

Relational Properties of Recursively Defined Domains

Andrew M. Pitts*

University of Cambridge Computer Laboratory
Cambridge CB2 3QG, UK

Abstract

This paper describes a mixed induction/co-induction property of relations on recursively defined domains. We work within a general framework for relations on domains and for actions of type constructors on relations introduced by O’Hearn and Tennent [20], and draw upon Freyd’s analysis [7] of recursive types in terms of a simultaneous initiality/finality property. The utility of the mixed induction/co-induction property is demonstrated by deriving a number of families of proof principles from it. One instance of the relational framework yields a family of induction principles for admissible subsets of general recursively defined domains which extends the principle of structural induction for inductively defined sets. Another instance of the framework yields the co-induction principle studied by the author in [22], by which equalities between elements of recursively defined domains may be proved via ‘bisimulations’.

1 Introduction

A characteristic feature of higher-order functional languages such as Standard ML [16] or Haskell [11] is the facilities they provide for declaring and using recursive datatypes. However, this powerful feature comes at a price: because few syntactic restrictions are placed upon the form of a datatype declaration, in general it can be hard to understand what the values of such a datatype denote, and correspondingly hard to reason about the observable behaviour under evaluation of expressions involving such datatypes. One way round this problem is to consider only restricted forms of datatype declaration, for example just those giving rise to inductively defined sets of finite values (lists, trees, etc.) for which there are easily understood reasoning techniques such as the principle of structural induction. This is too restrictive since it

cuts out ‘lazy’ datatypes (containing partial and potentially infinite values), which are an important functional programming tool and unavoidable by design in non-strict languages such as Haskell. Whilst properties of lazy datatypes have been considered on a case-by-case basis (see [31], for example), there are remarkably few reasoning principles in the literature that apply uniformly to all recursive datatypes. This paper attempts to improve this situation by proving a fundamental property of (abstract) relations on recursive datatypes. We work with the denotational semantics of datatype declarations given via recursively defined domains.

A key idea used in this paper, going back at least to [23, 15, 25, 26], is to take account of the fact that the action of various type constructors on domains can be extended to an action on relations. Traditionally one considers only certain relations on domains—ones that are sufficiently complete, or ‘admissible’. We need to go beyond this to arbitrary set-theoretic relations in order to capture the co-inductive properties of recursively defined domains. Indeed we will see that our results go through for a very abstract notion of ‘relation’ which includes, for example, continuous endofunctions of a domain.

A second crucial idea, present in Freyd’s recent work on recursive types [6, 7, 8], is to treat separately the positive and negative occurrences of a type in the body of the declaration that defines it. To explain this further, consider the ML declaration

$$\text{datatype } \mathbf{ty} = \mathbf{C}_1 \text{ of } \sigma_1 \mid \cdots \mid \mathbf{C}_n \text{ of } \sigma_n \quad (1)$$

where the types $\sigma_1, \dots, \sigma_n$ are built up from basic types (booleans, integers, characters, etc.) and the type variable \mathbf{ty} , using any combination of the product ($*$) and function space (\rightarrow) constructors. The denotational semantics of \mathbf{ty} is then the minimal solution, $\text{rec}\alpha.\Phi(\alpha)$, to the domain equation $\alpha = \Phi(\alpha)$, where $D \mapsto \Phi(D)$ is an operation on domains built up according to the structure of the body of the declaration (1), using appropriate domain constructors for the type constructors $*$, \rightarrow , and $|$. This opera-

*Research supported by UK SERC grant GR/G53279.

tion on domains can be extended to an operation on relations, sending a relation R on a domain D to a relation $\Phi(R)$ on the domain $\Phi(D)$. The potentially complex behaviour of expressions of type \mathbf{ty} stems in large part from the fact that the declared type can occur both positively and negatively in the body of the declaration (1). (Recall that an occurrence of \mathbf{ty} in σ_i is negative if it is to the left, hereditarily, of an odd number of function space constructors, and is positive otherwise.) For this reason, the operation $R \mapsto \Phi(R)$ will in general not be monotone for inclusions between relations. However, we can treat positive and negative occurrences separately in the semantics and then diagonalize. Replacing positive occurrences of \mathbf{ty} by a new type variable \mathbf{ty}^+ and negative ones by a different type variable \mathbf{ty}^- and then taking the semantics of the declaration body, we obtain a domain constructor of two variables, $(D^-, D^+) \mapsto \hat{\Phi}(D^-, D^+)$, from which the original one can be recovered, since $\Phi(D) = \hat{\Phi}(D, D)$. Just as for Φ , the operation $\hat{\Phi}$ can be extended to one on relations, $(R^-, R^+) \mapsto \hat{\Phi}(R^-, R^+)$; but now this operation is order-preserving in its right-hand argument and order-reversing in its left-hand one. If R^- and R^+ are both relations on a domain D , then $\hat{\Phi}(R^-, R^+)$ is a relation on $\hat{\Phi}(D, D)$, i.e. on $\Phi(D)$; if moreover there is an isomorphism $D \cong \Phi(D)$, we can identify the relation $\hat{\Phi}(R^-, R^+)$ with one on D again. In this way, for each solution $D \cong \Phi(D)$ of the domain equation corresponding to (1) we can associate a binary operation $b : \mathcal{R}(D) \times \mathcal{R}(D) \rightarrow \mathcal{R}(D)$ on the set $\mathcal{R}(D)$ of relations on D ; and for inclusions, \subset , between relations, b is order-preserving in its right-hand argument and order-reversing in its left-hand argument. For such an operation one can consider fixed points, $b(I, I) = I$, with a mixed inductive/co-inductive property—namely that if $R^- \subset b(R^+, R^-)$ and $b(R^-, R^+) \subset R^+$ then $R^- \subset I \subset R^+$. The main result of this paper (Theorem 2.9) is that when D is the recursively defined domain $\text{rec}\alpha.\Phi(\alpha)$, the identity relation I on D has this mixed inductive/co-inductive fixed point property (provided the relations R^+ in the positive place are suitably ‘admissible’). Of course we will deduce this property from the mixed initiality/finality property of recursively defined domains which Freyd has focused upon in his work on ‘algebraically compact’ categories [7, 8].

The above discussion does not make precise exactly what is a ‘relation’ on a domain. In fact we need remarkably few properties of relations in order to establish our main result. These properties can be conveniently axiomatized using the framework employed by O’Hearn and Tennent in their recent work [20] on

applying relational parametricity to the study of local-variable declarations. Thus our theorem about relations on recursively defined domains is in fact a family of properties, parameterized by the particular notion of ‘relation’ being used and by the particular way type constructors act on these relations. By varying these parameters we show that the main Theorem 2.9 contains the following reasoning principles as particular cases:

- An *induction principle* for admissible (i.e. chain-complete, $--$ -containing) subsets of a general recursively defined domain. This induction property coincides with structural induction when the recursively defined domain is the lift of an inductively defined set. More generally, if the recursively defined domain is given by a type constructor in which the defined type only occurs positively, then the induction property coincides with an *initial algebra induction principle* in the sense of Lehmann and Smyth [13]. In particular it includes the induction principle for the *fixed point object* of the lifting monad, studied by Crole and Pitts [3]. (Scott’s principle of induction for admissible properties of least fixed points of continuous functions is a consequence of this case.) We show that the induction property can yield an interesting proof principle even for problematic recursively defined domains, such as those that model untyped lambda calculus. See section 3.
- The *co-induction principle* described in [22], by which two elements of a recursively defined domain may be proved equal by constructing a ‘bisimulation’ relating them (or may be proved to lie in the order relation by constructing a ‘simulation’ relation). In contrast to the induction principle, the co-induction principle deals with arbitrary relations rather than just admissible ones. See section 4.
- Scott’s characterization of the identity function on a recursively defined domain as the least fixed point of a certain operator on continuous endofunctions of the domain. (See Definition 2.7.)

