Basic Research in Computer Science

BRICS RS-94-13 G. Winskel: Stable Bistructure Models of PCF

Stable Bistructure Models of PCF

Glynn Winskel

BRICS Report Series **RS-94-13**

ISSN 0909-0878 **May 1994**

See back inner page for a list of recent publications in the BRICS Report Series. Copies may be obtained by contacting:

**BRICS
Department of Computer Science
University of Aarhus
Ny Munkegade, building 540
DK - 8000 Aarhus C
Denmark**

**Telephone: +45 8942 3360
Telefax: +45 8942 3255
Internet: BRICS@brics.dk**

BRICS publications are in general accessible through WWW and anonymous FTP:

`http://www.brics.dk/`
`ftp ftp.brics.dk (cd pub/BRICS)`

Stable bistructure models of PCF

Preliminary draft

Glynn Winskel
BRICS[1], Comp. Sc. Dept.
Aarhus University
Aarhus, Denmark

Abstract

Stable bistructures are a generalisation of event structures to represent spaces of functions at higher types; the partial order of causal dependency is replaced by two orders, one associated with input and the other output in the behaviour of functions. They represent Berry's bidomains. The representation can proceed in two stages. Bistructures form a categorical model of Girard's linear logic consisting of a linear category together with a comonad. The comonad has a co-Kleisli category which is equivalent to a cartesian-closed full subcategory of Berry's bidomains. A main motivation for bidomains came from the full abstraction problem for Plotkin's functional language PCF. However, although the bidomain model incorporates both the Berry stable order and the Scott pointwise order, its PCF theory (those inequalities on terms which hold in the bidomain model) does not include that of the Scott model. With a simple modification we can obtain a new model of PCF, combining the Berry and Scott orders, which does not have this inadequacy.

1 Introduction

Plotkin's PCF is a programming language based on typed lambda calculus with recursion. It has proved difficult to obtain a domain theory which yields a fully abstract model for PCF, one where the inequations on terms given by the domain orderings coincide with those inequations obtained from the operational semantics, where it is said that one term approximates another if in any closed context at ground type its convergence implies the other's convergence to the same value. In particular, the obvious Scott model, based on domains and continuous functions ordered pointwise, is not fully abstract.

Event structures and related models can be used to represent domains in the semantics of programming languages. For example, event

[1]**B**asic **R**esearch **i**n **C**omputer **S**cience,Centre of the Danish National Research Foundation.

structures give a representation of Berry's cartesian closed category of dI-domains and stable functions [2, 18], while Berry and Curien's category of sequential algorithms on Kahn and Plotkin's concrete datastructures provides a domain theory in which the method of computation is represented explicitly [3]. These two domain theories can be used to give models of PCF. But they do not give order-extensional models; a model is *order-extensional* if elements of the function spaces are ordered according to the pointwise order. If we limit attention to extensional models, that is models where denotations correspond to functions between input and output values, then from Milner's results [12] it is known that any extensional model of PCF which is fully abstract is necessarily order-extensional.

Here we focus on stable-bistructure models of PCF, as a means to obtain order-extensional models which at the same time take some account of the way a function is computed. Stable bistructures were introduced in [17] as a generalisation of event structures to represent a full subcategory of Berry's bidomains [2]. Bidomains possess an intensional, stable ordering, based on the method of computation, and an extensional, pointwise ordering, inherited from Scott's domain theory; their morphisms are functions which respect both, a property shared by functions definable in PCF. The represention of bidomains goes via a view of bistructures as a categorical model of Girard's linear logic [8]. The model consists of a linear category of bistructures together with a comonad; a category of bistructures equivalent to a cartesian-closed full subcategory of bidomains is obtained as the coKleisli category of the model. The work here overlaps with that of [15] which shows bistructures form a model of classical linear logic.

In particular, it is shown how to correct an inadequacy of Berry's bidomains; although the bidomain model incorporates both the stable and pointwise order, its PCF theory (those inequalities on terms which hold in the bidomain model) does not include that of the Scott model (*cf.* [10]), essentially because the model fails to eliminate the first of Curien's examples in [7], Proposition 4.4.2. Adjoining an additional conflict relation to bistructures leads to a model of PCF bearing a satisfactory relationship to the Scott model. The proof of this fact is inspired by the work of Bucciarelli and Ehrhard in the context of making their coherence models extensional—see *e.g.* [5] Chapter 8.

Of course, it is hoped that bistructures have a more important role to play in understanding the extensional fully abstract model of PCF. Two

rather independent lines of work on sequentiality appear central here: one is the work on coherences and hypercoherences [5, 6]; the other the successes in obtaining *intensionally* fully abstract models of PCF [1, 9]—models which although not extensional do have the property that every finite element of the domains is definable by a PCF term. The work on hypercoherences combines smoothly with bistructures (as observed jointly with Gordon Plotkin), providing one route to extensional sequential models of PCF, though not directly to the fully abstract model of PCF.

2 PCF

PCF is a programming language based on LCF, Scott's logic of computable functions. It is a form of typed lambda calculus in which certain terms are singled out as programs.

The set of *types* is the least set containing o (for booleans), ι (for integers) and $(\sigma \to \tau)$ whenever it contains σ and τ. We write $(\sigma_1, \cdots, \sigma_n; \tau)$ for

$$(\sigma_1 \to \cdots (\sigma_n \to \tau) \cdots).$$

The types o and ι are called *ground types*.

Terms are produced from the following collection of constant functions with the indicated types:

$\underline{0}, \underline{1}, \cdots \underline{n}, \cdots :$ type ι	(numerals)
tt, ff : type o	(truth values)
$(+1), (-1)$: type $\iota \to \iota$	(increment and decrement by 1)
Z : type $\iota \to o$	(test for zero)
$\supset_\iota$: type $(o, \iota, \iota; \iota)$	(conditional with integer result)
$\supset_o$: type $(o, o, o; o)$	(conditional with boolean result)

Starting with the above collection of constants and countably many variables $v_i, i \in \omega$, for each type the terms are given by the formation rules:

1. Every variable x_i^σ is a term of type σ.
2. Every constant of type σ is a term of type σ.
3. if M is a term of type $\sigma \to \sigma$ then $Y_\sigma M$ (its least fixed point) is a term of type σ.

4. If M and N are terms of type $\sigma \to \tau$ and σ respectively then MN is a terms of type τ.

5. If M is a term of type τ then $\lambda v^\sigma M$ is one of type $\sigma \to \tau$.

The *programs* are closed terms of ground type. Intuitively they yield observable output, concrete values consisting of integers or truth values; other terms are significant only as subterms of programs.

An *operational semantics* is given to the language by defining eval, a partial function from programs to constants. It is defined using an *immediate reduction relation* $\to$ between terms: $\mathrm{eval}(M) = c$ iff $M \to^* c$, for any program M and constant c.

The immediate reduction relation is given by:

1. $(+1)\underline{n} \to \underline{n+1}$

2. $(-1)\underline{n+1} \to \underline{n}$

3. $\mathrm{Z}\underline{0} \to \mathrm{tt} \qquad \mathrm{Z}\underline{n+1} \to \mathrm{ff}$

4. $\supset_\sigma \mathrm{tt} MN \to M \qquad \supset_\sigma \mathrm{ff} MN \to N$, where $\sigma = o, \iota$

5. If $M \to M'$ then $\supset_\sigma M \to \supset_\sigma M'$ for M, M' of type o, and σ a type o or ι

6. If $M \to M'$ then $(MN) \to (M'N)$

7. If M is $(+1)$ or (-1) or Z and $N \to N'$ then $(MN) \to (MN')$

8. $Y_\sigma M \to M(Y_\sigma M)$

9. $((\lambda v\ M)N) \to [N/v]M$

The relation $\to$ is a partial function so eval is well-defined above.

