Technical Report

UCAM-CL-TR-51
ISSN 1476-2986

Number 51

UNIVERSITY OF CAMBRIDGE
Computer Laboratory

Using information systems to solve recursive domain equations effectively

Glynn Winskel, Kim Guldstrand Larsen

July 1984

15 JJ Thomson Avenue
Cambridge CB3 0FD
United Kingdom
phone +44 1223 763500

http://www.cl.cam.ac.uk/

Technical reports published by the University of Cambridge
Computer Laboratory are freely available via the Internet:

http://www.cl.cam.ac.uk/techreports/

ISSN 1476-2986

USING INFORMATION SYSTEMS TO SOLVE RECURSIVE DOMAIN EQUATIONS EFFECTIVELY

Kim Guldstrand Larsen
Department of Computer Science,
University of Edinburgh,
King's Buildings,
Mayfield Road,
Edinburgh.

Glynn Winskel
Computer Laboratory,
University of Cambridge,
Corn Exchange Street,
Cambridge.

Abstract.

This paper aims to make two main contributions. One is to show how to use the concrete nature of Scott's information systems to advantage in solving recursive domain equations. The method is based on the substructure relation between information systems. This essentially makes a complete partial order (cpo) of information systems. Standard domain constructions like function space can be made continuous on this cpo so the solution of recursive domain equations reduces to the more familiar construction of forming the least fixed-point of a continuous function. The second contribution again relies on the concrete nature of information systems, this time to develop a basic theory of effectively given information systems and through this present a simple treatment of effectively given domains.

0. Introduction.

The mathematical theory of semantics of programming languages founded by Dana Scott and Christopher Strachey has been based on partial orders of information. To give the idea, in a simple imperative language a command might be denoted by a partial function from a set of input states to a set of output states. When a function is more defined we think of it as having more information because because it tells us more about the input/output behaviour. In general of course the the partial orders of information can be considerably more complicated, reflecting more intricate types of information, like for instance that associated with a procedure which takes procedures as arguments. But the central idea is still the same ; increasing in the partial order means increasing information. Just as one thinks of computing an integer or a final state so can one think of computing an element in these more abstract partial orders of information. And just as one has computations from input states to output states one can have computations which from input in one partial order of information compute output in another.

Not all partial orders make sense when these intuitions are refined and not surprisingly to get the theory to work people had to narrow down to more specific kinds of partial order. The most general partial orders in use are *complete partial orders* (abbreviated to cpos). A cpo has a least element $\bot$, called "bottom", which stands for null information and satisfies a completeness axiom which we try to motivate. Imagine computing an element of information like a partial function. The information need not come in one indivisible lump but may instead be presented as an increasing chain of elements of information, with information accumulating as time goes on. In a cpo the accumulation of this information, even over infinite time, is represented by an element of the partial order too. Formally, a cpo must satisfy the condition that if $x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$ is an increasing ω–chain then it has a least upper bound (lub) $\bigsqcup_{n\in\omega} x_n$ in the partial order.

Thus computing a value in a cpo is associated with an increasing chain of elements. What if we want to compute output information from input information in this general setting? We would naturally expect to model this as a function from one cpo to another. However not all functions would be feasible from a computational viewpoint. To be feasible a function $f : D \to E$ from one cpo to another should be *continuous* in the following sense. Suppose information x in D is delivered in the form of a chain $x_0 \sqsubseteq x_1 \sqsubseteq \cdots \sqsubseteq x_n \sqsubseteq \cdots$, so $x = \bigsqcup_{n\in\omega} x_n$. More information as input should yield more information as output; in other words the function f should be *monotonic i.e.* $y \sqsubseteq z$ implies $f(y) \sqsubseteq f(z)$. Thus as output we expect $f(x_0) \sqsubseteq f(x_1) \sqsubseteq \cdots \sqsubseteq f(x_n) \sqsubseteq \cdots$ and ultimately $f(x)$. By monotonicity $\bigsqcup_{n\in\omega} f(x_n) \sqsubseteq f(x) = f(\bigsqcup_{n\in\omega} x_n)$. To be continuous f should satisfy $\bigsqcup_{n\in\omega} f(x_n) = f(\bigsqcup_{n\in\omega} x_n)$. Intuitively the ultimate output value should be no more than the limit of the values determined at finite stages in delivering the input, so we can approximate the ultimate output value arbitrarily closely by the output values at finite stages.

An important property of a continuous function $f : D \to D$ on a cpo is that it has a *least fixed point* given in a simple way. It is given by $\mathrm{fix} f = \bigsqcup_{n\in\omega} f^n(\bot)$. That this gives a fixed point follows directly from continuity because $f(\mathrm{fix} f) = f(\bigsqcup_{n\in\omega} f^n(\bot)) =$

$\bigsqcup_{n\in\omega} f^{n+1}(\bot) = \mathrm{fix}f$. That it is the least follows simply too. Assume x is another fixed point, so $f(x) = x$. Then $\bot \sqsubseteq x$ and inductively $f^n(\bot) \sqsubseteq x$ for all n, using monotonicity. Thus from its definition $\mathrm{fix}f \sqsubseteq x$. (In our work we shall need this result not just for cpos where the partial is over a set but also for *large* cpos which are over a class. However the proof is the same.)

