A Resumption Monad Transformer and its Applications in the Semantics of Concurrency[*]

Nikolaos S. Papaspyrou
(nickie@softlab.ntua.gr)

National Technical University of Athens,
Department of Electrical and Computer Engineering,
Software Engineering Laboratory,
Polytechnioupoli, 15780 Zografou, Athens, Greece.

Abstract

Resumptions are a valuable tool in the analysis and design of semantic models for concurrent programming languages, in which computations consist of sequences of atomic steps that may be interleaved. In this paper we consider a general notion of resumption, parameterized by the kind of computations that take place in the atomic steps. We define a monad transformer which, given a monad M that represents the atomic computations, constructs a monad $\mathsf{R}(M)$ for interleaved computations. Moreover, we use this monad transformer to define the denotational semantics of a simple imperative language supporting non-determinism and concurrency.

1 Introduction

Modern computer architectures and operating systems have made it practical to execute different parts of a program simultaneously. From the programmer's point of view, it is often not important whether the parts of a program are executed by different physical processors or by a single processor using a time-sharing strategy. New tools are needed to define the semantics of *concurrent programming languages*, which allow the parts of a program that execute simultaneously to interact with one another, typically using the same memory variables.

Resumptions have long been suggested as a model of interleaved computation in the semantics of concurrent programming languages. In brief, a resumption is either a computed value of some domain D or an atomic computation that results in a new resumption. An extensive treatment is offered in [dBak96] using the theory of complete metric spaces as the mathematical framework for domains. Many variations of resumption domains (also called *branching domains*) for specific instances of atomic computations are investigated there.

In this paper, we propose a structured generalization of this technique. We allow the atomic steps to perform any type of computation, represented by an arbitrary monad M. Thus, we define the *resumption monad transformer* R, which transforms monad M to a new monad $\mathsf{R}(M)$ representing interleaved computations. Domains constructed by $\mathsf{R}(M)$ satisfy the isomorphism

$$\mathsf{R}(M)(D) \quad \simeq \quad D + M(\mathsf{R}(M)(D))$$

which defines the essence of resumption domains. By introducing $\mathsf{R}(M)$ we obtain a general framework for reasoning about such domains. For example, the domain

[*]This technical report is based on work supported by the National Technical University of Athens, under the Programme for the Support of Basic Research “Archimedes”. Project title “SynSemAL: Syntax and Semantics of Artificial Languages”.

$$\mathbf{D} \;\simeq\; \mathbf{U} + (\mathbf{S} \to \mathsf{P}(\mathbf{S} \times \mathbf{D}))$$

that is used in [dBak96] for defining the semantics of non-uniform parallelism (ignoring some complexities related to the use of complete metric spaces) is exactly the same as the domain $\mathsf{R}(\mathsf{D}(\mathsf{P}))(\mathbf{U})$ that we use in Section 4 for the same purpose.

The rest of the paper is structured as follows. Section 2 contains brief introductions to category theory, monads and domain theory, that provide the mathematical background for this paper. In Section 3 we define the resumption monad transformer R and in Section 4 we use it to present the denotational semantics of a simple imperative language featuring non-determinism and concurrency. We conclude with Section 5.

2 Mathematical background

In this section we define the mathematical background that is necessary for the rest of the paper. The reader is referred to the related literature for a more informative introduction and the proofs of the theorems.

2.1 Category theory

Category theory was developed in an attempt to unify simple abstract concepts that were applicable in many branches of mathematics. Excellent introductions to category theory and its application in Computer Science can be found in [Pier90, Gogu91, Pier91, Aspe91, Barr96].

Definition 2.1 *A* category C *is a collection of* objects *and a collection of* arrows *satisfying the following properties:*

- *For each arrow f there is a* domain *object $\mathit{dom}(f)$ and a* codomain *object $\mathit{codom}(f)$, and by writing $f : x \to y$ it is indicated that $x = \mathit{dom}(f)$ and $y = \mathit{codom}(f)$.*
- *For every pair of arrows $f : x \to y$ and $g : y \to z$ there is a* composite *arrow $g \circ f : x \to z$.*
- *Composition of arrows is associative, i.e. for all arrows $f : x \to y$, $g : y \to z$ and $h : z \to w$ it is $h \circ (g \circ f) = (h \circ g) \circ f$.*
- *For each object x there is an* identity *arrow $\mathit{id}_x : x \to x$.*
- *Identity arrows are identities for arrow composition, i.e. for all arrows $f : x \to y$ it is $f \circ \mathit{id}_x = \mathit{id}_y \circ f = f$.*

Definition 2.2 *Two objects x and y of category C are* isomorphic *if there are arrows $f : x \to y$ and $g : y \to x$ such that $f \circ g = \mathit{id}_y$ and $g \circ f = \mathit{id}_x$. Arrows f and g are called* isomorphisms.

Properties of categories are commonly presented using *commuting diagrams*. A diagram is a graph whose nodes are objects and whose edges are arrows. A diagram *commutes* if for every pair of nodes and for every pair of paths connecting these two nodes the composition of arrows along the first path is equal to the composition of arrows along the second. An example of a commuting diagram, implying that $g \circ f = h$, is shown below.

$$\begin{tikzcd}
a \arrow[r, "f"] \arrow[dr, "h"'] & b \arrow[d, "g"] \\
 & c
\end{tikzcd}$$

Definition 2.3 *A* functor *F from category C to category D, written as $F : \mathsf{C} \to \mathsf{D}$, is a pair of mappings. Every object x in C is mapped to an object $F(x)$ in D and every arrow $f : x \to y$ in C is mapped to an arrow $F(f) : F(x) \to F(y)$ in D. Moreoever, the following properties must be satisfied:*

- $F(id_x) = id_{F(x)}$ *for all objects x in C.*
- $F(g \circ f) = F(g) \circ F(f)$ *for all arrows $f : x \to y$ and $g : y \to z$ in C.*

Definition 2.4 *An* endofunctor *on category C is a functor $F : \mathsf{C} \to \mathsf{C}$.*

Definition 2.5 *If $F : \mathsf{C} \to \mathsf{D}$ and $G : \mathsf{D} \to \mathsf{E}$ are functors, then their* composition *is a functor $G \circ F : \mathsf{C} \to \mathsf{E}$. It is defined by taking $(G \circ F)(x) = G(F(x))$ and $(G \circ F)(f) = G(F(f))$.*

Definition 2.6 *For every category C, an* identity functor *$Id_{\mathsf{C}} : \mathsf{C} \to \mathsf{C}$ can be defined by taking $Id_{\mathsf{C}}(x) = x$ and $Id_{\mathsf{C}}(f) = f$.*

Note that if $F : \mathsf{C} \to \mathsf{C}$ is an endofunctor and n is a positive natural number, the notation $F^n : \mathsf{C} \to \mathsf{C}$ can be used for the composition of F with itself n times. The notation can be extended so that $F^0 = Id_{\mathsf{C}}$.

Theorem 2.1 *Functors preserve isomorphisms.*

Theorem 2.2 *Identity functors are identities for functor composition, that is, if $F : \mathsf{C} \to \mathsf{D}$ is a functor, then $F \circ Id_{\mathsf{C}} = Id_{\mathsf{D}} \circ F = F$*

Definition 2.7 *If $F : \mathsf{C} \to \mathsf{D}$ and $G : \mathsf{C} \to \mathsf{D}$ are functors, then a* natural transformation *η between F and G, written as $\eta : F \dot{\to} G$ is a family of arrows in D. In this family, an arrow $\eta_x : F(x) \to G(x)$ in D is defined for every object x in C. Moreover, the following diagram must commute:*

$$\begin{tikzcd}
x \arrow[d, "f"'] & F(x) \arrow[r, "\eta_x"] \arrow[d, "F(f)"'] & G(x) \arrow[d, "G(f)"] \\
y & F(y) \arrow[r, "\eta_y"] & G(y)
\end{tikzcd}$$

2.2 Monads and monad transformers

The notion of *monad*, also called triple, is not new in the context of category theory. In Computer Science, monads became very popular in the 1990s. The categorical properties of monads are discussed in most books on category theory, e.g. in [Barr96]. For a comprehensive introductions to monads and their use in denotational semantics the user is referred to [Mogg90]. A somehow different approach to the definition of monads is found in [Wadl92], which expresses the current practice of monads in functional programming. The two approaches are equivalent. In this paper, the categorical approach (presented here) is used for the definition of monads, since it is much more elegant, and the functional approach (presented in Section 2.4) is used for describing the semantics of programming languages.

Definition 2.8 *A* monad *on a category C is a triple $\langle M, \eta, \mu \rangle$, where $M : \mathsf{C} \to \mathsf{C}$ is an endofunctor, $\eta : Id_{\mathsf{C}} \dot{\to} M$ and $\mu : M^2 \dot{\to} M$ are natural transformations. For all objects x in C, the following diagrams must commute.*

$$\begin{tikzcd}
M(x) \arrow[r, "\eta_{M(x)}"] \arrow[dr, "id_{M(x)}"'] & M^2(x) \arrow[d, "\mu_x"] & M(x) \arrow[l, "M(\eta_x)"'] \arrow[dl, "id_{M(x)}"] \\
 & M(x) &
\end{tikzcd}
\qquad
\begin{tikzcd}
M^3(x) \arrow[r, "M(\mu_x)"] \arrow[d, "\mu_{M(x)}"'] & M^2(x) \arrow[d, "\mu_x"] \\
M^2(x) \arrow[r, "\mu_x"] & M(x)
\end{tikzcd}$$

The transformation η is called the unit *of the monad, whereas the transformation μ is called the* multiplication *or* join.

The commutativity of these two diagrams is equivalent to the following three equations, commonly called the three *monad laws*:

$$\begin{array}{rcll}
\mu_x \circ \eta_{M(x)} & = & id_{M(x)} & \textit{(1st Monad Law)} \\
\mu_x \circ M(\eta_x) & = & id_{M(x)} & \textit{(2nd Monad Law)} \\
\mu_x \circ M(\mu_x) & = & \mu_x \circ \mu_{M(x)} & \textit{(3rd Monad Law)}
\end{array}$$

Definition 2.9 *If* C *is a category, a* monad transformer *on* C *is a mapping between monads on* C.[1]

2.3 Domain theory

The theory of domains was established by Scott and Strachey, in order to provide appropriate mathematical spaces on which to define the denotational semantics of programming languages. Introductions of various sizes and levels can be found in [Scot71, Scot82, Gunt90, Gunt92]. Various kinds of domains are commonly used in denotational semantics, the majority of them based on complete partial orders (cpo's). The variation used here is one of the possible options.

Definition 2.10 *A* partial order, *or* poset, *is a set D together with a binary relation $\sqsubseteq$ that is reflexive, anti-symmetric and transitive.*

Definition 2.11 *A subset $P \subseteq D$ of a poset D is* bounded *if there is a $x \in D$ such that $y \sqsubseteq x$ for all $y \in P$. In this case, x is an* upper bound *of P.*

Definition 2.12 *The* least upper bound *of a subset $P \subseteq D$, written as $\bigsqcup P$, is an upper bound of P such that, $\bigsqcup P \sqsubseteq x$ for all upper bounds x of P.*[2]

Definition 2.13 *A subset $P \subseteq D$ of a poset D is* directed *if every finite subset $F \subseteq P$ has an upper bound $x \in P$.*

Definition 2.14 *A poset D is* complete *if every directed subset $P \subseteq D$ has a least upper bound. A complete partial order is also called a* cpo.

Definition 2.15 *A* domain *is a cpo D with a bottom element, written as $\bot$. For all elements $x \in D$, it must be $\bot \sqsubseteq x$.*

[1] Many options for the definition of monad transformers have been suggested in literature. Given a category C, monads on C and *monad morphisms* (which have not been defined in this paper) form a category $\mathsf{Mon}(\mathsf{C})$. Monad transformers can be defined as mappings between objects in $\mathsf{Mon}(\mathsf{C})$, as endofunctors on $\mathsf{Mon}(\mathsf{C})$, as premonads on $\mathsf{Mon}(\mathsf{C})$ (i.e. endofunctors with a unit), and as monads on $\mathsf{Mon}(\mathsf{C})$. In this paper we have selected the first option.

[2] The notation $a \sqcup b$ is used as an abbreviation of $\bigsqcup\{\, a, b \,\}$.

Definition 2.16 *Every set S defines a* flat domain S^{o}*, whose underlying set is $S \cup \{ \perp \}$ and in which $x \sqsubseteq y$ iff $x = y$ or $x = \perp$.*

A number of useful domains can be defined at this point. The trivial domain **O** is the flat domain that corresponds to the empty set; it contains a single element $\perp$. A useful domain with a single ordinary element is $\mathbf{U} = \{ u \}^{\mathrm{o}}$. The natural numbers under their usual ordering $\leq$ form a poset ω which is not a cpo, since it is directed and does not have a least upper bound.

Definition 2.17 *If D is a poset, an ω-*chain *$(x_n)_{n \in \omega}$ in D is a set of elements $x_n \in D$ such that $n \leq m$ implies $x_n \sqsubseteq x_m$.*

Definition 2.18 *A function $f : D \to E$ between posets D and E is* monotone *if $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$.*

Definition 2.19 *A function $f : D \to E$ between posets D and E is* continuous *if it is monotone and $f(\bigsqcup P) = \bigsqcup \{ f(x) \mid x \in P \}$ for all directed $P \subseteq D$.*

Definition 2.20 *A function $f : D \to E$ between domains D and E is* strict *if $f(\perp) = \perp$.*

Definition 2.21 *A relation $\sqsubseteq$ can be defined for functions between domains D and E as follows. If $f, g : D \to E$, then $f \sqsubseteq g$ iff $f(x) \sqsubseteq g(x)$ for all $x \in D$.*

Theorem 2.3 *The set of continuous functions between D and E under the relation of Definition 2.21 is a domain. This domain is denoted by $D \to E$.*

Definition 2.22 *An element $x \in D$ is a* fixed point *of a function $f : D \to D$ if $x = f(x)$.*

Theorem 2.4 *If D is a domain and $f : D \to D$ is continuous, then f has a* least fixed point *$\mathit{fix}(f) \in D$, i.e. $\mathit{fix}(f) = f(\mathit{fix}(f))$ and $\mathit{fix}(f) \sqsubseteq x$ for all x such that $x = f(x)$. Furthermore, $\mathit{fix}(f) = \bigsqcup_{n \in \omega} f^n(\perp)$.*

Theorem 2.5 *For all $n \in \omega$, let $f_n : A \to B$. Let $x \in A$. Then $\left(\bigsqcup_{n \in \omega} f_n\right)(x) = \bigsqcup_{n \in \omega} f_n(x)$ if the least upper bounds on the left hand side exists.*

Theorem 2.6 *Domains and continuous functions form a category* Dom.

To simplify presentation, in category Dom we often omit the parentheses surrounding a function's argument, i.e. we write $f\ x$ instead of $f(x)$.

Definition 2.23 *A functor $F : \mathsf{Dom} \to \mathsf{Dom}$ is* locally monotone *if $f \sqsubseteq g$ implies $F(f) \sqsubseteq F(g)$, for all domains A and B and for all functions $f, g : A \to B$.*

Definition 2.24 *A functor $F : \mathsf{Dom} \to \mathsf{Dom}$ is* locally continuous *if it is locally monotone and $F(\bigsqcup P) = \bigsqcup \{ F(f) \mid f \in P \}$ for all domains A and B and for all directed $P \subseteq A \to B$.*

Definition 2.25 *If D and E are domains, then the* product *$D \times E$ is a domain. The elements of $D \times E$ are the pairs $\langle x, y \rangle$ with $x \in D$ and $y \in E$, and the ordering relation is defined as $\langle x_1, y_1 \rangle \sqsubseteq \langle x_2, y_2 \rangle \Leftrightarrow x_1 \sqsubseteq_D x_2 \wedge y_1 \sqsubseteq_E y_2$.*

Definition 2.26 *If $D \times E$ is a product domain, two continuous projection functions $\mathtt{fst} : D \times E \to D$ and $\mathtt{snd} : D \times E \to E$ can be defined as $\mathtt{fst}\ \langle x, y \rangle = x$ and $\mathtt{snd}\ \langle x, y \rangle = y$.*

Definition 2.27 *If D and E are domains, then the* separated sum *$D + E$ is a domain. The set of elements of $D + E$ is:*

$$\{ \langle x, 0\rangle \ | \ x \in D \} \ \cup \ \{ \langle y, 1\rangle \ | \ y \in E \} \ \cup \ \{ \bot_{D+E} \}$$

The ordering relation is defined separately for each kind of pairs, i.e. $\langle x_1, 0\rangle \sqsubseteq \langle x_2, 0\rangle \Leftrightarrow x_1 \sqsubseteq_D x_2$ and $\langle y_1, 1\rangle \sqsubseteq \langle y_2, 1\rangle \Leftrightarrow y_1 \sqsubseteq_E y_2$. In addition, $\bot_{D+E} \sqsubseteq z$ for all $z \in D + E$.

