A Coinduction Principle for Recursive Data Types Based on Bisimulation

Marcelo P. Fiore*
Department of Computer Science
Laboratory for Foundations of Computer Science
University of Edinburgh, The King's Buildings
Edinburgh EH9 3JZ, Scotland
<mf@dcs.ed.ac.uk>

Synopsis

The concept of bisimulation from concurrency theory [Par81, Mil89] is used to reason about recursively defined data types. From two strong-extensionality theorems stating that the equality (resp. inequality) relation is maximal among all bisimulations, a proof principle for the final coalgebra of an endofunctor on a category of data types (resp. domains) is obtained. As an application of the theory developed, an internal full abstraction result (in the sense of [AO93]) for the canonical model of the untyped call-by-value λ-calculus is proved.

Also, the operational notion of bisimulation and the denotational notion of final semantics are related by means of conditions under which both coincide.

Introduction

This paper provides foundations for a reasoning principle (coinduction) for establishing the equality of potentially infinite elements of self-referencing (or circular) data types. As it is well-known, such data types not only form the core of the denotational approach to the semantics of programming languages [SS71], but also arise explicitly as recursive data types in functional programming languages like Standard ML [MTH90] or Haskell [HPJW92]. In the latter context, the coinduction principle provides a powerful technique for establishing the equality of programs with values in recursive data types (see examples herein and in [Pit94]).

Our analysis is exclusively denotational in that the proof principle is derived from the characterising property of the mathematical interpretation (as final coalgebras) of

*Research partially supported by Fundación Antorchas and The British Council grant ARG 2281/14/6.

the recursive data types. To do this we borrow the notion of bisimulation from concurrency [Par80, Par81, Mil89] and apply it to data types.

In concurrency theory, the concept of bisimulation (in its various forms) plays two main roles:

1. it provides an appropriate notion of equivalence between processes, and
2. it induces a proof method (coinduction) due to Park which reads: "to prove two processes observationally equivalent, show that they are bisimilar".

In the theory of data types, we will see that bisimulation is useful for proving two elements of a recursive data type equal. Let us think of a bisimulation on a recursive data type as relating observationally equivalent elements of that type. As such relations, bisimulations have enough closure properties to guarantee that related elements are equal. Thus, the following coinduction principle is intuitively valid:

> If R is a bisimulation on the recursive data type U, and u and u' are elements of U for which $u \, R \, u'$, then $u = u'$.

We illustrate the previous discussion and the applicability of the coinduction principle with an example. The Haskell data type `IntStream` of integer streams (i.e. infinite lists of elements of the built-in type `Int` of integers) with constructor `Cons`,

```
data IntStream = Cons Int IntStream,
```

and selectors

```
headS ::  IntStream -> Int
headS (Cons h t) = h

tailS ::  IntStream -> IntStream
tailS (Cons h t) = t
```

can be given a denotational semantics using the *final coalgebra* of the endofunctor $\mathbb{Z} \times \_ : \mathbf{Set} \to \mathbf{Set}$ where $\mathbb{Z}$ is the set of integers, $\times$ is the cartesian product functor, and $\mathbf{Set}$ is the category of small sets and functions. By this we mean that the interpretations of `IntStream`, `headS` and `tailS`, say $\mathbb{Z}^\omega$, h and t, are characterised by the following universal property: for every set T equipped with a map $\tau : T \to \mathbb{Z} \times T$, there exists a unique map $[\![\_]\!] : T \to \mathbb{Z}^\omega$ such that

$$\begin{tikzcd} T \arrow[r, "\tau"] \arrow[d, "{[\![\_]\!]}"'] & \mathbb{Z} \times T \arrow[d, "\mathrm{id} \times {[\![\_]\!]}"] \\ \mathbb{Z}^\omega \arrow[r, "{\langle h, t \rangle}"'] & \mathbb{Z} \times \mathbb{Z}^\omega \end{tikzcd}$$

commutes (i.e. such that $\langle h,t\rangle \circ [\![\_]\!] = (\mathrm{id} \times [\![\_]\!]) \circ \tau$). We henceforth call any $\tau : T \to \mathbb{Z} \times T$ a *structure map* as it endows T with a stream-like structure with a head-like selector $\pi_1 \circ \tau : T \to \mathbb{Z}$ and a tail-like selector $\pi_2 \circ \tau : T \to T$ that can be used to successively read-off integers from any element of T; the canonical way to read-off a stream of integers from $s \in T$ is given by $[\![s]\!] \in$ `IntStream`. For example, consider $T = \mathbb{N}$ (the set of natural numbers) with structure map $\tau : \mathbb{N} \to \mathbb{Z} \times \mathbb{N}$ given by $\tau(n) \stackrel{def}{=} \big(f(n), g(n)\big)$ where $f : \mathbb{N} \to \mathbb{Z}$ and $g : \mathbb{N} \to \mathbb{N}$; then, for $n \in \mathbb{N}$, $[\![n]\!] \in$ `IntStream` is the stream $f(n), f\big(g(n)\big), \ldots, f\big(g^k(n)\big), \ldots$.

The universal property of streams provides a canonical interpretation (*final semantics*, see [RT93]) for certain operations on streams. For example, consider the operation

```
jump ::  Int -> Int -> IntStream
-- jump n m counts by m's from n
-- e.g. jump 2 3 = 2, 5, 8, 11, ...
jump n m = Cons n (jump (n+m) m)
```

The set of terms $T = \{\texttt{jump}\ n\ m \mid n, m \in \mathbb{Z}\}$ can be given a structure map $\tau : T \to \mathbb{Z} \times T$ by orienting the defining equation for `jump` from left to right. That is, setting $\tau(\texttt{jump}\ n\ m) = (n, \texttt{jump}\ (n+m)\ m)$. Then, by the above universal property, it follows that there exists a unique interpretation $[\![\_]\!] : T \to \mathbb{Z}^\omega$ such that for every $n, m \in \mathbb{Z}$,

$$h\,[\![\texttt{jump}\,n\,m]\!] = n,$$

and

$$t\,[\![\texttt{jump}\,n\,m]\!] = [\![\texttt{jump}\,(n+m)\,m]\!].$$

Hence, $\texttt{jump}\,n\,m$ denotes the stream $n, n+m, n+2m, \ldots, n+km, \ldots$.

Now consider the perfect (or alternate) shuffle [HU79] of two streams

```
shuffle ::  IntStream -> IntStream -> IntStream
shuffle s1 s2 = Cons (headS s1) (shuffle s2 (tailS s1))
```

and the values `nat` = `jump 0 1`, `even` = `jump 0 2`, and `odd` = `jump 1 2`. Intuitively, `nat` = `shuffle even odd`. But, how do we prove it?

For comparison we offer proofs by induction and coinduction (identifying programs with their denotations):

1. Inductive proof.

   By induction on $\mathbb{N}$ one can prove the following two lemmas:

   - For $n \in \mathbb{N}$, $\texttt{tailS}^n\ \texttt{nat} = \texttt{jump}\ n\ \texttt{1}$.
   - For $n \in \mathbb{N}$, $\texttt{tailS}^n\ (\texttt{shuffle even odd}) = \texttt{shuffle}\ (\texttt{jump}\ n\ \texttt{2})\ (\texttt{jump}\ (n+1)\ \texttt{2})$.

from which it follows that, for $n \in \mathbb{N}$,

$$\texttt{headS}\,(\texttt{tailS}^n\ \texttt{nat}) \quad = \quad \texttt{headS}\,\big(\texttt{tailS}^n\ (\texttt{shuffle even odd})\big).$$

Appealing to the following extensionality principle

$$(\eta) \quad [\forall\, n \in \mathbb{N}.\ \texttt{headS}\ (\ \texttt{tailS}\ ^n s) = \ \texttt{headS}\ (\ \texttt{tailS}\ ^n s')] \ \Rightarrow \ s = s'$$

we are done.

2. Coinductive proof.

   A bisimulation on $\mathbb{Z}^\omega$ is a relation $R \subseteq \mathbb{Z}^\omega \times \mathbb{Z}^\omega$ for which

$$s \ R\ s' \ \Rightarrow \ [\ \texttt{headS}\ (s) = \ \texttt{headS}\ (s')] \wedge [\ \texttt{tailS}\ (s)\ R\ \ \texttt{tailS}\ (s')].$$

   By the coinduction principle, we need to find a bisimulation relating `nat` and `shuffle even odd`.

   We show that

$$R \quad = \quad \big\{\big(\ \texttt{jump}\ k\ 1\ ,\ \texttt{shuffle}\,(\texttt{jump}\ k\ 2)\,(\texttt{jump}\ (k+1)\ 2)\ \big) \mid k \in \mathbb{N}\big\}$$

   is a bisimulation.

   Let $k \in \mathbb{N}$. First observe that,

$$\texttt{headS}\,(\texttt{jump}\ k\ 1) \ = \ k \ = \ \texttt{headS}\,\big(\texttt{shuffle}\,(\texttt{jump}\ k\ 2)\,(\texttt{jump}\ (k+1)\ 2)\big).$$

   Second,

$$\texttt{tailS}\,(\texttt{jump}\ k\ 1) \quad R \quad \texttt{tailS}\,\big(\texttt{shuffle}\,(\texttt{jump}\ k\ 2)\,(\texttt{jump}\ (k+1)\ 2)\big)$$

   because

$$\texttt{tailS}\,(\texttt{jump}\ k\ 1) \quad = \quad \texttt{jump}\ (k+1)\ 1$$

   and

$$\begin{aligned} &\texttt{tailS}\,\big(\texttt{shuffle}\,(\texttt{jump}\ k\ 2)\,(\texttt{jump}\ (k+1)\ 2)\big) \\ &\qquad = \ \texttt{shuffle}\,(\texttt{jump}\ (k+1)\ 2)\,\big(\texttt{tailS}\,(\texttt{jump}\ k\ 2)\big) \\ &\qquad = \ \texttt{shuffle}\,(\texttt{jump}\ (k+1)\ 2)\,(\texttt{jump}\ (k+2)\ 2). \end{aligned}$$

   Hence we are done.

