma,\, \mu_i\rangle \overset{h_i}{\sim} \langle b,\, \gamma_i\rangle$ and $\langle d_1,\, \sigma,\, \mu_i\rangle \overset{h_i}{\sim} \langle e_1,\, \gamma_i\rangle$. The result is already true for $i = 0$, let us suppose it is true for $0 \le i < n$. By induction hypothesis on the derivation $\langle a,\, \sigma,\, \mu_i\rangle {\Downarrow_{\mathsf{cl}}} (\mathsf{true},\, \nu_i)$, there exist j_i, δ_i and g_i an extension of h_i such that

$$\langle b,\, \gamma_i\rangle {\Downarrow_{\mathsf{ca}}} \langle j_i,\, \delta_i\rangle \qquad\qquad (\mathsf{true},\, \nu_i) \overset{h_1}{\sim} \langle j_i,\, \delta_i\rangle$$

The second condition implies $j_i = \mathsf{true}$ and $\nu_i \overset{g_i}{\to} \delta_i$. Moreover $d_1 \overset{g_i}{\to} e_1$ since $d_1 \overset{h_i}{\to} e_1$, therefore $\langle d_1,\, \sigma,\, \nu_i\rangle \overset{g_i}{\sim} \langle e_1,\, \delta_i\rangle$. By induction hypothesis on the derivation $\langle d_1,\, \sigma,\, \nu_i\rangle {\Downarrow_{\mathsf{cl}}} ((),\, \mu_{i+1})$, there exist k_i, γ_{i+1} and h_{i+1} an extension of g_i such that

$$\langle e_1,\, \delta_i\rangle {\Downarrow_{\mathsf{ca}}} \langle k_i,\, \gamma_{i+1}\rangle \qquad\qquad ((),\, \mu_{i+1}) \overset{h_{i+1}}{\sim} \langle k_i,\, \gamma_{i+1}\rangle$$

The second condition implies $k_i = ()$ and $\mu_{i+1} \overset{h_{i+1}}{\to} \gamma_{i+1}$. Moreover $a \overset{h_{i+1}}{\to} b$ since $a \overset{h_i}{\to} b$ and $d_1 \overset{h_{i+1}}{\to} e_1$ since $d_1 \overset{g_i}{\to} e_1$, therefore $\langle a,\, \sigma,\, \mu_{i+1}\rangle \overset{h_{i+1}}{\sim} \langle b,\, \gamma_{i+1}\rangle$ and $\langle d_1,\, \sigma,\, \mu_{i+1}\rangle \overset{h_{i+1}}{\sim} \langle e_1,\, \gamma_{i+1}\rangle$. This completes the recurrence. In particular, for $i = n - 1$, $\langle a,\, \sigma,\, \mu_n\rangle \overset{h_n}{\sim} \langle b,\, \gamma_n\rangle$. By induction hypothesis on the derivation $\langle a,\, \sigma,\, \mu_n\rangle {\Downarrow_{\mathsf{cl}}} (\mathsf{false},\, \nu_n)$, there exist j_n, δ_n and g an extension of h_n such that

$$\langle b,\, \gamma_n\rangle {\Downarrow_{\mathsf{ca}}} \langle j_n,\, \delta_n\rangle \qquad\qquad (\mathsf{false},\, \nu_n) \overset{g}{\sim} \langle j_n,\, \delta_n\rangle$$

The second condition implies $j_n = \mathsf{false}$, therefore by definition of $\Downarrow_{\mathsf{ca}}$,

$$\langle \mathsf{while}\ b\ \mathsf{do}\ e_1,\, \gamma_0\rangle \Downarrow_{\mathsf{ca}} \langle (),\, \delta_n\rangle \qquad (u,\, \nu) = ((),\, \nu_n) \overset{g}{\sim} \langle (),\, \delta_n\rangle$$

which completes this case.