We base the notion of a standard model for PCF on *(standard) type structure* consisting of:

1. A cpo D_σ for each type σ with D_ι being isomorphic to the flat cpo of integers and D_o to the flat cpo of booleans.

2. For all types σ and τ a two place application operations $\cdot : D_{\sigma\to\tau} \times D_\sigma \to D_\tau$ which is continuous and order-extensional *i.e.*

$$x \sqsubseteq x' \text{ iff } \forall y.\ x \cdot y \sqsubseteq x' \cdot y.$$

Condition 2 ensures that the element of $D_{\sigma\to\tau}$ are in 1-1 correspondence with a subset of the continuous functions $[D_\sigma \to D_\tau]$ so that the ordering on $D_{\sigma\to\tau}$ is the restriction of the pointwise ordering on functions.

With respect to a type structure the *environment* Env consists of all type-respecting functions ρ from variables into $\bigcup_\sigma D_\sigma$

A *standard model* for PCF consists of a type structure D and a semantics $\mathfrak{M}$ a type-respecting map giving values in D to terms with respect to an environment ρ. They are required to satisfy the following conditions:

1. The terms $\underline{n}$, $(+1)$, (-1), Z, $\supset_o$, $\supset_\iota$ and least fixed points get their usual interpretation.

2. $\mathfrak{M}[\![v]\!]\rho = \rho(v)$
   $\mathfrak{M}[\![MN]\!] = \mathfrak{M}[\![M]\!]\rho \cdot \mathfrak{M}[\![N]\!]\rho$
   $(\mathfrak{M}[\![\lambda v.M]\!]\rho) \cdot d = \mathfrak{M}[\![M]\!]\rho[d/v]$

—$\rho[d/v]$ is the environment obtained from ρ updated so the variable v is assigned the value d.

Not all type structures determine models; there may be simply not enough functions in the domains to support the semantics. An obvious standard model is obtained by taking the type structure so that $D_{\sigma\to\tau} = [D_\sigma \to D_\tau]$, all continuous functions from D_σ to D_τ, with the application operator just the ordinary application of functions. Many other models are possible and according to criteria derived from the operational semantics the obvious model is not the best.

The denotational semantics should "match" the operational semantics. Plotkin defined two natural operational relations. Terms are of interest only insofar as they are part of programs. For this reason it is natural to regard two terms as operationally equivalent if they can be freely substituted for each other in a program without affecting its behaviour. Formally define the equivalence relation between terms of the same type by : $M \approx N$ iff whenever $C[M]$ and $C[N]$ are programs eval$(C[M])$ and eval$(C[N])$ are both undefined or otherwise defined and equal. More generally an operational preorder can be defined by: $M \ll N$ iff whenever $C[M]$ and $C[N]$ are programs then whenever eval$(C[M])$ is c then so is eval$(C[N])$. Clearly $M \approx N$ iff $M \ll N$ and $N \ll M$. For a semantics $\mathfrak{M}$ the expected semantic counterparts of these two relations are the relations on terms given by $M \sqsubseteq_M N$ iff $\mathfrak{M}[\![M]\!]\rho \sqsubseteq \mathfrak{M}[\![N]\!]\rho$ for all ρ, and $M =_M N$ iff $M \sqsubseteq_M N$ and $N \sqsubseteq_M M$. In the circumstance

when the relations $\ll$ and $\sqsubseteq_M$ coincide the semantics $\mathfrak{M}$ is said to be *fully abstract*.

For a standard semantics $\mathfrak{M}$, Plotkin showed the denotational relations will be included in the corresponding operational ones. However the converse will not generally hold. In particular Plotkin showed the Scott model with the obvious semantics, based on taking $D_{\sigma\to\tau}$ as all continuous functions $[D_\sigma \to D_\tau]$ is not fully abstract. The counterexample depended on producing two terms which were operationally equivalent but denotationally distinct through acting differently on "parallel-or"—parallel-or is the continuous function on booleans with $\bot$ which extends the usual boolean disjunction and returns true if either argument is true, regardless of whether the other argument is $\bot$.

3 Event structures

As a rehearsal for the work on bistructures we will present the Scott model for PCF in terms of event structures. We will show that a full subcategory of Scott domains with continuous functions, *viz.* the coherent prime algebraic domains [13], is equivalent to a cartesian-closed category of event structures; the continuous function space of two prime algebraic domains can itself be represented by an event structure. The types of PCF can be then be interpreted as event structures. We will present the category of event structures as the coKleisli category of a model of Girard's linear logic.

Recall that an event structure is a structure $(E, \leq, \#)$ where

- E is a set of *events*,
- $\leq$ is a partial order of *causal dependency*
- $\#$ is a binary, irreflexive, symmetric relation of *conflict*

satisfying

$$e\#e' \leq e'' \Rightarrow e\#e''.$$

The *configurations* (or states) of such an event structure are subsets $x \subseteq E$ which are

- *consistent:* $\forall e_1, e_2 \in x.\ \neg e_1\#e_2$,
- *secured:* $\forall e, e' \in E.\ e' \leq e \in x \Rightarrow e' \in x$.

Ordered by inclusion, the configurations $(\Gamma(E), \subseteq)$, form a coherent prime algebraic domain [13]; such domains are precisely the infinitely distributive, coherent Scott domains [19].

An instance of the causal dependency ordering $e' \leq e$ when e and e' are distinct, is usually understood as meaning that the event e causally depends on the event e', that the event e can only occur after e' has ocurred. Usually we impose a finiteness axiom:

$$\{e' \mid e' \leq e\} \text{ is finite, for all events } e,$$

expressing that an event has finite causes. Though we won't do this here, in order to represent a portion of Scott domain theory via a model of intuitionistic linear logic (an example of a nonstable model of linear logic, *cf.* [11]).

The categorical model we have in mind is equivalent to the category of coherent prime algebraic domains, with additive functions (*i.e.* functions preserving arbitrary lubs). The category has as objects event structures (without the axiom of finite causes) and morphisms configurations of a "function space" of event structures, constructed as follows:
Let $E_i = (E_i, \leq_i, \#_i)$, $i = 0, 1$, be event structures. Define

$$E_0 \multimap E_1 = (E_0 \times E_1, \leq, \#)$$

where

$$\begin{aligned} (e_0, e_1) \leq (e_0', e_1') &\;\Leftrightarrow\; e_0' \leq_0 e_0 \;\&\; e_1 \leq_1 e_1', \\ (e_0, e_1) \# (e_0', e_1') &\;\Leftrightarrow\; \neg e_0 \#_0 e_0' \;\&\; e_1 \#_1 e_1' \end{aligned}$$

The configurations of $E_0 \multimap E_1$ correspond to additive functions from $\Gamma(E_0)$ to $\Gamma(E_1)$—additive functions are determined by their action on complete primes[2] which correspond to events. While the inclusion ordering on configuration reflects the pointwise ordering on functions; in particular, the function events (e_0, e_1) correspond to special step functions and the order $\leq$ to the pointwise order between them.

A morphism from E_0 to E_1 is defined to be a configuration of $E_0 \multimap E_1$. As such it is a relation between the events of E_0 and E_1. Composition in the category is that of relations. The category is a model of intuitionistic

[2]A complete prime of a Scott domain $(D, \sqsubseteq)$ is an element p for which whenever $p \sqsubseteq \bigsqcup X$ then $p \sqsubseteq x$ for some $x \in X$.

linear logic. Its tensor, for instance, is given in a coordinatewise fashion: For event structures $E_i = (E_i \leq_i, \#_i)$, for $i = 0, 1$, define

$$E_0 \otimes E_1 = (E_0 \times E_1, \leq, \#)$$

where

$$\begin{array}{rcl} (e_0, e_1) \leq (e_0', e_1') & \Leftrightarrow & e_0 \leq_0 e_0' \;\&\; e_1 \leq_1 e_1' \quad \text{and} \\ (e_0, e_1) \# (e_0', e_1') & \Leftrightarrow & e_0 \#_0 e_0' \text{ or } e_1 \#_1 e_1'. \end{array}$$

The comonad operation is

$$!E = (\Gamma(E)^0, \subseteq, \#_!)$$

for an event structure E, with *finite* configurations $\Gamma(E)^0$, on which conflict is defined by

$$x \#_! y \Leftrightarrow \exists e_1 \in x, e_2 \in y.\ e_1 \# e_2,$$

which is equivalent to incompatibility with respect to inclusion in $\Gamma(E)^0$. The continuous functions from $\Gamma(E_0)$ to $\Gamma(E_1)$, between configurations of event structures E_0, E_1, are in 1-1 correspondence with configurations of $!E_0 \multimap E_1$. A configuration y of $!E_0 \multimap E_1$ can be applied to a configuration x of E_0,

$$y.x = \{e \mid \exists x_0 \subset x.\ (x_0, e) \in y\},$$

yielding a configuration of E_1. The association of y to the function $(x \mapsto y \cdot x)$ gives the 1-1 correspondence with continuous functions from $\Gamma(E_0)$ to $\Gamma(E_1)$.