Definition 2.28 *If $D + E$ is a sum domain, two continuous injection functions $\mathbf{inl} : D \to D + E$ and $\mathbf{inr} : E \to D + E$ can be defined as $\mathbf{inl}\ x = \langle x, 0\rangle$ and $\mathbf{inr}\ y = \langle y, 1\rangle$.*

Definition 2.29 *If D, E and F are domains and $f_1 : D \to F$ and $f_2 : E \to F$ are continuous functions, then a continuous function $[\, f_1, f_2\,] : D + E \to F$ can be defined as:*

$$[\, f_1, f_2\,](z) \quad = \quad \begin{cases} \bot_F & , \text{ if } z = \bot_{D+E} \\ f_1(x) & , \text{ if } z = \langle x, 0\rangle \\ f_2(y) & , \text{ if } z = \langle y, 1\rangle \end{cases}$$

Theorem 2.7 *Let A, B and C be domains, $f : A \to C$ and $g : B \to C$ continuous functions. Then $[\, f, g\,] \circ \mathbf{inl} = f$ and $[\, f, g\,] \circ \mathbf{inr} = g$.*

Theorem 2.8 *Let A and B be domains. Then $[\, \mathbf{inl}, \mathbf{inr}\,] = \mathbf{id}_{A+B}$.*

Theorem 2.9 *Let A_1, B_1, C_1, A_2, B_2 and C_2 be domains. Let $f_1 : B_1 \to C_1$, $f_2 : A_1 \to B_1$, $g_1 : B_2 \to C_2$ and $g_2 : A_2 \to B_2$ be continuous functions. Then $[\, f_1 \circ f_2, g_1 \circ g_2\,] = [\, f_1, g_1\,] \circ [\, \mathbf{inl} \circ f_2, \mathbf{inr} \circ g_2\,]$.*

Theorem 2.10 *Let A, B, C and D be domains, $f : C \to D$, $g_1 : A \to C$ and $g_2 : B \to C$ continuous functions. If f is strict, then $f \circ [\, g_1, g_2\,] = [\, f \circ g_1, f \circ g_2\,]$.*

Powerdomains are the domain-theoretic equivalent of powersets. They have been introduced as a tool for modeling the semantics of non-deterministic programs and have been widely used for the semantics of concurrency. In this paper we avoid a full definition of powerdomains; the reader is referred to [Gunt92] for a detailed definition and a study of their categoric and domain-theoretic properties.[3]

Definition 2.30 *Let D be a domain. We write $D^\natural$ for the (convex)* powerdomain *of D.*

Definition 2.31 *Let D and E be domains and $f : D \to E$ a continuous function. We can define a continuous function $f^\natural : D^\natural \to E^\natural$.*

Theorem 2.11 *By taking $P(D) = D^\natural$ and $P(f) = f^\natural$ we can define an endofunctor $P : \mathsf{Dom} \to \mathsf{Dom}$, which is called the* powerdomain functor.

Definition 2.32 *Let D be a domain. We can define a continuous function $\{\!|\, \cdot \,|\!\} : D \to D^\natural$, which is called the* powerdomain singleton *function.*

Definition 2.33 *Let D be a domain. We can define a continuous binary operation $\uplus : D^\natural \times D^\natural \to D^\natural$, which is called the* powerdomain union. *Furthermore, this binary operation is associative, commutative and idempotent.*

[3]We also ignore the fact that our category Dom of domains is not appropriate for the definition of powerdomains. The categories SFP (of sequences of finite posets) or Bif (of bifinite domains) should be used instead. The reader is again referred to [Gunt92].

Definition 2.34 *Let D be a domain. We can define a continuous function* $\biguplus^{\natural} : D^{\natural\natural} \to D^{\natural}$*, which is called the* powerdomain big union *function.*

Theorem 2.12 *The powerdomain singleton is a natural transformation between the identity functor Id_{Dom} and the powerdomain functor P.*

Theorem 2.13 *The powerdomain big union is a natural transformation between the functors P^2 and P.*

Theorem 2.14 *The powerdomain functor P with the powerdomain singleton as the unit and the powerdomain big union as the join define a monad, which is called the* powerdomain monad.

2.4 Monads and computations

An alternative approach to the definition of monads has become very popular in the functional programming community. According to this, a monad on category Dom is defined as a triple $\langle M, \mathit{unit}_{\mathsf{M}}, *_{\mathsf{M}}\rangle$. In this triple M is a domain constructor, $\mathit{unit}_{\mathsf{M}} : D \to M(D)$ is a continuous function and $*_{\mathsf{M}} : M(A) \times (A \to M(B)) \to M(B)$ is a binary operation.

In the semantics of programming languages, domains constructed by monad M typically denote *computations*, e.g. the domain $M(D)$ denotes computations returning values of the domain D. The result of $\mathit{unit}_{\mathsf{M}}\ v$ is simply a computation returning the value v and the result of $m\ *_{\mathsf{M}}\ f$ is the combined computation of m, returning v, followed by computation $f(v)$. Monad transformers are useful to transform between different types of computations [Lian95, Lian98].

The following equations connect a monad $\langle M, \mathit{unit}_{\mathsf{M}}, *_{\mathsf{M}}\rangle$ defined using the functional approach with a monad $\langle M, \eta, \mu\rangle$ defined using the categorical approach.

$$\begin{array}{lcl} \mathit{unit}_{\mathsf{M}} & = & \eta \\ m\ *_{\mathsf{M}}\ f & = & (\mu \circ M(f))\ m \end{array} \qquad \begin{array}{lcl} \eta & = & \mathit{unit}_{\mathsf{M}} \\ \mu & = & \lambda m.\ m\ *_{\mathsf{M}}\ \mathit{id} \\ M(f) & = & \lambda m.\ m\ *_{\mathsf{M}}\ (\mathit{unit}_{\mathsf{M}} \circ f) \end{array}$$

In the functional approach, the three monad laws can be formulated as follows.

$$\begin{array}{lcl} m\ *_{\mathsf{M}}\ \mathit{unit}_{\mathsf{M}} & = & m \\ (\mathit{unit}_{\mathsf{M}}\ v)\ *_{\mathsf{M}}\ f & = & f\ v \\ m\ *_{\mathsf{M}}\ (\lambda v.\ (f\ v)\ *_{\mathsf{M}}\ g) & = & (m\ *_{\mathsf{M}}\ f)\ *_{\mathsf{M}}\ g \end{array}$$

An interesting remark is that these three laws are enough to prove that the equivalent (M, η, μ), as defined above, is indeed a monad, i.e. that M is a functor (preserves function identities and composition) and η and μ are natural transformations.

In this setting, it is useful to define two special classes of monads, equipped with additional operations that are useful for modeling the semantics of concurrency in programming languages.

Definition 2.35 *A* multi-monad *is a monad M with a binary operation $\|_{\mathsf{M}} : M(D) \times M(D) \to M(D)$, where D is a domain.*

Definition 2.36 *A* strong monad *is a monad M with a binary operation $\bowtie_{\mathsf{M}} : M(A) \times M(B) \to M(A \times B)$, where A and B are domains.*

The binary operation $\|$ of a multi-monad is used to express disjunction in computations. In other words, if M is a multimonad, D is a domain and $m_1, m_2 \in M(D)$ are two computations, the computation $m_1\ \|\ m_2$ indicates a (possibly non-deterministic) option between m_1 and m_2. Moreover, the binary operation $\bowtie$ of a strong monad is used to express conjunction in computations. Let M be a strong monad, let A and B be domains. If $m_1 \in M(A)$ and $m_2 \in M(B)$ are two computations, the computation $m_1 \bowtie m_2$ indicates that both m_1 and m_2 will be performed and their results will be combined. The option here relates to the order, if any, in which the two computations will be performed.

3 Resumption monad transformer

The notion of execution *interleaving* is a well known one in the theory of concurrency. In this context, computations are considered to be sequences of *atomic steps* the nature of which depends on our notion of computation. In isolation, these atomic steps are performed one after another until the computation is complete. Given two computations A and B, an interleaved computation of A and B consists of an arbitrary merging of the atomic steps that constitute A and B. Interleaving easily extends to more than two computations. The atomic steps of any computation must still be executed in the right order, but this process can be interrupted by the execution of atomic steps belonging to other computations.

Our primary goal is to define a monad transformer R capable of modelling generic interleaved computations. In this way, if we are given a monad M which models the computations taking place at the atomic steps, we can obtain a monad $\mathsf{R}(M)$ which models interleaved computations of such atomic steps. One possible solution to this problem is to use the long suggested technique of *resumptions*, illustrated in [Schm86, dBak96] for specific instances of M.

Generalizing this technique, the domain $\mathsf{R}(M)(D)$ of resumptions must satisfy the following isomorphism:

$$\mathsf{R}(M)(D) \quad \simeq \quad D + M(\mathsf{R}(M)(D))$$

In this domain, atomic steps are arbitrary computations defined by M. The left part of the sum represents an already evaluated result, i.e. a computation that consists of zero atomic steps. The right part represents a computation that requires at least one atomic step. The result of this atomic step is a new element of the resumption domain.

We start by considering an arbitrary locally continuous monad M on Dom. The rest of the section is organized as follows. In Section 3.1 we define an endofunctor $\mathbf{R}_M : \mathsf{Dom} \to \mathsf{Dom}$. In Section 3.2 we define two natural transformations $\mathit{unit} : \mathit{Id} \dot{\to} \mathbf{R}_M$ and $\mathit{join} : \mathbf{R}_M^2 \dot{\to} \mathbf{R}_M$ and in Section 3.3 we prove that $(\mathbf{R}_M, \mathit{unit}, \mathit{join})$ satisfies the three monad laws. In this way we define the monad transformer R. Next, in Section 3.4 we prove that $\mathsf{R}(M)(D)$ satisfies the aforementioned isomorphism by constructing the two components h^e and h^p of the isomorphism. Finally, in Section 3.5 we define a few additional operations on domains constructed by $\mathsf{R}(M)$.

3.1 Functor $\mathbf{R}_M$

We start by defining for each domain D an endofunctor $\mathbf{F}_{M,D} : \mathsf{Dom} \to \mathsf{Dom}$, and some auxiliary functions. The domain $\mathsf{R}(M)(D)$ that we are trying to define is a fixed point of $\mathbf{F}_{M,D}$.

Definition 3.1 *Let D, A and B be domains and $f : A \to B$ a continuous function. We define the following mappings:*

$$\begin{aligned} \mathbf{F}_{M,D}(X) &= D + M(X) \\ \mathbf{F}_{M,D}(f) &= [\, \mathit{inl}, \mathit{inr} \circ M(f)\,] \end{aligned}$$

Lemma 3.1 $\mathbf{F}_{M,D}(f) \circ \mathit{inr} = \mathit{inr} \circ M(f)$

Proof Immediate from Definition 3.1. □

Theorem 3.1 $\mathbf{F}_{M,D} : \mathsf{Dom} \to \mathsf{Dom}$ *is a functor.*

Proof We must prove that $\mathbf{F}_{M,D}$ preserves identities and the composition of continuous functions.

1. Let X be a domain.

$$\begin{array}{l} \quad \mathbf{F}_{M,D}(id_X) \\ = \langle \textit{ Definition of } \mathbf{F}_{M,D} \textit{ (3.1) } \rangle \\ \quad [\, inl, inr \circ M(id_X)\,] \\ = \langle\ M \textit{ is a functor } \rangle \\ \quad [\, inl, inr \circ id_{M(X)}\,] \\ = \langle \textit{ Composition with identity } \rangle \\ \quad [\, inl, inr\,] \\ = \langle \textit{ Theorem 2.8 } \rangle \\ \quad id_{\mathbf{F}_{M,D}(X)} \end{array}$$

2. Let A and B be domains, $f : A \to B$ and $g : B \to C$ continuous functions.

$$\begin{array}{l} \quad \mathbf{F}_{M,D}(g \circ f) \\ = \langle \textit{ Definition of } \mathbf{F}_{M,D} \textit{ (3.1) } \rangle \\ \quad [\, inl, inr \circ M(g \circ f)\,] \\ = \langle\ M \textit{ is a functor } \rangle \\ \quad [\, inl, inr \circ M(g) \circ M(f)\,] \\ = \langle \textit{ Theorem 2.9 } \rangle \\ \quad [\, inl, inr \circ M(g)\,] \circ [\, inl, inr \circ M(f)\,] \\ = \langle \textit{ Definition of } \mathbf{F}_{M,D} \textit{ (3.1) } \rangle \\ \quad \mathbf{F}_{M,D}(g) \circ \mathbf{F}_{M,D}(f) \end{array}$$

□

It is not hard to prove that the functor $\mathbf{F}_{M,D}$ is locally monotone and locally continuous. This result comes easily, since monad M has these two properties and $\mathbf{F}_{M,D}$ is defined in terms of M, using only basic domain operations which preserve monotonicity and continuity.

Lemma 3.2 *Functor* $\mathbf{F}_{M,D} : \mathsf{Dom} \to \mathsf{Dom}$ *is locally monotone.*

Proof Let A and B be domains, let $f, g : A \to B$ be continuous functions with $f \sqsubseteq g$. We prove that $\mathbf{F}_{M,D}(f) \sqsubseteq \mathbf{F}_{M,D}(g)$. Let $x \in \mathbf{F}_{M,D}(A) = D + M(X)$. By case analysis on z.

1. Case $z = \bot$. Then

$$\begin{array}{l} \quad \mathbf{F}_{M,D}(f)\ z \\ = \langle \textit{ Assumption } \rangle \\ \quad \mathbf{F}_{M,D}(f)\ \bot \\ = \langle \textit{ Definition of } \mathbf{F}_{M,D}(f) \textit{ (3.1) } \rangle \\ \quad [\, inl, inr \circ M(f)\,]\ \bot \\ = \langle \textit{ Definition of selection } \rangle \\ \quad \bot \\ = \langle \textit{ Definition of selection } \rangle \\ \quad [\, inl, inr \circ M(g)\,]\ \bot \\ = \langle \textit{ Definition of } \mathbf{F}_{M,D}(g) \textit{ (3.1) } \rangle \\ \quad \mathbf{F}_{M,D}(g)\ \bot \\ = \langle \textit{ Assumption } \rangle \\ \quad \mathbf{F}_{M,D}(g)\ z \end{array}$$

2. Case $z = \mathit{inl}\ t$ for some $t \in D$. Then

$$\begin{aligned}
&\mathbf{F}_{M,D}(f)\ z \\
=\ &\langle\ \textit{Assumption}\ \rangle \\
&\mathbf{F}_{M,D}(f)\ (\mathit{inl}\ t) \\
=\ &\langle\ \textit{Definition of}\ \mathbf{F}_{M,D}(f)\ \textit{(3.1)}\ \rangle \\
&[\,\mathit{inl}, \mathit{inr} \circ M(f)\,]\ (\mathit{inl}\ t) \\
=\ &\langle\ \textit{Definition of selection}\ \rangle \\
&\mathit{inl}\ t \\
=\ &\langle\ \textit{Definition of selection}\ \rangle \\
&[\,\mathit{inl}, \mathit{inr} \circ M(g)\,]\ (\mathit{inl}\ t) \\
=\ &\langle\ \textit{Definition of}\ \mathbf{F}_{M,D}(g)\ \textit{(3.1)}\ \rangle \\
&\mathbf{F}_{M,D}(g)\ (\mathit{inl}\ t) \\
=\ &\langle\ \textit{Assumption}\ \rangle \\
&\mathbf{F}_{M,D}(g)\ z
\end{aligned}$$

3. Case $z = \mathit{inr}\ w$ for some $w \in M(A)$. Then

$$\begin{aligned}
&\mathbf{F}_{M,D}(f)\ z \\
=\ &\langle\ \textit{Assumption}\ \rangle \\
&\mathbf{F}_{M,D}(f)\ (\mathit{inr}\ w) \\
=\ &\langle\ \textit{Definition of}\ \mathbf{F}_{M,D}(f)\ \textit{(3.1)}\ \rangle \\
&[\,\mathit{inl}, \mathit{inr} \circ M(f)\,]\ (\mathit{inr}\ w) \\
=\ &\langle\ \textit{Definition of selection}\ \rangle \\
&(\mathit{inr} \circ M(f))\ w \\
=\ &\langle\ \textit{Composition}\ \rangle \\
&\mathit{inr}\ (M(f)\ w) \\
\sqsubseteq\ &\langle\ M\ \textit{is locally monotone, monotonicity of}\ \mathit{inr}\ \rangle \\
&\mathit{inr}\ (M(g)\ w) \\
=\ &\langle\ \textit{Composition}\ \rangle \\
&(\mathit{inr} \circ M(g))\ w \\
=\ &\langle\ \textit{Definition of selection}\ \rangle \\
&[\,\mathit{inl}, \mathit{inr} \circ M(g)\,]\ (\mathit{inr}\ w) \\
=\ &\langle\ \textit{Definition of}\ \mathbf{F}_{M,D}(g)\ \textit{(3.1)}\ \rangle \\
&\mathbf{F}_{M,D}(g)\ (\mathit{inr}\ w) \\
=\ &\langle\ \textit{Assumption}\ \rangle \\
&\mathbf{F}_{M,D}(g)\ z
\end{aligned}$$

□

Lemma 3.3 *Functor* $\mathbf{F}_{M,D} : \mathsf{Dom} \to \mathsf{Dom}$ *is locally continuous.*

Proof Let A and B be domains and let $P \subseteq A \to B$ be a directed subset. We need to prove that $\mathbf{F}_{M,D}(\bigsqcup P) = \bigsqcup \{\, \mathbf{F}_{M,D}(f) \mid f \in P \,\}$.

$$\begin{aligned}
&\mathbf{F}_{M,D}(\textstyle\bigsqcup P) \\
=\ &\langle\ \textit{Definition of}\ \mathbf{F}_{M,D}\ \textit{(3.1)}\ \rangle \\
&D + M(\textstyle\bigsqcup P) \\
=\ &\langle\ M\ \textit{is locally continuous}\ \rangle \\
&D + \textstyle\bigsqcup \{\, M(f) \mid f \in P \,\} \\
=\ &\langle\ \textit{Continuity of}\ +\ \textit{domain operator,}\ \{\, M(f) \mid f \in P \,\}\ \textit{is directed}\ \rangle
\end{aligned}$$

$\bigsqcup\{\,[\,inl, inr \circ g\,] \mid g \in \{\,M(f) \mid f \in P\,\}\,\}$
= $\langle$ *Simplification* $\rangle$
$\bigsqcup\{\,[\,inl, inr \circ M(f)\,] \mid f \in P\,\}$
= $\langle$ *Definition of* $\mathbf{F}_{M,D}$ *(3.1)* $\rangle$
$\bigsqcup\{\,\mathbf{F}_{M,D}(f) \mid f \in P\,\}$ $\square$

The two functions ι^e and ι^p are useful in the definition of $\mathbf{R}_M(D)$. They define an embedding and a projection between the domains $\mathbf{O}$ and $\mathbf{F}_{M,D}(\mathbf{O})$.