The above inductive proof does not only involve more calculations than the coinductive one but, more fundamentally, appeals to the extensionality principle (η). However, how is extensionality justified? One possibility, is to represent $\mathbb{Z}^\omega$ as $\mathbb{N} \Rightarrow \mathbb{Z}$ (the set of functions from $\mathbb{N}$ to $\mathbb{Z}$) with structure map given by $h(s) = s(0)$ and $t(s) = \lambda n \in \mathbb{N}.\ s(n+1)$, and invoke extensionality for them. Another justification, without committing the interpretation of `IntStream` to any particular representation, follows from the coinduction principle: assuming the antecedent of (η), the relation

$$\big\{ \big(\, \mathtt{tailS}^{\,n}(s),\ \mathtt{tailS}^{\,n}(s')\big) \mid n \in \mathbb{N} \big\}$$

is a bisimulation (as the reader should check) and therefore $s = s'$.

Thus, for some data types coinduction seems to be a more appealing proof principle than induction. This is even more noticeable when reasoning about infinite trees, where the extensionality principle is not as neat as the one for streams.

Further applications of coinduction in computer science can be found in relational semantics [MT88].

To model recursive data types we adopt a non-standard approach recently revived in the work of Peter Freyd [Fre91, Fre92]. The meaning of a recursive data type will be a final coalgebra. We recall the basic definitions. Let $\mathcal{C}$ be a category (which we think of as the universe of denotations for the data types) and let F be an endofunctor on it (which we think of as the denotation of a data type constructor with a free type variable). An F-*coalgebra* (or simply, coalgebra when F is clear from the context) is a pair (T, τ) consisting of an object $T \in |\, \mathcal{C}\, |$ together with a structure map $\tau : T \to FT$ in $\mathcal{C}$; a coalgebra *homomorphism* $h : (T, \tau) \to (U, \mu)$ is a morphism $h : T \to U$ such that

$$\begin{tikzcd} T \arrow[r, "\tau"] \arrow[d, "h"'] & FT \arrow[d, "Fh"] \\ U \arrow[r, "\mu"'] & FU \end{tikzcd}$$

commutes. Moreover, a coalgebra (U, μ) is said to be *final* if for every coalgebra (T, τ) there exists a unique homomorphism $(T, \tau) \to (U, \mu)$; i.e. if it is terminal in the category of coalgebras and homomorphims.

Observe again that, in this view, `IntStream` has been interpreted as the final coalgebra for the endofunctor $\mathbb{Z} \times \_$ on **Set**. Other examples of data types naturally arising as coalgebras can be found in [AM82].

It has been customary to interpret recursive types as initial algebras (the notion dual to that of final coalgebra). We remark that in certain categories of domains this traditional view and the viewpoint adopted here coincide for initial algebras and final coalgebras are canonically isomorphic (see e.g. [Fre90, Smy91, Fre92, FP92, Fio94]).

The mysterious definition of bisimulation in our example is an instance of an abstract notion of bisimulation on a coalgebra (taken from [AM89]) motivated by concurrency theory. This has two important methodological consequences.

1. We are able to provide bisimulations for recursive data types by characterising the abstract notion for concrete examples; this is done for (finite and/or infinite) trees in Section 1, and for natural numbers, Cantor space and languages in Section 4.

2. We are able to test whether a relation satisfying certain closure properties can be regarded as a bisimulation; this is exemplified in the proof of the internal full abstraction result in Section 8.

A central result of the paper (Principle 6.9) is an abstract coinduction principle (along the lines of the one presented above) for proving that one element of a recursively defined domain approximates another. We begin in Section 1 presenting our theory in the light of an example. This section aims at a general audience and is mainly intended to serve as a guide for the subsequent abstract development. Sections 2–4 assume some familiarity with basic category theory (as, for example, in [Pie91, LS94]). In Section 2 an abstract definition of bisimulation is introduced. Examples of bisimulation are presented by characterising the abstract notion for various data type constructors. Section 3 concentrates on final semantics (i.e. the interpretation induced by the unique homomorphism into the final coalgebra) and analyses its relation with bisimulation. In Section 4 we state the strong-extensionality theorem from which the proof principle is derived. Sections 5–7 rework the theory to adapt it to recursive domain equations; generalising the previous treatment to the setting of order-enriched categories. In particular, ordered categorical bisimulations are defined and an order strong-extensionality theorem for them is proved. In Section 8 we apply the theory to obtain an internal full abstraction result for the canonical model of the untyped call-by-value λ-calculus. This section is technically involved; in particular the technique used to solve recursive type equations of mixed-variance functors, namely via algebraic compactness, is merely sketched (for a thorough treatment consult [Fio94, Chapters 6 and 7]). Finally, in Section 9 we indicate plans for future work.

There are some relationships between this and the work of [RT93] and [Pit94, Pit93].

In [RT93], Rutten and Turi study final semantics and strong extensionality for the categories of non-well-founded sets, complete metric spaces and ω-complete pointed partial orders (cppos). In the case of cppos, they define ordered bisimulations and prove an order strong-extensionality theorem for them. Our treatment generalises theirs in two respects: first, we work with order-enriched categories (though we remark that their approach could easily be accommodated in this setting); second, as Rutten pointed out, every ordered bisimulation in the sense of [RT93] is an ordered categorical bisimulation in our sense (Definition 5.1) whilst the converse does not hold (Example 5.4 gives a counterexample).

In [Pit94], Pitts considered domains recursively defined using the cartesian product, disjoint union, partial function space, and convex powerdomain constructors; introduced a notion of *simulation* over these domains; and used it to characterise their approximation order. In this way he provided a *coinductive* proof principle for proving that an element of these recursively defined domains approximates another; viz. by exhibiting a simulation relating the two elements. Later, in [Pit93], he generalised the previous result by establishing a mixed induction/coinduction property of relations on a recursive domain. This he achieved by working with a category of domains equipped with a *relational structure* (a general framework for relations on domains) and by drawing upon the simultaneous initiality/finality property of recursive domains. In this context the notion of (bi)simulation is induced by an extension of the data type endofunctor to the relational structure, which is subject to conditions but is not prescribed by the data type. Our approach differs from his in a couple of points. First, we do not just concentrate on categories of domains but rather study the coinduction principle for recursive data types in arbitrary categories (to encompass domains we develop an *order-enriched* theory). Second, as we follow [AM89], we consider bisimulations on recursive data types as relations (within the category of data types) with closure properties solely determined by the action of the data type constructor.

Recently, in [HJ95], Hermida and Jacobs have provided an abstract analysis of induction and coinduction in the spirit of categorical logic. Roughly, in a rich setting given by a category of data types equipped with a category of predicates/relations over it (i.e. a *fibration* with structure) they express induction/coinduction principles categorically as preservation of initial-algebras/final-coalgebras. This work has unified aspects of our approach and that of [Pit93] in that the action of the type constructors on predicates/relations is determined by the endofunctor defining the data type (as we do here), whereas their setting of predicates/relations in a fibration refines Pitts' relational structures.

1 A coinduction principle based on bisimulation: an example

An instance of our main theorem is presented in the view of an example. For this purpose we consider (one-sorted) signatures. A signature Ω is an $\mathbb{N}$-indexed set $\langle \Omega_n \rangle_{n\in\mathbb{N}}$ of (disjoint) operator symbols where the intended meaning of $o \in \Omega_n$ is that o is an operator symbol of arity n.

Every signature Ω induces the endofunctor $F_\Omega(\_) = \coprod_{n\in\mathbb{N}} \Omega_n \times (\_)^n$ on **Set** where $\coprod$ and $\times$ are respectively the coproduct and product functors. In passing, notice that F_Ω-coalgebras (T,τ) can be regarded as *guarded substitutions* where every $x \in T$ (thought of as a variable) is assign to the term $o(x_1,\ldots,x_n)$ whenever $\tau(x) = \big(o,(x_1,\ldots,x_n)\big) \in \Omega_n \times T^n$.

We introduce the notion of bisimulation on F_Ω-coalgebras:

Definition 1.1 A relation $R \subseteq T\times T$ is a *categorical F_Ω-bisimulation* on the F_Ω-coalgebra (T,τ) if there exists an F_Ω-coalgebra structure ρ on R such that

$$\begin{tikzcd} R \arrow[r, "\pi_1"] \arrow[d, "\rho"'] & T \arrow[d, "\tau"] & R \arrow[l, "\pi_2"'] \arrow[d, "\rho"] \\ F_\Omega R \arrow[r, "F_\Omega \pi_1"'] & F_\Omega T & F_\Omega R \arrow[l, "F_\Omega \pi_2"] \end{tikzcd}$$

commutes $\big($i.e. such that the projections $\pi_1, \pi_2 : R \to T$ become homomorphisms $(R,\rho) \to (T,\tau)\big)$. $\square$

In order to describe categorical bisimulations in more elementary terms we need some operations induced by a coalgebra $\tau : T \to F_\Omega T$. For $t \in T$ with $\tau(t) = \big(o, (t_1, \ldots, t_n)\big) \in \Omega_n \times T^n$, we set

- $\mathbf{root}_\tau(t) \stackrel{def}{=} o$; and
- for $1 \leq i \leq n$, $\mathbf{subtree}_\tau(t, i) \stackrel{def}{=} t_i$.