Linear types σ, given by

$$\sigma ::= o \mid \iota \mid \sigma \multimap \sigma' \mid\ !\sigma$$

can be interpreted as event structures C_σ. The event structure C_o representing the booleans has two events t, f, corresponding to $\mathtt{tt}, \mathtt{ff}$, related by the conflict relation and with the identity relation as causal dependency. Similarly, C_ι is the event structure representing the flat domain of integers; it has events ω, the natural numbers, pairwise in conflict, with the identity relation as its causal dependency. A type $\sigma \multimap \sigma'$ denotes the event structure $C_\sigma \multimap C_{\sigma'}$, while $!\sigma$ denotes the event structure $!C_\sigma$. Throughout, we will identify a PCF type $\sigma \to \sigma'$ with the linear type $!\sigma \multimap \sigma'$.

We can reformulate the Scott model of PCF via the representation using event structures. In the model a PCF type σ is interpreted as the cpo of configurations $\Gamma(C_\sigma)$. For M a term of type σ, and for ρ an environment, we define $\mathcal{C}[\![M]\!]\rho \in \Gamma(C_\sigma)$. For example,

$$\begin{aligned}\mathcal{C}[\![MN]\!]\rho =& \{e \mid \exists d \subseteq \mathcal{C}[\![N]\!]\rho \ \&\ (d,e) \in \mathcal{C}[\![M]\!]\rho\}\\ \mathcal{C}[\![\lambda v M]\!]\rho =& \{(d,e) \mid e \in \mathcal{C}[\![M]\!]\rho[d/v]\},\end{aligned}$$

reflecting the way application and currying are represented in event structures.

It is also possible to give an event structure representation of the stable model of PCF (see *e.g.* [18]), though this won't be a standard model in our sense because it isn't order-extensional. However we can obtain a stable model, which is at the same time order-extensional, by working with bistructures.

4 Bistructures

In the last section we obtained a model of PCF from event structures, but at a price. In this model the order $\leq$ can no longer be thought of as causal dependency, $\leq$ does not correspond to precedence in time, and we have lost one of the key intuitions behind event structures. The loss stems from the definition of the order $\leq$ for $(E_0 \multimap E_1)$ of event structures $E_i = (E_i, \leq_i, \#_i), i = 0, 1$. Its events are ordered by:

$$(e_0, e_1) \leq (e_0', e_1') \ \Leftrightarrow\ e_0' \leq_0 e_0 \ \&\ e_1 \leq_1 e_1'$$

The reversal in the $\leq_0$ order can lead to $\leq$ violating the axiom of finite causes, even though $\leq_0$ and $\leq_1$ do not: an infinite, ascending chain of events in E_0 can give rise to an infinite, *descending* chain in $E_0 \multimap E_1$.

However, if we factor $\leq$ into two orderings, one associated with input (on the left) and output (on the right) we can expose two finitary orderings. Define

$$\begin{aligned}(e_0, e_1) \leq^L (e_0', e_1') &\Leftrightarrow e_0' \leq_0 e_0 \ \&\ e_1 = e_1'\\ (e_0, e_1) \leq^R (e_0', e_1') &\Leftrightarrow e_0' = e_0 \ \&\ e_1 \leq_1 e_1'.\end{aligned}$$

Then, it is clear that $\leq$ factors as

$$(e_0, e_1) \leq (e_0', e_1') \Leftrightarrow (e_0, e_1) \leq^L (e_0', e_1) \ \&\ (e_0', e_1) \leq^R (e_0', e_1'),$$

and that this factorisation is unique. Provided the orderings of E_0 and E_1 are finitary, then so are $\leq^R$ and the converse ordering $\geq^L$. Their finitary nature reflects their expressing temporal precedence: increasing with respect to $\leq^R$ corresponds to producing further output over time; decreasing with respect to $\leq^L$ is associated with inspecting further input. This factorisation is the first step towards the definition of bistructures. We obtain further encouragement when we notice a suggestive characterisation of the stable additive functions. We have already observed that additive functions from $\Gamma(E_0)$ to $\Gamma(E_1)$ correspond to configurations of $E_0 \multimap E_1$, *i.e.* to subsets of events $x \subseteq E_0 \times E_1$ which are $\leq$-downwards closed and consistent. Consider the $\leq^L$ maximal events $M(x)$ of such a configuration. The set $M(x)$ will correspond to what Girard calls the *trace* of the function. The associated function will be *stable* if whenever events $(e_0, e_1), (e_0', e_1)$ are in $M(x)$ then either e_0 and e_0' are equal or in conflict. Just as inclusion between traces determines the stable order, so an inclusion $M(x) \subseteq M(y)$, for configurations x, y for stable additive functions, means that the functions are in the stable order.

We can jointly enforce the consistency required of all configurations of $(E_0 \multimap E_1)$ and the extra property of stability by introducing a new conflict relation. For two events $(e_0, e_1), (e_0', e_1')$ of the linear function space, write $(e_0, e_1) \smile (e_0', e_1')$ iff they are distinct and

$$\neg e_0 (\# \cup 1_{E_0}) e_0' \ \& \ e_1 (\# \cup 1_{E_1}) e_1'.$$

Stable, additive functions from $\Gamma(E_0)$ to $\Gamma(E_1)$ correspond to configurations x of $E_0 \multimap E_1$ such that

- x is consistent with respect to the relation $\smile$, in the sense that

$$\forall e, e' \in M(x).\ \neg e \smile e',$$

- x is downwards closed with respect to $\leq$.

The new conflict relation $\smile$ imposes stability, and we call it *stable conflict*. Unlike $\#$ it is not preserved upwards with respect to $\leq$. However we observe that it satisfies the weaker axioms: $\downarrow^L \subseteq \asymp$ and $\uparrow^R \subseteq \frown$.[3]

We have not yet specified how to repeat the function-space construction at higher orders, when the event structures come already equipped

[3]We shall use Girard's notation: $\asymp$ is the reflexive closure of the irreflexive relation $\smile$, and $\frown$, the complement of $\smile$, is the reflexive closure of the irreflexive relation $\frown$. It is clear that specifying one relation determines all the others.

with $\leq^L$ and $\leq^R$ orders. The definition of stable bistructures (based on that in [17]) arose in extending these results beyond the first order.

Definition: A *(stable) bistructure* consists of

$$(E, \leq^L, \leq^R, \smile)$$

where E is a countable set, $\leq^L, \leq^R$ are partial orders on E, $\smile$ is a binary irreflexive symmetric relation on E, called *stable conflict*, for which:

1. Defining $\leq = (\leq^L \cup \leq^R)^*$, we have the unique factorisation property:
$$e \leq e' \Rightarrow \exists!\ e''.\ e\ \leq^L e'' \leq^R e'$$
[It follows that $\leq$ is a partial order.]

2. Defining $\preceq = (\geq^L \cup \leq^R)^*$,
   (a) $\{e' \mid\ e' \preceq\ e\}$ is finite, for all e,
   (b) $\preceq$ is a partial order.