Although Theorem 2.9 is therefore quite powerful, the proof we give of it is surprisingly simple. In fact it is the last property in the above list which is used to derive the theorem, together with the uniformity property for least fixed points of continuous functions (‘Plotkin’s Axiom’—see [10, section 2.3]). Section 2 sets up the particular framework for relations we use and presents the theorem. Then in sections 3 and 4

we show how to derive the induction and co-induction principles mentioned above. The bulk of this paper is about proving *properties* of relations on recursively defined domains; in section 5 we indicate briefly how the techniques of the paper can be used to *construct* relations. This permits one to extend the main result of the paper to parameterized recursive domains and (as will be described elsewhere) leads to simplified proofs of ‘computational adequacy’ of denotational semantics for functional languages with recursive types.

2 Relational structures

One would like reasoning principles for recursively defined domains that apply to as wide a class of properties of domains as possible. We will achieve this aim via an axiomatic framework for relations on domains. The framework is surprisingly simple—remarkably few properties of a general notion of ‘relation’ are needed to establish quite powerful induction and co-induction principles. Accordingly there is a diversity of examples which fit into this framework. The structure for relations we use is based on that used by O’Hearn and Tennent [20] in their work on applying relational parametricity to the study of local-variable declarations. To it we add an abstract treatment of the crucial property of relations that in LCF [9] is called *admissibility* (meaning ‘admitting induction’).

Our domain-theoretic terminology and notation generally follows that of the survey article of Gunter and Scott [10]. We will use the term *cpo* to mean a partially ordered set possessing least upper bounds of all countable chains, but not necessarily possessing a least element. A *pointed* cpo, or *cpo* for short, is a cpo D that does possess a least element, denoted $-_D$ (or just $-$). This simple notion of semantic domain is sufficient for our purposes here, since we are interested in recursively defined domains for modelling recursive datatypes in (strict or lazy) higher order functional programming languages. Thus we will only consider the following constructions on cppo’s: $(-)_-$ (*lifting*), $(-) \times (-)$ (*cartesian product*), $(-) \otimes (-)$ (*smash product*), $(-) \oplus (-)$ (*coalesced sum*), $(-) \rightarrow (-)$ (*continuous function space*), and $(-) \multimap (-)$ (*strict continuous function space*).

Let \mathcal{Cpo}_- denote the category of cppo’s and strict continuous functions. We will use the notation $f : D \multimap E$ to indicate that f is such a function between cppo’s. *Strict* continuous functions play a prominent role because, as is well known, the above constructions on domains can be extended to well-behaved (i.e. functorial and continuous) constructions on such functions.

Thus lifting determines a functor $\mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$, the cartesian and smash products determine functors $\mathcal{Cpo}_- \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$, and the strict and non-strict continuous function spaces determine functors $\mathcal{Cpo}_-^{\text{op}} \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$; moreover, all these functors are *locally continuous*, in the sense that their action on morphisms preserves least upper bounds of countable chains.

Definition 2.1 (Cf. [20, section 2].) A *relational structure* for \mathcal{Cpo}_- , \mathcal{R} , is specified by the following data:

- For each cppo D , a set $\mathcal{R}(D)$ (of ‘ \mathcal{R} -relations on D ’).
- For each strict continuous function $f : D \multimap E$ between cppo’s, a binary relation between elements $R \in \mathcal{R}(D)$ and elements $S \in \mathcal{R}(E)$, written

$$f : R \subset_{\mathcal{R}} S$$

(or just $f : R \subset S$). These binary relations are required to satisfy the following properties:

- (a) $id_D : R \subset R$, for all $R \in \mathcal{R}(D)$;
- (b) if f and g are composable, $f : R \subset S$ and $g : S \subset T$, then $gf : R \subset T$.

Note in particular that the binary relation $R \subset_{\mathcal{R}} R'$ on $\mathcal{R}(D)$ which is defined to hold if and only if $id_D : R \subset R'$, is a preorder.

We say that \mathcal{R} is a *unitary* relational structure if each $\mathcal{R}(D)$ contains a distinguished element I_D (called the ‘identity \mathcal{R} -relation on D ’) such that

- (c) $f : I_D \subset I_E$, for all $f : D \multimap E$.

Here are three examples of unitary relational structures for \mathcal{Cpo}_- which will be relevant to this paper.

Examples 2.2 (i) ‘**Strict**’ relations. Fix a natural number $n \geq 1$. For each cppo D let $\mathcal{R}(D)$ consist of all subsets of the n -fold smash product $D \otimes \cdots \otimes D$ containing $-$. The distinguished identity relation is defined to be $I_D = \{(x, \dots, x) \mid x \neq -\} \cup \{-\}$. The relation $f : R \subset S$ is defined to hold if and only if for all $u \in R$, $(f \otimes \cdots \otimes f)(u) \in S$. Conditions (a)–(c) are easily verified. We will use this example, for the case $n = 1$, to derive an induction principle for recursively defined domains in section 3.

(ii) **‘Non-strict’ relations.** Fix a natural number $n \geq 1$, and for each cppo D let $\mathcal{R}(D)$ consist of all subsets of the n -fold product $D \times \cdots \times D$. The distinguished identity relation is defined to be $I_D = \{(x_1, \dots, x_n) \mid x_1 \sqsubseteq x_2 \sqsubseteq \cdots \sqsubseteq x_n\}$. The relation $f : R \subset S$ is defined to hold if and only if for all $u \in R$, $(f \times \cdots \times f)(u) \in S$. Once again, conditions (a)–(c) are easily verified. We will use this example, for the case $n = 2$, to derive a co-induction principle for recursively defined domains in section 4. (Taking I_D instead to be the diagonal relation leads to a co-induction principle phrased in terms of two-sided ‘bisimulations’.)

(iii) **Endofunctions as relations.** For each cppo D , let $\mathcal{R}(D)$ consist of all strict continuous functions from D to itself. The identity \mathcal{R} -relation is defined to be the identity function. For $f : D \multimap E$, the relation $f : R \subset S$ is defined to hold iff for all $x \in D$, $f(R(x)) = S(f(x))$.

The unitary relational structures of Definition 2.1 are a special case of the categorical reflexive graphs used by O’Hearn and Tennent in [20], which concentrates on the case of binary relations. The specialization has two aspects. First, we have given the definition of a relational structure for the category of cppo’s and strict continuous functions, \mathcal{Cpo}_- , rather than for a general category. Secondly, we have defined a *unary* relational structure (notwithstanding the fact that the examples above involve n -ary relations). An n -ary relational structure would be specified by sets $\mathcal{R}(D_1, \dots, D_n)$ for each n -tuple of cppo’s, and by relations $f_1, \dots, f_n : R \subset S$ (where $f_i : D_i \multimap E_i$, for $i = 1, \dots, n$), satisfying analogues of conditions (a) and (b).