Proposition 1.2 A relation $R \subseteq T \times T$ is a categorical bisimulation on $\tau : T \to F_\Omega T$ if and only if $t \; R \; t'$ implies

$$[\mathbf{root}_\tau(t) = \mathbf{root}_\tau(t') \in \Omega_n] \;\wedge\; [\forall\, 1 \leq i \leq n.\; \mathbf{subtree}_\tau(t,i) \; R \; \mathbf{subtree}_\tau(t',i)].$$

Proof:

($\Rightarrow$) Let $\rho : R \to F_\Omega R$ be such that, for $i = 1, 2$, $\tau \circ \pi_i = (F_\Omega \pi_i) \circ \rho$.

Assume $t \; R \; t'$ and let $\rho(t, t') = \Big(r, \big((s_1, s'_1), \ldots, (s_n, s'_n)\big)\Big)$. Since,

$$\tau t = (F_\Omega \pi_1)\big(\rho(t,t')\big) = \big(r, (s_1, \ldots, s_n)\big) \quad \text{and} \quad \tau t' = (F_\Omega \pi_2)\big(\rho(t,t')\big) = \big(r, (s'_1, \ldots, s'_n)\big),$$

it follows that $\mathbf{root}_\tau(t) = r = \mathbf{root}_\tau(t')$ and $\mathbf{subtree}_\tau(t,i) = s_i \; R \; s'_i = \mathbf{subtree}_\tau(t',i)$ for $1 \leq i \leq n$.

($\Leftarrow$) For $t, t' \in T$, let $\rho(t,t') = \Big(\mathbf{root}_\tau(t), \big\langle\big(\mathbf{subtree}_\tau(t,i), \mathbf{subtree}_\tau(t',i)\big)\big\rangle_{1 \leq i \leq n}\Big)$.

The mapping ρ makes R into an F_Ω-coalgebra (i.e. a map $R \to \coprod_{n \in \mathbb{N}} \Omega_n \times R^n$) because for every $t \; R \; t'$, $\mathbf{subtree}_\tau(t,i) \; R \; \mathbf{subtree}_\tau(t',i)$.

Moreover, by construction, $\tau \circ \pi_1 = (F_\Omega \pi_1) \circ \rho$; whilst $\tau \circ \pi_2 = (F_\Omega \pi_2) \circ \rho$ because for every $t \; R \; t'$, $\mathbf{root}_\tau(t) = \mathbf{root}_\tau(t')$. □

Corollary 1.3 For every F_Ω-coalgebra, the equality relation is a categorical F_Ω-bisimulation. □

The coalgebras for which equality reduces to bisimulation are identified:

Definition 1.4 (c.f. [Acz88]) A coalgebra is *strongly extensional* if the equality relation is maximal among all its categorical bisimulations. □

Theorem 1.5 Final F_Ω-coalgebras are strongly extensional.

Proof: By Corollary 1.3, the equality relation is a categorical bisimulation. Moreover, if $\mu : U \to F_\Omega U$ is a final coalgebra and $R \subseteq U \times U$ is a categorical bisimulation then there exists a coalgebra $\rho : R \to F_\Omega R$ such that π_1 and π_2 are homomorphisms $(R, \rho) \to (U, \mu)$. By the universal property of final coalgebras, it follows that $\pi_1 = \pi_2$, and hence $u \; R \; u'$ implies $u = u'$. □

As a corollary the following proof principle holds:

Principle 1.6 If R is a categorical F_Ω-bisimulation on the final coalgebra U, and u and u' are elements of U for which $u \; R \; u'$, then $u = u'$. □

The final F_Ω-coalgebra can be described as the set of finite and/or infinite trees generated by the signature Ω. Explicitly, an Ω-*tree* (see [Cou83, AM82]) is a partial function $t : {\mathbb{N}_{>0}}^* \rightharpoonup \coprod_{n\in\mathbb{N}} \Omega_n$ such that its domain of definition, $\mathrm{dom}(t)$, satisfies the following conditions:

1. $\mathrm{dom}(t)$ is non-empty and *prefix-closed* $\big($i.e. $\alpha i \in \mathrm{dom}(t) \;\Rightarrow\; \alpha \in \mathrm{dom}(t)\big)$;
2. for $\alpha \in \mathrm{dom}(t)$, if $t(\alpha) \in \Omega_n$ then, for $i \in \mathbb{N}$, $\alpha i \in \mathrm{dom}(t) \Leftrightarrow 1 \leq i \leq n$.

And, the final F_Ω-coalgebra is the set of Ω-*trees*, T^ω_Ω, with structure map

$$T^\omega_\Omega \to \coprod_{n\in\mathbb{N}} \Omega_n \times (T^\omega_\Omega)^n : t \mapsto \big(t(\epsilon), \big\langle \lambda\alpha \in {\mathbb{N}_{>0}}^*.\, t(i\alpha)\big\rangle_{1\leq i\leq n}\big).$$

Hence, the coinduction principle reads:

Coinduction principle for finite and/or infinite trees:

Suppose that R is a binary relation on T^ω_Ω satisfying

$$t \; R \; t' \;\Rightarrow\; [\mathbf{root}(t) = \mathbf{root}(t') \in \Omega_n] \;\wedge\; [\forall\, 1 \leq i \leq n.\, \mathbf{subtree}(t, i) \; R \; \mathbf{subtree}(t', i)]$$

Then, $t \; R \; t' \Rightarrow t = t'$.

2 Categorical bisimulation

Interleaving models of concurrent computation are generally based on transition systems. A transition system is a triple $(A, S, \to)$ where A is a set of actions, S is a set of states, and $\to \subseteq S \times A \times S$ is the transition relation. In general, we write $p \xrightarrow{a} q$ for $(p, a, q) \in \to$ and read it as "after performing the action a, the state p evolves to the state q". Transition systems with a set of actions A can equivalently be defined as coalgebras for the endofunctor $\mathcal{P}(A \times \_)$ on **Set** (where $\mathcal{P}$ denotes the powerset endofunctor), since the map sending a transition system $(A, S, \to)$ to the coalgebra $\sigma : p \mapsto \{(a, q) \mid p \xrightarrow{a} q\}$ establishes a bijection.

The notion of bisimulation is introduced in order to identify observationally equivalent states. Here we are only concerned with what is known as *strong* bisimulation. A relation $R \subseteq S \times S$ is called a bisimulation if $p \; R \; q$ implies that

- for every $p \xrightarrow{a} p'$, there exists a state q' such that $q \xrightarrow{a} q'$ and $p' \; R \; q'$, and
- for every $q \xrightarrow{a} q'$, there exists a state p' such that $p \xrightarrow{a} p'$ and $p' \; R \; q'$.

Any relation $R \subseteq S \times S$ can be given a coalgebra structure $\rho : R \to \mathcal{P}(A \times R)$ mapping $(p, q) \mapsto \big\{\big(a, (p', q')\big) \mid p \xrightarrow{a} p',\ q \xrightarrow{a} q',\ p' \; R \; q'\big\}$. Moreover, if R is a bisimulation then the projections from R to S are coalgebra homomorphisms from (R, ρ) to (S, σ). Conversely, every relation with a coalgebra structure making the projections into coalgebra homomorphisms is a bisimulation. This fact motivates the definition of bisimulation below.

Remark. Unless otherwise stated we assume our category of discourse to be a *partial cartesian category of partial maps*. This concept is defined in Appendix A; but, on a first reading, the reader may skip such details as long as he keeps in mind that this includes all cartesian categories (i.e. categories with finite products) and the *usual* categories of sets with structure and partial functions equipped with finite cartesian products.

Convention. Monos (i.e. morphisms $m : A \to B$ such that for every $x, y : X \to A$, $m \circ x = m \circ y$ implies $x = y$) are denoted with $\rightarrowtail$.

Definition 2.1 A relation $r : R \rightarrowtail S \times S$ is a *categorical bisimulation* on the coalgebra (S, σ) if there exists a coalgebra structure ρ on R such that $\pi_i \circ r : (R, \rho) \to (S, \sigma)$ for $i = 1, 2$. □

The above definition generalises that of [AM89] in that we consider endofunctors over arbitrary (partial) cartesian categories (of partial maps). This will allow us to discuss examples and establish theorems in wide generality. Concerning examples of bisimulations we have:

Convention. **Pfn** denotes the category of small sets and partial functions, and **1** and 2 denote canonical one and two-element sets.

Example 2.2 (Type constructor for natural numbers, $\mathbf{1} + \_ : \mathbf{Pfn} \to \mathbf{Pfn}$)
$R \subseteq N \times N$ is a categorical bisimulation on $p : N \rightharpoonup \mathbf{1} + N$ if and only if $n \; R \; n'$ implies

- $p(n)$ and $p(n')$ are both undefined, or
- $p(n)$ and $p(n')$ are both defined, and either $p(n) \in \mathbf{1} \ni p(n')$, or $p(n) \in N \ni p(n')$ and $p(n) \; R \; p(n')$. □

Example 2.3 (Type constructor for cantor space, $\mathbf{2} \times \_ : \mathbf{Set} \to \mathbf{Set}$)
$R \subseteq S \times S$ is a categorical bisimulation on $\langle h, t \rangle : S \to 2 \times S$ if and only if $s \; R \; s'$ implies $h(s) = h(s')$ and $t(s) \; R \; t(s')$. □

More generally, for a signature $\Omega = \langle \Omega_n \rangle_{n \in \mathbb{N}}$ (see Section 1) we can consider Ω-coalgebras and partial Ω-coalgebras:

Example 2.4 (Type constructor for infinite trees, $F_\Omega(\_) : \mathbf{Set} \to \mathbf{Set}$)
$R \subseteq T \times T$ is a categorical bisimulation on $\tau : T \to F_\Omega T$ if and only if $t \; R \; t'$ implies

$$\tau t = (r, s) \in \Omega_n \times T^n \ni (r', s') = \tau t'$$

with $r = r'$ and $s_i \; R \; s'_i$ for every $1 \leq i \leq n$. □

Example 2.5 (Type constructor for finite trees, $F_\Omega(\_) : \mathbf{Pfn} \to \mathbf{Pfn}$)
$R \subseteq T \times T$ is a categorical bisimulation on $\tau : T \rightharpoonup F_\Omega T$ if and only if $t \; R \; t'$ implies

- τt and $\tau t'$ are both undefined, or

- τt and $\tau t'$ are both defined, and

$$\tau t = (r, s) \in \Omega_n \times T^n \ni (r', s') = \tau t'$$

with $r = r'$ and $s_i \; R \; s'_i$ for every $1 \leq i \leq n$. $\square$

Example 2.6 (Type constructor for deterministic automata, $\mathbf{2} \times (A \Rightarrow \_) :$ Set $\rightarrow$ Set)
A deterministic automaton [HU79] consist of a set of states and a set of transitions from state to state that occur on input symbols chosen from an alphabet. For each input symbol there is exactly one transition out of each state. Some states are designated as accepting states.