3. (a) $\downarrow^L \subseteq \succ$ (b) $\uparrow^R \subseteq \frown$
The two compatibility relations mean
$$e \downarrow^L e' \ \Leftrightarrow\ \exists e''.\ e''\ \leq^L\ e\ \&\ e'' \leq^L\ e'$$
$$e \uparrow^R e' \ \Leftrightarrow\ \exists e''.\ e\ \leq^R\ e''\ \&\ e' \leq^R\ e''.$$

Ordinary, countable event structures, $(E, \leq, \#)$, satisfying the axiom of finite causes, yield special bistructures $(E, 1_E, \leq, \#)$, in which the $\leq^L$ order is degenerate.

Let $E = (E, \leq^L, \leq^R, \smile)$ be a bistructure. As in the motivating discussion, for $x \subseteq E$, we take $M(x)$ to consist of the $\leq^L$-maximal events in x, and define an *extensional configuration* of E, to be a subset $x \subseteq E$ which is

- *consistent:* $\forall e_1, e_2 \in M(x).\ e_1 \frown e_2$,
- $\leq$*-downwards closed:* $\forall e, e'.\ e' \leq e \in x \Rightarrow e' \in x$.

Alternatively, we can choose to regard the "trace" $M(x)$ itself as a configuration: A *stable configuration* of the bistructure E is a subset $y \subseteq E$ which is

- *consistent:* $\forall e_1, e_2 \in y.\ e_1 \frown e_2$,
- *secured:* $\forall e \in y \forall e' \leq^R e \exists\ e''.\ e' \leq^L e'' \in y$.

Notice that e'' is unique in any consistent set because $\downarrow^L \subseteq \asymp$. The securedness condition is quite intuitive. It says that for any output, lesser output must have arisen previously through the same or lesser input. The finiteness axiom 2(a) on bistructures, says that the succession of inspecting input, delivering output, inspecting more input, and so on, leading up to a particular output for some input, is a finite procedure. The intuition that event structures capture temporal precedence, lost in the representation of coherent prime algebraic domains and continuous functions, is regained in this modified form for bistructures.

The two forms of configuration are intimately related. For $x \subseteq E$, define its $\leq$-downwards closure to be

$$\lceil x \rceil = \{e \in E \mid \exists e' \in x.\ e \leq e'\}.$$

Proposition 1 *For y a stable configuration, $\lceil y \rceil$ is an extensional configuration. For x an extensional configuration, $M(x)$ is a stable configuration. Moreover, $\lceil - \rceil$ and $M(-)$ are mutual inverses, giving a 1-1 correspondence between stable configurations and extensional configurations.*

By Proposition 1, extensional and stable configurations carry the same information. However, here it's more convenient to work with extensional configurations. We write $\Gamma(E)$ for the set of extensional configurations of a bistructure E; write $\Gamma(E)^0$ for its finite configurations. (For a development paralleling that here, but based on stable configurations see [15].)

The inclusion orders on the two kinds of configurations, extensional and stable, correspond to the extensional and stable orders of a Berry bidomain:

The order $\sqsubseteq$ is given by inclusion on *extensional* configurations, and on *stable* configurations by

$$x \sqsubseteq y \text{ iff } \forall e \in x \exists e' \in y.\ e \leq^L e'.$$

The order $\sqsubseteq^R$ is given by inclusion on *stable* configurations, and on *extensional* configurations by

$$x \sqsubseteq^R y \text{ iff } M(x) \subseteq M(y).$$

Proposition 2 $(\Gamma(E), \sqsubseteq^R, \sqsubseteq)$ *is a Berry bidomain in which* $\sqsubseteq^R$ *is the stable and* $\sqsubseteq$ *the extensional order.*

Another order on configurations, $\sqsubseteq^L$, will come to play an important role. For $x, y \in \Gamma(E)$ define

$$x \sqsubseteq^L y \Leftrightarrow x \sqsubseteq y \;\&\; (\forall y' \in \Gamma(E).\; x \sqsubseteq y' \;\&\; y' \sqsubseteq^R y \Rightarrow y = y').$$

Thus, $x \sqsubseteq^L y$ means y is a $\sqsubseteq^R$-minimum configuration such that $x \sqsubseteq y$. In characterising $\sqsubseteq^L$ we use:

Notation: Let $E = (E, \leq^L, \leq^R, \smile)$ be a bistructure.

Let $x \in \Gamma(E)$. Let $e \in x$. Then, by Proposition 1, $e \leq e'$, for some $e' \in M(x)$. Factorising $\leq$, we obtain $e \leq^L e'' \leq^R e'$, and as $M(x)$ is a stable configuration there is a unique $e_{max} \in M(x)$ such that $e'' \leq^L e_{max}$. This shows that any event e in an extensional configuration x is $\leq^L$-dominated by a unique event e_{max} in $M(x)$. We write $m(e, x)$ for this event e_{max}.

For $x \in \Gamma(E)$, we define the relativised relation $\preceq_x$ as the reflexive, transitive closure of $\preceq_x^1$ where, for $e, e' \in x$,

$$e \preceq_x^1 e' \Leftrightarrow_{def} \exists e'' \in M(x).\; e'' \leq^L e \;\&\; e'' \leq^R e'.$$

We characterise compatibility with respect to $\sqsubseteq^R$ and the order $\sqsubseteq^L$:

Lemma 3 *For a bistructure* E*, let* $x, y \in \Gamma(E)$*,*

$$(i)\; x \uparrow^R y \;\&\; e \in M(x) \cap M(y) \Rightarrow (\forall e' \in E.\; e' \preceq_x e \Leftrightarrow e' \preceq_y e)$$
$$(ii)\; x \sqsubseteq^L y \Leftrightarrow x \sqsubseteq y \;\&\; \forall e' \in M(y) \exists e \in M(x).\; e' \preceq_y m(e, y).$$

In particular, we obtain a unique factorisation for configurations:

$$x \sqsubseteq y \Rightarrow \exists! z.\; x \sqsubseteq^L z \sqsubseteq^R y$$

—the intermediate value z is such that $M(z)$ is the stable configuration generated by the $\leq^L$-image of x in $M(y)$. The factorisation restricts to finite configurations.

5 A category of bistructures

Morphisms between bistructures will correspond to configurations of a linear function space.

Assuming

$$E_i = (E_i, \leq_i^L, \leq_i^R, \smile_i), \quad i \ = 0, 1,$$

are bistructures, define

$$E_0 \multimap E_1 = (E_0 \times E_1, \leq^L, \leq^R, \smile)$$

where

$$\begin{array}{rcl} (e_0, e_1) \leq^L (e_0', e_1') & \Leftrightarrow & e_0' \leq^R e_0 \ \& \ e_1 \leq^L e_1' \\ (e_0, e_1) \leq^R (e_0', e_1') & \Leftrightarrow & e_0' \leq^L e_0 \ \& \ e_1 \leq_1^R e_1' \end{array}$$

and

$$(e_0, e_1) \asymp (e_0', e_1') \ \Leftrightarrow \ e_0 \frown_0 e_0' \ \& \ e_1 \asymp_1 e_1'.$$

We define the category of stable bistructures by taking morphisms E_0 to E_1 to be extensional configurations of $E_0 \multimap E_1$, composed as relations. Of course, we should show that this composition is well-defined and has identities.

Lemma 4 *Let α be an extensional configuration of $E_0 \multimap E_1$ and β an extensional configuration of $E_1 \multimap E_2$. Then their relational composition $\beta \circ \alpha$ is an extensional configuration of $E_0 \multimap E_2$. Furthermore*

$$M(\beta \circ \alpha) = M(\beta) \circ M(\alpha).$$

A bistructure of the form $E \multimap E$ has the relation

$$Id_E = \{(e, e') \mid e' \leq e\}$$

as an extensional configuration. It is an identity for composition. Furthermore $M(Id_E)$ is the identity relation on events of E.