Definition 2.3 Let \mathcal{R} be a relational structure for cppo’s. An \mathcal{R} -relation $R \in \mathcal{R}(D)$ is called *admissible* if for all other \mathcal{R} -relations $S \in \mathcal{R}(E)$ the subset

$$[S, R] \stackrel{\text{def}}{=} \{f \mid f : S \subset R\}$$

of the cppo $E \multimap D$ contains $-$ and is closed under taking least upper bounds of countable chains. If \mathcal{R} is a unitary relational structure and each identity \mathcal{R} -relation, I_D , is admissible, we say that \mathcal{R} itself is *admissible*.

Remark 2.4 Although we have defined admissibility in terms of the concrete structure of cppo’s, the definition can in fact be phrased purely in terms of the *monoidal closed* structure $1_-, \otimes, \multimap$ on the category \mathcal{Cpo}_- . (See [14, Chapter VII] for an introduction

to this categorical notion.) Here 1_- is the lift of a one-element set: it is a unit for the smash product operation on cppo’s. Using the adjunction between $(-)\otimes E$ and $E \multimap (-)$, morphisms $1_- \multimap (E \multimap D)$ correspond naturally to morphisms $E \cong (1_- \otimes E) \multimap D$. Then $R \in \mathcal{R}(D)$ is admissible if and only if for each $S \in \mathcal{R}(E)$ there is a morphism $m : [S, R] \multimap (E \multimap D)$ with the property that composition with m induces a bijection between morphisms $1_- \multimap [S, R]$ and morphisms $f : E \multimap D$ for which $f : S \subset R$ holds. (Such an m is necessarily a monomorphism, hence determines a sub-cppo of $(E \multimap D)$ —namely the one given in Definition 2.3.)

Example 2.5 When \mathcal{R} is the (unitary) relational structure of Example 2.2(i), it can be shown that $R \in \mathcal{R}(D)$ is admissible if and only if the subset $R \subseteq D \otimes \cdots \otimes D$ is chain-complete. Note that since each I_D is admissible in this case, \mathcal{R} is by definition an admissible relational structure. Similarly, for Example 2.2(ii), $R \in \mathcal{R}(D)$ is admissible if and only if it is a chain-complete subset of $D \times \cdots \times D$ containing $(-, \dots, -)$, and \mathcal{R} is itself an admissible relational structure. For Example 2.2(iii), *any* \mathcal{R} -relation is admissible (and so in particular \mathcal{R} is itself admissible).

Given a particular relational structure for cppo’s, \mathcal{R} , we are interested in lifting functorial constructions on cppo’s to constructions on \mathcal{R} -relations. The basic requirement is that the construction on \mathcal{R} -relations respects the relation $- : - \subset_{\mathcal{R}} -$ (and for *unitary* relational structures, that the construction preserve identity \mathcal{R} -relations). However, in certain important examples such actions may be subject to certain admissibility conditions. We will see an example of this for the relational structure considered in section 4 for coinduction. The following definition captures the properties we require of a functor $\mathcal{Cpo}_-^{\text{op}} \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$ in order to establish our main result about relations on recursively defined domains (Theorem 2.9).

Definition 2.6 Let \mathcal{R} be an a unitary relational structure for cppo’s, and let $F : \mathcal{Cpo}_-^{\text{op}} \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$ be a functor. A (unitary) *admissible action of F on \mathcal{R} -relations* is an operation mapping each pair of \mathcal{R} -relations $R \in \mathcal{R}(D)$ and $S \in \mathcal{R}(E)$ to an \mathcal{R} -relation $F(R, S) \in \mathcal{R}(F(D, E))$, satisfying:

- (a) if $f : R' \subset R$ and $g : S \subset S'$ with R and S' admissible, then $F(f, g) : F(R, S) \subset F(R', S')$ with $F(R', S')$ admissible;
- (b) $F(I_D, I_E) = I_{F(D, E)}$.

We gave the above definition for functors of two arguments which are contravariant in their first argument and covariant in their second, because, as recent work of Freyd [6, 7, 8] emphasises, it is precisely this kind of functor which is relevant for recursive domain definitions whose bodies contain negative as well as positive occurrences of the defined domain. (An occurrence is negative if to the left, hereditarily, of an odd number of function space constructors, and is positive otherwise.) Thus solutions to a domain equation $\alpha = \Phi(\alpha)$ are the same as solutions to $\alpha = \hat{\Phi}(\alpha, \alpha)$, where $\hat{\Phi}(\alpha^-, \alpha^+)$ is obtained from $\Phi(\alpha)$ by replacing positive occurrences of α by a variable α^+ and negative ones by a distinct variable α^- . The point of this separation of variables is that unlike Φ , $\hat{\Phi}$ has a continuous, functorial action on strict continuous functions: using the functoriality of the basic domain constructions, we can proceed by induction on the structure of $\hat{\Phi}$ to define a locally continuous functor $\hat{\Phi} : \mathcal{Cpo}_{-}^{\text{op}} \times \mathcal{Cpo}_{-} \rightarrow \mathcal{Cpo}_{-}$.

Definition 2.7 Let $F : \mathcal{Cpo}_{-}^{\text{op}} \times \mathcal{Cpo}_{-} \rightarrow \mathcal{Cpo}_{-}$ be a locally continuous functor. An *invariant* for F is a cppo D equipped with an isomorphism $i : F(D, D) \cong D$. We will say that such an invariant is *minimal* if $id_D : D \multimap D$ is the least fixed point of the continuous function $\delta : (D \multimap D) \rightarrow (D \multimap D)$ given by $\delta(\epsilon) = iF(\epsilon, \epsilon)i^{-1}$.

The existence of a minimal invariant for a locally continuous functor follows by applying Scott's original construction of recursively defined domains via colimits of chains of embedding-projection pairs (within the category-theoretic setting subsequently developed by Smyth and Plotkin [30]). In particular, when $F = \hat{\Phi}$ is the functor associated to a domain constructor Φ as above, then the minimal invariant for F is the recursively defined domain $\text{rec}\alpha.\Phi(\alpha)$, i.e. the solution to $\alpha = \Phi(\alpha)$ constructed via any of the several methods available in the literature (see [30, 28] for example).

The concept of minimal invariant given by Freyd in [7] is apparently stronger than that given in Definition 2.7, in that it requires id_D to be the *unique* fixed point of δ . However, for locally continuous functors this strengthening is a corollary of the following result, which is the most useful 'universal property' for recursively defined domains that has been found to date.