Definition 3.2 *Let D be a domain. We define the pair of functions $\iota^e : \mathbf{O} \to \mathbf{F}_{M,D}(\mathbf{O})$ and $\iota^p : \mathbf{F}_{M,D}(\mathbf{O}) \to \mathbf{O}$ to be equal to $\bot$.*

Lemma 3.4 $\iota^p \circ \iota^e = id_{\mathbf{O}}$

Proof Let $z \in \mathbf{O}$. Then

$(\iota^p \circ \iota^e)\ z$
= $\langle$ *Composition* $\rangle$
$\iota^p\ (\iota^e\ z)$
= $\langle$ *Definition of* ι^p *(3.2)* $\rangle$
$\bot$
= $\langle$ $\mathbf{O}$ *has only one element* $\rangle$
z
= $\langle$ *Identity* $\rangle$
$id_{\mathbf{O}}\ z$ $\square$

Lemma 3.5 $\iota^e \circ \iota^p \sqsubseteq id_{\mathbf{F}_{M,D}(\mathbf{O})}$

Proof Let $z \in \mathbf{F}_{M,D}(\mathbf{O})$. Then

$(\iota^e \circ \iota^p)\ z$
= $\langle$ *Composition* $\rangle$
$\iota^e\ (\iota^p\ z)$
= $\langle$ *Definition of* ι^e *(3.2)* $\rangle$
$\bot$
$\sqsubseteq$ $\langle$ *Bottom element* $\rangle$
z
= $\langle$ *Identity* $\rangle$
$id_{\mathbf{F}_{M,D}(\mathbf{O})}\ z$ $\square$

We proceed by defining a mapping of objects and a mapping of functions, which will define the endofunctor $\mathbf{R}_M : \mathsf{Dom} \to \mathsf{Dom}$ at the end of this section. This is the key definition of our work.

Definition 3.3 *Let D be a domain. The domain $\mathbf{R}_M(D)$ is the set*

$$\mathbf{R}_M(D) \;=\; \{\,(x_n)_{n\in\omega} \mid \forall n \in \omega.\ x_n \in \mathbf{F}^n_{M,D}(\mathbf{O}) \,\wedge\, x_n = \mathbf{F}^n_{M,D}(\iota^p)(x_{n+1})\,\}$$

with its elements ordered pointwise:

$$(x_n)_{n\in\omega} \sqsubseteq_{\mathbf{R}_M(D)} (y_n)_{n\in\omega} \;\Leftrightarrow\; \forall n \in \omega.\ x_n \sqsubseteq_{\mathbf{F}^n_{M,D}(\mathbf{O})} y_n$$

The elements of the domain $\mathbf{R}_M(D)$ are infinite sequences, indexed by the set of natural numbers ω. The n-th element of the sequence is an element of the domain $\mathbf{F}^n_{M,D}(\mathbf{O})$. Such elements represent *finite approximations* of resumption computations: if a given resumption computation terminates in less than n steps, its n-th approximation is able to compute its result accurately; otherwise it produces $\perp$. The condition $x_n = \mathbf{F}^n_{M,D}(\iota^p)(x_{n+1})$ states that the elements of the infinite sequence must indeed be approximations: the result of projecting the $(n+1)$-th appoximation (an element of $\mathbf{F}^{n+1}_{M,D}(\mathbf{O})$) to an element of $\mathbf{F}^n_{M,D}(\mathbf{O})$ must be equal to the n-th approximation.

Before we can define the mapping of functions that corresponds to $\mathbf{R}_M$, it is necessary to define a few families of auxiliary functions. The first is the family of functions $f^D_{m,n}$ which map between different approximations of a resumption computation.

Definition 3.4 *Let D be a domain. For all $m, n \in \omega$, we define a function $f^D_{m,n} : \mathbf{F}^m_{M,D}(\mathbf{O}) \to \mathbf{F}^n_{M,D}(\mathbf{O})$ by:*

$$\begin{array}{lcll} f^D_{m,n} & = & id_{\mathbf{F}^n_{M,D}(\mathbf{O})} & \text{, if } m = n \\ f^D_{m,n+1} & = & f^D_{m,n} \circ \mathbf{F}^n_{M,D}(\iota^p) & \text{, if } m \leq n \\ f^D_{m+1,n} & = & \mathbf{F}^m_{M,D}(\iota^e) \circ f^D_{m,n} & \text{, if } m \geq n \end{array}$$

Lemma 3.6 *For all $m, n \in \omega$, $f^D_{m,n} \circ \mathbf{F}^n_{M,D}(\iota^p) \sqsubseteq f^D_{m,n+1}$.*

Proof

1. If $m \leq n$.

   $f^D_{m,n} \circ \mathbf{F}^n_{M,D}(\iota^p)$
   = ⟨ *Definition of $f^D_{m,n+1}$ (3.4), $m \leq n$* ⟩
   $f^D_{m,n+1}$

2. If $m > n$, by induction on $m - n$.

   (a) Base case. If $m = n + 1$ then

   $f^D_{m,n} \circ \mathbf{F}^m_{M,D}(\iota^p)$
   = ⟨ *Assumption* ⟩
   $f^D_{n+1,n} \circ \mathbf{F}^{n+1}_{M,D}(\iota^p)$
   = ⟨ *Definition of $f^D_{n+1,n}$ (3.4)* ⟩
   $\mathbf{F}^{n+1}_{M,D}(\iota^e) \circ f^D_{n,n} \circ \mathbf{F}^{n+1}_{M,D}(\iota^p)$
   = ⟨ *Definition of $f^D_{n,n}$ (3.4)* ⟩
   $\mathbf{F}^{n+1}_{M,D}(\iota^e) \circ id_{\mathbf{F}^n_{M,D}(\mathbf{O})} \circ \mathbf{F}^{n+1}_{M,D}(\iota^p)$
   = ⟨ *Composition with identity* ⟩
   $\mathbf{F}^{n+1}_{M,D}(\iota^e) \circ \mathbf{F}^{n+1}_{M,D}(\iota^p)$
   = ⟨ *$\mathbf{F}^{n+1}_{M,D}$ is a functor (Theorem 3.1)* ⟩
   $\mathbf{F}^{n+1}_{M,D}(\iota^e \circ \iota^p)$
   $\sqsubseteq$ ⟨ *Lemma 3.4, $\mathbf{F}^{n+1}_{M,D}$ is locally monotone (Lemma 3.2)* ⟩
   $\mathbf{F}^{n+1}_{M,D}(id_{\mathbf{O}})$
   = ⟨ *$\mathbf{F}^{n+1}_{M,D}$ is a functor (Theorem 3.1)* ⟩
   $id_{\mathbf{F}^{n+1}_{M,D}(\mathbf{O})}$
   = ⟨ *Definition of $f^D_{n+1,n+1}$ (3.4)* ⟩
   $f^D_{n+1,n+1}$
   = ⟨ *Assumption* ⟩
   $f^D_{m,n+1}$

(b) Let us assume that it is true for $m = n + k$ for some $k > 0$. Then, for $m = n + k + 1$ we have

$f^D_{m,n} \circ \mathbf{F}^n_{M,D}(\iota^p)$
= ⟨ *Assumption* ⟩
$f^D_{n+k+1,n} \circ \mathbf{F}^n_{M,D}(\iota^p)$
= ⟨ *Definition of* $f^D_{n+k+1,n}$ *(3.4),* $n + k \geq n$ ⟩
$\mathbf{F}^{n+k}_{M,D}(\iota^e) \circ f^D_{n+k,n} \circ \mathbf{F}^n_{M,D}(\iota^p)$
$\sqsubseteq$ ⟨ *Inductive hypothesis* ⟩
$\mathbf{F}^{n+k}_{M,D}(\iota^e) \circ f^D_{n+k,n+1}$
= ⟨ *Definition of* $f^D_{n+k+1,n+1}$ *(3.4),* $n + k \geq n + 1$ ⟩
$f^D_{n+k+1,n+1}$
= ⟨ *Assumption* ⟩
$f^D_{m,n+1}$ □

Lemma 3.7 *For all* $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$ *and for all* $m, n \in \omega$, $f^D_{m,n}\ x_n \sqsubseteq x_m$.

Proof

1. If $m = n$ then

$f^D_{m,n}\ x_n$
= ⟨ *Assumption* ⟩
$f^D_{n,n}\ x_n$
= ⟨ *Definition of* $f^D_{n,n}$ *(3.4)* ⟩
$id_{\mathbf{F}^n_{M,D}(\mathbf{O})}\ x_n$
= ⟨ *Identity* ⟩
x_n
= ⟨ *Assumption* ⟩
x_m

2. If $m < n$, by induction on $n - m$. Let us assume that it is true for $n = m + k$ for some $k \geq 0$. Then for $n = m + k + 1$ we have

$f^D_{m,n}\ x_n$
= ⟨ *Assumption* ⟩
$f^D_{m,m+k+1}\ x_{m+k+1}$
= ⟨ *Definition of* $f^D_{m,m+k+1}$ *(3.4),* $m \leq m + k$ ⟩
$(f^D_{m,m+k} \circ \mathbf{F}^{m+k}_{M,D}(\iota^p))\ x_{m+k+1}$
= ⟨ *Composition* ⟩
$f^D_{m,m+k}\ (\mathbf{F}^{m+k}_{M,D}(\iota^p)\ x_{m+k+1})$
= ⟨ $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$ *(Definition 3.3)* ⟩
$f^D_{m,m+k}\ x_{m+k}$
$\sqsubseteq$ ⟨ *Inductive hypothesis* ⟩
x_m

3. If $m > n$, by induction on $m - n$. Let us assume that it is true for $m = n + k$ for some $k \geq 0$. Then for $m = n + k + 1$ we have

$f^D_{m,n}\ x_n$

= ⟨ *Assumption* ⟩
  $f^D_{n+k+1,n}\ x_n$
= ⟨ *Definition of* $f^D_{n+k+1,n}$ *(3.4),* $n+k \geq n$ ⟩
  $(\mathbf{F}^{n+k}_{M,D}(\iota^e) \circ f^D_{n+k,n})\ x_n$
= ⟨ *Composition* ⟩
  $\mathbf{F}^{n+k}_{M,D}(\iota^e)\ (f^D_{n+k,n}\ x_n)$
$\sqsubseteq$ ⟨ *Inductive hypothesis, monotonicity* ⟩
  $\mathbf{F}^{n+k}_{M,D}(\iota^e)\ x_{n+k}$
= ⟨ $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$ *(Definition 3.3)* ⟩
  $\mathbf{F}^{n+k}_{M,D}(\iota^e)\ (\mathbf{F}^{n+k}_{M,D}(\iota^p)\ x_{n+k+1})$
= ⟨ *Composition* ⟩
  $(\mathbf{F}^{n+k}_{M,D}(\iota^e) \circ \mathbf{F}^{n+k}_{M,D}(\iota^p)\ x_{n+k+1}$
= ⟨ $\mathbf{F}^{n+k}_{M,D}$ *is a functor (Theorem 3.1)* ⟩
  $\mathbf{F}^{n+k}_{M,D}(\iota^e \circ \iota^p\ x_{n+k+1}$
$\sqsubseteq$ ⟨ *Lemma 3.5,* $\mathbf{F}^{n+k}_{M,D}$ *is locally monotone (Lemma 3.2)* ⟩
  $id_{\mathbf{F}_{M,D}(\mathbf{O})}\ x_{n+k+1}$
= ⟨ *Identity* ⟩
  x_{n+k+1}
= ⟨ *Assumption* ⟩
  x_m □

The families of μ^e_n and μ^p_n functions are also related with mappings between resumption computations and their approximations. The former embeds an approximation requiring less than n steps to an element of the domain $\mathbf{R}_M(D)$, while the latter projects an element of the domain $\mathbf{R}_M(D)$ to its n-th approximation.

Definition 3.5 *Let* D *be a domain,* $n \in \omega$, $z \in \mathbf{F}^n_{M,D}(\mathbf{O})$ *and* $(x_m)_{m\in\omega} \in \mathbf{R}_M(D)$. *We define the pair of functions* $\mu^e_n : \mathbf{F}^n_{M,D}(\mathbf{O}) \to \mathbf{R}_M(D)$ *and* $\mu^p_n : \mathbf{R}_M(D) \to \mathbf{F}^n_{M,D}(\mathbf{O})$ *as follows:*

$$\begin{array}{lcl} \mu^e_n\ z & = & (f^D_{m,n}\ z)_{m\in\omega} \\ \mu^p_n\ (x_m)_{m\in\omega} & = & x_n \end{array}$$

We are now ready to define the mapping of functions required by the functor $\mathbf{R}_M$. Instead of defining this mapping directly in terms of elements of the resumption domain $\mathbf{R}_M(D)$, we use the family of functions $\zeta^{A,B}_n$ and define it in terms of the finite approximations.

Definition 3.6 *Let* A *and* B *be domains and* $f : A \to B$ *a continuous function. For all* $n \in \omega$ *we define a continuous function* $\zeta^{A,B}_n\ f : \mathbf{F}^n_{M,A}(\mathbf{O}) \to \mathbf{F}^n_{M,B}(\mathbf{O})$ *by:*

$$\begin{array}{lcl} \zeta^{A,B}_0\ f & = & \bot \\ \zeta^{A,B}_{n+1}\ f & = & [\, inl \circ f, inr \circ M(\zeta^{A,B}_n\ f)\,] \end{array}$$

Definition 3.7 *Let* A *and* B *be domains and* $f : A \to B$ *a continuous function. We define a continuous function* $\mathbf{R}_M(f) : \mathbf{R}_M(A) \to \mathbf{R}_M(B)$ *by:*

$$\mathbf{R}_M(f)\ (x_n)_{n\in\omega} \quad = \quad (\zeta^{A,B}_n\ f\ x_n)_{n\in\omega}$$

The central result of this section is Theorem 3.2 in which we prove that $\mathbf{R}_M$ is a functor. For doing so, we make use of the following lemmata.

Lemma 3.8 *For all* $n \in \omega$, $\mu^e_n \circ \mathbf{F}^n_{M,D}(\iota^p) \sqsubseteq \mu^e_{n+1}$.