Proof: This proof is trickier than might be thought. One proof relies on Lemma 5 of [15] which shows that relational composition of stable

configurations yields a stable configuration. An argument very similar to the proof of Lemma 5, yields

$$\lceil \beta \circ \alpha \rceil = \lceil \beta \rceil \circ \lceil \alpha \rceil$$

for *stable* configurations—here it is helpful to observe that for stable configurations x,

$$\lceil x \rceil = \{e' \mid \exists e \in x.\ e' \leq^L e\}.$$

It follows that the composition of *extensional* configurations α, β is well-defined, being

$$\beta \circ \alpha = \lceil M(\beta) \rceil \circ \lceil M(\alpha) \rceil = \lceil M(\beta) \circ M(\alpha) \rceil$$

where $M(\beta) \circ M(\alpha)$ is a stable configuration by Lemma 5. Applying M, we obtain $M(\beta \circ \alpha) = M(\beta) \circ M(\alpha)$. □

Morphisms on bistructures determine (special) extensional, linear (= stable and additive) functions on bidomains:

Proposition 5 *Let $F \in \Gamma(E_0 \multimap E_1)$ and $x \in \Gamma(E_0)$. Defining*

$$(\Gamma F)(x) = \{b \mid \exists a \in x.\ (a, b) \in F\}$$

yields a configuration of E_1. The function $\Gamma F : \Gamma(E_0) \to \Gamma(E_1)$ is linear with respect to $\sqsubseteq^R$ and continuous with respect to $\sqsubseteq$. The operation Γ is a faithful functor from the category of bistructures to bidomains.

5.1 Categorical constructions

Throughout this section assume $E_i = (E_i, \leq_i^L, \leq_i^R, \#_i)$ are bistructures, for $i = 0, 1$.

Define their *tensor* $E_0 \otimes E_1$ to be $(E_0 \times E_1, \leq^L, \leq^R, \smile)$ where

$$\begin{aligned}
(e_0, e_1) \leq^L (e_0', e_1') &\ \Leftrightarrow\ e_0 \leq_0^L e_0' \ \&\ e_1 \leq_1^L e_1' \\
(e_0, e_1) \leq^R (e_0', e_1') &\ \Leftrightarrow\ e_0 \leq_0^R e_0' \ \&\ e_1 \leq_1^R e_1' \\
(e_0, e_1) \smile (e_0', e_1') &\ \Leftrightarrow\ e_0 \smile_0 e_0' \text{ or } e_1 \smile_1 e_1'.
\end{aligned}$$

This construction extends to a functor; for $\alpha_0 : E_0 \to E_0'$ and $\alpha_1 : E_1 \to E_1'$, define define $\alpha_0 \otimes \alpha_1 : E_0 \otimes E_1 \to E_0' \otimes E_1'$ by

$$((e_0, e_1), (e_0', e_1')) \in \alpha_0 \otimes \alpha_1 \Leftrightarrow (e_0, e_0') \in \alpha_0 \ \&\ (e_1, e_1') \in \alpha_1.$$

This operation is easily checked to be well-defined and because morphisms compose as relations the functor laws follow directly.

The unit of tensor, $\mathbf{1}$, has a single event on which $\leq^L$ and $\leq^R$ are the identity, and the conflict relation is empty:

$$\mathbf{1} = (\{\bullet\}, 1, 1, \emptyset).$$

Monoidal-closure follows from the isomorphism

$$(E_0 \otimes E_1 \multimap E_2) \cong (E_0 \multimap (E_1 \multimap E_2))$$

natural in event structures E_0, E_2.

The *product* in the category bistructures is given by

$$E_0 \times E_1 = (E_0 \cup E_1, \leq^L, \leq^R, \smile)$$

obtained as the juxtaposition of the two bistructures, assumed disjoint, so for example $\smile = \smile_0 \cup \smile_1$ iff they are injections of conflicting events in one or the other component.

The *coproduct* is given by

$$E_0 \oplus E_1 = (E_0 \cup E_1, \leq^L, \leq^R, \smile)$$

again obtained as the juxtaposition of the two bistructures, made disjoint, but this time extending conflict across the two components, so

$$\smile = \smile_0 \cup \smile_1 \cup (E_0 \times E_1) \cup (E_1 \times E_0).$$

In fact, bistructures form a model of *classical* linear logic [15]. Define *linear negation*, the involution of linear logic, by

$$E^\perp = (E, \geq^R, \geq^L, \frown)$$

where $E = (E, \leq^L, \leq^R, \smile)$. Clearly $(E^\perp)^\perp = E$.

To get the *exponential* $!E$, of a bistructure E, we use the orderings $\leq^L, \leq^R$ on configurations introduced in Section 4. Define

$$!E = (\Gamma(E)^0, \sqsubseteq^L, \sqsubseteq^R, \not\uparrow^R)$$

where $x \uparrow^R y \Leftrightarrow_{def} \exists z \in \Gamma(E).\ x, y \subseteq z$.

Lemma 6 *$!E$ is a bistructure. The operation $!$ extends to a functor: for $\alpha : E \to E'$, define $!\alpha : !E \to !E'$ by taking*

$$!\alpha = \{(x, y) \in \Gamma(E)^0 \times \Gamma(E')^0 \mid y \subseteq \alpha x\}.$$

A configuration y of a bistructure $!E_0 \multimap E_1$ may be applied to a configuration x of E_0:

$$y \cdot x = \{e \mid \exists x_0 \subseteq x.\ (x_0, e) \in y\}$$

This operation of *application* yields a configuration of E_1. It shows how the configuration y determines a function $x \mapsto y \cdot x$. In fact, $!E_0 \multimap E_1$ has configurations in 1-1 correspondence with elements in the function space

$$[(\Gamma(E_0), \sqsubseteq^R, \sqsubseteq) \to (\Gamma(E_1), \sqsubseteq^R, \sqsubseteq)]$$

in the cartesian-closed category of Berry's bidomains:

Proposition 7 *Let E_0, E_1 be bistructures. For $y \in \Gamma(!E_0 \multimap E_1)$ and $x \in \Gamma(E_0)$ define*

$$\widehat{y}(x) = \{e \mid \exists x_0 \subseteq x.\ (x_0, e) \in y\}.$$

Then $\widehat{y}$ is a function $\Gamma(E_0) \to \Gamma(E_1)$ which is continuous with respect to $\sqsubseteq$ and stable with respect to $\sqsubseteq^R$. In fact, $y \mapsto \widehat{y}$ is a 1-1 correspondence between configurations of $!E_0 \multimap E_1$ and such functions.

More completely:

Theorem 8 *The category* **BS** *forms a linear category in the sense of [16]. The exponential $!$ forms a comonad on the category* **BS**. *Together they form a model of classical linear logic (a Girard category in the sense of [16]). The associated co-Kleisli category is equivalent to a cartesian-closed full subcategory of Berry's bidomains, where morphisms are continuous with respect to the extensional order and stable with respect to the stable order.*

In particular, $!$ is a comonad. Its *counit* $\epsilon_E : !E \to E$ is given by

$$(x, e) \in \epsilon_E \Leftrightarrow e \in x,$$

where x is an event of $!E$ and e an event of E. Its *comultiplication*: $\mu_E : !E \rightarrow !!E$ is given by

$$(x, X) \in \mu_E \Leftrightarrow \cup X \subseteq x,$$

where x is an event of $!E$ and X an event of $!!E$. There is also a natural isomorphism

$$\alpha_{E_0,E_1} : !E_0 \otimes !E_1 \cong !(E_0 \times E_1)$$

where

$$((x_0, x_1), y) \in \alpha_{E_0,E_1} \Leftrightarrow y \subseteq x_0 \cup x_1$$

for events (x_0, x_1) of $!E_0 \otimes !E_1$ and y of $!(E_0 \times E_1)$—we assume that the events of E_0 and E_1 are disjoint.