Theorem 2.8 (Freyd) *Let F , D and i be as in Definition 2.7. For each pair of cppo's A, B and each pair of strict continuous functions*

$$f : A \multimap F(B, A) \quad g : F(A, B) \multimap B$$

there are unique strict continuous functions $h : A \multimap D$ and $k : D \multimap B$ such that

$$h = iF(k, h)f \tag{2}$$

$$k = gF(h, k)i^{-1} \tag{3}$$

Proof Consider the function $\phi : (A \multimap D) \times (D \multimap B) \rightarrow (A \multimap D) \times (D \multimap B)$ given by $\phi(h, k) = (iF(k, h)f, gF(h, k)i^{-1})$. This is a continuous function because F is locally continuous; taking its least fixed point yields a pair of strict functions (h, k) with the required property. To see that this is the only such pair, suppose (h', k') is another and consider the function $\epsilon : (D \multimap D) \rightarrow (A \multimap D) \times (D \multimap B)$ defined by $\epsilon(\epsilon) = (\epsilon h', k' \epsilon)$. Clearly ϵ is continuous, and it is strict because k' is strict. Moreover the assumption that h' and k' satisfy (2) and (3), together with the functoriality of F , imply that

$$\phi \epsilon = \epsilon \delta \tag{4}$$

(where δ was defined in Definition 2.7). Since ϵ is strict, it follows from (4) that $\text{fix}(\phi) = \epsilon \text{fix}(\delta)$, where we use fix to denote the least fixed point operator. (Here we are using 'Plotkin's Axiom', i.e. the *uniformity* of least fixed points of continuous functions on cppo's; this property is easily established by fixed point induction—see Gunter and Scott [10, Theorem 2.3].) But by minimality of D , $\text{fix}(\delta) = id_{D \multimap D}$. Hence

$$(h, k) \stackrel{\text{def}}{=} \text{fix}(\phi) = \epsilon id = (h', k')$$

as required. \square

We now state our main result.

Theorem 2.9 *If \mathcal{R} is an admissible unitary relational structure for cppo's and $F : \mathcal{Cpo}_{-}^{\text{op}} \times \mathcal{Cpo}_{-} \rightarrow \mathcal{Cpo}_{-}$ is a locally continuous functor with an admissible action on \mathcal{R} -relations, then the minimal invariant for F , $i : F(D, D) \cong D$, satisfies the following rule for all $R^-, R^+ \in \mathcal{R}(D)$ with R^+ admissible:*

$$\frac{i^{-1} : R^- \subset F(R^+, R^-) \quad i : F(R^-, R^+) \subset R^+}{R^- \subset_{\mathcal{R}} I_D \subset_{\mathcal{R}} R^+} \tag{5}$$

(Recall from Definition 2.1 that the preorder $R \subset_{\mathcal{R}} R'$ holds if and only if $id_D : R \subset R'$.)

The theorem can be established as a special case of the following slightly more general result.

Proposition 2.10 *Let \mathcal{R} , F , D and i be as in the statement of Theorem 2.9. Given strict continuous functions $f : A \multimap F(B, A)$ and $g : F(A, B) \multimap B$,*

let $h : A \multimap D$ and $k : D \multimap B$ be the unique strict continuous functions defined from f and g as in Theorem 2.8. Then for any $R^- \in \mathcal{R}(A)$ and $R^+ \in \mathcal{R}(B)$ with R^+ admissible, if

$$f : R^- \subset F(R^+, R^-) \quad \text{and} \quad g : F(R^-, R^+) \subset R^+$$

then

$$h : R^- \subset I_D \quad \text{and} \quad k : I_D \subset R^+.$$

Proof Let E be the cppo $(A \multimap D) \times (D \multimap B)$. Recall from the proof of Theorem 2.8 that (h, k) is the least fixed point of the continuous function $\phi : E \rightarrow E$ given by $\phi(h', k') = (iF(k', h')f, gF(h', k')i^{-1})$. So we have to show that $\text{fix}(\phi)$ lies in the subset of E given by $[R^-, I_D] \times [I_D, R^+] = \{(h', k') \mid h' : R^- \subset I_D \text{ and } k' : I_D \subset R^+\}$. Since R^+ and I_D are admissible (the latter because \mathcal{R} is itself admissible), this subset contains $(-, -)$ and is closed under least upper bounds of countable chains. So it suffices to verify that $\phi : E \rightarrow E$ maps the subset into itself.

But if $h' : R^- \subset I_D$ and $k' : I_D \subset R^+$, then since I_D and R^+ are admissible, by property (a) of Definition 2.6

$$\begin{aligned} F(k', h') & : F(R^+, R^-) \subset F(I_D, I_D) \\ F(h', k') & : F(I_D, I_D) \subset F(R^-, R^+). \end{aligned}$$

By property (b) of that definition $F(I_D, I_D) = I_{F(D, D)}$, so that 2.1(c) yields

$$\begin{aligned} i & : F(I_D, I_D) \subset I_D \\ i^{-1} & : I_D \subset F(I_D, I_D). \end{aligned}$$

Using 2.1(b) to compose these relations with those given by hypothesis on f and g , we have $iF(k', h')f : R^- \subset I_D$ and $gF(h', k')i^{-1} : I_D \subset R^+$. Thus ϕ does indeed map the subset $[R^-, I_D] \times [I_D, R^+]$ into itself. \square

Theorem 2.9 is the special case of this proposition when $A = D = B$, $f = i^{-1}$, and $g = i$, in which case the uniquely determined h and k are both id_D .

There are two basic parameters that can be varied in Theorem 2.9: the particular relational structure \mathcal{R} , and the definition of the action of the various domain constructors on \mathcal{R} -relations. The next two sections show that suitable choices for these parameters yield general induction and co-induction principles for recursively defined domains as instances of the theorem.

Remark 2.11 Theorem 2.9 has a converse: if D is a cppo for which there is an isomorphism $i : F(D, D) \cong$

D satisfying (5) for any choice of \mathcal{R} and \mathcal{R} -action for F , then D is a minimal invariant for F . (The minimal invariant is unique up to isomorphism, because of Theorem 2.8.) For we can take \mathcal{R} as in Example 2.2(iii), with \mathcal{R} -relations on D given by strict continuous endofunctions of D . The preorder $R \subset_{\mathcal{R}} R'$ holds if and only if R and R' are equal endomorphisms. Moreover, all \mathcal{R} -relations are admissible, and any locally continuous F has an (admissible) \mathcal{R} -action given by the action on the functor of morphisms in \mathcal{Cpo}_- . So an isomorphism $i : F(D, D) \cong D$ satisfies (5) for this \mathcal{R} if and only if for all $e^-, e^+ : D \multimap D$, if $i^{-1}e^- = F(e^+, e^-)i^{-1}$ and $iF(e^-, e^+) = e^+i$, then $e^- = id_D = e^+$. Taking $e^- = e^+$, we have that id_D is the unique fixed point of the function $e \mapsto iF(e, e)i^{-1}$, and hence in particular is the least fixed point of that function. So D is a minimal invariant for F .

3 Induction

We show how to derive from Theorem 2.9 a family of induction principles for subsets of recursively defined cppo's. To do this we use the unitary relational structure for cppo's given in Example 2.2(i) in the case $n = 1$. Thus for each cppo D , $\mathcal{R}(D)$ consists of all subsets of D that contain $-$; $f : R \subset S$ holds just in case f maps R into S ; and the identity \mathcal{R} -relation I_D is the whole of D .