Deterministic automata with input alphabet A can be formally described as coalgebras for the functor $2 \times (A \Rightarrow \_) : \mathbf{Set} \rightarrow \mathbf{Set}$. For example, $\langle \alpha, \delta \rangle : S \rightarrow 2 \times (A \Rightarrow S)$ describes the automaton with set of states S, accepting predicate α and transition function δ.

$R \subseteq S \times S$ is a categorical bisimulation on $\langle \alpha, \delta \rangle$ if and only if $p \; R \; q$ implies $\alpha(p) = \alpha(q)$ and $\delta(p)(a) \; R \; \delta(q)(a)$ for every $a \in A$. $\square$

3 Final semantics

As mentioned in the introduction, recursive data types can be modelled by final coalgebras. This automatically provides a final semantics since every coalgebra can be canonically interpreted in the final one via the unique homomorphism (denoted as $[\![\_]\!]$). As a step towards understanding the scope of the abstract notion of bisimulation (Definition 2.1) its relation with final semantics is studied.

Remark. The proofs of Propositions 3.1 and 3.3 below are omitted as they are, respectively, instances of Propositions 7.1 and 7.3.

Two elements are said to be bisimilar whenever there exists a bisimulation relating them.

Proposition 3.1 Bisimilar elements have the same final semantics. $\square$

For a deterministic automaton, the final semantics corresponds to the language accepted by the automaton. The final coalgebra is

$$\begin{array}{rcl} \mathcal{P}(A^*) & \rightarrow & 2 \times (A \Rightarrow \mathcal{P}(A^*)) \\ L & \mapsto & (\epsilon \in L, \lambda a.\{w \mid a.w \in L\}) \end{array}$$

and, for $\langle \alpha, \delta \rangle : S \rightarrow 2 \times (A \Rightarrow S)$ and $p \in S$,

- $\epsilon \in [\![p]\!] \Leftrightarrow \alpha(p)$, and
- $a.w \in [\![p]\!] \Leftrightarrow w \in [\![\delta(p)(a)]\!]$.

Then, applying Proposition 3.1, we conclude that bisimilar states in a deterministic automaton accept the same language. As is well-known the converse also holds. In fact, since $2 \times (A \Rightarrow \_) : \mathbf{Set} \to \mathbf{Set}$ preserves pullbacks, it follows from very general considerations:

Definition 3.2 The pair $r_1, r_2 : R \to S$ is a *weak kernel pair* (*kernel pair*) of $f : S \to U$ if $f \circ r_1 = f \circ r_2$ and, for every $p, q : T \to S$, whenever $f \circ p = f \circ q$ there exists (a unique) r such that $r_1 \circ r = p$ and $r_2 \circ r = q$. □

Proposition 3.3 Consider a category with kernel pairs and an endofunctor on it mapping kernel pairs to weak kernel pairs. Then, for every coalgebra, elements with the same final semantics are bisimilar. □

In general, the converse of Proposition 3.1 does not hold. To provide a counterexample, consider a simple kind of machine whose states have one of the following three capabilities: they are in deadlock (i.e. no capabilities), they can evolve to another state, or they can produce an observation. These machines can be represented as coalgebras $\sigma : S \rightharpoonup \mathbf{1} + S$ where S is the set of states, and for $p \in S$, we have that $\sigma(p)$ is undefined if and only if p is in deadlock, p evolves to q if and only if $\sigma(p) = q$ and $\sigma(p) \in \mathbf{1}$ if and only if p produces an observation. The final machine is $\mathbf{pred} : \mathbb{N} \to \mathbf{1} + \mathbb{N}$ and, for every element $p \in S$, the final semantics is $[\![p]\!] = \mu n[\sigma^{n+1}(p) \in \mathbf{1}]$ where μ is the minimisation operator of recursion theory. For the machine whose set of states is $\{\mathbf{nil}, \mathbf{loop}\}$ where $\mathbf{nil}$ is in deadlock and $\mathbf{loop}$ evolves to itself we have that $[\![\mathbf{nil}]\!] = [\![\mathbf{loop}]\!]$. But, by the characterisation of bisimulations given in Example 2.2, **nil** and **loop** cannot be bisimilar.

This should be interpreted as the inability of the final semantics to discriminate between some different operational behaviours. This happens because the final semantics is very abstract. As will be shown, the final coalgebra does not have distinct bisimilar elements.

4 Strong-extensionality theorem

The theorem justifying the coinduction principle is stated. We start by internalising the notion of equality.

Definition 4.1 The *equality relation* $e_S : =_S \rightarrowtail S \times S$ is defined as the equaliser of $\pi_1, \pi_2 : S \times S \to S$. □

Remark. The equality relation could have been defined to be $\langle \mathrm{id}_S, \mathrm{id}_S \rangle : S \rightarrowtail S \times S$. However, the previous definition was chosen for two reasons: it generalises to categories of partial maps and it provides the intuition for the generalisation we carry out in Section 6.

Proposition 4.2 For every coalgebra, the equality relation is a categorical bisimulation. □

Theorem 4.3 (Strong extensionality) Final coalgebras are strongly extensional (according to Definition 1.4). □

Remark. The proofs of Proposition 4.2 and Theorem 4.3 are omitted as they are instances of Proposition 6.3 and Theorem 6.8.

Convention. Given monos $m : D \rightarrowtail A$ and $n : E \rightarrowtail A$, we write $m \subseteq n$ (or even $D \subseteq E$) whenever m factors through n.

A map $p : 1 \rightarrowtail S$ is said to be an element of S. Given a relation $r : R \rightarrowtail S \times S$ and elements $p, q : 1 \rightarrowtail S$ we write $p\ r\ q$ (or even $p\ R\ q$) whenever $\langle p, q\rangle \subseteq r$.

As a corollary the following proof principle holds:

Principle 4.4 If R is a categorical bisimulation on the final coalgebra U, and u and u' are elements of U for which $u\ R\ u'$, then $u = u'$. □

This abstract formulation instantiated to the examples of Section 2 yields the following coinduction principles:

Finite and/or infinite trees:

Recalling that for a signature Ω, the final coalgebra of $F_\Omega(\_) : \mathbf{Set} \to \mathbf{Set}$ is T_Ω^ω, the set of (finite and infinite) Ω-*trees* and that the final coalgebra of $F_\Omega(\_) : \mathbf{Pfn} \to \mathbf{Pfn}$ is T_Ω, the Ω-*word algebra*; the principle reads:

Suppose that R is a binary relation on T_Ω or T_Ω^ω satisfying

$$t\ R\ t' \ \Rightarrow\ [\mathbf{root}(t) = \mathbf{root}(t') \in \Omega_n] \ \wedge\ [\forall\, 1 \leq i \leq n.\ \mathbf{subtree}(t,i)\ R\ \mathbf{subtree}(t',i)]$$

Then, $t\ R\ t' \Rightarrow t = t'$.

Taking

$$\Omega = \langle \Omega_0 = \{\mathbf{0}\}, \Omega_1 = \{\mathbf{succ}\}, \Omega_n = \emptyset \text{ for } n \geq 2\rangle \text{ in } \mathbf{Pfn}$$

and

$$\Omega = \langle \Omega_0 = \emptyset, \Omega_1 = 2, \Omega_n = \emptyset \text{ for } n \geq 2\rangle \text{ in } \mathbf{Set}$$

we obtain the principles provided in [Smy91] (though derived there in a more indirect fashion) for $\mathbb{N}$ and 2^ω.

Natural numbers:

Suppose that R is a binary relation on $\mathbb{N}$ satisfying

$$n\ R\ n' \ \Rightarrow\ [n = \mathbf{0} = n'] \ \vee\ [\mathbf{pred}(n)\ R\ \mathbf{pred}(n')]$$

Then, $n\ R\ n' \Rightarrow n = n'$.

Cantor space:

Suppose that R is a binary relation on 2^ω such that

$$s \; R \; s' \;\Rightarrow\; [\mathbf{head}(s) = \mathbf{head}(s')] \;\wedge\; [\mathbf{tail}(s) \; R \; \mathbf{tail}(s')]$$

Then, $s \; R \; s' \Rightarrow s = s'$.

The following coinduction principle seems to be new (though its usefulness is still to be proved).

Languages:

Suppose that R is a binary relation on $\mathcal{P}(A^*)$ for which

$$L \; R \; L' \;\Rightarrow\; [\epsilon \in L \Leftrightarrow \epsilon \in L'] \;\wedge\; [\forall \, a \in A. \; \{w \mid a.w \in L\} \; R \; \{w \mid a.w \in L'\}]$$

Then, $L \; R \; L' \Rightarrow L = L'$.

So far we have demonstrated that coinduction principles are not exclusive to a particular category of data types. Next, the approach is extended to categories of domains as solutions to recursive data types are more frequently encountered in this setting.