6 PCF in bidomains

Linear types can be interpreted as bistructures straightforwardly: The event structures of Section 3, interpreting the types o and ι of booleans and integers, correspond to special bistructures, with degenerate $\leq^L$ relations; linear types $\sigma \multimap \sigma'$ and $!\sigma$ are interpreted by the associated constructions on bistructures. For M a term of type σ, and any environment ρ , we can define, by structural induction, the denotation of M in an environment ρ lies in the cpo of extensional configurations of the bistructure for the type of M. Application between an extensional configuration y of type $\sigma \rightarrow \sigma'$ and an extensional configuration of type σ is as defined in Section 5.1. The semantic definition holds no surprises; in particular, the clauses for applications and λ-abstractions follow that earlier for event-structure semantics in Section 3.

But, as Jim and Meyer point out in [10] there is a problem with the bidomain, and so bistructure model of PCF. The theory it induces on PCF is not even comparable with that of the Scott model. Despite cuting down to stable functions we are no closer to the fully abstract model for PCF, and, as argued in [10], it is hard to extend PCF to make the bidomain (and bistructure) model fully abstract. The two PCF terms

$$\mathbf{lor} \equiv \lambda x \lambda y (\supset_o x \ \mathrm{tt} (\supset_o y \ \mathrm{tt} \ \mathrm{ff})) \text{ and}$$
$$\mathbf{ror} \equiv \lambda x \lambda y (\supset_o y \ \mathrm{tt} (\supset_o x \ \mathrm{tt} \ \mathrm{ff}))$$

for disjunction, one first evaluating its left argument and the other first its right argument are denoted by the configurations lor and ror respectively

in the bistructure model. The restriction to stable functions excludes the parallel-or function so in bistructures *lor* and *ror* have no upper bound with respect to either the stable or extensional order. For this reason the bistructure model has a configuration representing an *ortester*, a function returning tt on *lor* and ff on *ror*. The following term will denote an *ortester-tester*

$$F \equiv \lambda h \supset_o (h\ \mathbf{lor})(\supset_o (h\ \mathbf{ror})\Omega_o \mathrm{tt})\Omega_o$$

—it yields tt precisely when h is an ortester, and otherwise gives $\perp$, the denotation of $\Omega_o \equiv Y_o(\lambda v^o v)$. According to the bistructure model where an ortester is represented, F, with type σ, will not receive the same denotation as $\Omega_\sigma \equiv Y_\sigma(\lambda v^\sigma v)$. However, according to the Scott model F and Ω_σ will be identified, *i.e.* $\mathcal{C}[\![F]\!] = \mathcal{C}[\![\Omega_\sigma]\!]$. This is because there cannot be an ortester in the Scott model; by monotonicity, it would have to send parallel-or to a truth value bigger than both tt and ff, which is absurd.

It follows that the PCF theory in bistructures does not include the PCF theory in the Scott model; according to the Scott semantics the ortester-tester F is equivalent to Ω_σ, which not so in the bistructure semantics. In fact, the theories in the Scott and bistructure semantics are incomparable. In the paper [14], Plotkin exhibits two terms which are operationally equivalent but whose denotations act differently on parallel-or. According to the bistructure semantics the two terms are equivalent (bistructures rule out parallel-or, the only continuous function on which the two terms differ), but they are not equivalent according to the Scott semantics.

7 Extending bistructures

The incomparability in two PCF theories arises because, in limiting attention to stable conflict $\smile$, we have lost track of the conflict $\#$ in event-structures. In the event-structure model, the events *lor* and *ror* corresponding to the functions **lor** and **ror** are not in conflict which ensures that the two events (lor, t) and (ror, f) are in conflict in the function space. So in the event-structure model undesirable functions like ortesters are not represented. One way to correct this problem with bistructures, and bidomains, is to reinstate $\#$ as extra structure on bistructures, marrying together the constructions of Sections 3 and 5.

An *extended bistructure* is a structure $E = (E, \leq^L, \leq^R, \smile, \#)$ where

- $(E, \leq^L, \leq^R, \smile)$ is a stable bistructure, and
- *extensional conflict* $\#$ is a binary, irreflexive, symmetric relation on E satisfying
$$e\#e' \leq e'' \Rightarrow e\#e''.$$

A *configuration* is an extensional configuration x of the bistructure which also satisfies
$$\forall e_1, e_2 \in x.\ \neg e_1\#e_2.$$
We write $\Gamma(E)$ for the set of configurations.

We can now imitate Section 5, to obtain a new model of PCF. As the *linear function space* of extended bistructures $E_i = (E_i, \leq_i^L, \leq_i^R, \smile_i, \#_i)$ for $i = 0, 1,$, we take
$$E_0 \multimap E_1 = (E, \leq^L, \leq^R, \smile, \#)$$
where $(E, \leq^L, \leq^R, \smile)$ is the linear function space of bistructures, and
$$(e_0, e_1)\#(e_0', e_1') \Leftrightarrow \neg e_0\#_0 e_0' \ \&\ e_1\#_1 e_1'.$$
As before, morphisms are configurations of the function space which compose as relations.

The category is monoidal-closed with respect to a tensor of two extended bistructures, given as in Sections 3 and 5. Precisely, two extended bistructures $E_i = (E_i, \leq_i^L, \leq_i^R, \smile_i, \#_i)$ for $i = 0, 1$, have *tensor*
$$E_0 \otimes E_1 = (E, \leq^L, \leq^R, \smile, \#)$$
where $(E, \leq^L, \leq^R, \smile)$ is the tensor of the bistructures and
$$(e_0, e_1)\#(e_0', e_1') \Leftrightarrow e_0\#_0 e_0' \text{ or } e_1\#_1 e_1'.$$
Tensor extends to a functor just as in Section 5.

Products are obtained by disjoint juxtaposition, and coproducts similarly, but extending the two conflict relations across the event structures.

As the *exponential* of an extended bistructure $E = (E, \leq^L, \leq^R, \smile, \#)$, we take
$$!E = (\Gamma(E)^0, \sqsubseteq^L, \sqsubseteq^R, \Subset^R, \#_!)$$

where

$$x\#_!y \Leftrightarrow \exists e_1 \in x, e_2 \in y.\ e_1\#e_2.$$

It extends to a functor as in Section 5. The same definitions of counit and comultiplication suffice to form a comonad and still have natural isomorphism α as defined there. The natural isomorphism α, where

$$\alpha_{E_0,E_1} :!E_0 \otimes !E_1 \cong !(E_0 \times E_1),$$

together with monoidal-closure ensures that the coKleisli category of ! is cartesian-closed. We obtain a model of intuitionistic linear logic.

Linear types σ can be interpreted as extended bistructures B_σ. The extended bistructure B_o, representing the booleans, has the two events t, f, corresponding to tt, ff, related by both conflict relations and with the identity relation as $\leq^L$, $\leq^R$. Similarly, B_ι is the event structure representing the flat domain of integers; it has events ω, the natural numbers, pairwise in conflict according to both conflict relations, with the identity relation for $\leq^L$ and $\leq^R$. A type $\sigma \multimap \sigma'$ denotes the extended bistructure $B_\sigma \multimap B_{\sigma'}$, while $!\sigma$ denotes the extended bistructure $!B_\sigma$.

We obtain a standard model for PCF by interpreting a PCF type σ as the cpo of configurations $\Gamma(B_\sigma)$, ordered by inclusion. Application between a configuration y of type $\sigma \to \sigma'$ and a configuration of type σ is again given by

$$y \cdot x = \{e \mid \exists x_0 \subseteq x.\ (x_0, e) \in y\}.$$

For M a term of type σ, and any environment ρ, we define, by structural induction, $\mathcal{B}[\![M]\!]\rho$ to be an element of $\Gamma(B_\sigma)$. For example,

$$\begin{aligned}\mathcal{B}[\![MN]\!]\rho &= \{e \mid \exists d \subseteq \mathcal{B}[\![N]\!]\rho\ \&\ (d,e) \in \mathcal{B}[\![M]\!]\rho\}\\ \mathcal{B}[\![\lambda v M]\!]\rho &= \{(d,e) \mid e \in \mathcal{B}[\![M]\!]\rho[d/v]\}.\end{aligned}$$

Now it can be checked that the configurations lor and ror denoting the PCF terms **lor** and **ror** are not in extensional conflict in $B_{!(o,o;o)}$, so that an ortester taking lor to tt and ror to ff can not be represented by any configuration of the extended bistructure model.