Proof Let $z \in \mathbf{F}^{n+1}_{M,D}(\mathbf{O})$. Then

$(\mu^e_n \circ \mathbf{F}^n_{M,D}(\iota^p))\ z$
= ⟨ *Composition* ⟩
$\mu^e_n\ (\mathbf{F}^n_{M,D}(\iota^p)\ z)$
= ⟨ *Definition of* μ^e_n *(3.5)* ⟩
$(f^D_{m,n}\ (\mathbf{F}^n_{M,D}(\iota^p)\ z))_{m\in\omega}$
= ⟨ *Composition* ⟩
$((f^D_{m,n} \circ \mathbf{F}^n_{M,D}(\iota^p))\ z)_{m\in\omega}$
$\sqsubseteq$ ⟨ *Lemma 3.6* ⟩
$(f^D_{m,n+1}\ z)_{m\in\omega}$
= ⟨ *Definition of* μ^e_{n+1} *(3.5)* ⟩
$\mu^e_{n+1}\ z$ □

Lemma 3.9 *Let* A *and* B *be domains,* $f : A \to B$ *a continuous function. Then for all* $n \in \omega$,

$$\zeta^{A,B}_{n+1}\ f \circ \mathit{inr} = \mathit{inr} \circ M(\zeta^{A,B}_n\ f)$$

Proof Immediate from Definition 3.6. □

Lemma 3.10 *For all* $x \in \mathbf{R}_M(D)$, $(\mu^p_n\ x)_{n\in\omega} = x$.

Proof Let us assume that $x = (x_m)_{m\in\omega}$. Then

$(\mu^p_n\ x)_{n\in\omega}$
= ⟨ *Assumption* ⟩
$(\mu^p_n\ (x_m)_{m\in\omega})_{n\in\omega}$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
$(x_n)_{n\in\omega}$
= ⟨ *Assumption* ⟩
x □

Lemma 3.11 *For all* $m, n \in \omega$, $\mu^p_m \circ \mu^e_n = f^D_{m,n}$.

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. Then

$(\mu^p_m \circ \mu^e_n)\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$\mu^p_m\ (\mu^e_n\ (x_n)_{n\in\omega})$
= ⟨ *Definition of* μ^e_n *(3.5)* ⟩
$\mu^p_m\ (f^D_{m',n}\ (x_n)_{n\in\omega})_{m'\in\omega}$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
$f^D_{m,n}\ (x_n)_{n\in\omega}$ □

Lemma 3.12 *For all* $n \in \omega$, $\mu^p_n \circ \mu^e_n = \mathit{id}_{\mathbf{F}^n_{M,D}(\mathbf{O})}$.

Proof Directly from Lemma 3.11. □

Lemma 3.13 *For all* $n \in \omega$, $\mu^e_m \circ \mu^p_n \sqsubseteq \mathit{id}_{\mathbf{R}_M(D)}$.

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. Then

$(\mu^e_m \circ \mu^p_n)\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$\mu^e_m\ (\mu^p_n\ (x_n)_{n\in\omega})$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
$\mu^e_m\ x_n$
= ⟨ *Definition of* μ^e_n *(3.5)* ⟩
$(f^D_{m,n}\ x_n)_{m\in\omega}$
$\sqsubseteq$ ⟨ *Lemma 3.7, definition of* $\sqsubseteq_{\mathbf{R}_M(D)}$ *(3.3)* ⟩
$(x_m)_{m\in\omega}$
= ⟨ *Identity* ⟩
$\mathit{id}_{\mathbf{R}_M(D)}\ (x_m)_{m\in\omega}$ □

Lemma 3.14 *Let* A *and* B *be domains,* $f : A \to B$ *a continuous function. Then for all* $n \in \omega$,

$$\mu^p_n \circ \mathbf{R}_M(f) = \zeta^{A,B}_n\ f \circ \mu^p_n$$

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(A)$. Then

$(\mu^p_n \circ \mathbf{R}_M(f))\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$\mu^p_n\ (\mathbf{R}_M(f)\ (x_n)_{n\in\omega})$
= ⟨ *Definition of* $\mathbf{R}_M$ *(3.7)* ⟩
$\mu^p_n\ (\zeta^{A,B}_n\ f\ x_n)_{n\in\omega}$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
$\zeta^{A,B}_n\ f\ x_n$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
$\zeta^{A,B}_n\ f\ (\mu^p_n\ (x_n)_{n\in\omega})$
= ⟨ *Composition* ⟩
$(\zeta^{A,B}_n\ f \circ \mu^p_n)\ (x_n)_{n\in\omega}$ □

Lemma 3.15 *Let* A *be a domain. Then for all* $n \in \omega$, $\zeta^{A,A}_n\ \mathit{id}_A = \mathit{id}_{\mathbf{F}^n_{M,A}(\mathbf{O})}$.

Proof By induction on n. If $n = 0$ then $\zeta^{A,A}_0\ \mathit{id}_A = \bot = \mathit{id}_{\mathbf{F}^0_{M,A}(\mathbf{O})}$. Let us assume that it is true for some $n \in \omega$. Then

$\zeta^{A,A}_{n+1}\ \mathit{id}_A$
= ⟨ *Definition of* ζ *(3.6)* ⟩
$[\,\mathit{inl} \circ \mathit{id}_A, \mathit{inr} \circ M(\zeta^{A,A}_n\ \mathit{id}_A)\,]$
= ⟨ *Induction hypothesis* ⟩
$[\,\mathit{inl} \circ \mathit{id}_A, \mathit{inr} \circ M(\mathit{id}_{\mathbf{F}^n_{M,A}(\mathbf{O})})\,]$
= ⟨ M *is a functor* ⟩
$[\,\mathit{inl} \circ \mathit{id}_A, \mathit{inr} \circ \mathit{id}_{M(\mathbf{F}^n_{M,A}(\mathbf{O}))}\,]$
= ⟨ *Composition with identity* ⟩
$[\,\mathit{inl}, \mathit{inr}\,]$
= ⟨ *Theorem 2.8* ⟩
$\mathit{id}_{\mathbf{F}^{n+1}_{M,D}(X)}$ □

Lemma 3.16 *Let A, B and C be domains, $f : A \to B$ and $g : B \to C$ continuous functions. Then for all $n \in \omega$, $\zeta_n^{A,C}\ (g \circ f) = \zeta_n^{B,C}\ g \circ \zeta_n^{A,B}\ f$.*

Proof By induction on n. If $n = 0$ then $\zeta_0^{A,C}\ (g \circ f) = \bot = \bot \circ \bot = \zeta_0^{B,C}\ g \circ \zeta_0^{A,B}\ f$. Let us assume that it is true for some $n \in \omega$. Then

$\zeta_{n+1}^{A,C}\ (g \circ f)$
= $\langle$ *Definition of ζ (3.6)* $\rangle$
$[\, inl \circ g \circ f, inr \circ M(\zeta_n^{A,C}\ (g \circ f))\,]$
= $\langle$ *Induction hypothesis* $\rangle$
$[\, inl \circ g \circ f, inr \circ M(\zeta_n^{B,C}\ g \circ \zeta_n^{A,B}\ f)\,]$
= $\langle$ *M is a functor* $\rangle$
$[\, inl \circ g \circ f, inr \circ M(\zeta_n^{B,C}\ g) \circ M(\zeta_n^{A,B}\ f))\,]$
= $\langle$ *Theorem 2.9* $\rangle$
$[\, inl \circ g, inr \circ M(\zeta_n^{B,C}\ g)\,] \circ [\, inl \circ f, inr \circ M(\zeta_n^{A,B}\ f)\,]$
= $\langle$ *Definition of ζ (3.6)* $\rangle$
$\zeta_{n+1}^{B,C}\ g \circ \zeta_{n+1}^{A,B}\ f$ □

We can now proceed with the proof of Theorem 3.2.

Theorem 3.2 $\mathbf{R}_M$: $\mathsf{Dom} \to \mathsf{Dom}$ *is a functor.*

Proof We must prove that $\mathbf{R}_M$ preserves identities and the composition of continuous functions.

1. Let X be a domain and $(x_n)_{n\in\omega} \in \mathbf{R}_M(X)$.

   $\mathbf{R}_M(id_X)\ (x_n)_{n\in\omega}$
   = $\langle$ *Definition of $\mathbf{R}_M$ (3.7)* $\rangle$
   $(\zeta_n^{X,X}\ id_X\ x_n)_{n\in\omega}$
   = $\langle$ *Lemma 3.15* $\rangle$
   $(id_{\mathbf{F}_{M,X}^n(\mathbf{O})}\ x_n)_{n\in\omega}$
   = $\langle$ *Identity function* $\rangle$
   $(x_n)_{n\in\omega}$
   = $\langle$ *Identity function* $\rangle$
   $id_{\mathbf{R}_M(X)}\ (x_n)_{n\in\omega}$

2. Let A and B be domains, $f : A \to B$ and $g : B \to C$ continuous functions and $(x_n)_{n\in\omega} \in \mathbf{R}_M(X)$.

   $\mathbf{R}_M(g \circ f)\ (x_n)_{n\in\omega}$
   = $\langle$ *Definition of $\mathbf{R}_M$ (3.7)* $\rangle$
   $(\zeta_n^{A,C}\ (g \circ f)\ x_n)_{n\in\omega}$
   = $\langle$ *Lemma 3.16* $\rangle$
   $((\zeta_n^{B,C}\ g \circ \zeta_n^{A,B}\ f)\ x_n)_{n\in\omega}$
   = $\langle$ *Composition* $\rangle$
   $(\zeta_n^{B,C}\ g\ (\zeta_n^{A,B}\ f\ x_n))_{n\in\omega}$
   = $\langle$ *Definition of $\mathbf{R}_M$ (3.7)* $\rangle$
   $\mathbf{R}_M(g)\ (\zeta_n^{A,B}\ f\ x_n)_{n\in\omega}$
   = $\langle$ *Definition of $\mathbf{R}_M$ (3.7)* $\rangle$
   $\mathbf{R}_M(g)\ (\mathbf{R}_M(f)\ (x_n)_{n\in\omega})$
   = $\langle$ *Composition* $\rangle$
   $(\mathbf{R}_M(g) \circ \mathbf{R}_M(f))\ (x_n)_{n\in\omega}$ □

3.2 Unit and join

Having defined $\mathbf{R}_M$ as a functor, we now define the two monad operations *unit* and *join*. For each one, we prove that it is a natural transformation.

The *unit* function maps an element $d \in D$ to a resumption computation, using the family of auxiliary functions η. All approximations in the resumption computation are equal to *inl* d (except for the trivial approximation of zero steps).

Definition 3.8 *Let D be a domain. For all $n \in \omega$ we define a continuous function $\eta_n^D : D \to \mathbf{F}^n_{M,D}(\mathbf{O})$ by:*

$$\begin{array}{lcl} \eta_0^D & = & \perp \\ \eta_{n+1}^D & = & \mathit{inl} \end{array}$$

Definition 3.9 *Let D be a domain and $d \in D$. We define the function $\mathit{unit}_D : D \to \mathbf{R}_M(D)$ by:*

$$\mathit{unit}_D\ d \quad = \quad (\eta_n^D\ d)_{n\in\omega}$$

The following lemma is useful in proving that *unit* is a natural transformation.

Lemma 3.17 *Let A and B be domains, $f : A \to B$ a continuous function. Then, for all $n \in \omega$,*
$\zeta_n^{A,B}\ f \circ \eta_n^A = \eta_n^B \circ f$

Proof By induction on n. If $n = 0$ then $\zeta_0^{A,B}\ f \circ \eta_n^A = \perp \circ \perp = \perp = \perp \circ f = \eta_0^B \circ f$. Let us assume that it is true for some $n \in \omega$. Then

$\zeta_{n+1}^{A,B}\ f \circ \eta_{n+1}^A$
= ⟨ *Definitions of ζ (3.6) and η (3.8)* ⟩
$[\,\mathit{inl} \circ f, \mathit{inr} \circ M(\zeta_n^{A,B})\,] \circ \mathit{inl}$
= ⟨ *Theorem 2.7* ⟩
$\mathit{inl} \circ f$
= ⟨ *Definition of η (3.8)* ⟩
$\eta_n^B \circ f$ □

Theorem 3.3 *$\mathit{unit} : \mathit{Id} \dot{\to} \mathbf{R}_M$ is a natural transformation.*

Proof Let A and B be domains and $f : A \to B$ a continuous function. We must show that $\mathit{unit}_B \circ f = \mathbf{R}_M(f) \circ \mathit{unit}_A$. Let $a \in A$.

$(\mathit{unit}_B \circ f)\ a$
= ⟨ *Composition* ⟩
$\mathit{unit}_B\ (f\ a)$
= ⟨ *Definition of unit (3.9)* ⟩
$(\eta_n^D\ (f\ a))_{n\in\omega}$
= ⟨ *Composition* ⟩
$((\eta_n^D \circ f)\ a)_{n\in\omega}$
= ⟨ *Lemma 3.17* ⟩
$((\zeta_n^{A,B} \circ \eta_n^A)\ a)_{n\in\omega}$
= ⟨ *Composition* ⟩
$(\zeta_n^{A,B}\ (\eta_n^A\ a))_{n\in\omega}$
= ⟨ *Definition of $\mathbf{R}_M$ (3.7)* ⟩
$\mathbf{R}_M(f)\ (\eta_n^A\ a)_{n\in\omega}$
= ⟨ *Definition of unit (3.9)* ⟩

$\mathbf{R}_M(f)\ (\mathit{unit}_A\ a)$
= $\langle$ *Composition* $\rangle$
$(\mathbf{R}_M(f) \circ \mathit{unit}_A)\ a$ $\square$

The definition of the join function requires the family of functions ξ which associates corresponding approximations in the domains $\mathbf{R}_M(D)$ and D.

Definition 3.10 *Let D be a domain. For all $n \in \omega$ we define a continuous function $\xi_n^D : \mathbf{F}^n_{M,\mathbf{R}_M(D)}(\mathbf{O}) \to \mathbf{F}^n_{M,D}(\mathbf{O})$ by:*

$$\begin{array}{lcl} \xi_0^D & = & \bot \\ \xi_{n+1}^D f & = & [\,\mu^p_{n+1}, \mathit{inr} \circ M(\xi_n^D)\,] \end{array}$$

Definition 3.11 *Let D be a domain and $(x_n)_{n\in\omega} \in \mathbf{R}^2_M(D)$. We define the function $\mathit{join}_D : \mathbf{R}^2_M(D) \to \mathbf{R}_M(D)$ by:*

$$\mathit{join}_D\ (x_n)_{n\in\omega} \quad = \quad (\xi_n^D\ x_n)_{n\in\omega}$$

For proving that join is a natural transformation, we need the following lemmata.

Lemma 3.18 *Let D be a domain. Then for all $n \in \omega$, $\xi_{n+1}^D \circ \mathit{inr} = \mathit{inr} \circ M(\xi_n^D)$.*

Proof Immediate from Definition 3.10. $\square$

Lemma 3.19 *Let A and B be domains, $f : A \to B$ a continuous function. Then for all $n \in \omega$,*
$\xi_n^B \circ \zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f)) = \zeta_n^{A,B}\ f \circ \xi_n^A$

Proof By induction on n. If $n = 0$ then $\xi_0^B \circ \zeta_0^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f)) = \bot \circ \bot = \zeta_0^{A,B}\ f \circ \xi_0^A$. Let us assume that it is true for some $n \in \omega$. Then

$\xi_{n+1}^B \circ \zeta_{n+1}^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f))$
= $\langle$ *Definitions of ξ (3.10) and ζ (3.6)* $\rangle$
$[\,\mu^p_{n+1}, \mathit{inr} \circ M(\xi_n^B)\,] \circ [\,\mathit{inl} \circ \mathbf{R}_M(f), \mathit{inr} \circ M(\zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f)))\,]$
= $\langle$ *Theorem 2.9* $\rangle$
$[\,\mu^p_{n+1} \circ \mathbf{R}_M(f), \mathit{inr} \circ M(\xi_n^B) \circ M(\zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f)))\,]$
= $\langle$ *M is a functor* $\rangle$
$[\,\mu^p_{n+1} \circ \mathbf{R}_M(f), \mathit{inr} \circ M(\xi_n^B \circ \zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f)))\,]$
= $\langle$ *Induction hypothesis* $\rangle$
$[\,\mu^p_{n+1} \circ \mathbf{R}_M(f), \mathit{inr} \circ M(\zeta_n^{A,B}\ f \circ \xi_n^A)\,]$
= $\langle$ *M is a functor* $\rangle$
$[\,\mu^p_{n+1} \circ \mathbf{R}_M(f), \mathit{inr} \circ M(\zeta_n^{A,B}\ f) \circ M(\xi_n^A)\,]$
= $\langle$ *Lemma 3.9* $\rangle$
$[\,\mu^p_{n+1} \circ \mathbf{R}_M(f), \zeta_{n+1}^{A,B}\ f \circ \mathit{inr} \circ M(\xi_n^A)\,]$
= $\langle$ *Lemma 3.14* $\rangle$
$[\,\zeta_{n+1}^{A,B}\ f \circ \mu^p_{n+1}, \zeta_{n+1}^{A,B}\ f \circ \mathit{inr} \circ M(\xi_n^A)\,]$
= $\langle$ *Theorem 2.10, $\zeta_{n+1}^{A,B}\ f$ is strict* ***[!!!]*** $\rangle$
$\zeta_{n+1}^{A,B}\ f \circ [\,\mu^p_{n+1}, \mathit{inr} \circ M(\xi_n^A)\,]$
= $\langle$ *Definition of ξ (3.10)* $\rangle$
$\zeta_{n+1}^{A,B}\ f \circ \xi_{n+1}^A$ $\square$

Theorem 3.4 *join* $: \mathbf{R}_M^2 \dot{\to} \mathbf{R}_M$ *is a natural transformation.*

Proof Let A and B be domains and $f : A \to B$ a continuous function. We must show that $\mathit{join}_B \circ \mathbf{R}_M(\mathbf{R}_M(f)) = \mathbf{R}_M(f) \circ \mathit{join}_A$. Let $(x_n)_{n\in\omega} \in \mathbf{R}_M^2(A)$.