Definition 3.1 We endow the cppo constructors $-$, \times , \otimes , \oplus , \rightarrow , and \multimap with an action on \mathcal{R} -relations as follows. For $R \in \mathcal{R}(D)$ and $S \in \mathcal{R}(E)$ define $R_- \in \mathcal{R}(D_-)$, $R \times S \in \mathcal{R}(D \times E)$, $R \otimes S \in \mathcal{R}(D \otimes E)$, $R \oplus S \in \mathcal{R}(D \oplus E)$, $R \rightarrow S \in \mathcal{R}(D \rightarrow E)$, and $R \multimap S \in \mathcal{R}(D \multimap E)$, as follows.

$$\begin{aligned} R_- & \stackrel{def}{=} R \cup \{-\} \\ R \times S & \stackrel{def}{=} \{(x, y) \mid x \in R \text{ and } y \in S\} \\ R \otimes S & \stackrel{def}{=} \{(x, y) \mid - \neq x \in R \text{ and } \\ & \quad - \neq y \in S\} \cup \{-\} \\ R \oplus S & \stackrel{def}{=} \{\text{inl}(x) \mid - \neq x \in R\} \cup \\ & \quad \{\text{inr}(y) \mid - \neq y \in S\} \cup \{-\} \\ R \rightarrow S & \stackrel{def}{=} \{f \mid \forall x \in R \quad f(x) \in S\} \\ R \multimap S & \stackrel{def}{=} (R \rightarrow S) \cap (D \multimap E) \end{aligned}$$

Let $\Phi(\alpha)$ be a cppo-constructor built up from the variable α and constants ranging over cppo's, using $-$, \times , \otimes , \oplus , \rightarrow , and \multimap . Recall from the previous section that by separating the positive and negative

occurrences of α , one can associate with Φ a locally continuous functor $\hat{\Phi} : \mathcal{Cpo}_-^{\text{op}} \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$. We can use the above definitions to endow this functor with an admissible action on \mathcal{R} -relations in the sense of Definition 2.6. Recall from 2.5 that for this \mathcal{R} , $R \in \mathcal{R}(D)$ is admissible if and only if it is a chain-closed subset of D , and that \mathcal{R} is itself admissible. So we can apply Theorem 2.9 for this particular relational structure and with $F = \hat{\Phi}$. Note that since $R^- \subset_{\mathcal{R}} I_D$ holds for any R^- , the first half of the conclusion of the rule (5) tells us nothing in this case. This suggests that we take $R^- = I_D$. The second half of the conclusion tells us that the subset R^+ is the whole of D . Moreover the first part of the hypothesis of (5) (*viz.* $i^{-1} : R^- \subset \hat{\Phi}(R^+, R^-)$) is automatically satisfied when $R^- = I_D$, since it is easily verified that the \mathcal{R} -action of $\hat{\Phi}$ has the property that $\hat{\Phi}(R, I_E) = I_{\hat{\Phi}(D, E)}$, for all $R \in \mathcal{R}(D)$. So the content of Theorem 2.9 in this case is the following result:

Theorem 3.2 (Induction Property for cppo's) *Let \mathcal{R} and Φ be as above, and let $D = \text{reca}.\Phi(\alpha)$ be the cppo recursively defined by $\alpha = \Phi(\alpha)$, with associated isomorphism $i : \Phi(D) \cong D$. For any subset $R \subseteq D$ containing $-$, let $\hat{\Phi}(R) \subseteq \Phi(D)$ be $\hat{\Phi}(I_D, R)$. (Thus $\hat{\Phi}(R)$ can be defined directly by induction on the structure of Φ using Definition 3.1.) Then D satisfies the following rule of induction for any chain-complete subset $R \subseteq D$ containing $-$:*

$$\frac{\forall u \in \Phi(D)(u \in \hat{\Phi}(R) \Rightarrow i(u) \in R)}{\forall x \in D.x \in R}$$

Example 3.3 (Lazy Lambda Calculus) Let $\Phi(\alpha)$ be $(\alpha \rightarrow \alpha)_-$. In this case $D = \text{reca}.\Phi(\alpha)$ is the canonical model of the lazy lambda calculus studied by Abramsky and Ong [1, 2]. Let $x, y \mapsto x \cdot y$ denote the application function on D . Thus when $x = -$, $x \cdot y = -$; and when $x \neq -$, $x \cdot y = f(y)$ where $f \in (D \rightarrow D)$ is the unique element such that $x = i(f)$. If $R \in \mathcal{R}(D)$, then applying Definition 3.1, one has that $\hat{\Phi}(I_D, R)$ is the subset of $(D \rightarrow D)_-$ containing $-$ and all elements of the form f where $f \in (D \rightarrow D)$ satisfies that $f(y) \in R$ for all $y \in D$. Thus for each $x \in D$, $i^{-1}(x) \in \hat{\Phi}(I_D, R)$ if and only if $\forall y \in D(x \cdot y \in R)$. Note that $i : \hat{\Phi}(I_D, R) \subset R$ holds if and only if for all $x \in D$, $i^{-1}(x) \in \hat{\Phi}(I_D, R)$ implies $x \in R$. So Theorem 3.2 yields the following induction principle for chain-complete subsets $R \subseteq D$ containing $-$:

$$\frac{\forall x(\forall y(x \cdot y \in R) \Rightarrow x \in R)}{\forall x(x \in R)}$$

This principle looks like a kind of (impredicative!) ‘structural induction’ for D , in which a ‘function’ $x \in D$ is decomposed into its vector of values $(x \cdot y \mid y \in D)$. Pretty though it is, the author has yet to find useful applications of this principle.

Example 3.4 (Fixpoint induction) When $\Phi(\alpha) = \alpha_-$, $\text{reca}.\Phi(\alpha)$ is the ordinal $\omega + 1$, which is a *fixpoint object* for the lifting monad on the category of cppo's, in the sense of Crole and Pitts [3]. In this case Theorem 3.2 yields the induction principle which motivated the one for fixpoint objects of strong monads in general, studied in *loc. cit.* Scott's principle of induction for proving admissible properties of least fixed points of continuous functions is a formal consequence of this instance of the theorem.

Example 3.5 (Structural induction) If $\Phi(\alpha)$ is built up from α and *flat* cppo's (i.e. lifts of discretely ordered sets) just using \otimes and \oplus , then $\text{reca}.\Phi(\alpha) = X_-$ is also a flat cppo. Indeed, X is the initial algebra for an appropriate endofunctor on the category of sets and functions (the functor being built up using the set operations of cartesian product and disjoint union according to the structure of Φ). In this case the induction principle of Theorem 3.2 coincides with the principle of *initial algebra induction* studied by Lehmann and Smyth [13, Section 5.2]. As is well known, for various choices of such Φ , X yields inductively defined sets of numbers, lists, trees, *etc.*, and initial algebra induction coincides with the corresponding principle of structural induction.

Remark 3.6 Jensen [12] also derives a family of induction principles for recursively defined domains starting from their minimal invariance property. Our Induction Property differs from Jensen's in a couple of respects. First, although the proof of 3.2 ultimately depends on properties of continuous endomorphisms of the recursively defined domain, the induction hypothesis is stated purely in terms of subsets of the domain, whereas Jensen's principle in general contains hypotheses still involving endomorphisms. Secondly, for the problematic case of a constructor $\Phi(\alpha)$ involving both positive and negative occurrences of α , such as Example 3.3, Jensen's principle can be vacuous (in the sense that the conclusion occurs as part of the induction hypothesis) when 3.2 is not. Nevertheless, 3.2 does not yield useful information about $\text{reca}.\Phi(\alpha)$ in all cases. For example, when $\Phi(\alpha)$ is $\alpha \rightarrow K$ (with K some fixed cppo) then $\hat{\Phi}(I_D, R) = I_{D \rightarrow K}$; so the hypothesis $i : \hat{\Phi}(I_D, R) \subset R$ is just $i : I_{D \rightarrow K} \subset R$, which is the same as $R = D$, since i is an isomorphism. So the induction principle is vacuous in this case.