It's reassuring that we have avoided ortesters in the extended-bistructure model. But of course we want that the PCF theory of the Scott model is included in that of the extended-bistructure model. The argument that this is so goes via an embedding between the interpretations the two models give to types. Here it is useful to refer to the representation of the Scott model by event structures given in Section 3.

Definition: By structural induction on linear types σ, define functions θ_σ from the events of B_σ to the events of C_σ as follows:

- θ_o, θ_ι are the identity functions on events.
- $\theta_{!\sigma}(x) = \lceil \theta_\sigma x \rceil$, for events x of $B_{!\sigma}$.
- $\theta_{\sigma_0 \multimap \sigma_1}((e_0, e_1)) = (\theta_{\sigma_0}(e_0), \theta_{\sigma_0}(e_1))$, for events (e_0, e_1) of $B_{\sigma_0 \multimap \sigma_1}$.

Proposition 9 *For any linear type σ, the function θ_σ is injective from the events of B_σ to the events of C_σ and such that*

$$\begin{array}{rcl} e \leq e' \text{ in } B_\sigma & \Leftrightarrow & \theta_\sigma(e) \leq \theta_\sigma(e') \text{ in } C_\sigma \\ e \# e' \text{ in } B_\sigma & \Leftrightarrow & \theta_\sigma(e) \# \theta_\sigma(e') \text{ in } C_\sigma \end{array}$$

where e, e' are events of B_σ.

Proof: By structural induction on σ. □

Notation: We call environments in the event-structure semantics of Section 3 $\mathcal{C}$-environments, while those in the extended-bistructure semantics will be $\mathcal{B}$-environments. For ρ a $\mathcal{B}$-environment, we write $\theta \circ \rho$ for the environment sending variable x^σ to $\lceil \theta_\sigma \rho(x) \rceil$—we'll drop the types on θ whenever we can get away with it.

Lemma 10 *For M a PCF term of type σ, ρ a $\mathcal{B}$-environment,*

$$e \in \mathcal{B}[\![M]\!]\rho \Leftrightarrow \theta(e) \in \mathcal{C}[\![M]\!]\theta \circ \rho.$$

Proof: We use a result of Berry: In any least fixed point semantics of PCF, which $\mathcal{B}[\![-]\!]$ and $\mathcal{C}[\![-]\!]$ are, the denotation of any term is the least upper bound of the denotations of its Böhm tree approximations (see [4], Theorem 4.3.1). A Böhm tree is a well-typed term of PCF of the form

- $\Omega_\sigma \equiv Y_\sigma(\lambda v^\sigma v^\sigma)$, or
- $\lambda v_1 \cdots v_n \, u BT_1 \cdots BT_m$, where u is a variable or PCF constant and $BT_1, \cdots, BT_m$ are themselves Böhm trees.

By Berry's result it is sufficient to verify the statement of the lemma for terms M which are Böhm trees. This we do by structural induction

on Böhm trees. Consider the cases according to the form of the Böhm tree.

Case Ω*:* If the Böhm tree is Ω, then both $\mathcal{B}[\![\Omega]\!]$ and $\mathcal{C}[\![\Omega]\!]\theta \circ \rho$ are the empty set.

Case v*, a variable:* This case follows directly from the semantics of variables.

Case $f\overrightarrow{BT}$*, where* f *is a PCF constant:* At ground and first-order types σ the events is both models coincide and θ_σ is the identity on events. PCF constants f (remember these do not include the fixed-point operators Y_σ) receive the same set of events as denotations in the two models, and hence $\mathcal{B}[\![f]\!]\rho = \mathcal{C}[\![f]\!]\theta \circ \rho$, for an arbitrary $\mathcal{B}$-environment ρ. Now, consider a term $f\overrightarrow{BT}$, for f a PCF constant, where, by induction, for any argument BT, we have for a $\mathcal{B}$-environment ρ, that

$$d \subseteq \mathcal{B}[\![BT]\!]\rho \Leftrightarrow \theta d \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho.$$

Here BT must have type ι or o where θ is the identity. Thus

$$d \subseteq \mathcal{B}[\![BT]\!]\rho \Leftrightarrow d \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho$$

Now, using self-evident vector notation, we see that for any $\mathcal{B}$-environment ρ,

$$\begin{aligned} e \in \mathcal{B}[\![f\overrightarrow{BT}]\!]\rho &\Leftrightarrow \exists \overrightarrow{d \subseteq \mathcal{B}[\![BT]\!]\rho}.\ (\overrightarrow{d}, e) \in \mathcal{B}[\![f]\!]\rho \\ &\Leftrightarrow \exists \overrightarrow{d \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho}.\ (\overrightarrow{d}, e) \in \mathcal{C}[\![f]\!]\theta \circ \rho \\ &\Leftrightarrow e \in \mathcal{C}[\![f\overrightarrow{BT}]\!]\theta \circ \rho \ \ \textit{i.e.}\ \ \theta(e) \in \mathcal{C}[\![f\overrightarrow{BT}]\!]\theta \circ \rho. \end{aligned}$$

Case $v\overrightarrow{BT}$*, where* v *is a variable:* We consider just vBT—the argument is essentially the same for $v\overrightarrow{BT}$. We require

$$e \in \mathcal{B}[\![v\ BT]\!]\rho \ \Leftrightarrow\ \theta(e) \in \mathcal{C}[\![v\ BT]\!]\theta \circ \rho.$$

"$\Rightarrow$" Suppose $e \in \mathcal{B}[\![v\ BT]\!]\rho$. Then for some $d \in \Gamma(B_\sigma)^0$, with σ the type of BT,

$$d \subseteq \mathcal{B}[\![BT]\!]\rho\ \&\ (d, e) \in \rho(v).$$

Thus $(\theta(d), \theta(e)) \in \theta \circ \rho(v)$. By induction, $\theta d \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho$ so $\theta(d) \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho$. But now, from the definition of the semantics of $\mathcal{C}[\![v\ BT]\!]\theta \circ \rho$,

we obtain $\theta(e) \in \mathcal{C}[\![v\ BT]\!]\theta \circ \rho$.

"$\Leftarrow$" Suppose $\theta(e) \in \mathcal{C}[\![v\ BT]\!]\theta \circ \rho$. Then for some $d \in \Gamma(C_\sigma)^0$, with σ the type of BT,

$$d \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho\ \&\ (d, \theta(e)) \in \theta(\rho(v)).$$

Now, $(d, \theta(e)) \in \theta(\rho(v)) = \lceil \theta\rho(v) \rceil$ implies there is $(d_0, e_0) \in \rho(v)$ with

$$\theta(d_0) \subseteq d\ \&\ \theta(e) \leq \theta(e_0).$$

Hence $\theta d_0 \subseteq \mathcal{C}[\![BT]\!]\theta \circ \rho$. By induction, $d_0 \subseteq \mathcal{B}[\![BT]\!]\rho$. Thus $e_0 \in \mathcal{B}[\![v\ BT]\!]\rho$, and because $e \leq e_0$ (by Proposition 9), $e \in \mathcal{B}[\![v\ BT]\!]\rho$, as required.