5 Ordered categorical bisimulation

In the vein of [SP82, Fre90, FP92, Fio94] our analysis is not committed to a particular category of domains (i.e. certain kind of partial orders) but based on the structure such a category should have. In this respect, (order-)enriched category theory is fundamental and so it is adopted as the main mathematical tool.

To keep the paper self contained, the enriched notions used throughout are provided. (For a thorough treatment of enriched category theory consult [Kel82].)

Let **Poset** (**Cpo**) be the category of small posets (cpos —posets, possibly without a least element, closed under lubs of ω-chains) and monotone (continuous) functions.

A **Poset**-*category* (**Cpo**-*category*) $\mathcal{C}$ is a category whose hom-sets $\mathcal{C}(A, B)$ come equipped with a partial order (ω-complete partial order) $\leq_{A,B}$ with respect to which composition of morphisms $\_ \circ = : \mathcal{C}(B, C) \times \mathcal{C}(A, B) \to \mathcal{C}(A, C)$ is a monotone (continuous) operation.

The immediate examples of **Poset**-categories are **Poset** and **Cpo** where each hom-set is ordered pointwise; even more, **Cpo** is a **Cpo**-category. More examples are obtained from the observation that every category *freely* induces a **Poset**-category (**Cpo**-category) by discretely ordering each hom-set.

For **Poset**-categories (**Cpo**-categories) $\mathcal{A}$ and $\mathcal{B}$, a **Poset**-*functor* (**Cpo**-*functor*) $F : \mathcal{A} \to \mathcal{B}$ is a functorial mapping such that for every $A, A' \in |\,\mathcal{A}\,|$ the assignment $F_{A,A'} : \mathcal{A}(A, A') \to \mathcal{B}(FA, FA')$ is monotone (continuous).

Categorical bisimulation is adapted to the order-enriched setting:

Definition 5.1 In a **Poset**-category, a relation $r : R \rightarrowtail S \times S$ is called an *ordered categorical bisimulation* on the F-coalgebra (S, σ) if there exists a coalgebra structure ρ on R such that

$$\begin{tikzcd} R \arrow[r, "r_1"] \arrow[d, "\rho"'] \arrow[dr, phantom, "\geq"] & S \arrow[d, "\sigma"] & R \arrow[l, "r_2"'] \arrow[d, "\rho"] \\ FR \arrow[r, "Fr_1"'] & FS & FR \arrow[l, "Fr_2"] \end{tikzcd}$$

where $r_i \stackrel{def}{=} \pi_i \circ r$ $(i = 1, 2)$. □

Remark. When considering (partial) products in **Poset**-categories, the pairing of morphisms is assumed to be monotone.

Convention. A morphism $f : A \to B$ is called a *lax-homomorphism* from the F-coalgebra (A, α) to the F-coalgebra (B, β) whenever $Ff \circ \alpha \geq \beta \circ f$. (Thus, r_1 in the definition above is a lax-homomorphism from (R, ρ) to (S, σ).)

For examples, consider:

Example 5.2 ($\mathbf{1} + \_ : \mathbf{Pfn} \to \mathbf{Pfn}$)
$R \subseteq N \times N$ is an ordered categorical bisimulation on $p : N \rightharpoonup \mathbf{1} + N$ if and only if $n\ R\ m$ implies

- if $p(n)$ is defined then so is $p(m)$ and either $p(n) \in \mathbf{1} \ni p(m)$, or $p(n) \in N \ni p(m)$ and $p(n)\ R\ p(m)$, and
- if $p(n)$ is undefined then, whenever $p(m) \in N$, there exists k such that $k\ R\ p(m)$.□

Example 5.3 ($\mathcal{P}_A(\_) : \mathbf{SFP}_\perp \to \mathbf{SFP}_\perp$, c.f. [Pit94, Section 5])
For a set A, let $\mathcal{P}_A(\_)$ be the functor $\{\emptyset\}_\perp \oplus \mathcal{P}^\natural(A_\perp \otimes \_)$, where $(\_)_\perp$, $\oplus$, $\otimes$ and $\mathcal{P}^\natural(\_)$ are the lifting, coalesced sum, smash product and convex powerdomain [Plo76] functors.

Then, for every set S, the (strict) coalgebras $S_\perp \multimap \mathcal{P}_A(S_\perp)$ can be viewed as those *transition systems with divergence* [Abr91] $(S, \to \subseteq S \times A \times S, \uparrow \subseteq S)$ such that, for every $p \in S$, if $\{(a, q) \mid p \xrightarrow{a} q\}$ is infinite then $p \in \uparrow$. This is because the map sending $(S, \to, \uparrow)$ to

$$\sigma : p \mapsto \begin{cases} \perp & \text{if } p = \perp \\ \{(a, q) \mid p \xrightarrow{a} q\} \cup \{\perp \mid p \in \uparrow\} & \text{if } p \neq \perp \end{cases}$$

establishes a bijective correspondence.

For $R \subseteq S \times S$, if $R_\perp \rightarrowtail S_\perp \otimes S_\perp$ is an ordered categorical bisimulation on σ then R is a *partial bisimulation* [Abr91] on $(S, \to, \uparrow)$; that is, $p\ R\ q$ implies

$$\begin{array}{ll} & \forall\, p \xrightarrow{a} p'.\, \exists\, q'.\, q \xrightarrow{a} q' \ \wedge\ p'\ R\ q' \\ \wedge & \\ & p \downarrow \Rightarrow\ q \downarrow\ \wedge \forall\, q \xrightarrow{a} q'.\, \exists\, p'.\, p \xrightarrow{a} p' \ \wedge\ p'\ R\ q', \end{array}$$

where $p \downarrow \Leftrightarrow p \notin \uparrow$. □

Ordered categorical bisimulations resemble the notion of 'bisimulation up to' [Mil89]:

Example 5.4 Let $\omega^\top$ be ω with a top element and let $(\_)_\perp : \mathbf{Cpo} \to \mathbf{Cpo}$ be the lifting functor.

$R \subseteq \omega^\top \times \omega^\top$ (viewed as a discrete cpo) is an ordered categorical bisimulation on the coalgebra $\mathbf{pred} : \omega^\top \to (\omega^\top)_\perp$ if and only if $n\ R\ m$ implies $n = 0 = m$, or both $m > 0$ and there exists k for which $\mathbf{pred}(n) \leq k\ R\ \mathbf{pred}(m)$. □

6 Order strong-extensionality theorem

An order-enriched version of the strong-extensionality theorem is obtained. To internalise the inequality, the 2-categorical notion of inserter is used in the context of order-enriched categories.

Definition 6.1 In a **Poset**-category, $x : X \to A$ is an *inserter* of the ordered pair $(f : A \to B, g : A \to B)$ if $f \circ x \leq g \circ x$ and for every $y : Y \to A$ such that $f \circ y \leq g \circ y$, there exists a unique u for which $y = x \circ u$. □

Inserters behave like equalisers where equality has been replaced by inequality. For example, essentially the same proof that shows equalisers to be monos shows inserters to be monos as well.

For a cpo D, define $\leq_D$ to be the subcpo of $D \times D$ with underlying set $\{(x, y) \mid x \leq y\}$. In **Cpo** and in **pCpo**, the category of cpos and partial continuous functions [Plo85] (i.e. partial functions whose domain of definition is Scott-open and such that its restriction to the domain of definition is continuous), $\leq_D \rightarrowtail D \times D$ is the inserter of (π_1, π_2). This motivates the abstract definition of inequality:

Definition 6.2 In a **Poset**-category, the *inequality relation* $l_S : \leq_S \rightarrowtail S \times S$ is defined as the inserter of (π_1, π_2). □

Proposition 6.3 For every coalgebra, the inequality relation is an ordered categorical bisimulation.

PROOF: For a coalgebra (S, σ), let $\delta : S \rightarrowtail \leq_S$ be the unique morphism, given by the universal property of inserters, for which $l_S \circ \delta = \langle \mathrm{id}_S, \mathrm{id}_S \rangle$; and give $\leq_S$ the following coalgebra structure, $F\delta \circ \sigma \circ \pi_2 \circ l_S$. □

Definition 6.4 A coalgebra is *order strongly extensional* if the inequality relation is maximal among all its ordered categorical bisimulations. □

In the **Poset**-enriched setting, final coalgebras need not be order strongly extensional. Thus, a stronger notion is needed:

Definition 6.5 In a **Poset**-category, a coalgebra is *maximally-final* if it is final and the unique homomorphism from any other coalgebra is maximal among all lax-homomorphisms. □

Proposition 6.6 Maximally-final coalgebras are order strongly extensional.

PROOF: If r is a categorical ordered bisimulation on a maximally-final coalgebra U, then $r_1 \leq r_2$. And, by the universal property of inserters, it follows that $r \subseteq l_U$. $\square$

The above proposition is useful only if there are simple criteria to recognise maximally-final coalgebras. In particular, the best one can hope for is that final coalgebras of **Poset**-endofunctors are automatically maximally-final. Unfortunately, this is not the case. For example, let $\mathcal{Z}$ be the **Poset**-category induced by the ordered monoid $(\mathbb{Z}, 0, +, \leq)$. The **Poset**-endofunctor $2\_ : \mathcal{Z} \to \mathcal{Z}$ that multiplies morphisms by 2, has 0 as a final coalgebra. In fact, $-n : n \to 0$ is the unique mediating homomorphism. If 0 were to be maximally final, then for every lax-homomorphism $m : n \to 0$ the inequality $m \leq -n$ should hold. But, on the contrary, $m : n \to 0$ is a lax-homomorphism if and only if $m \geq -n$. It then seems inevitable to abandon **Poset**-enrichment and move onto the more comfortable **Cpo**-enriched case:

Proposition 6.7 ([Plo91]) Final coalgebras of **Cpo**-endofunctors are maximally-final.