$(\mathit{join}_B \circ \mathbf{R}_M(\mathbf{R}_M(f)))\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$\mathit{join}_B\ (\mathbf{R}_M(\mathbf{R}_M(f))\ (x_n)_{n\in\omega})$
= ⟨ *Definition of* $\mathbf{R}_M$ *(3.7)* ⟩
$\mathit{join}_B\ (\zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f))\ x_n)_{n\in\omega}$
= ⟨ *Definition of join (3.11)* ⟩
$(\xi_n^B\ (\zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f))\ x_n))_{n\in\omega}$
= ⟨ *Composition* ⟩
$((\xi_n^B \circ \zeta_n^{\mathbf{R}_M(A),\mathbf{R}_M(B)}\ (\mathbf{R}_M(f)))\ x_n)_{n\in\omega}$
= ⟨ *Lemma 3.19* ⟩
$((\zeta_n^{A,B}\ f \circ \xi_n^A)\ x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$(\zeta_n^{A,B}\ f\ (\xi_n^A\ x_n))_{n\in\omega}$
= ⟨ *Definition of* $\mathbf{R}_M(f)$ *(3.7)* ⟩
$\mathbf{R}_M(f)\ (\xi_n^A\ x_n)_{n\in\omega}$
= ⟨ *Definition of join (3.11)* ⟩
$\mathbf{R}_M(f)\ (\mathit{join}_A\ (x_n)_{n\in\omega})$
= ⟨ *Composition* ⟩
$(\mathbf{R}_M(f) \circ \mathit{join}_A)\ (x_n)_{n\in\omega}$ □

3.3 Monad $\mathsf{R}(M)$

In this section we prove that functor $\mathbf{R}_M$ together with the natural transformations *unit* and *join* defines a monad. The three theorems of this section verify the three monad laws and the following lemmata are necessary for proving them. Let D be a domain.

Lemma 3.20 *For all* $n \in \omega$, $\xi_n^D \circ \eta_n^{\mathbf{R}_M(D)} = \mu_n^p$.

Proof By a degenerate induction on n. If $n = 0$ then $\xi_n^D \circ \eta_n^{\mathbf{R}_M(D)} = \bot \circ \bot = \bot = \mu_n^p$. Also

$\xi_{n+1}^D \circ \eta_{n+1}^{\mathbf{R}_M(D)}$
= ⟨ *Definition of* ξ *(3.10)* ⟩
$[\,\mu_{n+1}^p, \mathit{inr} \circ M(\xi_n^D)\,] \circ \eta_{n+1}^{\mathbf{R}_M(D)}$
= ⟨ *Definition of* η *(3.8)* ⟩
$[\,\mu_{n+1}^p, \mathit{inr} \circ M(\xi_n^D)\,] \circ \mathit{inl}$
= ⟨ *Theorem 2.7* ⟩
μ_{n+1}^p □

Lemma 3.21 *For all* $n \in \omega$, $\mu_{n+1}^p \circ \mathit{unit}_D = \mathit{inl}$.

Proof Let $d \in D$. Then

$(\mu_{n+1}^p \circ \mathit{unit}_D)\ d$
= ⟨ *Composition* ⟩
$\mu_{n+1}^p\ (\mathit{unit}_D\ d)$

= ⟨ *Definition of* unit *(3.9)* ⟩
  $\mu^p_{n+1}\ (\eta^D_m\ d)_{m\in\omega}$
= ⟨ *Definition of* μ^p_{n+1} *(3.5)* ⟩
  $\eta^D_{n+1}\ d$
= ⟨ *Definition of* η *(3.8)* ⟩
  $\mathit{inl}\ d$ □

Lemma 3.22 *For all* $n \in \omega$, $\xi^D_n \circ \zeta^{D,\mathbf{R}_M(D)}_n\ \mathit{unit}_D = \mathit{id}_{\mathbf{F}^n_{M,D}(\mathbf{O})}$.

Proof By induction on n. If $n = 0$ then $\xi^D_0 \circ \zeta^{D,\mathbf{R}_M(D)}_0\ \mathit{unit}_D = \bot \circ \bot = \bot = \mathit{id}_{\mathbf{F}^0_{M,D}(\mathbf{O})}$. Let us assume that it is true for some $n \in \omega$. Then

  $\xi^D_{n+1} \circ \zeta^{D,\mathbf{R}_M(D)}_{n+1}\ \mathit{unit}_D$
= ⟨ *Definitions of* ζ *(3.6) and* ξ *(3.10)* ⟩
  $[\,\mu^p_{n+1}, \mathit{inr} \circ M(\xi^D_n)\,] \circ [\,\mathit{inl} \circ \mathit{unit}_D, \mathit{inr} \circ M(\zeta^{D,\mathbf{R}_M(D)}_n\ \mathit{unit}_D)\,]$
= ⟨ *Theorem 2.9* ⟩
  $[\,\mu^p_{n+1} \circ \mathit{unit}_D, \mathit{inr} \circ M(\xi^D_n) \circ M(\zeta^{D,\mathbf{R}_M(D)}_n\ \mathit{unit}_D)\,]$
= ⟨ M *is functor* ⟩
  $[\,\mu^p_{n+1} \circ \mathit{unit}_D, \mathit{inr} \circ M(\xi^D_n \circ \zeta^{D,\mathbf{R}_M(D)}_n\ \mathit{unit}_D)\,]$
= ⟨ *Induction hypothesis* ⟩
  $[\,\mu^p_{n+1} \circ \mathit{unit}_D, \mathit{inr} \circ M(\mathit{id}_{\mathbf{F}^n_{M,D}(\mathbf{O})})\,]$
= ⟨ M *is functor* ⟩
  $[\,\mu^p_{n+1} \circ \mathit{unit}_D, \mathit{inr} \circ \mathit{id}_{M(\mathbf{F}^n_{M,D}(\mathbf{O}))}\,]$
= ⟨ *Composition with identity* ⟩
  $[\,\mu^p_{n+1} \circ \mathit{unit}_D, \mathit{inr}\,]$
= ⟨ *Lemma 3.21* ⟩
  $[\,\mathit{inl}, \mathit{inr}\,]$
= ⟨ *Theorem 2.8* ⟩
  $\mathit{id}_{\mathbf{F}^{n+1}_{M,D}(\mathbf{O})}$ □

Lemma 3.23 *For all* $n \in \omega$, $\mu^p_n \circ \mathit{join}_D = \xi^D_n \circ \mu^p_n$.

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}^2_M(D)$. Then

  $(\mu^p_n \circ \mathit{join}_D)\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
  $\mu^p_n\ (\mathit{join}_D\ (x_n)_{n\in\omega})$
= ⟨ *Definition of join (3.11)* ⟩
  $\mu^p_n\ (\xi^D_n\ x_n)_{n\in\omega}$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
  $\xi^D_n\ x_n$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
  $\xi^D_n\ (\mu^p_n(x_n)_{n\in\omega})$
= ⟨ *Composition* ⟩
  $(\xi^D_n \circ \mu^p_n)\ (x_n)_{n\in\omega}$ □

Lemma 3.24 *For all* $d \in \omega$*,* $\xi_n^D \circ \zeta_n^{\mathbf{R}_M^2(D), \mathbf{R}_M(D)}\ \mathit{join}_D = \xi_n^D \circ \xi_n^{\mathbf{R}_M(D)}$.

Proof By induction on n. If $n = 0$ then $\xi_0^D \circ \zeta_0^{\mathbf{R}_M^2(D), \mathbf{R}_M(D)}\ \mathit{join}_D = \bot \circ \bot = \xi_0^D \circ \xi_0^{\mathbf{R}_M(D)}$. Let us assume that it is true for some $n \in \omega$. Then

$\xi_{n+1}^D \circ \zeta_{n+1}^{\mathbf{R}_M^2(D), \mathbf{R}_M(D)}\ \mathit{join}_D$
= $\langle$ *Definitions of* ξ *(3.10) and* ζ *(3.6)* $\rangle$
$[\,\mu_{n+1}^p, \mathit{inr} \circ M(\xi_n^D)\,] \circ [\,\mathit{inl} \circ \mathit{join}_D, \mathit{inr} \circ M(\zeta_n^{\mathbf{R}_M^2(D), \mathbf{R}_M(D)}\ \mathit{join}_D)\,]$
= $\langle$ *Theorem 2.9* $\rangle$
$[\,\mu_{n+1}^p \circ \mathit{join}_D, \mathit{inr} \circ M(\xi_n^D) \circ M(\zeta_n^{\mathbf{R}_M^2(D), \mathbf{R}_M(D)}\ \mathit{join}_D)\,]$
= $\langle$ M *is functor* $\rangle$
$[\,\mu_{n+1}^p \circ \mathit{join}_D, \mathit{inr} \circ M(\xi_n^D \circ \zeta_n^{\mathbf{R}_M^2(D), \mathbf{R}_M(D)}\ \mathit{join}_D)\,]$
= $\langle$ *Inductive hypothesis* $\rangle$
$[\,\mu_{n+1}^p \circ \mathit{join}_D, \mathit{inr} \circ M(\xi_n^D \circ \xi_n^{\mathbf{R}_M(D)})\,]$
= $\langle$ M *is functor* $\rangle$
$[\,\mu_{n+1}^p \circ \mathit{join}_D, \mathit{inr} \circ M(\xi_n^D) \circ M(\xi_n^{\mathbf{R}_M(D)})\,]$
= $\langle$ *Lemma 3.18* $\rangle$
$[\,\mu_{n+1}^p \circ \mathit{join}_D, \xi_{n+1}^D \circ \mathit{inr} \circ M(\xi_n^{\mathbf{R}_M(D)})\,]$
= $\langle$ *Lemma 3.23* $\rangle$
$[\,\xi_{n+1}^D \circ \mu_{n+1}^p, \xi_{n+1}^D \circ \mathit{inr} \circ M(\xi_n^{\mathbf{R}_M(D)})\,]$
= $\langle$ *Theorem 2.10,* ξ_{n+1}^D *is strict* ***[!!!]*** $\rangle$
$\xi_{n+1}^D \circ [\,\mu_{n+1}^p, \mathit{inr} \circ M(\xi_n^{\mathbf{R}_M(D)})\,]$
= $\langle$ *Definition of* ξ *(3.10)* $\rangle$
$\xi_{n+1}^D \circ \xi_{n+1}^{\mathbf{R}_M(D)}$ □

We can now proceed by proving the three monad laws.

Theorem 3.5 (1st Monad Law) $\mathit{join}_D \circ \mathit{unit}_{\mathbf{R}_M(D)} = \mathit{id}_{\mathbf{R}_M(D)}$

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. Then

$(\mathit{join}_D \circ \mathit{unit}_{\mathbf{R}_M(D)})\ (x_n)_{n\in\omega}$
= $\langle$ *Composition* $\rangle$
$\mathit{join}_D\ (\mathit{unit}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega})$
= $\langle$ *Definition of* unit *(3.9)* $\rangle$
$\mathit{join}_D\ (\eta_n^{\mathbf{R}_M(D)}\ (x_m)_{m\in\omega})_{n\in\omega}$
= $\langle$ *Definition of join (3.11)* $\rangle$
$(\xi_n^D\ (\eta_n^{\mathbf{R}_M(D)}\ (x_m)_{m\in\omega}))_{n\in\omega}$
= $\langle$ *Composition* $\rangle$
$((\xi_n^D \circ \eta_n^{\mathbf{R}_M(D)})\ (x_m)_{m\in\omega})_{n\in\omega}$
= $\langle$ *Lemma 3.20* $\rangle$
$(\mu_n^p\ (x_m)_{m\in\omega})_{n\in\omega}$
= $\langle$ *Definition of* μ_n^p *(3.5)* $\rangle$
$(x_n)_{n\in\omega}$
= $\langle$ *Identity* $\rangle$
$\mathit{id}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega}$ □

Theorem 3.6 (2nd Monad Law) $\mathit{join}_D \circ \mathbf{R}_M(\mathit{unit}_D) = \mathit{id}_{\mathbf{R}_M(D)}$

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. Then

$(\mathit{join}_D \circ \mathbf{R}_M(\mathit{unit}_D))\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$\mathit{join}_D\ (\mathbf{R}_M(\mathit{unit}_D)\ (x_n)_{n\in\omega})$
= ⟨ *Definition of* $\mathbf{R}_M$ *(3.7)* ⟩
$\mathit{join}_D\ (\zeta_n^{D,\mathbf{R}_M(D)}\ \mathit{unit}_D\ x_n)_{n\in\omega}$
= ⟨ *Definition of join (3.11)* ⟩
$(\xi_n^D\ (\zeta_n^{D,\mathbf{R}_M(D)}\ \mathit{unit}_D\ x_n))_{n\in\omega}$
= ⟨ *Composition* ⟩
$((\xi_n^D \circ \zeta_n^{D,\mathbf{R}_M(D)}\ \mathit{unit}_D)\ x_n)_{n\in\omega}$
= ⟨ *Lemma 3.22* ⟩
$(\mathit{id}_{\mathbf{F}^n_{M,D}(\mathbf{O})}\ x_n)_{n\in\omega}$
= ⟨ *Identity* ⟩
$(x_n)_{n\in\omega}$
= ⟨ *Identity* ⟩
$\mathit{id}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega}$ □

Theorem 3.7 (3rd Monad Law) $\mathit{join}_D \circ \mathbf{R}_M(\mathit{join}_D) = \mathit{join}_D \circ \mathit{join}_{\mathbf{R}_M(D)}$

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}^3_M(D)$. Then

$(\mathit{join}_D \circ \mathbf{R}_M(\mathit{join}_D))\ (x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$\mathit{join}_D\ (\mathbf{R}_M(\mathit{join}_D)\ (x_n)_{n\in\omega})$
= ⟨ *Definition of* $\mathbf{R}_M$ *(3.7)* ⟩
$\mathit{join}_D\ (\zeta_n^{\mathbf{R}^2_M(D),\mathbf{R}_M(D)}\ \mathit{join}_D\ x_n)_{n\in\omega}$
= ⟨ *Definition of join (3.11)* ⟩
$(\xi_n^D\ (\zeta_n^{\mathbf{R}^2_M(D),\mathbf{R}_M(D)}\ \mathit{join}_D\ x_n))_{n\in\omega}$
= ⟨ *Composition* ⟩
$((\xi_n^D \circ \zeta_n^{\mathbf{R}^2_M(D),\mathbf{R}_M(D)}\ \mathit{join}_D)\ x_n)_{n\in\omega}$
= ⟨ *Lemma 3.24* ⟩
$((\xi_n^D \circ \xi_n^{\mathbf{R}_M(D)})\ x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$(\xi_n^D\ (\xi_n^{\mathbf{R}_M(D)}\ x_n))_{n\in\omega}$
= ⟨ *Definition of join (3.11)* ⟩
$\mathit{join}_D\ (\xi_n^{\mathbf{R}_M(D)}\ x_n)_{n\in\omega}$
= ⟨ *Definition of join (3.11)* ⟩
$\mathit{join}_D\ (\mathit{join}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega})$
= ⟨ *Composition* ⟩
$(\mathit{join}_D \circ \mathit{join}_{\mathbf{R}_M(D)})\ (x_n)_{n\in\omega}$ □

Having established that $\mathbf{R}_M$ satisfies the three monad laws, we can now conclude the definition of the resumption monad transformer R.

Definition 3.12 *The* resumption monad transformer R *is defined by the mapping* $\mathsf{R}(M) = \mathbf{R}_M$.

3.4 Isomorphism

Let D be a domain. In this section, we define the pair of functions h^e and h^p that establish the isomorphism between domains $\mathbf{R}_M(D)$ and $D + M(\mathbf{R}_M(D))$. Using these functions, it is possible to define an operation in one of these two domains and obtain the corresponding operation on the other domain by applying h^e and h^p appropriately.