Case $\lambda\overrightarrow{v}\ u\overrightarrow{BT}$, where u is a PCF constant or variable. The cases where $\overrightarrow{v}$ is an empty list have already been covered above. From the semantics, for a $\mathcal{B}$-environment ρ,

$$\begin{aligned}
(\overrightarrow{d}, e) \in \mathcal{B}[\![\lambda\overrightarrow{v}u\overrightarrow{BT}]\!]\rho &\Leftrightarrow e \in \mathcal{B}[\![u\overrightarrow{BT}]\!]\rho[\overrightarrow{d/z}] \\
&\Leftrightarrow \theta(e) \in \mathcal{C}[\![u\overrightarrow{BT}]\!]\theta \circ (\rho[\overrightarrow{s/z}]) \\
&\text{— this case of the induction hypothesis} \\
&\quad \text{has already been covered,} \\
&\Leftrightarrow \theta(e) \in \mathcal{C}[\![u\overrightarrow{BT}]\!](\theta \circ \rho)[\overrightarrow{\theta(d)/z}] \\
&\Leftrightarrow (\overrightarrow{\theta(d)}, \theta(e)) \in \mathcal{C}[\![\lambda\overrightarrow{v}u\overrightarrow{BT}]\!]\theta \circ \rho \\
&\Leftrightarrow \theta((\overrightarrow{d}, e)) \in \mathcal{C}[\![\lambda\overrightarrow{v}u\overrightarrow{BT}]\!]\theta \circ \rho.
\end{aligned}$$

□

As a corollary we can relate the theories of the two models, the event-stucture model (representing the Scott model) and the bistructure model. For PCF terms M, N of the same type, we have the following preorders induced by the semantics:

$M \sqsubseteq_B N \Leftrightarrow \mathcal{B}[\![M]\!]\rho \sqsubseteq \mathcal{B}[\![N]\!]\rho$ for all $\mathcal{B}$-environments ρ, and
$M \sqsubseteq_C N \Leftrightarrow \mathcal{C}[\![M]\!]\rho \sqsubseteq \mathcal{C}[\![N]\!]\rho$ for all $\mathcal{C}$-environments ρ.

Theorem 11 *For PCF terms M, N of the same type:*

$$M \sqsubseteq_C N \Rightarrow M \sqsubseteq_B N$$

Proof: Assume $M \sqsubseteq_C N$. For ρ a $\mathcal{B}$-environment, argue:

$$\begin{aligned} e \in \mathcal{B}[\![M]\!]\rho &\Rightarrow \theta(e) \in \mathcal{C}[\![M]\!]\theta \circ \rho \text{ by Lemma 10} \\ &\Rightarrow \theta(e) \in \mathcal{C}[\![N]\!]\theta \circ \rho \text{ as } M \sqsubseteq_C N, \\ &\Rightarrow e \in \mathcal{B}[\![N]\!]\rho \text{ by Lemma 10.} \end{aligned}$$

Thus $\mathcal{B}[\![M]\!]\rho \subseteq \mathcal{B}[\![N]\!]\rho$ for an arbitrary $\mathcal{B}$-environment ρ, *i.e.* $M \sqsubseteq_B N$.□

8 Concluding remarks

Can the syntax of PCF be extended to make the extended-bistructures model fully abstract? More significantly, can bistructures help construct *extensional* fully abstract models for pure PCF? Gordon Plotkin and I have had some preliminary success adapting ideas like those here to a sequential, rather than just a stable model, by replacing the conflict relations of extended bistructures by Ehrhard's hypercoherences (in a stable and extensional form)—presently, it seems we can eliminate all but the third of Curien's examples in Proposition 4.4.2 [7].

References

[1] Abramsky, S., Jagadeesan, R., and Malacaria, P., Games and full abstraction for PCF (first and second preliminary announcement). Linear Logic and Types electronic newsgroups, 1993.

[2] Berry, G., Modèles complètement adéquats et stables des lambda-calculs typés. Thèse de Doctorat d'Etat, Université de Paris VII, 1979.

[3] Berry, G.,and Curien, P-L., Sequential algorithms on concrete data structures. Theoretical Computer Science 20, pp.265-321, 1982.

[4] Berry, G., Curien, P-L., and Levy, J-J., The full-abstraction problem: state of the art. Proc. French-US Seminar on the Applications of Algebra to Language Definition and Compilation, Fontainebleau 1982, Cambridge University Press, 1985.

[5] Bucciarelli, A., Sequential models of PCF: some contributions to the domain-theoretic approach to full abstraction. PhD thesis in Computer Science, University of Pisa, March 1993.

[6] Ehrhard, T., Hypercoherences: a strongly stable model of linear logic. Mathematical Structures in Computer Science, 1993.

[7] Curien, P-L., Categorical combinators, sequential algorithms, and functional programming. Second edition, Birkhäuser, 1993.

[8] Girard, J-Y., Linear logic. Theoretical Computer Science 50, 1987.

[9] Hyland, J.M.E., and Ong, C-H.L., Games and full abstraction for PCF. Linear Logic and Types electronic newsgroups, 1993.

[10] Jim, T., and Meyer, A., Full abstraction and the context lemma. Invited lecture, proceedings of Theoretical Aspects of Computer Software, Sendai, Springer Lecture Notes in Computer Science 526, 1991.

[11] Hoofman, R., Nonstable models of linear logic. PhD thesis, University of Utrecht, 1992.

[12] Milner, A.R.G., Fully abstract models of typed lambda-calculi. Theoretical Computer Science 4(1), pp. 1-23, 1977.

[13] Nielsen, M., Plotkin, G.D., and Winskel, G., Petri nets, Event structures and Domains, part 1. Theoretical Computer Science, vol. 13, 1981.

[14] Plotkin, G.D., LCF considered as a programming language. Theoretical Computer Science 5(3), pp.223-256, 1977.

[15] Plotkin, G.D., and Winskel, G., Bistructures, bidomains and linear logic. To appear in the proceedings of ICALP 94, Springer Lecture Notes in Computer Science, 1994.

[16] Seely, R.A.G., Linear logic, $*$-autonomous categories and cofree coalgebras. Contempory Mathematics, vol.92, 1989.

[17] Winskel, G., Events in Computation. PhD thesis, University of Edinburgh, available as a Comp. Sc. report, 1980.

[18] Winskel, G., Event structures. Lectures for the Advanced Course on Petri nets, September 1986, Springer Lecture Notes in C.S., vol.255, 1987.

[19] Winskel, G., An introduction to event structures. In the lecture notes for the REX summerschool in temporal logic, May 88, in Springer Lecture Notes in C.S., vol.354, 1989.

Recent Publications in the BRICS Report Series

RS-94-4 **Nils Klarlund and Michael I. Schwartzbach.** ***Graphs and Decidable Transductions based on Edge Constraints*****. February 1994, 19 pp. Appears in:** ***Trees in Algebra and Programming CAAP '94*** **(ed. S. Tison), LNCS 787, 1994.**

RS-94-5 **Peter D. Mosses.** ***Unified Algebras and Abstract Syntax*****. March 1994, 21 pp. To appear in:** ***Recent Trends in Data Type Specification*** **(ed. F. Orejas), LNCS 785, 1994.**

RS-94-6 **Mogens Nielsen and Christian Clausen.** ***Bisimulations, Games and Logic*****. April 1994, 37 pp. Full version of paper to appear in:** ***New Results and Trends in Computer Science*****, LNCS, 1994.**

RS-94-7 **André Joyal, Mogens Nielsen, and Glynn Winskel.** ***Bisimulation from Open Maps*****. May 1994, 42 pp. Journal version of LICS '93 paper.**

RS-94-8 **Javier Esparza and Mogens Nielsen.** ***Decidability Issues for Petri Nets*****. 1994, 23 pp.**

RS-94-9 **Gordon Plotkin and Glynn Winskel.** ***Bistructures, Bidomains and Linear Logic*****. May 1994, 16 pp. To appear in the proceedings of ICALP '94, LNCS, 1994.**

RS-94-10 **Jakob Jensen, Michael Jørgensen, and Nils Klarlund.** ***Monadic second-order Logic for Parameterized Verification*****. May 1994.**

RS-94-11 **Nils Klarlund.** ***A Homomorphism Concept for ω-Regularity*****. May 1994.**

RS-94-12 **Glynn Winskel and Mogens Nielsen.** ***Models for Concurrency*****. May 1994, 144 pp. To appear as a chapter for the** ***Handbook of Logic and the Foundations of Computer Science*****, Oxford University Press.**

RS-94-13 **Glynn Winskel.** ***Stable Bistructure Models of PCF*****. May 1994, 26 pp.** ***Preliminary draft*****. Invited lecture for MFCS 94. To appear in the LNCS proceedings of MFCS 94.**