PROOF: Let $\mathcal{K}$ be a **Cpo**-category and let F be a **Cpo**-endofunctor on it. Let (U, μ) be a final coalgebra and let f be a lax-homomorphism from a coalgebra (S, σ) to (U, μ).

Define $\Phi : \mathcal{K}(S, U) \to \mathcal{K}(S, U)$ as the continuous mapping $x \mapsto \mu^{-1} \circ (Fx) \circ \sigma$. Then, $\langle \Phi^n(f) \rangle$ is an ω-chain and

$$\begin{aligned} \bigsqcup \Phi^{n+1}(f) &= \bigsqcup \mu^{-1} \circ F\big(\Phi^n(f)\big) \circ \sigma \\ &= \mu^{-1} \circ F\big(\bigsqcup \Phi^n(f)\big) \circ \sigma. \end{aligned}$$

Thus, $\bigsqcup \Phi^n(f)$ is the mediating morphism from (S, σ) to (U, μ) and $f \leq \bigsqcup \Phi^n(f)$. $\square$

Propositions 6.6 and 6.7 imply:

Theorem 6.8 (Order strong extensionality) Final coalgebras on a **Cpo**-endofunctor are order strongly extensional. $\square$

Finally, the proof principle reads as follows:

Principle 6.9 For a **Cpo**-endofunctor, if R is an ordered categorical bisimulation on the final coalgebra U, and u and u' are elements of U for which $u\ R\ u'$, then $u \leq u'$. $\square$

7 Final semantics in the order-enriched setting

We establish generalisations of Propositions 3.1 and 3.3 in the order-enriched setting.

Proposition 7.1 The final semantics into maximally-final coalgebras transforms ordered bisimilarity into inequality.

PROOF: Let U be a maximally-final coalgebra, and let p and q be elements of a coalgebra S.

If R is an ordered categorical bisimulation on S then, by the universal property of maximally-final coalgebras, it follows that $[\![\_]\!] \circ \pi_1 \circ R \leq [\![\_]\!] \circ \pi_2 \circ R$. Moreover, if $p\ R\ q$ then, writing $\langle\!\langle p, q \rangle\!\rangle$ for the unique element of R such that $\langle p, q\rangle = R \circ \langle\!\langle p, q \rangle\!\rangle$, it follows that $[\![p]\!] = [\![\_]\!] \circ \pi_1 \circ R \circ \langle\!\langle p, q \rangle\!\rangle \leq [\![\_]\!] \circ \pi_2 \circ R \circ \langle\!\langle p, q \rangle\!\rangle = [\![q]\!]$. $\square$

Definition 7.2 In a **Poset**-category, the ordered pair $(r_1 : R \to S, r_2 : R \to S)$ is a *(right-strict) bistrict lax-kernel pair* of $f : S \to U$ if $f \circ r_1 \leq f \circ r_2$ and, for every $p, q : T \to S$, whenever $f \circ p \leq f \circ q$ there exists a unique r such that $(r_1 \circ r \geq p)$ $r_1 \circ r = p$ and $r_2 \circ r = q$. $\square$

Proposition 7.3 Consider a **Poset**-category with bistrict lax-kernel pairs and a **Poset**-endofunctor on it mapping bistrict lax-kernel pairs to weak right-strict lax-kernel pairs. Then, for every coalgebra, elements with less than or equal final semantics are ordered bisimilar.

PROOF: Let (U, μ) be a final F-coalgebra. For an F-coalgebra (S, σ) let $(r_1 : R \to S, r_2 : R \to S)$ be a bistrict lax-kernel pair of $[\![\_]\!] : S \to U$. Then R is a relation (i.e. $\langle r_1, r_2\rangle : R \rightarrowtail S \times S$). Let us see that it is an ordered bisimulation,

$$F[\![\_]\!] \circ \sigma \circ r_1 \ = \ \mu \circ [\![\_]\!] \circ r_1 \ \leq \ \mu \circ [\![\_]\!] \circ r_2 \ = \ F[\![\_]\!] \circ \sigma \circ r_2$$

and, since (Fr_1, Fr_2) is a weak right-strict lax-kernel pair of $F[\![\_]\!]$, it follows that there exists a ρ such that $Fr_1 \circ \rho \geq \sigma \circ r_1$ and $Fr_2 \circ \rho = \sigma \circ r_2$.

Finally, by the universal property of bistrict lax-kernel pairs, for two elements p and q of S, if $[\![p]\!] \leq [\![q]\!]$ then $p\ R\ q$. $\square$

Remark. For categories of partial maps the above argument breaks down since we know that R is a subobject of the categorical product in the category of partial maps (if it exists) but we cannot guarantee that R is a subobject of the categorical product in the subcategory of total maps (see remark after Definition 2.1 and Appendix A).

For examples of final semantics for applicative and non-deterministic languages see [TJ93].

8 Applicative bisimulation for the untyped λ_V-calculus

As another application of the theory developed, an internal full abstraction result for the canonical model of the untyped *call-by-value* λ-calculus is proved.

In [AO93], and in [Pit94], as an example of coinduction for recursively defined domains, such a result is provided for the untyped *lazy* λ-calculus. Their proof relies on the way the model is constructed, namely as the colimit of a certain ω-chain of approximations. Instead, in accordance with [Pit93], it is shown that this kind of result follows from the universal property of the model.

First, the theory needed to solve domain equations for bifunctors in the spirit of [Fre91] is sketched (for details see [Fio94, Chapters 6 and 7]). Along the way, it is observed that a coinduction principle is also available for bifunctors.

Definition 8.1 (c.f. [Fre91]) A category is said to be *(***Cpo***-)algebraically compact* if every (**Cpo**-)endofunctor F on it has a free algebra $\alpha : FA \to A$, in the sense that is an initial algebra such that $\alpha^{-1} : A \to FA$ is a final coalgebra. $\square$

Examples of algebraically compact categories can be found in [Fre92]. Here we just recall that **pCpo** is **Cpo**-algebraically compact and therefore so is $\mathbf{pCpo}^{op} \times \mathbf{pCpo}$ (see [FP92, Fio94]).

Next we consider parameterised free algebras. Let $\mathcal{K}$ be a category and let $\mathcal{L}$ be an algebraically compact category. Given $G : \mathcal{K} \times \mathcal{L} \to \mathcal{L}$, for every $K \in\ \mid \mathcal{K} \mid$ let $\iota^G_K : G(K, G^\dagger K) \to G^\dagger K$ be a free $G(K, \_)$-algebra. Then, $G^\dagger$ extends to a functor from $\mathcal{K}$ to $\mathcal{L}$ where for every $f : K \to A$, we define $G^\dagger f$ to be the unique morphism making

$$\begin{tikzcd}
G(K, G^\dagger K) \arrow[r, "\iota^G_K"] \arrow[d, "G(f, G^\dagger f)"'] & G^\dagger K \arrow[d, "G^\dagger f"] \\
G(A, G^\dagger A) \arrow[r, "\iota^G_A"'] & G^\dagger A
\end{tikzcd}$$

commute.

The key lemma (specifically item 2) that allows us to apply the strong-extensionality theorem to bifunctors follows:

Lemma 8.2 Let $\mathcal{K}$ and $\mathcal{L}$ be algebraically compact categories, and let $F : \mathcal{K} \times \mathcal{L} \to \mathcal{K}$ and $G : \mathcal{K} \times \mathcal{L} \to \mathcal{L}$ be functors.

1. **(Bekič's lemma for algebraically compact categories** [Fre92])
   If (A, α) is a free $F(\_, G^\dagger\_)$-algebra then $\big((A, G^\dagger A), (\alpha, \iota^G_A)\big)$ is a free $\langle F, G\rangle$-algebra.

2. If $\big((K, L), (\kappa, \lambda)\big)$ is a free $\langle F, G\rangle$-algebra then (L, λ) is a free $G(K, \_)$-algebra.

PROOF: (2) Let (A, α) be a free $F(\_, G^\dagger\_)$-algebra.

By Bekič's lemma, there exists an isomorphism $(k, l) \in \mathcal{K} \times \mathcal{L}$ such that

$$\begin{tikzcd}
F(A, G^\dagger A) \arrow[r, "\alpha"] \arrow[d, "F(k, l)"'] & A \arrow[d, "k"] \\
F(K, L) \arrow[r, "\kappa"'] & K
\end{tikzcd}
\qquad
\begin{tikzcd}
G(A, G^\dagger A) \arrow[r, "\iota^G_A"] \arrow[d, "G(k, l)"'] & G^\dagger A \arrow[d, "l"] \\
G(K, L) \arrow[r, "\lambda"'] & L
\end{tikzcd}$$

Then, $l \circ G^{\dagger}(k^{-1})$ is a $G(K, \_)$-algebra isomorphism from $(G^{\dagger}K, \iota_K^G)$ to (L, λ). $\square$

Convention. We write $\breve{\mathcal{K}}$ for $\mathcal{K}^{op} \times \mathcal{K}$ and for a bifunctor $F : \breve{\mathcal{K}} \to \mathcal{K}$ we define $\breve{F} : \breve{\mathcal{K}} \to \breve{\mathcal{K}} : (f', f) \mapsto \big(F(f, f'), F(f', f)\big)$.

Also, $\_ \mathrel{=\!\!\!\rightharpoonup} =$ is defined to be the bifunctor $\mathbf{pCpo}(\_, =) : \mathbf{p\breve{C}po} \to \mathbf{pCpo}$.

The canonical model of the untyped call-by-value λ-calculus is specified next:

Proposition 8.3 In $\mathbf{pCpo}$, there exists an isomorphism $\delta : D \to D \mathrel{=\!\!\!\rightharpoonup} D$ such that (D, δ) is a free $D \mathrel{=\!\!\!\rightharpoonup} \_$-coalgebra.