4 Co-induction

In this section we indicate briefly how to derive a co-inductive property for recursively defined domains from Theorem 2.9. The co-induction principle for recursively defined cppo's established in [22] is an instance of this property. We use the unitary relational structure for cppo's given by the $n = 2$ case of Example 2.2(ii). Thus for each cppo D , $\mathcal{R}(D)$ consists of all binary relations on D , i.e. subsets of $D \times D$; $f : R \subset S$ holds if and only if for all $(x_1, x_2) \in R$, $(f(x_1), f(x_2)) \in S$; and I_D is the order relation on D , \sqsubseteq_D .

Definition 4.1 Given $R \in \mathcal{R}(D)$ and $S \in \mathcal{R}(E)$, define $R_- \in \mathcal{R}(D_-)$, $R \times S \in \mathcal{R}(D \times E)$, $R \otimes S \in \mathcal{R}(D \otimes E)$, $R \oplus S \in \mathcal{R}(D \oplus E)$, $R \rightarrow S \in \mathcal{R}(D \rightarrow E)$, and $R \multimap S \in \mathcal{R}(D \multimap E)$ as follows:

- (a) $(u_1, u_2) \in R_-$ if and only if $u_1 \neq -$ implies $u_2 \neq -$ and $(u_1, u_2) \in R$.
- (b) $((x_1, y_1), (x_2, y_2)) \in R \times S$ if and only if $(x_1, x_2) \in R$ and $(y_1, y_2) \in S$.
- (c) $(u_1, u_2) \in R \otimes S$ if and only if whenever $u_1 = (x_1, y_1)$ for some $x_1 \neq -$ and $y_1 \neq -$, then $u_2 = (x_2, y_2)$ for some $x_2 \neq -$ and $y_2 \neq -$ with $(x_1, x_2) \in R$ and $(y_1, y_2) \in S$.
- (d) $(u_1, u_2) \in R \oplus S$ if and only if both
 - if $u_1 = \text{inl}(x_1)$ for some $x_1 \neq -$, then $u_2 = \text{inl}(x_2)$ for some $x_2 \neq -$ with $(x_1, x_2) \in R$; and
 - if $u_1 = \text{inr}(y_1)$ for some $y_1 \neq -$, then $u_2 = \text{inr}(y_2)$ for some $y_2 \neq -$ with $(y_1, y_2) \in S$.
- (e) $(f_1, f_2) \in R \rightarrow S$ if and only if for all $x \in D$, $(f_1(x), f_2(x)) \in S$.
- (f) $(f_1, f_2) \in R \multimap S$ if and only if for all $x \neq -$, $(f_1(x), f_2(x)) \in S$.

If $\Phi(\alpha)$ is a cppo-constructor built up from the variable α and constants ranging over cppo's, using $-$, \times , \otimes , \oplus , \rightarrow , and \multimap , then we can use the above definitions to endow the associated locally continuous functor $\hat{\Phi} : \mathcal{Cppo}^{\text{op}} \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$ with an admissible action on \mathcal{R} -relations in the sense of Definition 2.6. In this case, the admissibility restriction in part (a) of that definition is needed because of the properties of $R, S \mapsto R \multimap S$ as defined in clause (f) above. (Recall from 2.5 that for this \mathcal{R} , $R \in \mathcal{R}(D)$ is admissible if and only if it is a chain-closed subset of D , and that \mathcal{R} is

itself admissible.) So we can apply Theorem 2.9 for this particular relational structure and with $F = \hat{\Phi}$. In fact, we consider the special case when the admissible \mathcal{R} -relation R^+ is just I_D . So the second part of the conclusion of (5) (*viz.* $I_D \subset_{\mathcal{R}} R^+$) is automatic. But in fact in this case the second half of the hypothesis (*viz.* $i : \hat{\Phi}(R^-, R^+) \subset R^+$) is also automatic, since it is easily verified that the \mathcal{R} -action of $\hat{\Phi}$ has the property that $\hat{\Phi}(R, I_E) = I_{\hat{\Phi}(D, E)}$, for all $R \in \mathcal{R}(D)$. (Ensuring this property of the action on relations was the reason for choosing the given definitions of $R \rightarrow S$ and $R \multimap S$ which throw away the relation R , rather than choosing to relate functions that send R -related arguments to S -related results.)

So the content of Theorem 2.9 in this case is the following result, where given $R \in \mathcal{R}(D)$ we write $\Phi(R)$ for $\hat{\Phi}(I_D, R) \in \mathcal{R}(\hat{\Phi}(D, D)) = \mathcal{R}(\Phi(D))$. (The 'only if' part of the theorem follows from the fact that $R = I_D$ satisfies (6).)

Theorem 4.2 (Co-induction Property) *Let $\Phi(\alpha)$ be a cppo constructor as above, and let $D = \text{reca}.\Phi(\alpha)$ be the corresponding recursively defined cppo with associated isomorphism $i : \Phi(D) \cong D$. Then for all $x, x' \in D$, $x \sqsubseteq_D x'$ if (and only if) $(x, x') \in R$ for some subset $R \subseteq D \times D$ satisfying:*

$$\forall (y, y') \in R. (i^{-1}(y), i^{-1}(y')) \in \Phi(R) \quad (6)$$

The co-induction principle for recursively defined cppo's established in [22, Theorem 2.5] can be deduced from this theorem by restricting attention to cppo constructors just involving the constructions $(-)_-$, $(-) \otimes (-)$, $(-) \oplus (-)$, and $((-) \multimap (-))_-$. As shown in *loc. cit.*, binary relations satisfying (6) give a uniform notion of 'simulation' (for the cppo-constructors considered), which includes for example (one-sided) *applicative bisimulation*, used by Abramsky [1] in connection with the lazy lambda calculus. The existence of a simulation containing two elements of a recursive cppo establishes that the order relation holds between them. There is a more symmetric notion of 'bisimulation' that establishes directly that two elements are equal. This can be obtained by changing the identity \mathcal{R} -relations of the relational structure used in this section from $I_D = \sqsubseteq_D$ to equality relations $I_D = \{(x, x) \mid x \in D\}$. In order to retain the necessary property $\hat{\Phi}(R, I_E) = I_{\hat{\Phi}(D, E)}$ one has to 'symmetrize' the action of cppo-constructors on \mathcal{R} -relations in Definition 4.1. For example R_- would now contain (u_1, u_2) if and only if

$$\begin{aligned} u_1 \neq - \text{ implies } u_2 \neq - \text{ and } (u_1, u_2) \in R, \text{ and} \\ u_2 \neq - \text{ implies } u_1 \neq - \text{ and } (u_1, u_2) \in R. \end{aligned}$$

With these changes Theorem 4.2 remains valid as stated except that $x \sqsubseteq_D x'$ is replaced by $x = x'$.

We refer the reader to [22], and to [27, 4] for further discussion and applications of this kind of co-induction principle.

5 Parameterized recursive domains

The results in this paper exploit the fact that various simple domain constructors have well-behaved actions on relations. The solution of recursive domain equations with parameters provides an important source of more complicated domain constructors. Thus if $\Psi(\alpha, \beta)$ is built up from variables α and β using $-$, \times , \otimes , \oplus , \rightarrow , and \multimap , we get a unary domain constructor

$$\Phi(\alpha) \stackrel{\text{def}}{=} \text{rec}\beta.\Psi(\alpha, \beta)$$

whose value $\Phi(D)$ at a cppo D is the minimal solution of the domain equation $\beta = \Psi(D, \beta)$. It is well known that such a Φ determines a locally continuous functor $\mathcal{Cpo}_-^{\text{op}} \times \mathcal{Cpo}_- \rightarrow \mathcal{Cpo}_-$. So to apply Theorem 2.9 to derive reasoning principles for recursively defined cppo's involving such constructors, one has to show how the action of $\Psi(\alpha, \beta)$ on pairs of relations gives rise to an action of $\text{rec}\beta.\Psi(\alpha, \beta)$ on single relations.