Cpos and continuous functions provide a category in which one can do a great deal of denotational semantics but, in general, cpos are much too crude to support a theory of computability. Before we can talk about a computable element of a cpo it must first be presented in an effective way. Most of the cpos which arise in practice are *ω–algebraic*, which means that they have a countable basis of *finite* (or isolated, or compact) elements, and it is via an enumeration of this basis that a concept of computable element can be developed. The definition of finite element rests on the concept of a *directed* subset of a cpo. A subset S of a cpo D is said to be *directed* iff it is non–null and satisfies $\forall s, t \in S \exists u \in S.\ s \sqsubseteq u\ \&\ t \sqsubseteq u$. An element x of a cpo D is *finite* iff $x \sqsubseteq \bigsqcup S \Rightarrow \exists s \in S.\ x \sqsubseteq s$, for all directed sets S. Intuitively a finite element is a element of information which if it is realised—used or produced—in a computation is realised in finite time. The set of finite elements of a cpo D is written as D^0. A cpo D is *algebraic* iff every directed set has a lub and for all elements x the set $\{e \sqsubseteq x \mid e \in D^0\}$ is directed with lub x; it is said to be ω–algebraic if D^0 is countable. With respect to an enumeration of the countable basis, we say an element is *computable* if it is the lub of a recursively enumerable set of finite elements. (There are theories of computability for cpos which are not algebraic—see *e.g.* [Sm]—but algebraic cpos seem to do when information is discrete; algebraicity can fail for the cpos of probability distributions in [S–D].)

The category of algebraic cpos has many nice properties but unfortunately lacks one essential characteristic; algebraic cpos with continuous functions do not form a cartesian–closed category because the function space of two algebraic cpos need not be algebraic. As the function–space construction is used again and again in denotational semantics this might be a serious drawback. Fortunately there are cartesian–closed (full) subcategories of algebraic cpos, two of which are widely used.

One is the category of SFP objects and we shall say more on this in the conclusion. The other, the one we shall be most concerned with here, is that of *consistently complete* algebraic cpos with continuous functions. A cpo is consistently complete iff every compatible set has a least upper bound. For X a subset of a partial order, we shall write $X\uparrow$ to mean X is a compatible subset *i.e.* $\exists y \forall x \in X.\ x \sqsubseteq y$. Consistently complete algebraic cpos are often called *domains* and we shall write **Dom** for the category of domains and ω–**Dom** for the subcategory of domains with a countable basis. (The reader is warned that "domain" is sometimes used very loosely and can refer to just a cpo.)

Domains have an appealing and suggestive representation as information systems introduced by Dana Scott in [S]. An information system can be viewed as prescription, or program, saying how to build a domain. In more detail an information system consists of a set of tokens, to be thought of as assertions, or propositions, one might make about a

computation, which are related by entailment and consistency relations. So information systems are well known and understood. What is novel is their tie–up with domains. An information system determines a domain with elements those sets of tokens which are consistent and closed with respect to the entailment relation; the ordering is just set inclusion. Through them the elements of a domain are seen as the logically closed and consistent sets of assertions. The information associated with an a computation is determined by the type of assertions one chooses to make about it, so previously abstract domain constructions can now be viewed as constructions on the logical apparatus used to describe computations.

There are technical advantages to working with information systems rather than directly with domains. As pointed out in [S] properties of domains can be derived rather than postulated and the representation makes them more amenable. In this paper we are specifically concerned with solving domain equations using the representation of information systems. Because information systems are based concretely on sets and relations they can be ordered to form a cpo. All the usual domain constructions have their counterparts as continuous operations on information systems. Recursive domain equations can then be solved using just the simple results we have given here for finding the least fixed point of a continuous function. The treatment is elementary, goes over readily to a theory of computability in information systems, and the results translate over to domains.

This paper owes much to Dana Scott's work. It is hoped that it will serve as a useful companion to the introduction to information systems in [S]. However note we shall give definitions which are slightly different from those given in [S] stemming from the fact that we do not assume that the token sets always contain a distinguished element Δ (standing for the always true assertion). In all cases the domains associated with constructions here and those of the same name in [S] will be isomorphic.

1. Information systems.

An information system consists of a set of tokens, a consistency predicate and an entailment relation. Tokens are the units of information which may be valid of a computation. Those tokens which are taken to be relevant to a computation can vary according to which aspects one wishes to capture. They might be parts of bit patterns, at a concrete level, or the input–output pairs of a function being computed at a more abstract level.