The definition of the embedding function h^e is straightforward. We make use of a family of auxiliary functions θ, which construct the necessary approximations.

Definition 3.13 *For all $n \in \omega$ we define a function $\theta^D_n : \mathbf{F}_{M,D}(\mathbf{R}_M(D)) \to \mathbf{F}^n_{M,D}(\mathbf{O})$ by:*

$$\begin{aligned} \theta^D_0 &= \bot \\ \theta^D_{n+1} &= [\, \mathit{inl}, \mathit{inr} \circ M(\mu^p_n) \,] \end{aligned}$$

Definition 3.14 *Let $z \in \mathbf{F}_{M,D}(\mathbf{R}_M(D))$. We define the function $h^e : \mathbf{F}_{M,D}(\mathbf{R}_M(D)) \to \mathbf{R}_M(D)$ by:*

$$h^e\ z \ = \ (\theta^D_n\ z)_{n\in\omega}$$

On the other hand, the definition of the projection function h^p is more complicated. It first requires the definition of an additional domain $\mathbf{Q}_M(D)$ whose elements are infinite sequences of computations yielding approximations (we will call them *approximate computations* for short). We also find it helpful to define a family of auxiliary functions σ for associating elements of $\mathbf{Q}_M(D)$ with approximations in $\mathbf{R}_M(D)$.

Definition 3.15 *The domain $\mathbf{Q}_M(D)$ is the set*

$$\mathbf{Q}_M(D) \ = \ \{\, (z_n)_{n\in\omega} \ \mid \ \forall n \in \omega.\ z_n \in M(\mathbf{F}^n_{M,D}(\mathbf{O})) \ \wedge \ z_n = M(\mathbf{F}^n_{M,D}(\iota^p))(z_{n+1}) \,\}$$

with its elements ordered pointwise:

$$(z_n)_{n\in\omega} \sqsubseteq_{\mathbf{Q}_M(D)} (w_n)_{n\in\omega} \ \Leftrightarrow \ \forall n \in \omega.\ z_n \sqsubseteq_{M(\mathbf{F}^n_{M,D}(\mathbf{O}))} w_n$$

Definition 3.16 *Let $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$. For all $n \in \omega$, we define a function $\sigma^D_n : \mathbf{Q}_M(D) \to \mathbf{F}^n_{M,D}(\mathbf{O})$ by:*

$$\begin{aligned} \sigma^D_0\ (z_m)_{m\in\omega} &= \bot \\ \sigma^D_{n+1}\ (z_m)_{m\in\omega} &= \mathit{inr}\ z_n \end{aligned}$$

Furthermore, the definition of h^p requires the proof of Lemma 3.25, which states that elements of $\mathbf{R}_M(D)$ come in three distinct forms. This lemma is crucial in the definition of h^p and in the proofs of several theorems that follow.

Lemma 3.25 *Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. Then exactly one of the following is true:*

1. *For all $n \in \omega$, $x_n = \bot$.*
2. *There exists a $t \in D$ such that for all $n \in \omega$, $x_n = \eta^D_n\ t$.*
3. *There exists a $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ such that for all $n \in \omega$, $x_n = \sigma^D_n\ (z_m)_{m\in\omega}$.*

Proof It is obvious that the three alternatives are mutually exclusive, so it suffices to show that one of them is true. We know that $x_0 \in \mathbf{F}^0_{M,D}(\mathbf{O}) = \mathbf{O}$, and therefore $x_0 = \bot$. Also, $x_1 \in \mathbf{F}_{M,D}(\mathbf{O}) = D + M(\mathbf{O})$. If $n > 0$, we notice that

x_n
= ⟨ *Definition of $\mathbf{R}_M(D)$ (3.3)* ⟩
$\mathbf{F}^n_{M,D}(\iota^p)(x_{n+1})$

= $\langle$ *Composition* $\rangle$
$\mathbf{F}_{M,D}(\mathbf{F}^{n-1}_{M,D}(\iota^p))(x_{n+1})$
= $\langle$ *Definition of* $\mathbf{F}_{M,D}$ *(3.1)* $\rangle$
$[\, \mathit{inl}, \mathit{inr} \circ M(\mathbf{F}^{n-1}_{M,D}(\iota^p)) \,]\; x_{n+1}$

We proceed by case analysis on x_1.

1. Case $x_1 = \bot$. We will show that the first alternative is true, i.e. for all $n \in \omega$, $x_n = \bot$. We already know it for $n = 0$ and $n = 1$. Let us assume that it is true for some $n > 0$. Then, from the previous remark we obtain $\bot = [\, \mathit{inl}, \mathit{inr} \circ M(\mathbf{F}^{n-1}_{M,D}(\iota^p)) \,]\; x_{n+1}$. If $x_{n+1} = \mathit{inl}\ d$ for some $d \in D$, then we obtain $\bot = \mathit{inl}\ d$ which is a contradiction. Similarly, if $x_{n+1} = \mathit{inr}\ m$ for some $m \in M(\mathbf{F}^{n}_{M,D}(\mathbf{O}))$, then we obtain $\bot = \mathit{inr}\ (M(\mathbf{F}^{n-1}_{M,D}(\iota^p))\ m)$. It follows that $x_{n+1} = \bot$.

2. Case $x_1 = \mathit{inl}\ t$ for some $t \in D$. We will show that the second alternative is true, i.e. for all $n \in \omega$, $x_n = \eta^D_n\ t$. For $n = 0$, we know that $x_0 = \bot = \eta^D_0\ t$. Also for $n = 1$, we know that $x_1 = \mathit{inl}\ t = \eta^D_1\ t$. Let us assume that it is true for some $n > 0$. Then, from the previous remark we obtain $\mathit{inl}\ t = [\, \mathit{inl}, \mathit{inr} \circ M(\mathbf{F}^{n-1}_{M,D}(\iota^p)) \,]\; x_{n+1}$. If $x_{n+1} = \bot$, then we obtain $\mathit{inl}\ t = \bot$ which is a contradiction. Similarly, if $x_{n+1} = \mathit{inr}\ m$ for some $m \in M(\mathbf{F}^{n}_{M,D}(\mathbf{O}))$, then we obtain $\mathit{inl}\ t = \mathit{inr}\ (M(\mathbf{F}^{n-1}_{M,D}(\iota^p))\ m)$. Finally, if $x_{n+1} = \mathit{inl}\ t'$ for some $t' \in D$, then we obtain $\mathit{inl}\ t = \mathit{inl}\ t'$. Therefore $t' = t$ and $x_{n+1} = \mathit{inl}\ t$.

3. Case $x_1 = \mathit{inr}\ w$ for some $w \in M(\mathbf{O})$. We will show that the third alternative is true by constructing a $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ such that for all $n \in \omega$, $x_n = \sigma^D_n\ (z_m)_{m\in\omega}$. For $n = 1$, we know that $x_1 = \mathit{inr}\ w$. Let us take $z_0 = w$. Furthermore, let us assume that for some $n > 0$ we have $x_n = \mathit{inr}\ z_{n-1}$. Then, from the previous remark we obtain $\mathit{inr}\ z_{n-1} = [\, \mathit{inl}, \mathit{inr} \circ M(\mathbf{F}^{n-1}_{M,D}(\iota^p)) \,]\; x_{n+1}$. If $x_{n+1} = \bot$, then we obtain $\mathit{inr}\ z_{n-1} = \bot$ which is a contradiction. Similarly, if $x_{n+1} = \mathit{inl}\ d$ for some $d \in D$, then we obtain $\mathit{inr}\ z_{n-1} = \mathit{inl}\ d$. Finally, if $x_{n+1} = \mathit{inr}\ w'$ for some $w' \in M(\mathbf{F}^{n}_{M,D}(\mathbf{O}))$, then we obtain $\mathit{inr}\ z_{n-1} = \mathit{inr}\ (M(\mathbf{F}^{n-1}_{M,D}(\iota^p))\ w')$ and therefore $z_{n-1} = M(\mathbf{F}^{n-1}_{M,D}(\iota^p))\ w'$. It suffices therefore to take $z_n = w'$ and by definition we obtain that $x_{n+1} = \mathit{inr}\ z_n$ and $z_{n-1} = M(\mathbf{F}^{n-1}_{M,D}(\iota^p))\ z_n$. In this way we can construct a $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ such that for all $n > 0$, $x_n = \mathit{inr}\ z_{n-1} = \sigma^D_n\ (z_m)_{m\in\omega}$. And obviously, for $n = 0$ we know that $x_0 = \bot = \sigma^D_0\ (z_m)_{m\in\omega}$. □

We can now proceed with the definition of h^p, based on the three cases of Lemma 3.25. For the first two cases, the definition is easy. In the third case, each approximate computation z_n is mapped to a computation $M(\mu^e_n)\ z_n \in M(\mathbf{R}_M(D))$ and the least upper bound of this infinite series of computations is taken.

Definition 3.17 *We define the function* $h^p : \mathbf{R}_M(D) \to \mathbf{F}_{M,D}(\mathbf{R}_M(D))$ *by case analysis on its argument* $(x_n)_{n\in\omega}$ *based on Lemma 3.25:*

1. *If for all* $n \in \omega$, $x_n = \bot$, *then*
$$h^p\ (x_n)_{n\in\omega} = \bot$$
2. *If there exists a* $t \in D$ *such that for all* $n \in \omega$, $x_n = \eta^D_n\ t$, *then*
$$h^p\ (x_n)_{n\in\omega} = \mathit{inl}\ t$$
3. *If there exists a* $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ *such that for all* $n \in \omega$, $x_n = \sigma^D_n\ (z_m)_{m\in\omega}$, *then*
$$h^p\ (x_n)_{n\in\omega} = \mathit{inr}\ \left(\bigsqcup_{n\in\omega} M(\mu^e_n)\ z_n \right)$$

In order to ensure that the least upper bound in the third case of the previous definition exists, we prove Lemma 3.26 which states that $M(\mu^e_n)\ z_n$ form an ω-chain.

Lemma 3.26 *Let* $(z_n)_{n\in\omega} \in \mathbf{Q}_M(D)$*. For all* $n \in \omega,\ \ M(\mu^e_n)\ z_n \sqsubseteq M(\mu^e_{n+1})\ z_{n+1}$.

Proof We have:

$M(\mu^e_n)\ z_n$
= $\langle$ *Definition of* $\mathbf{Q}_M(D)$ *(3.15)* $\rangle$
$M(\mu^e_n)\ (M(\mathbf{F}^n_{M,D}(\iota^p))\ z_{n+1})$
= $\langle$ *Composition* $\rangle$
$(M(\mu^e_n) \circ M(\mathbf{F}^n_{M,D}(\iota^p)))\ z_{n+1}$
= $\langle$ M *is a functor* $\rangle$
$M(\mu^e_n \circ \mathbf{F}^n_{M,D}(\iota^p))\ z_{n+1}$
= $\langle$ *Lemma 3.8,* M *is locally monotone* $\rangle$
$M(\mu^e_{n+1})\ z_{n+1}$ □

The following lemmata are necessary for proving the central theorems of this section.

Lemma 3.27 *For all* $t \in D$*, for all* $n \in \omega,\ \ \theta^D_n\ (\mathit{inl}\ t) = \eta^D_n\ t$.

Proof By a degenerate induction on n. If $n = 0$ then $\theta^D_0\ (\mathit{inl}\ t) = \bot\ (\mathit{inl}\ t) = \bot = \bot\ t = \eta^D_0\ t$. Also

$\theta^D_{n+1}\ (\mathit{inl}\ t)$
= $\langle$ *Definition of* θ *(3.13)* $\rangle$
$[\,\mathit{inl}, \mathit{inr} \circ M(\mu^p_n)\,]\ (\mathit{inl}\ t)$
= $\langle$ *Definition of selection* $\rangle$
$\mathit{inl}\ t$
= $\langle$ *Definition of* η *(3.8)* $\rangle$
$\eta^D_{n+1}\ t$ □

Lemma 3.28 *For all* $w \in M(\mathbf{R}_M(D))$*, for all* $n \in \omega,\ \ \theta^D_n\ (\mathit{inr}\ w) = \sigma^D_n\ (M(\mu^p_m)\ w)_{m\in\omega}$.

Proof By a degenerate induction on n. If $n = 0$ then $\theta^D_0\ (\mathit{inr}\ w) = \bot\ (\mathit{inr}\ w) = \bot = \bot\ (M(\mu^p_m)\ w)_{m\in\omega} = \sigma^D_0\ (M(\mu^p_m)\ w)_{m\in\omega}$. Also

$\theta^D_{n+1}\ (\mathit{inr}\ w)$
= $\langle$ *Definition of* θ *(3.13)* $\rangle$
$[\,\mathit{inl}, \mathit{inr} \circ M(\mu^p_n)\,]\ (\mathit{inr}\ w)$
= $\langle$ *Definition of selection* $\rangle$
$(\mathit{inr} \circ M(\mu^p_n))\ w$
= $\langle$ *Composition* $\rangle$
$\mathit{inr}\ (M(\mu^p_n)\ w)$
= $\langle$ *Definition of* σ *(3.16)* $\rangle$
$\sigma^D_{n+1}\ (M(\mu^p_m)\ w)_{m\in\omega}$ □

Lemma 3.29 $\bigsqcup_{n\in\omega} \mu^e_n \circ \mu^p_n = \mathit{id}_{\mathbf{R}_M(D)}$

Proof We must first show that for all $n \in \omega$, $\mu^e_n \circ \mu^p_n \sqsubseteq \mathit{id}_{\mathbf{R}_M(D)}$. This follows immediately from Lemma 3.13. Then we must show that for all $f : \mathbf{R}_M(D) \to \mathbf{R}_M(D)$, if for all $n \in \omega$, $\mu^e_n \circ \mu^p_n \sqsubseteq f$ then $\mathit{id}_{\mathbf{R}_M(D)} \sqsubseteq f$. Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. We have

$\mathit{id}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega}$
= $\langle$ *Identity* $\rangle$
$(x_n)_{n\in\omega}$

= ⟨ *Identity* ⟩
$(\mathit{id}_{\mathbf{F}^n_{M,D}(\mathbf{O})}\ x_n)_{n\in\omega}$
= ⟨ *Lemma 3.12* ⟩
$((\mu^p_n \circ \mu^e_n)\ x_n)_{n\in\omega}$
= ⟨ *Composition* ⟩
$(\mu^p_n\ (\mu^e_n\ x_n))_{n\in\omega}$
= ⟨ *Definition of* μ^p_n *(3.5)* ⟩
$(\mu^p_n\ (\mu^e_n\ (\mu^p_n\ (x_m)_{m\in\omega})))_{n\in\omega}$
= ⟨ *Composition* ⟩
$((\mu^p_n \circ \mu^e_n \circ \mu^p_n)\ (x_m)_{m\in\omega})_{n\in\omega}$
$\sqsubseteq$ ⟨ *Assumption, monotonicity, definition of* $\mathbf{R}_M(D)$ *(3.3)* ⟩
$((\mu^p_n \circ f)\ (x_m)_{m\in\omega})_{n\in\omega}$
= ⟨ *Composition* ⟩
$(\mu^p_n\ (f\ (x_m)_{m\in\omega}))_{n\in\omega}$
= ⟨ *Lemma 3.10* ⟩
$f\ (x_n)_{n\in\omega}$ □

Lemma 3.30 *For all* $w \in M(\mathbf{R}_M(D))$, $\bigsqcup_{n\in\omega} M(\mu^e_n \circ \mu^p_n)\ w = w$.

Proof
$\bigsqcup_{n\in\omega} M(\mu^e_n \circ \mu^p_n)\ w$
= ⟨ *Theorem 2.5* ⟩
$\left(\bigsqcup_{n\in\omega} M(\mu^e_n \circ \mu^p_n)\right)\ w$
= ⟨ M *is locally continuous* ⟩
$M\left(\bigsqcup_{n\in\omega} \mu^e_n \circ \mu^p_n\right)\ w$
= ⟨ *Lemma 3.29* ⟩
$M(\mathit{id}_{\mathbf{R}_M(D)})\ w$
= ⟨ M *is a functor* ⟩
$\mathit{id}_{M(\mathbf{R}_M(D))}\ w$
= ⟨ *Identity* ⟩
w □

Lemma 3.31 *Let* $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$. *For all* $m \in \omega$, $\bigsqcup_{n\in\omega} M(f^D_{m,n})\ z_n = z_m$.

Proof We must first show that for all $n \in \omega$, $M(f^D_{m,n})\ z_n \sqsubseteq z_m$.