Proof: Since $\mathbf{p\breve{C}po}$ is $\mathbf{Cpo}$-algebraically compact it follows that $\_ \mathrel{\breve{=\!\!\!\rightharpoonup}} = : \mathbf{p\breve{C}po} \to \mathbf{p\breve{C}po}$ has a free algebra of the form $\big((D, D), (\delta, \delta^{-1})\big)$ —see [Fio94]— and, by Lemma 8.2 (2), (D, δ) is a free $D \mathrel{=\!\!\!\rightharpoonup} \_$-coalgebra. $\square$

Remark. D (as in the above proposition) can be obtained as the colimit in $\mathbf{pCpo}$ of the ω-chain $\langle \delta_n : D_n \to D_{n+1} \rangle$ defined as

$$D_n = \begin{cases} \emptyset & \text{if } n = 0 \\ D_{n-1} \mathrel{=\!\!\!\rightharpoonup} D_{n-1} & \text{if } n \geq 1 \end{cases} \qquad \delta_n = \begin{cases} \emptyset & \text{if } n = 0 \\ {\delta_{n-1}}^R \mathrel{=\!\!\!\rightharpoonup} \delta_{n-1} & \text{if } n \geq 1 \end{cases}$$

where $(\delta_n, {\delta_n}^R)$ is an embedding-projection pair.

Let P be a cpo. For the endofunctor $P \mathrel{=\!\!\!\rightharpoonup} \_ : \mathbf{pCpo} \to \mathbf{pCpo}$, a relation $R \subseteq S \times S$ (viewed as a discrete cpo) is a categorical bisimulation on the coalgebra $\sigma : S \rightharpoonup P \mathrel{=\!\!\!\rightharpoonup} S$ if and only if $s \; R \; s'$ implies that for every $p \in P$, either $\sigma(s)(p)$ and $\sigma(s')(p)$ are both undefined, or they are both defined and $\sigma(s)(p) \; R \; \sigma(s')(p)$. The notion of bisimulation for the applicative structures considered here is obtained by making the notion of categorical bisimulation asymmetric (in the same way that Kleene's equality is generalised to Kleene's inequality).

Definition 8.4 A relation $R \subseteq S \times S$ is an *applicative bisimulation* on $\sigma : S \rightharpoonup P \mathrel{=\!\!\!\rightharpoonup} S$ if $s \; R \; s'$ implies that for every $p \in P$, if $\sigma(s)(p)$ is defined then so is $\sigma(s')(p)$ and $\sigma(s)(p) \; R \; \sigma(s')(p)$. $\square$

Convention. For a subset X of a cpo P we write $\overline{X}$ for its closure under lubs of ω-chains and $\hat{X}$ for $\{\bigsqcup x_n \mid \langle x_n \rangle \text{ is an } \omega\text{-chain in } X\}$.

Proposition 8.5

1. If $R \subseteq S \times S$ is an applicative bisimulation on $\sigma : S \rightharpoonup P \mathrel{=\!\!\!\rightharpoonup} S$ then so is $\hat{R}$.

2. If $\{R_i \subseteq S \times S \mid i \in I\}$ is a family of applicative bisimulations on $\sigma : S \rightharpoonup P \mathrel{=\!\!\!\rightharpoonup} S$ then $\bigcup_{i \in I} R_i$ is an applicative bisimulation on σ.

PROOF:

(1) Let $s \hat{R} s'$ and let $\langle x_n, {x_n}' \rangle$ be an ω-chain in R with $\bigsqcup x_n = s$ and $\bigsqcup {x_n}' = s'$.

For $p \in P$, if $\sigma(s)(p) = \sigma(\bigsqcup x_n)(p)$ is defined then there exists n_0 such that for all $n \geq n_0$, $\sigma(x_n)(p)$ is defined and $\bigsqcup_{n \geq n_0} \sigma(x_n)(p) = \sigma(\bigsqcup x_n)(p)$. Then, since R is an applicative bisimulation, for all $n \geq n_0$, $\sigma({x_n}')(p)$ is defined and $\sigma(x_n)(p) \; R \; \sigma({x_n}')(p)$. Thus, $\bigsqcup_{n \geq n_0} \sigma({x_n}')(p) = \sigma(\bigsqcup {x_n}')(p)$ is defined and

$$\sigma(s)(p) = \bigsqcup_{n \geq n_0} \sigma(x_n)(p) \;\; \hat{R} \;\; \bigsqcup_{n \geq n_0} \sigma({x_n}')(p) = \sigma(s')(p).$$

(2) Straightforward. $\square$

Corollary 8.6 If $R \subseteq S \times S$ is an applicative bisimulation on $\sigma : S \rightharpoonup P \Rightarrow S$ then so is $\overline{R}$.

PROOF: Define

$$\begin{array}{rcll} R_0 & = & R, & \\ R_{k+1} & = & \hat{R}_k & \text{for } k \text{ an ordinal}, \\ R_l & = & \bigcup_{k<l} R_k & \text{for } l \text{ a limit ordinal}. \end{array}$$

Then, by transfinite induction using Proposition 8.5, R_k is an applicative bisimulation for every k. The result follows because $\overline{R} = R_k$ for some (sufficiently large) k. $\square$

Every applicative bisimulation on a cppo induces an ordered categorical bisimulation containing it:

Proposition 8.7 If $R \subseteq S_\perp \times S_\perp$ is an applicative bisimulation on $\sigma : S_\perp \rightharpoonup P \Rightarrow S_\perp$ then $\overline{R} \cup (\{\perp\} \times S_\perp) \rightarrowtail S_\perp \times S_\perp$ is an ordered categorical bisimulation on $(S_\perp, \sigma)$.

PROOF: Writing R' for $\overline{R} \cup (\{\perp\} \times S_\perp)$, the continuous function $\rho : R' \to P \rightharpoonup R'$, where $\rho(s, s')$ is the partial continuous function mapping

$$p \mapsto \begin{cases} \big(\sigma(s)(p), \sigma(s')(p)\big) & \text{if } \sigma(s)(p) \downarrow \text{ and } \sigma(s')(p) \downarrow \\ \big(\perp, \sigma(s')(p)\big) & \text{if } \sigma(s)(p) \uparrow \text{ and } \sigma(s')(p) \downarrow \end{cases}$$

shows that R' is an ordered categorical bisimulation. $\square$

The definition of applicative bisimulation straightforwardly implies:

Proposition 8.8 For every $\sigma : S \rightharpoonup P \rightharpoonup S$, the inequality relation $\leq_S$ is an applicative bisimulation. $\square$

Finally, the result is stated and proved.

Theorem 8.9 (Internal full abstraction) The inequality relation $\leq_D$ is maximal among all applicative bisimulations on $\delta : D \to D \rightharpoonup D$ as in Proposition 8.3.

PROOF: By Proposition 8.8, $\leq_D$ is an applicative bisimulation. It is the maximal such because if R is an applicative bisimulation on (D, δ) then, by Proposition 8.7 and the order strong-extensionality theorem, $\overline{R} \cup (\{\perp\} \times D) \rightarrowtail \leq_D$. Thus, $R \subseteq \leq_D$. $\square$

9 Concluding remarks and future work

We have concentrated on the impact of one of the fundamental concepts in concurrency theory, namely bisimulation, on the theory of data types. We have related an abstract formulation of bisimulation and the final coalgebra approach to the semantics of data types: we established conditions under which the operational notion of bisimulation and the denotational notion of final semantics coincide and provided a coinduction principle for recursive data types.

Further study of the relationship between order strong-extensionality and finality in $\mathbf{Cppo}_\perp$ can be found in [Rut93]. There, final coalgebras of $\mathbf{Cpo}$-endofunctors mapping bistrict lax-kernel pairs to weak right-strict lax-kernel pairs are characterised as order strongly-extensional dense-epis.

In view of the ideas exposed throughout, one of the more important open issues is whether the theory of data types can provide some feedback into the theory of concurrency. In this respect, for example, the study of the category of $\mathcal{P}(A \times \_)$-coalgebras (regarded as a category of transition systems with set of actions A) might provide new insight into models for concurrency. To a certain extent, this has already occurred since this category of transition systems has the interesting property of having bisimulation-preserving maps, it differs from the one considered in [WN94] and, as far as we know, it has not yet been studied.

Another intriguing connection appears with the work of Amadio and Cardelli [AC93]. They considered a simply-typed λ-calculus with recursive types and define two recursive types to be equal if they induce the same infinite tree obtained by unfolding the recursive type. This is a form of final semantics and testing for equality reduces to finding a bisimulation. This might not be completely surprising since on page 25 of the paper, while presenting the most interesting rule (contractiveness) for characterising equality, they point out that it "was also inspired by a standard proof technique for bisimulation". It would be interesting to investigate the precise relation between bisimulation and contractiveness in the context of type equality and subtyping.

Acknowledgements

This work was done under the supervision of Gordon Plotkin and Barry Jay. I would like to thank them both for their involvement and guidance. I would also like to thank Andy Pitts and Jan Rutten for comments on previous drafts, to one of the LICS referees for providing useful comments, and to Thomas Streicher for detecting an error in a previous version. Finally, I am grateful to the IC referees for their detailed comments that led to a much better presentation.

A Categories of partial maps

For a thorough treatment of categories of partial maps consult [RR88] and [Fio94, Chapter 3].

A.1 Partial maps

Definition A.1 ([Mog86, Ros86]) A *domain structure* is a pair $(\mathcal{K}, \mathcal{D})$ consisting of a category $\mathcal{K}$ and a well-powered category $\mathcal{D}$, such that $\mathcal{D}$ is a full-on-objects subcategory of $\mathcal{K}$ all of whose morphisms are monos in $\mathcal{K}$, with the following closure property: every diagram $X \xrightarrow{f} A \overset{n}{\leftarrowtail} D$ with $f \in \mathcal{K}$ and $n \in \mathcal{D}$ has a pullback in $\mathcal{K}$ and if $X \overset{f^{-1}(n)}{\leftarrowtail} f^{-1}(D) \xrightarrow{n^* f} D$ is any such pullback then $f^{-1}(n) \in \mathcal{D}$. $\square$

Convention. $\mathcal{K}$ is called the category of *total* maps, and $\mathcal{D}$ is called the category of *admissible* monos.