Given $R \in \mathcal{R}(D)$, how do we define $\text{rec}\beta.\Psi(R, \beta) \in \mathcal{R}(\Phi(D))$? Since β may occur both positively and negatively in Ψ , it is not in general the case that $S \mapsto \Psi(R, S)$ is monotone for the inclusion preorder $\subset_{\mathcal{R}}$ between \mathcal{R} -relations. Therefore the relation cannot be given simply by an inductive definition. A traditional solution to this problem is to resort to the detailed construction of the recursively defined domain $\text{rec}\beta.\Psi(D, \beta)$ as a colimit of a chain of embedding-projections; the relation $\text{rec}\beta.\Psi(R, \beta)$ can be built up as an inverse limit simultaneously with the construction of the recursively defined domain (see [24], [18, Chapter 5]). The methods employed in the proof of Proposition 2.10 can be extended to yield an alternative, more synthetic construction of $\text{rec}\beta.\Psi(R, \beta)$, as follows.

First one separates positive and negative occurrences of β in Ψ , obtaining $\hat{\Psi}(\alpha, \beta^-, \beta^+)$. The admissible action of Ψ yields a mapping

$$(S^-, S^+) \mapsto (\hat{\Psi}(R, S^+, S^-), \hat{\Psi}(R, S^-, S^+)) \quad (7)$$

Assuming \mathcal{R} -relations can be transported along isomorphisms (as is the case for the relational structures considered in this paper), (7) gives rise to a monotone operator on $\mathcal{R}(\Phi(D))^{\text{op}} \times \mathcal{R}^{\text{adm}}(\Phi(D))$,

where \mathcal{R}^{adm} denotes the subcollection of admissible \mathcal{R} -relations. For sufficiently complete relational structures $\mathcal{R}(\Phi(D))$, and hence also $\mathcal{R}(\Phi(D))^{\text{op}} \times \mathcal{R}^{\text{adm}}(\Phi(D))$, will be (equivalent to) a complete lattice. (This is the case for the examples in 2.2, which are all closed under taking set-theoretic intersection.) Therefore one can form the least fixed point of the monotone operator, obtaining a pair of \mathcal{R} -relations (Δ^-, Δ^+) with Δ^+ admissible.

Next, one shows that Δ^- and Δ^+ are equal (or, more precisely, are equivalent elements of the preorder $\mathcal{R}(\Phi(D))$). In fact $\Delta^+ \subset_{\mathcal{R}} \Delta^-$ follows easily from the definition of (Δ^-, Δ^+) and the symmetry of the operator (7). The converse, i.e. that $\text{id}_{\Phi(D)} : \Delta^- \subset \Delta^+$, can be established much as in the proof of Proposition 2.10: by the minimal invariance property (Definition 2.7), $\text{id}_{\Phi(D)}$ is the least fixed point of a continuous operator δ ; but δ can be shown to map the admissible subset $\{f \mid f : \Delta^- \subset \Delta^+\}$ into itself; so $\text{id}_{\Phi(D)}$ is contained in the subset.

We define $\text{rec}\beta.\Psi(R, \beta) \stackrel{\text{def}}{=} \Delta^- = \Delta^+$. The definition of (Δ^-, Δ^+) as a least fixed point implies the following properties for $\text{rec}\beta.\Psi(R, \beta)$, where $i : \Psi(D, \Phi(D)) \cong \Phi(D)$ is the isomorphism exhibiting $\Phi(D)$ as the minimal solution of the domain equation $\beta = \Psi(D, \beta)$:

- (a) $i^{-1} : \text{rec}\beta.\Psi(R, \beta) \subset \Psi(R, \text{rec}\beta.\Psi(R, \beta))$ and $i : \Psi(R, \text{rec}\beta.\Psi(R, \beta)) \subset \text{rec}\beta.\Psi(R, \beta)$;
- (b) for all $S^-, S^+ \in \mathcal{R}(\Phi(D))$ with S^+ admissible, if

$$\begin{aligned} i^{-1} & : S^- \subset \hat{\Psi}(R, S^+, S^-) \\ i & : \hat{\Psi}(R, S^-, S^+) \subset S^+, \end{aligned}$$

then

$$S^- \subset_{\mathcal{R}} \text{rec}\beta.\Psi(R, \beta) \subset_{\mathcal{R}} S^+.$$

These properties can be used to deduce that $R \mapsto \text{rec}\beta.\Psi(R, \beta)$ determines an admissible action on relations in the sense of Definition 2.6.

In fact Theorem 2.9 can be deduced from the above construction. For property (b) implies that $\text{rec}\beta.\Psi(R, \beta)$ is the unique \mathcal{R} -relation with the fixed point property (a). So when $R = I_D$, since $I_{\Phi(D)}$ has this fixed point property, we must have $\text{rec}\beta.\Psi(I_D, \beta) = I_{\Phi(D)}$, and then the rule (5) is just (b).

The method of constructing relations on recursively defined domains which we have outlined in this section can be applied to give simplified proofs of ‘computational adequacy’ of denotational semantics for functional languages with recursive datatypes. Such results involve showing that a language expression evaluates to canonical form if (and only if) its denotation

is not $-$. This can be done by constructing a suitable ‘computability relation’ between domain elements and language expressions. The properties required of the computability relation involve both positive and negative occurrences of the relation, making the demonstration of its existence the hard part of the proof of computational adequacy. In fact, for a suitable choice of relational structure \mathcal{R} , the requirements on the computability relation are just those of the fixed point property (a) (for a suitable choice of Ψ). So it can be constructed using the above techniques—separation of variables, simultaneous inductive definition of positive and negative versions of the relation, followed by a proof by fixed point induction that the two versions actually coincide. The details of this application of the methods in this paper to proofs of computational adequacy will be given elsewhere.

6 Conclusion

The results in this paper show that a number of quite powerful families of reasoning principles for the denotational semantics of recursive datatypes in functional languages can be derived from some simple properties of abstract relations on domains. The reasoning principles apply to general forms of recursive declaration with unrestricted positive and negative occurrences of the defined type. That being said, for some ‘problematic’ recursive definitions, a particular reasoning principle may be hard to apply (see 3.3) or even vacuous (see 3.6).

With a view to implementing these proof principles in mechanized proof assistants, note that the principle for $\text{rec}\alpha.\Phi(\alpha)$ in a particular family can be generated automatically according to the structure of $\Phi(\alpha)$. Moreover, the statement of the principles is independent of any detailed construction of recursively defined domains within set theory or higher order logic. Indeed the proof of Theorem 2.9 ultimately relies upon two quite elementary facts: Scott’s minimal invariance property of recursively defined domains (Definition 2.7), and the uniformity property for least fixed points of continuous functions (‘Plotkin’s Axiom’—see [10, section 2.3]), from which Freyd’s ‘algebraic compactness’ property for recursively defined domains (Theorem 2.8) is derived.