1. If $m = n$ then

   $M(f^D_{m,n})\ z_n$
   = ⟨ *Assumption* ⟩
   $M(f^D_{n,n})\ z_n$
   = ⟨ *Definition of* $f^D_{n,n}$ *(3.4)* ⟩
   $M(\mathit{id}_{\mathbf{F}^n_{M,D}(\mathbf{O})})\ z_n$

= ⟨ M *is a functor* ⟩
$\quad id_{M(\mathbf{F}^n_{M,D}(\mathbf{O}))}\ z_n$
= ⟨ *Identity* ⟩
$\quad z_n$
= ⟨ *Assumption* ⟩
$\quad z_m$

2. If $m < n$, by induction on $n - m$. Let us assume that it is true for $n = m + k$ for some $k \geq 0$. Then for $n = m + k + 1$ we have

$\quad M(f^D_{m,n})\ z_n$
= ⟨ *Assumption* ⟩
$\quad M(f^D_{m,m+k+1})\ z_{m+k+1}$
= ⟨ *Definition of $f^D_{m,m+k+1}$ (3.4), $m \leq m + k$* ⟩
$\quad M(f^D_{m,m+k} \circ \mathbf{F}^{m+k}_{M,D}(\iota^p))\ z_{m+k+1}$
= ⟨ M *is a functor* ⟩
$\quad (M(f^D_{m,m+k}) \circ M(\mathbf{F}^{m+k}_{M,D}(\iota^p)))\ z_{m+k+1}$
= ⟨ *Composition* ⟩
$\quad M(f^D_{m,m+k})\ (M(\mathbf{F}^{m+k}_{M,D}(\iota^p))\ z_{m+k+1})$
= ⟨ $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ ⟩
$\quad M(f^D_{m,m+k})\ z_{m+k}$
= ⟨ *Inductive hypothesis* ⟩
$\quad z_m$

3. If $m > n$, by induction on $m - n$. Let us assume that it is true for $m = n + k$ for some $k \geq 0$. Then for $m = n + k + 1$ we have

$\quad M(f^D_{m,n})\ z_n$
= ⟨ *Assumption* ⟩
$\quad M(f^D_{n+k+1,n})\ z_n$
= ⟨ *Definition of $f^D_{n+k+1,n}$ (3.4), $n + k \geq n$* ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(\iota^e) \circ f^D_{n+k,n})\ z_n$
= ⟨ M *is a functor* ⟩
$\quad (M(\mathbf{F}^{n+k}_{M,D}(\iota^e)) \circ M(f^D_{n+k,n}))\ z_n$
= ⟨ *Composition* ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(\iota^e))\ (M(f^D_{n+k,n})\ z_n)$
$\sqsubseteq$ ⟨ *Inductive hypothesis, monotonicity* ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(\iota^e))\ z_{n+k}$
= ⟨ $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(\iota^e))\ (M(\mathbf{F}^{n_k}_{M,D}(\iota^p))\ z_{n+k+1})$
= ⟨ *Composition* ⟩
$\quad (M(\mathbf{F}^{n+k}_{M,D}(\iota^e)) \circ M(\mathbf{F}^{n_k}_{M,D}(\iota^p)))\ z_{n+k+1}$
= ⟨ M *is a functor* ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(\iota^e) \circ \mathbf{F}^{n_k}_{M,D}(\iota^p))\ z_{n+k+1}$
= ⟨ $\mathbf{F}^{n+k}_{M,D}$ *is a functor (Theorem 3.1)* ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(\iota^e \circ \iota^p))\ z_{n+k+1}$
$\sqsubseteq$ ⟨ *Lemma 3.5, M, $\mathbf{F}^{n+k}_{M,D}$ are locally monotone* ⟩
$\quad M(\mathbf{F}^{n+k}_{M,D}(id_{\mathbf{F}_{M,D}(\mathbf{O})}))\ z_{n+k+1}$

= ⟨ $\mathbf{F}^{n+k}_{M,D}$ *is a functor (Theorem 3.1)* ⟩
  $M(id_{\mathbf{F}^{n+k+1}_{M,D}(\mathbf{O})})\ z_{n+k+1}$
= ⟨ M *is a functor* ⟩
  $id_{M(\mathbf{F}^{n+k+1}_{M,D}(\mathbf{O}))}\ z_{n+k+1}$
= ⟨ *Identity* ⟩
  z_{n+k+1}
= ⟨ *Assumption* ⟩
  z_m □

At this point, we can proceed to Theorem 3.8 and Theorem 3.9, our central results in this section. These two theorems conclude that functions h^e and h^p define indeed an isomorphism between the domains $\mathbf{R}_M(D)$ and $D + M(\mathbf{R}_M(D))$.

Theorem 3.8 $h^p \circ h^e = id_{\mathbf{F}_{M,D}(\mathbf{R}_M(D))}$
Proof Let $z \in \mathbf{F}_{M,D}(\mathbf{R}_M(D)) = D + M(\mathbf{R}_M(D))$. By case analysis on z.

1. Case $z = \bot$. Then

  $(h^p \circ h^e)\ z$
= ⟨ *Assumption* ⟩
  $(h^p \circ h^e)\ \bot$
= ⟨ *Composition* ⟩
  $h^p\ (h^e\ \bot)$
= ⟨ *Definition of* h^e *(3.14)* ⟩
  $h^p\ (\bot)_{n\in\omega}$
= ⟨ *Definition of* h^p *(3.17)* ⟩
  $\bot$
= ⟨ *Assumption* ⟩
  z

2. Case $z = inl\ t$ for some $t \in D$. Then

  $(h^p \circ h^e)\ z$
= ⟨ *Assumption* ⟩
  $(h^p \circ h^e)\ (inl\ t)$
= ⟨ *Composition* ⟩
  $h^p\ (h^e\ (inl\ t))$
= ⟨ *Definition of* h^e *(3.14)* ⟩
  $h^p\ (\theta^D_n\ (inl\ t))_{n\in\omega}$
= ⟨ *Lemma 3.27* ⟩
  $h^p\ (\eta^D_n\ t)_{n\in\omega}$
= ⟨ *Definition of* h^p *(3.17)* ⟩
  $inl\ t$
= ⟨ *Assumption* ⟩
  z

3. Case $z = inr\ w$ for some $w \in M(\mathbf{R}_M(D))$. Then

  $(h^p \circ h^e)\ z$

= ⟨ *Assumption* ⟩
$(h^p \circ h^e)\ (\mathit{inr}\ w)$
= ⟨ *Composition* ⟩
$h^p\ (h^e\ (\mathit{inr}\ w))$
= ⟨ *Definition of h^e (3.14)* ⟩
$h^p\ (\theta_n^D\ (\mathit{inr}\ w))_{n\in\omega}$
= ⟨ *Lemma 3.28* ⟩
$h^p\ (\sigma_n^D\ (M(\mu_m^p)\ w)_{m\in\omega})_{n\in\omega}$
= ⟨ *Definition of h^p (3.17)* ⟩
$\mathit{inr}\ \left(\bigsqcup_{n\in\omega} M(\mu_n^e)\ (M(\mu_n^p)\ w)\right)$
= ⟨ *Composition* ⟩
$\mathit{inr}\ \left(\bigsqcup_{n\in\omega} (M(\mu_n^e) \circ M(\mu_n^p))\ w\right)$
= ⟨ *M is a functor* ⟩
$\mathit{inr}\ \left(\bigsqcup_{n\in\omega} M(\mu_n^e \circ \mu_n^p)\ w\right)$
= ⟨ *Lemma 3.30* ⟩
$\mathit{inr}\ w$
= ⟨ *Assumption* ⟩
z □

Theorem 3.9 $h^e \circ h^p = \mathit{id}_{\mathbf{R}_M(D)}$

Proof Let $(x_n)_{n\in\omega} \in \mathbf{R}_M(D)$. By case analysis on $(x_n)_{n\in\omega}$ based on Lemma 3.25:

1. If for all $n \in \omega$, $x_n = \bot$, then

$(h^e \circ h^p)\ (x_n)_{n\in\omega}$
= ⟨ *Assumption* ⟩
$(h^e \circ h^p)\ (\bot)_{n\in\omega}$
= ⟨ *Composition* ⟩
$h^e\ (h^p\ (\bot)_{n\in\omega})$
= ⟨ *Definition of h^p (3.17)* ⟩
$h^e\ \bot$
= ⟨ *Definition of h^e (3.14)* ⟩
$(\theta_n^D\ \bot)_{n\in\omega}$
= ⟨ *Definition of θ (3.13)* ⟩
$(\bot)_{n\in\omega}$
= ⟨ *Assumption* ⟩
$(x_n)_{n\in\omega}$
= ⟨ *Identity* ⟩
$\mathit{id}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega}$

2. If there exists a $t \in D$ such that for all $n \in \omega$, $x_n = \eta_n^D\ t$, then

$(h^e \circ h^p)\ (x_n)_{n\in\omega}$

= ⟨ *Assumption* ⟩
$(h^e \circ h^p)\ (\eta_n^D\ t)_{n\in\omega}$
= ⟨ *Composition* ⟩
$h^e\ (h^p\ (\eta_n^D\ t)_{n\in\omega})$
= ⟨ *Definition of h^p (3.17)* ⟩
$h^e\ (\mathit{inl}\ t)$
= ⟨ *Definition of h^e (3.14)* ⟩
$(\theta_n^D\ (\mathit{inl}\ t))_{n\in\omega}$
= ⟨ *Lemma 3.27* ⟩
$(\eta_n^D\ t)_{n\in\omega}$
= ⟨ *Assumption* ⟩
$(x_n)_{n\in\omega}$
= ⟨ *Identity* ⟩
$\mathit{id}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega}$

3. If there exists a $(z_m)_{m\in\omega} \in \mathbf{Q}_M(D)$ such that for all $n \in \omega$, $x_n = \sigma_n^D\ (z_m)_{m\in\omega}$, then

$(h^e \circ h^p)\ (x_n)_{n\in\omega}$
= ⟨ *Assumption* ⟩
$(h^e \circ h^p)\ (\sigma_n^D\ (z_m)_{m\in\omega})_{n\in\omega}$
= ⟨ *Composition* ⟩
$h^e\ (h^p\ (\sigma_n^D\ (z_m)_{m\in\omega})_{n\in\omega})$
= ⟨ *Definition of h^p (3.17)* ⟩
$$h^e\ \left(\mathit{inr}\ \left(\bigsqcup_{n\in\omega} M(\mu_n^e)\ z_n\right)\right)$$
= ⟨ *Definition of h^e (3.14)* ⟩
$$\left(\theta_n^D\ \left(\mathit{inr}\ \left(\bigsqcup_{n\in\omega} M(\mu_n^e)\ z_n\right)\right)\right)_{n\in\omega}$$
= ⟨ *Lemma 3.28* ⟩
$$\left(\sigma_n^D\ \left(M(\mu_m^p)\ \left(\bigsqcup_{n'\in\omega} M(\mu_{n'}^e)\ z_{n'}\right)\right)_{m\in\omega}\right)_{n\in\omega}$$
= ⟨ $M(\mu_m^p)$ *is continuous* ⟩
$$\left(\sigma_n^D\ \left(\bigsqcup_{n'\in\omega} M(\mu_m^p)\ (M(\mu_{n'}^e)\ z_{n'})\right)_{m\in\omega}\right)_{n\in\omega}$$
= ⟨ *Composition* ⟩
$$\left(\sigma_n^D\ \left(\bigsqcup_{n'\in\omega} (M(\mu_m^p) \circ M(\mu_{n'}^e))\ z_{n'}\right)_{m\in\omega}\right)_{n\in\omega}$$
= ⟨ M *is functor* ⟩
$$\left(\sigma_n^D\ \left(\bigsqcup_{n'\in\omega} M(\mu_m^p \circ \mu_{n'}^e)\ z_{n'}\right)_{m\in\omega}\right)_{n\in\omega}$$
= ⟨ *Lemma 3.11* ⟩
$$\left(\sigma_n^D\ \left(\bigsqcup_{n'\in\omega} M(f_{m,n'}^D)\ z_{n'}\right)_{m\in\omega}\right)_{n\in\omega}$$

= ⟨ *Lemma 3.31* ⟩
$(\sigma_n^D\ (z_m)_{m\in\omega})_{n\in\omega}$
= ⟨ *Assumption* ⟩
$(x_n)_{n\in\omega}$
= ⟨ *Identity* ⟩
$\mathit{id}_{\mathbf{R}_M(D)}\ (x_n)_{n\in\omega}$ □

3.5 Additional operations

In this section we define two functions, step and run, which convert a non interleaved computation of type $M(A)$ to an interleaved computation of type $\mathsf{R}(M)(A)$ and vice-versa. The names of these functions indicate their behaviour. The first converts a whole computation to a single atomic *step* in an interleaved computation. The second *runs* the whole sequence of atomic steps of an interleaved computation without allowing other computations to intervene.

In the rest of this section, we assume that (M, η, μ) is a monad and that D is a domain.

Definition 3.18 $\mathit{step}_D : M(D) \to \mathsf{R}(M)(D)$ *is the continuous function defined by:*

$$\mathit{step}_D \quad = \quad h^e \circ \mathit{inr} \circ M(h^e \circ \mathit{inl})$$

Definition 3.19 $\mathit{run}_D : \mathsf{R}(M)(D) \to M(D)$ *is the continuous function defined by:*

$$\mathit{run}_D \quad = \quad \mathit{fix}\ (\lambda\, g.\ [\,\eta_D, \mu_D \circ M(g)\,] \circ h^p)$$

The following theorem states that the composition of run and step, in this order, yields identity. The reverse composition does not yield identity, since it forces an interleaved computation to be executed in one atomic step (it will be used in Section 4 for defining the semantics of $\langle s \rangle$).

Theorem 3.10 $\mathit{run}_D \circ \mathit{step}_D = \mathit{id}_{M(D)}$

Proof

$\mathit{run}_D \circ \mathit{step}_D$
= ⟨ *Unfolding fix in the definition of* run_D *(3.19)* ⟩
$[\,\eta_D, \mu_D \circ M(\mathit{run}_D)\,] \circ h^p \circ \mathit{step}_D$
= ⟨ *Definition of* step_D *(3.18)* ⟩
$[\,\eta_D, \mu_D \circ M(\mathit{run}_D)\,] \circ h^p \circ h^e \circ \mathit{inr} \circ M(h^e \circ \mathit{inl})$
= ⟨ *Theorem 3.8* ⟩
$[\,\eta_D, \mu_D \circ M(\mathit{run}_D)\,] \circ \mathit{id}_{\mathbf{F}_{M,D}(\mathbf{R}_M(D))} \circ \mathit{inr} \circ M(h^e \circ \mathit{inl})$
= ⟨ *Composition with identity* ⟩
$[\,\eta_D, \mu_D \circ M(\mathit{run}_D)\,] \circ \mathit{inr} \circ M(h^e \circ \mathit{inl})$
= ⟨ *Theorem 2.7* ⟩
$\mu_D \circ M(\mathit{run}_D) \circ M(h^e \circ \mathit{inl})$
= ⟨ M *is a functor* ⟩
$\mu_D \circ M(\mathit{run}_D \circ h^e \circ \mathit{inl})$
= ⟨ *Unfolding fix in the definition of* run_D *(3.19)* ⟩
$\mu_D \circ M([\,\eta_D, \mu_D \circ M(\mathit{run}_D)\,] \circ h^p \circ h^e \circ \mathit{inl})$
= ⟨ *Theorem 3.8* ⟩
$\mu_D \circ M([\,\eta_D, \mu_D \circ M(\mathit{run}_D)\,] \circ \mathit{id}_{\mathbf{F}_{M,D}(\mathbf{R}_M(D))} \circ \mathit{inl})$
= ⟨ *Composition with identity* ⟩

$\mu_D \circ M([\eta_D, \mu_D \circ M(\mathit{run}_D)] \circ \mathit{inl})$
$= \langle$ *Theorem 2.7* $\rangle$
$\mu_D \circ M(\eta_D)$
$= \langle$ *M is a monad, 2nd Monad Law* $\rangle$
$\mathit{id}_{M(D)}$ $\Box$

Function prom, which lifts a computation of type $\mathsf{R}(M)(D)$ to a computation of type $M(\mathsf{R}(M)(D))$, is useful in the rest of this section where we establish that $\mathsf{R}(M)(D)$ can be defined as a multi-monad and a strong-monad. These two properties of $\mathsf{R}(M)(D)$ will also be used in Section 4.

Definition 3.20 $\mathit{prom}_D : \mathsf{R}(M)(D) \to M(\mathsf{R}(M)(D))$ *is the continuous function defined by:*

$$\mathit{prom}_D \ = \ [\eta_{\mathbf{R}_M(D)} \circ \mathit{inl}, \mathit{id}_{M(\mathbf{R}_M(D))}] \circ h^p$$

Let us now assume that M is a multi-monad and that $\|_{\mathsf{M}}$ is a *non-deterministic option* operator for computations represented by monad M. It is easy to extend this behaviour to the monad $\mathsf{R}(M)$.