($\Leftarrow$) Let $\delta_n = \delta$. The derivation of $\Downarrow_{\mathsf{ca}}$ for e shows that

$$\begin{array}{ll} \langle b,\, \gamma_i\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{true},\, \delta_i\rangle & \langle e_1,\, \delta_i\rangle \Downarrow_{\mathsf{ca}} \langle k_i,\, \gamma_{i+1}\rangle,\ 0 \le i < n \\ \langle b,\, \gamma_n\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{false},\, \delta_n\rangle & i = () \end{array}$$

for some $n \ge 0, \gamma_1, \ldots, \gamma_n, \delta_0, \ldots, \delta_{n-1}$. Let us prove by recurrence on $0 \le i < n$ that there exists h_i, μ_i such that $\langle a,\, \sigma,\, \mu_i\rangle \overset{h_i}{\sim} \langle b,\, \gamma_i\rangle$ and $\langle d_1,\, \sigma\rangle\mu_i \overset{h_i}{\sim} \langle e_1,\, \gamma_i\rangle$. The result is already true for $i = 0$, let us suppose it is true for $0 \le i < n$. By induction hypothesis on the derivation $\langle b,\, \gamma_i\rangle \Downarrow_{\mathsf{ca}} \langle \mathsf{true},\, \delta_i\rangle$, there exist v_i, ν_i and g_i an extension of h_i such that

$$\langle a,\, \sigma,\, \mu_i\rangle \Downarrow_{\mathsf{cl}} (v_i,\, \nu_i) \qquad (v_i,\, \nu_i) \overset{h_1}{\sim} \langle \mathsf{true},\, \delta_i\rangle$$

The second condition implies $v_i = \mathsf{true}$ and $\nu_i \overset{g_i}{\to} \delta_i$. Moreover $d_1 \overset{g_i}{\to} e_1$ since $d_1 \overset{h_i}{\to} e_1$, therefore $\langle d_1,\, \sigma,\, \nu_i\rangle \overset{g_i}{\sim} \langle e_1,\, \delta_i\rangle$. By induction hypothesis on the derivation $\langle e_1,\, \delta_i\rangle \Downarrow_{\mathsf{cl}} ((),\, \gamma_{i+1})$, there exist w_i, μ_{i+1} and h_{i+1} an extension of g_i such that

$$\langle d_1,\, \sigma,\, \nu_i\rangle \Downarrow_{\mathsf{cl}} (w_i,\, \mu_{i+1}) \qquad (w_i,\, \mu_{i+1}) \overset{h_{i+1}}{\sim} \langle (),\, \gamma_{i+1}\rangle$$

The second condition implies $w_i = ()$ and $\mu_{i+1} \overset{h_{i+1}}{\to} \gamma_{i+1}$. Moreover $a \overset{h_{i+1}}{\to} b$ since $a \overset{h_i}{\to} b$ and $d_1 \overset{h_{i+1}}{\to} e_1$ since $d_1 \overset{g_i}{\to} e_1$, therefore $\langle a,\, \sigma,\, \mu_{i+1}\rangle \overset{h_{i+1}}{\sim} \langle b,\, \gamma_{i+1}\rangle$ and $\langle d_1,\, \sigma,\, \mu_{i+1}\rangle \overset{h_{i+1}}{\sim} \langle e_1,\, \gamma_{i+1}\rangle$. This completes the recurrence. In particular, for $i = n-1$, $\langle a,\, \sigma,\, \mu_n\rangle \overset{h_n}{\sim} \langle b,\, \gamma_n\rangle$. By induction hypothesis on the derivation $\langle b,\, \gamma_n\rangle \Downarrow_{\mathsf{ca}} (\mathsf{false},\, \nu_n)$, there exist v_n, δ_n and g an extension of h_n such that

$$\langle a,\, \sigma,\, \mu_n\rangle \Downarrow_{\mathsf{cl}} (v_n,\, \nu_n) \qquad (v_n,\, \nu_n) \overset{g}{\sim} \langle \mathsf{false},\, \delta_n\rangle$$

The second condition implies $v_n = \mathsf{false}$, therefore by definition of $\Downarrow_{\mathsf{cl}}$,

$$\langle \mathsf{while}\ a\ \mathsf{do}\ d_1,\, \sigma,\, \mu_0\rangle \Downarrow_{\mathsf{ca}} ((),\, \nu_n) \qquad ((),\, \nu_n) \overset{g}{\sim} \langle (),\, \delta_n\rangle = \langle i,\, \delta\rangle$$

which completes this case and the proof. □