We have not considered recursively defined domains involving powerdomains in this paper. [22, Section 5] extends the co-induction principle (Theorem 4.2) to recursively defined domains involving the convex powerdomain construction, $P^{\natural}(-)$. To deduce this extended principle from Theorem 2.9 one would have

to restrict the relational structure used in section 4 to *bifinite* domains D , for which there is a construction of $P^{\natural}(D)$ that is sufficiently concrete to permit a well-behaved action of $P^{\natural}(-)$ on binary relations to be defined (see [22, Lemma 5.2]). It would be interesting to investigate whether the induction principle of Theorem 3.2 can be extended to include use of $P^{\natural}(-)$. (The admissibility considerations which are present in the induction principle make this a more complicated problem than is the case for co-induction.)

A more subtle question is whether the results in this paper can be generalized by replacing the role of lifting and the category \mathcal{Cpo}_- by an arbitrary *strong monad* and its associated category of algebras—thereby fitting these results into Moggi’s [17] monadic approach to denotational semantics and the associated approach to program logic introduced by the author in [21]. Simpson [29] studies the key properties of minimal invariance and uniformity of fixpoints in this setting, and their relationship to the notion of a *fixpoint object* for the monad in the sense of Crole and Pitts [3] (see also Mulry [19]). His results require a commutativity condition to hold of the monad. Without that condition the analogue of Freyd’s Theorem 2.8 loses its usefulness, simply because some type constructors (in particular, the product constructor) may fail to be functorial on the Kleisli category of the monad. Unfortunately many monads that one would like to consider (such as those for notions of computation involving side-effects on mutable state) are not commutative. It remains a topic of future research to see the extent to which inductive and co-inductive relational properties can be developed for datatypes involving such monads, and whether the Evaluation Logic of [21] can be extended to encompass formal versions of these principles.

References

- [1] S. Abramsky, *The Lazy Lambda Calculus*. In: D. Turner (ed.), *Research Topics in Functional Programming* (Addison-Wesley, 1990), pp 65–116.
- [2] S. Abramsky and C.-H. L. Ong, *Full Abstraction in the Lazy Lambda Calculus*, Information and Computation, *to appear*.
- [3] R. L. Crole and A. M. Pitts, *New foundations for fixpoint computations: FIX-hyperdoctrines and the FIX-logic*, Information and Computation 98(1992) 171–210.

- [4] M. P. Fiore, *A Coinduction Principle for Recursive Datatypes Based on Bisimulation*. In this volume.
- [5] M. P. Fourman, P. T. Johnstone and A. M. Pitts (eds), *Applications of Categories in Computer Science*, L.M.S. Lecture Note Series 177 (Cambridge University Press, 1992).
- [6] P. J. Freyd, *Recursive Types Reduced to Inductive Types*. In: *Proc. 5th LICS Symp., Philadelphia* (IEEE Computer Society Press, Washington, 1990), pp 498–508.
- [7] P. J. Freyd, *Algebraically Complete Categories*. In: A. Carboni *et al* (eds), *Proc. 1990 Como Category Theory Conference*, Lec. Notes in Math. Vol. 1488 (Springer-Verlag, Berlin, 1991), pp 95–104.
- [8] P. J. Freyd, *Remarks on algebraically compact categories*. In [5], pp 95–106.
- [9] M. Gordon, R. Milner and C. P. Wadsworth, *Edinburgh LCF*, Lecture Notes in Computer Science, Vol. 78 (Springer-Verlag, Berlin, 1979).
- [10] C. A. Gunter and D. S. Scott, *Semantic Domains*. In: J. van Leeuwen (ed.), *Handbook of Theoretical Computer Science* (Elsevier Science Publishers B.V., 1990), pp 634–674.
- [11] P. Hudak, S. Peyton Jones and P. Wadler (eds), *Report on the Programming Language Haskell: Version 1.1*, Technical Report, Yale Univ. and Glasgow Univ., August 1991.
- [12] F. V. Jensen, *Inductive Inference in Reflexive Domains*, Univ. Edinburgh Dept. Computer Science Report No. CSR 86-81, May 1981.
- [13] D. J. Lehmann and M. B. Smyth, *Algebraic specification of datatypes: a synthetic approach*, Math. Systems Theory 14(1981) 97–139.
- [14] S. MacLane, *Categories for the Working Mathematician* (Springer-Verlag, New York, 1971).
- [15] R. E. Milne, *The formal semantics of computer languages and their implementations*, Ph.D. Thesis, Univ. Cambridge, 1973.
- [16] R. Milner, M. Tofte and R. Harper, *The Definition of Standard ML* (MIT Press, 1990).
- [17] E. Moggi, *Notions of Computation and Monads*, Information and Computation 93(1991) 55–92.
- [18] K. Mulmuley, *Full abstraction and semantic equivalence* (MIT Press, Cambridge Mass., 1987).
- [19] P. S. Mulry, *Strong monads, algebras and fixed points*. In [5], pp 202–216.
- [20] P. W. O’Hearn and R. D. Tennent, *Relational Parametricity and Local Variables*. In: *Conf. Record 20th Symp. on Principles of Programming Languages*, Charleston, 1993 (ACM, New York, 1993), pp 171–184.
- [21] A. M. Pitts, *Evaluation Logic*. In: G. Birtwistle (ed.), *IVth Higher Order Workshop, Banff 1990*, Workshops in Computing (Springer-Verlag, Berlin, 1991), pp 162–189.
- [22] A. M. Pitts, *A Co-induction Principle for Recursively Defined Domains*, Theoretical Computer Science, *to appear*. (Available as Univ. Cambridge Computer Laboratory Tech. Rept. No. 252, April 1992.)
- [23] G. D. Plotkin, *Lambda definability and logical relations*, Memorandum SAI-RM-4, Univ. Edinburgh School of Artificial Intelligence, October 1973.
- [24] G. D. Plotkin, *Lectures on Predomains and Partial Functions*. Notes for a course at CSLI, Stanford University, 1985.
- [25] J. C. Reynolds, *On the relation between direct and continuation semantics*. In: J. Loeckx (ed.), *Proc. 2nd Int. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 14 (Springer, Berlin, 1974), pp 141–156.
- [26] J. C. Reynolds, *Types, Abstraction and Parametric Polymorphism*. In: R. E. A. Mason (ed.), *Information Processing 83* (Elsevier Science Publishers B.V., Amsterdam, 1983), pp 513–523.
- [27] J. J. M. M. Rutten, *A Structural Co-induction Theorem*. In: *Proc. Math. Foundations of Programming Language Semantics*, New Orleans, 1993.
- [28] D. S. Scott, *Domains for Denotational Semantics*. In M. Nielsen and E. M. Schmidt (eds), *Proc. 9th Int. Coll. on Automata, Languages and Programming*, Lecture Notes in Computer Science, Vol. 140 (Springer, Berlin, 1982), pp 577–613.
- [29] A. K. Simpson, *Recursive Types in Kleisli Categories*, preprint, University of Edinburgh Department of Computer Science, July 1992.

- [30] M. B. Smyth and G. D. Plotkin, *The Category-Theoretic Solution of Recursive Domain Equations*, SIAM J. Computing 11(1982) 761–783.
- [31] S. Thompson, *A Logic for Miranda*, Formal Aspects of Computing 1(1989) 339–365.