Definition A.2 The *category of partial maps* $p(\mathcal{K}, \mathcal{D})$ induced by a domain structure $(\mathcal{K}, \mathcal{D})$ has the same objects as $\mathcal{K}$; a partial map $[m, f] : A \rightharpoonup B$ is an equivalence class of spans $A \overset{m}{\leftarrowtail} D \xrightarrow{f} B$ in $\mathcal{D} \times \mathcal{K}$, where two spans $A \overset{m}{\leftarrowtail} D \xrightarrow{f} B$ and $A \overset{n}{\leftarrowtail} E \xrightarrow{g} B$ are equivalent if and only if $m = n \circ i$ and $f = g \circ i$ for some isomorphism $i : D \cong E$. Composition of partial maps is given as for relations, by pullback; identities have the form $[\mathrm{id}_A, \mathrm{id}_A]$. $\square$

Convention. Whenever we write $p(\mathcal{K}, \mathcal{D})$ it is assumed that $(\mathcal{K}, \mathcal{D})$ is a domain structure; when $\mathcal{D}$ is clear from the context we simply write $p\mathcal{K}$.

To distinguish total maps (i.e. morphisms in $\mathcal{K}$) from partial maps (i.e. morphisms in $p\mathcal{K}$) we denote the former with $\rightarrow$ and the latter with $\rightharpoonup$.

The motivating example of a domain structure is $(\mathbf{Cpo}, \mathbf{\Sigma})$ where $\mathbf{\Sigma}$ is the subcategory of $\mathbf{Cpo}$ consisting of all those order-reflecting monos whose subobjects are Scott-open. We have: $p(\mathbf{Cpo}, \mathbf{\Sigma}) \cong \mathbf{pCpo}$.

A.2 Binary partial products

Let $\mathcal{K}$ have binary products. The *partial pairing* of $u = [m, f] : B \rightharpoonup A_1$ and $v = [n, g] : B \rightharpoonup A_2$ is the partial map $B \rightharpoonup A_1 \times A_2$ defined by

$$\langle u, v \rangle \quad = \quad \left[m \cap n, \left(f \circ m^{-1}(n), g \circ n^{-1}(m)\right)\right]$$

where $(\_, =)$ is the pairing of total maps. The product functor on $\mathcal{K}$ extends to a *partial product* functor $\_ \times = : p\mathcal{K} \times p\mathcal{K} \rightarrow p\mathcal{K}$ sending a pair of objects (A, B) to $A \times B$ and a pair of partial maps (u, v) to $\langle u \circ \pi_1, v \circ \pi_2 \rangle$.

Definition A.3 A category of partial maps $p(\mathcal{K}, \mathcal{D})$ is *partial cartesian* if its subcategory of total maps $\mathcal{K}$ is cartesian. $\square$

References

[Abr91] S. Abramsky. A domain equation for bisimulation. *Information and Computation*, 92:161–218, 1991.

[AC93] R.M. Amadio and L. Cardelli. Subtyping recursive types. *ACM Transactions on Programming Languages and Systems*, 15(4):575–631, 1993.

[Acz88] P. Aczel. *Non-well-founded Sets*. C.S.L.I. Lecture Notes; no. 14, 1988.

[AM82] M. Arbib and E. Manes. Parametrized data types do not need highly constrained parameters. *Information and Control*, 52:139–158, 1982.

[AM89] P. Aczel and P.F. Mendler. A final coalgebra theorem. In D.H. Pitt, D.E. Rydeheard, P. Dybjer, A.M. Pitts, and A. Poigné, editors, *Category Theory and Computer Science*, volume 389 of *Springer-Verlag*, pages 357–365, September 1989.

[AO93] S. Abramsky and C.-H. L. Ong. Full abstraction in the lazy lambda calculus. *Information and Computation*, 105:159–267, 1993.

[Cou83] B. Courcelle. Fundamental properties of infinite trees. *Theoretical Computer Science*, 25:95–169, 1983.

[Fio94] M.P. Fiore. *Axiomatic Domain Theory in Categories of Partial Maps*. PhD thesis, University of Edinburgh, 1994. To be published by Cambridge University Press in the Distinguished Dissertations Series.

[FP92] M.P. Fiore and G.D. Plotkin. On compactness and **Cpo**-enriched categories. In G. Winskel, editor, *Proceedings of the CLICS Workshop (23-27 March 1992)*, volume 397-II of *DAIMI PB*, pages 571–584. Computer Science Department, Aarhus University, May 1992.

[Fre90] P.J. Freyd. Recursive types reduced to inductive types. In *5^{th} LICS Conf.*, pages 498–507. IEEE, Computer Society Press, 1990.

[Fre91] P.J. Freyd. Algebraically complete categories. In A. Carboni, M. C. Pedicchio, and G. Rosolini, editors, *Category Theory*, volume 1488 of *Lecture Notes in Mathematics*, pages 131–156. Springer-Verlag, 1991.

[Fre92] P.J. Freyd. Remarks on algebraically compact categories. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of Categories in Computer Science*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1992.

[Gor80] M.J.C. Gordon. The denotational semantics of sequential machines. *Information Processing Letters*, 10(1):1–3, February 1980.

[HJ95] C. Hermida and B. Jacobs. Induction and coinduction via subset types and quotient types. In P. Dybjer and R. Pollack, editors, *Informal proceedings of the Joint CLICS-TYPES Workshop on Categories and Type Theory*, May 1995. (Available as Technical Report 85, Programming Methodology Group, Göteborg University and Chalmers University of Technology).

[HPJW92] P. Huday, S. Peyton Jones, and P. Wadler. Report on the functional programming language Haskell, Version 1.2. *SIGPLAN Notices*, 27(5), May 1992.

[HU79] J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 1979.

[Kel82] G.M. Kelly. *Basic Concepts of Enriched Category Theory.* Cambridge University Press, 1982.

[LS94] F.W. Lawvere and S.H. Schanuel. *Conceptual Mathematics: A first introduction to categories.* Buffalo Workshop Press, 1994.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice Hall, 1989.

[Mog86] E. Moggi. Categories of partial morphisms and the partial lambda-calculus. In *Proceedings Workshop on Category Theory and Computer Programming, Guildford 1985*, volume 240 of *Lecture Notes in Computer Science*, pages 242–251. Springer-Verlag, 1986.

[MT88] R. Milner and M. Tofte. Co-induction in relational semantics. Technical Report ECS-LFCS-88-65, University of Edinburgh, Department of Computer Science, October 1988.

[MTH90] R. Milner, M. Tofte, and R. Harper. *The Definition of Standard ML.* MIT Press, 1990.

[Par80] D.M. Park. On the semantics of fair parallelism. In D. Bjørner, editor, *Abstract Software Specifications*, volume 86 of *Lecture Notes in Computer Science*, pages 504–526. Springer-Verlag, 1980.

[Par81] D.M. Park. Concurrency and automata on infinite sequences. *Lecture Notes in Computer Science*, 104:167–183, 1981.

[Pie91] B.C. Pierce. *Basic Category Theory for Computer Scientists.* MIT Press, 1991.

[Pit93] A.M. Pitts. Relational properties of recursively defined datatypes. In *8^{th} LICS Conf.* IEEE, Computer Society Press, 1993.

[Pit94] A.M. Pitts. A co-induction principle for recursively defined domains. *Theoretical Computer Science*, 124:195–219, 1994.

[Plo76] G.D. Plotkin. A powerdomain construction. *SIAM Journal of Computing*, 5(3):452–487, 1976.

[Plo85] G.D. Plotkin. Denotational semantics with partial functions. Lecture at C.S.L.I. Summer School, 1985.

[Plo91] G.D. Plotkin. Some notes on recursive domain equations. Handwritten notes for the Domain Theory PG Course. University of Edinburgh, 1991.

[Ros86] G. Rosolini. *Continuity and Effectiveness in Topoi*. PhD thesis, University of Oxford, 1986.

[RR88] E. Robinson and G. Rosolini. Categories of partial maps. *Information and Computation*, 79:95–130, 1988.

[RT93] J. Rutten and D. Turi. On the foundations of final semantics: Non-standard sets, metric spaces, partial orders. In J.W. de Bakker, W.P. de Roever, and Rozenberg, editors, *Proc. of the REX workshop, Semantics: Foundations and Applications*, volume 666 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993.

[Rut93] J. Rutten. A structural co-induction theorem. In *Proc. of the 9^{th} Conference on the Mathematical Foundations of Programming Semantics*, Lecture Notes in Computer Science. Springer-Verlag, 1993. To appear.

[Smy91] M.B. Smyth. I-categories and duality. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of Categories in Computer Science*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 270–287. Cambridge University Press, 1991.

[SP82] M.B. Smyth and G.D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11(4):761–783, November 1982.

[SS71] D.S. Scott and C. Strachey. Towards a mathematical semantics for computer languages. Technical Report PRG-6, Oxford Univ. Computing Lab., 1971.

[TJ93] D. Turi and B. Jacobs. On final semantics for applicative and non-deterministic languages. In *CTCS-5 (Category Theory and Computer Science Fifth Biennial Meeting)*, pages 52–57. CWI, September 1993.

[WN94] G. Winskel and M. Nielsen. Models for concurrency. Technical Report RS-94-12, BRICS Research Series, May 1994. To appear as a chapter in the Handbook of Logic in Computer Science.