Definition 3.21 *Let M be a multi-monad. Let D be a domain. We define the binary operation $\|_{\mathsf{R(M)}}$: $\mathsf{R}(M)(D) \times \mathsf{R}(M)(D) \to \mathsf{R}(M)(D)$ by:*

$$x \ \|_{\mathsf{R(M)}} \ y \ = \ h^e \ (\mathit{inr} \ (\mathit{prom} \ x \ \|_{\mathsf{M}} \ \mathit{prom} \ y))$$

Monad $\mathsf{R}(M)$ with $\|_{\mathsf{R(M)}}$ is a multi-monad.

Furthermore, we can introduce a way to create a new interleaved computation of type $\mathsf{R}(M)(A \times B)$ given two existing computations of types $\mathsf{R}(M)(A)$ and $\mathsf{R}(M)(B)$. Here we prefer to use monads M and $\mathsf{R}(M)$ in the functional way. If one of the two computations does not require the execution of any atomic step, i.e. if one of the two computations has already been completed, then the other computation is executed and the two results are combined. Otherwise, if both computations require at least one atomic step, we choose non-deterministically which computation will start executing.

Definition 3.22 *Let M be a multi-monad. Let A and B be domains. We define the binary operation $\bowtie_{\mathsf{R(M)}}$: $\mathsf{R}(M)(A) \times \mathsf{R}(M)(B) \to \mathsf{R}(M)(A \times B)$ by:*

$$\begin{aligned}
\bowtie_{\mathsf{R(M)}} \ = \ & \mathit{fix} \ (\lambda\, g.\ \lambda\, \langle x, y\rangle. \\
& \quad [\lambda\, v_x.\ y \ *_{\mathsf{R(M)}} \ (\lambda\, v_y.\ \mathit{unit}_{\mathsf{R(M)}} \ \langle v_x, v_y\rangle), \lambda\, m_x. \\
& \quad [\lambda\, v_y.\ x \ *_{\mathsf{R(M)}} \ (\lambda\, v_x.\ \mathit{unit}_{\mathsf{R(M)}} \ \langle v_x, v_y\rangle), \lambda\, m_y. \\
& \qquad h^e \ (\mathit{inr} \ (m_x \ *_{\mathsf{M}} \ (\lambda\, x'.\ \mathit{unit}_{\mathsf{M}} \ (g \ \langle x', y\rangle)) \ \|_{\mathsf{M}} \\
& \qquad\qquad m_y \ *_{\mathsf{M}} \ (\lambda\, y'.\ \mathit{unit}_{\mathsf{M}} \ (g \ \langle x, y'\rangle))))] \ (h^p \ y)] \ (h^p \ x))
\end{aligned}$$

Monad $\mathsf{R}(M)$ with $\bowtie_{\mathsf{R(M)}}$ is a strong monad.

4 Semantics of concurrency

Consider the simple imperative language whose abstract syntax is given below.

s ::= **skip** | x **:=** e | s **;** s | **if** e **then** s **else** s | **while** e **do** s

It features an empty statement, assignment, sequential composition of statements, a structure for conditional and one more for *while* loops. The symbol $x \in \mathbf{Var}$ represents a variable. The language of expressions e is not important for the purpose of this paper and has therefore been omitted.

We define the denotational semantics of this language, assuming that the values of expressions are elements of the semantic domain $\mathbf{V}$. The *program state,* mapping variables to their current values, is an element of the domain $\mathbf{S} = \mathbf{Var} \to \mathbf{V}$.

As a provision for what will follow, we define a monad transformer D implementing the *direct semantics* approach.[4] If M is a monad, we define the monad $\mathsf{D}(M)$ as:

$$\begin{array}{lcl} \mathsf{D}(M)(D) & = & \mathbf{S} \to M(D \times \mathbf{S}) \\ \mathit{unit}_{\mathsf{D(M)}}\, v & = & \lambda\, s.\ \mathit{unit}_{\mathsf{M}}\, \langle v, s \rangle \\ m\ *_{\mathsf{D(M)}}\ f & = & \lambda\, s.\, m\ s\ *_{\mathsf{M}}\ (\lambda\, \langle v, s' \rangle.\ f\ v\ s') \end{array}$$

State computations created by the direct semantics monad transformer are functions (elements of $\mathsf{D}(M)(D)$) that take the initial program state (an element of $\mathbf{S}$) and return a stateless computation that yields the computed value (an element of D) and the final program state (an element of $\mathbf{S}$). The implementations of $\mathit{unit}_{\mathsf{D(M)}}$ and $*_{\mathsf{D(M)}}$ carry out the propagation of the program state.

We also define an operation for the assignment of values to variables.[5]

$$\begin{array}{lcl} \mathit{store}_{\mathsf{D}} & : & \mathbf{Var} \to \mathbf{V} \to \mathsf{D}(M)(\mathbf{U}) \\ \mathit{store}_{\mathsf{D}}\ x\ v & = & \lambda\, s.\ \mathit{unit}_{\mathsf{M}}\, \langle u, s\{x \mapsto v\} \rangle \end{array}$$

By taking the *identity monad* Id as the argument of D, we obtain the monad M that models our simple notion of computation (ordinary direct semantics).

$$\mathsf{M} \ = \ \mathsf{D}(\mathsf{Id})$$

The meaning of a statement s is a computation $[\![s]\!]$ of type $\mathsf{M}(\mathbf{U})$. Non-termination is represented by the bottom element. We also assume that the meaning of an expression e is a computation $[\![e]\!]$ of type $\mathsf{M}(\mathbf{V})$. The semantic function for the statements of our simple imperative language is completely straightforward.

$$\begin{array}{l} [\![\mathbf{skip}]\!] \ = \ \mathit{unit}\ u \\ [\![x \mathbf{:=} e]\!] \ = \ [\![e]\!] \ * \ (\mathit{store}\ x) \\ [\![s_1\ \mathbf{;}\ s_2]\!] \ = \ [\![s_1]\!] \ * \ (\lambda\, u.\ [\![s_2]\!]) \\ [\![\mathbf{if}\ e\ \mathbf{then}\ s_1\ \mathbf{else}\ s_2]\!] \ = \ [\![e]\!] \ * \ (\lambda\, b.\ \text{if}\ b\ \text{then}\ [\![s_1]\!]\ \text{else}\ [\![s_2]\!]) \\ [\![\mathbf{while}\ e\ \mathbf{do}\ s]\!] \ = \ \mathit{fix}\ (\lambda\, g.\ [\![e]\!] \ * \ (\lambda\, b.\ \text{if}\ b\ \text{then}\ [\![s]\!] \ * \ (\lambda\, u.\ g)\ \text{else}\ \mathit{unit}\ u)) \end{array}$$

Let us now introduce non-determinism and concurrency in our language, by extending it with three new constructs.

$$s \ ::= \ \ldots \mid \ s \oplus s \ \mid \ s \parallel s \ \mid \ \langle s \rangle$$

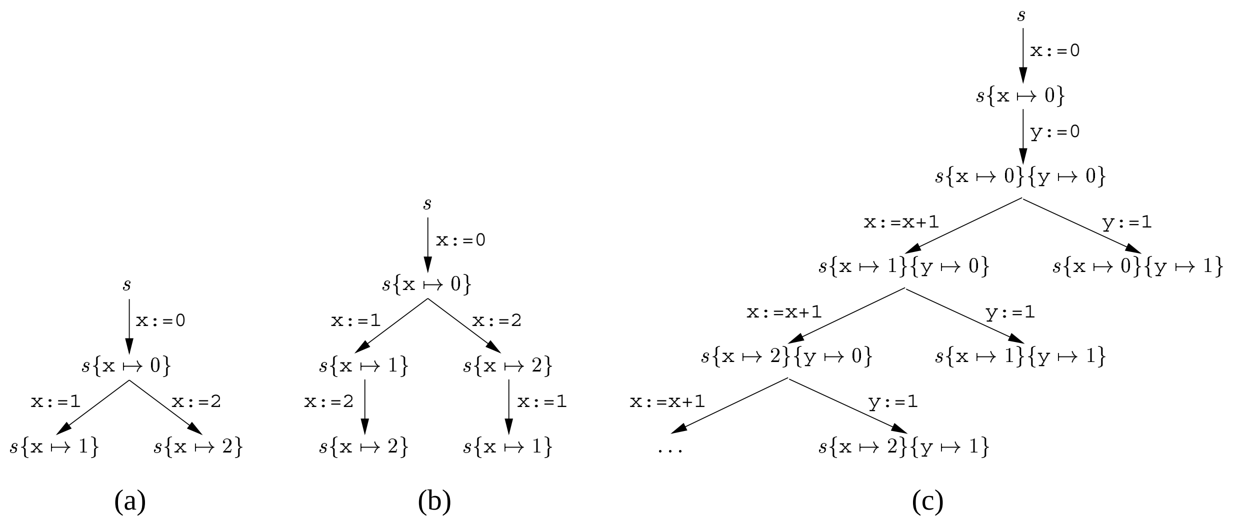
Operator $\oplus$ executes exactly one of the statements that are given as its operands. The selection is non-deterministic. On the other hand, operator $\parallel$ executes both statements that are given as its operands in an interleaved way. Finally, the construct $\langle s \rangle$ executes the statement s in a single atomic step, with no interleaving permitted during its execution.

Before we proceed with the semantics of our extended language, we have to modify the definition of M. By giving the powerdomain monad P as the argument of D, we obtain a multi-monad that can support non-determinism.

$$\mathsf{M} \ = \ \mathsf{D}(\mathsf{P})$$

[4]This is the state monad transformer, as defined in [Lian95, Lian98].

[5]If A and B are domains, $f : A \to B$, $a \in A$ and $b \in B$, we use the notation $f\{a \mapsto b\}$ to denote a function $f' : A \to B$ such that $f'(a) = b$ and, for all $x \neq a$, $f'(x) = f(x)$.



(a) `x:=0; (x:=1` $\oplus$ `x:=2)`
(b) `x:=0; (x:=1` $\|$ `x:=2)`
(c) `x:=0; y:=0;` **`while`** `y=0` **`do`** `(x:=x+1` $\oplus$ `y:=1)`

Figure 1: Three examples of interleaved computations.

The option operator $\|_{\mathsf{M}}$ is defined as:

$$m_1 \;\|_{\mathsf{M}}\; m_2 \;=\; \lambda\, s.\, (m_1\ s) \uplus^{\natural} (m_2\ s)$$

where $\uplus^{\natural}$ is the union operation on powerdomains.

In the semantics of the extended language, we use the monad $\mathsf{R}(\mathsf{M})$ to model interleaved computations. According to Definition 3.21, $\mathsf{R}(\mathsf{M})$ is a multi-monad equiped with a non-deterministic option operator $\|_{\mathsf{R}(\mathsf{M})}$. Also, according to Definition 3.22, $\mathsf{R}(\mathsf{M})$ is a strong monad and operator $\bowtie_{\mathsf{R}(\mathsf{M})}$ can be used to model the interleaving of computations. Furthermore, the *store* operation can easily be lifted to the new domain of computations.

$$\begin{array}{lcl} \mathit{store}_{\mathsf{R}} & : & \mathbf{Var} \to \mathbf{V} \to \mathsf{R}(\mathsf{D}(M))(\mathbf{U}) \\ \mathit{store}_{\mathsf{R}}\ x\ v & = & \mathit{step}\ (\mathit{store}_{\mathsf{D}}\ x\ v) \end{array}$$

The equations defining the meaning of existing language constructs do not require any changes, except for the implicit change that the meanings of statements and expressions are now elements of the semantic domains $\mathsf{R}(\mathsf{M})(\mathbf{U})$ and $\mathsf{R}(\mathsf{M})(\mathbf{V})$ respectively. On the other hand, the semantics of the additional constructs can be easily expressed in terms of $\mathsf{R}(\mathsf{M})$ operations.

$$\begin{array}{lcl} [\![s_1 \oplus s_2]\!] & = & [\![s_1]\!] \;\|_{\mathsf{R}(\mathsf{M})}\; [\![s_2]\!] \\ [\![s_1 \parallel s_2]\!] & = & [\![s_1]\!] \bowtie_{\mathsf{R}(\mathsf{M})} [\![s_2]\!] * (\lambda\, p.\ \mathit{unit}\ u) \\ [\![\langle s \rangle]\!] & = & \mathit{step}\ (\mathit{run}\ [\![s]\!]) \end{array}$$

Figure 1 shows three examples of resumption computations, in the form of directed graphs. Nodes in the graph represent program states, where s denotes an arbitrary initial state. Edges are labeled with the atomic computations (assignments) which transform one program state to another. The non-deterministic behaviour of operators $\oplus$ and $\|$ leads to the presence of branches in the graphs. Also notice the **while** statement in example (c), which may lead to an infinitely long sequence of program states. The terminal nodes in the graph that corresponds to the denotation of program *prog*, i.e. those with no departing edges, represent the final program states: if function *run* is applied to the resumption computation $[\![\mathit{prog}]\!]\ s$, the resulting element of the powerdomain $\mathsf{P}(\mathbf{U} \times \mathbf{S})$ will contain just these program states.

5 Conclusion

This paper defines a general theoretical framework for formalizing the semantics of interleaved computation in concurrent programming languages. The atomic steps in an interleaved computation may themselves be arbitrary computations represented by a given monad M. Furthermore, it is argued that the use of monads enhances the modularity and elegance of the semantics and facilitates the introduction of additional features in a principled way.

Apart from its application in the semantics of concurrency, the resumption monad transformer can be used in the semantics of deterministic languages with unspecified evaluation order, such as Algol and C. The present research was motivated by problems encountered in the formalization of ANSI C [Papa98, Papa01]. A Haskell implementation of the resumption monad transformer, based on the isomorphism between $\mathsf{R}(M)(D)$ and $D + M(\mathsf{R}(M)(D))$, has been used in [Papa00] to define the denotational semantics of an expression language with side effects under a variety of possible evaluation strategies.

References

[Aspe91] A. Asperti and G. Longo, *Categories, Types, and Structures: An Introduction to Category Theory for the Working Computer Scientist*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1991.

[Barr96] M. Barr and C. Wells, *Category Theory for Computing Science*, Prentice-Hall International Series in Computer Science, Prentice Hall, New York, NY, 2nd edition, 1996.

[dBak96] J. de Bakker and E. de Vink, *Control Flow Semantics*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1996.

[Gogu91] J. A. Goguen, "A Categorical Manifesto", *Mathematical Structures in Computer Science*, vol. 1, pp. 49–68, 1991.

[Gunt90] C. A. Gunter and D. S. Scott, "Semantic Domains", in J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, vol. B, chapter 12, pp. 633–674, Elsevier Science Publishers B.V., 1990.

[Gunt92] C. A. Gunter, *Semantics of Programming Languages: Structures and Techniques*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1992.

[Lian95] S. Liang, P. Hudak and M. Jones, "Monad Transformers and Modular Interpreters", in *Conference Record of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL'95)*, San Francisco, CA, January 1995.

[Lian98] S. Liang, *Modular Monadic Semantics and Compilation*, Ph.D. thesis, Yale University, Department of Computer Science, May 1998.

[Mogg90] E. Moggi, "An Abstract View of Programming Languages", Technical Report ECS-LFCS-90-113, University of Edinburgh, Laboratory for Foundations of Computer Science, 1990.

[Papa98] N. S. Papaspyrou, *A Formal Semantics for the C Programming Language*, Ph.D. thesis, National Technical University of Athens, Software Engineering Laboratory, February 1998.

[Papa00] N. S. Papaspyrou, "A Study of Evaluation Order Semantics in Expressions with Side Effects", *Journal of Functional Programming*, vol. 10, no. 3, pp. 227–244, May 2000.

[Papa01] N. S. Papaspyrou, "Denotational Semantics of ANSI C", *Computer Standards and Interfaces*, vol. 23, no. 3, pp. 169–185, July 2001.

[Pier90] B. C. Pierce, "A Taste of Category Theory for Computer Scientists", Technical Report CMU-CS-90-113R, Carnegie Mellon University, School of Computer Science, September 1990.

[Pier91] B. Pierce, *Basic Category Theory for Computer Scientists*, Foundations of Computing Series, MIT Press, Cambridge, MA, 1991.

[Schm86] D. A. Schmidt, *Denotational Semantics: A Methodology for Language Development*, Allyn and Bacon, Newton, MA, 1986.

[Scot71] D. Scott and C. Strachey, "Towards a Mathematical Semantics for Computer Languages", in *Proceedings of the Symposium on Computers and Automata*, pp. 19–46, Brooklyn, NY, 1971, Polytechnic Press.

[Scot82] D. S. Scott, "Domains for Denotational Semantics", in *International Colloquium on Automata, Languages and Programs*, vol. 140 of *Lecture Notes in Computer Science*, pp. 577–613, Berlin, Germany, 1982, Springer Verlag.

[Wadl92] P. Wadler, "The Essence of Functional Programming", in *Proceedings of the 19th Annual Symposium on Principles of Programming Languages (POPL'92)*, pp. 1–14, January 1992.