Tokens are in logical relationships with each other. In general not all subsets of tokens will be consistent—the validity of some may exclude the validity of others. For example a program computing a function on the integers cannot output two different numbers for the same input. We express the consistency of a set of tokens through a consistency predicate on finite subsets of tokens. And sometimes sets of tokens may entail others. For instance two tokens will entail a third if this represents their conjunction. We express this aspect through an entailment relation between sets of tokens.

countable

1.1 Definition. An *information system* is defined to be a structure $\mathbf{A} = (A, \mathrm{Con}, \vdash)$, where A is a set (the *tokens*), Con is a non-null subset of $\mathrm{Fin}(A)$ (the *consistent* sets) and $\vdash$ is a subset of $\mathrm{Con} \times A$ (the *entailment relation*) which satisfy:

(i) $X \subseteq Y \in \mathrm{Con} \Rightarrow X \in \mathrm{Con}$
(ii) $a \in A \Rightarrow \{a\} \in \mathrm{Con}$
(iii) $X \vdash a \Rightarrow X \cup \{a\} \in \mathrm{Con}$
(iv) $X \in \mathrm{Con} \ \ \& \ \ a \in X \Rightarrow X \vdash a$
(v) $(X, Y \in \mathrm{Con} \ \ \& \ \ \forall b \in Y.\ X \vdash b \ \ \& \ \ Y \vdash c) \Rightarrow X \vdash c.$

Remark. Note, that unlike [S], we do not assume that the token sets always contain a distinguished element Δ standing for the always true assertion.

An information system determines a family of subsets of tokens, called its *elements*. Think of the tokens as assertions about computations—assume that a token which is once true of a computation remains true of it. Intuitively an element of an information system is the set of tokens that can be truthfully asserted about a possible computation. This set of tokens can be viewed as the information content of the computation. As such the tokens should not contradict each other—they should be consistent—and should be closed under entailment. Of course this is once it has been decided to view the computation at the level of detail of the information system; in general there may be more than one information system used to view a computation corresponding to the different levels of abstraction involved.

1.2 Definition. The *elements*, $|\mathbf{A}|$, of an information system $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ are those subsets x of A which are

(i) *finitely consistent:* $X \subseteq^{fin} x \Rightarrow X \in \mathrm{Con}$
(ii) $\vdash$*–closed:* $X \subseteq x \ \ \& \ \ X \vdash a \Rightarrow a \in x.$

1.3 Lemma. *Let $\mathbb{A} = (A, Con, \vdash)$ be an information system. Suppose $X \in Con$ and let Y be a finite subset of A.*

(i) If $X \vdash b$ for every $b \in Y$ then $X \cup Y \in Con$ and $Y \in Con$.

(ii) The set $\overline{X} = \{a \in A \mid X \vdash a\}$ is an element of $\mathbb{A}$.

Proof. (i) Suppose $X \vdash b$ for every $b \in Y$. We show $X \cup Y \in$ Con and $Y \in$ Con by a simple induction on the size of Y. Clearly it holds when Y is null. Suppose Y is non–null, containing a token b', and $X \vdash b$ for all $b \in Y$. Then $X \vdash b$ for all $b \in Y \setminus \{b'\}$ so by induction $X \cup (Y \setminus \{b'\}) \in$ Con. By (iv) and (v), $X \cup (Y \setminus \{b'\}) \vdash b'$. By (iii), $X \cup Y \in$ Con. By (i), $Y \in$ Con too.

(ii) It follows from (i) that $\overline{X} = \{a \mid X \vdash a\}$ is consistent in the sense of 1.2(i). It is $\vdash$–closed because if $Y \subseteq \{a \mid X \vdash a\}$ and $Y \vdash a'$ then $X \vdash a'$ by axiom (v) in the definition of information systems. ∎

1.4 Notation. The entailment relation, between consistent sets and tokens, extends in a natural way to a relation between consistent sets. Let X and Y be consistent sets of an information system $(A, \mathrm{Con}, \vdash)$. We write $X \vdash Y$ as an abbreviation for $\forall a \in Y. X \vdash a$.

For X any subset of the tokens of an information system write
$\overline{X} =_{def} \{a \mid \exists Z \subseteq X. Z \vdash a\}$.

Thus an information system determines a family of sets. Such families have a simple characterisation.

1.5 Definition. A *closed family* of sets is a non–null set $\mathbb{F}$ of sets which satisfies

(i) If S is a directed subset of $(\mathbb{F}, \subseteq)$ then $\bigcup S \in \mathbb{F}$ and

(ii) If U is a non–null subset of $\mathbb{F}$ then $\bigcap U \in \mathbb{F}$.

Remark. Closed families are mentioned in [Grä] and [S1]. Peter Aczel has pointed out an approach to domains virtually the same as the one here [A]; the only difference is that his is based on closed families but as we now see there is a 1–1 correspondence between information systems and closed families.

1.6 Theorem.
(i) Let $\mathbb{A}$ be an information system. Then $|\mathbb{A}|$ is a closed family of sets.
(ii) Let $\mathbb{F}$ be a closed family of sets. Define

$$A_{\mathbb{F}} = \bigcup \mathbb{F} ,$$

$$X \in Con_{\mathbb{F}} \Leftrightarrow (\exists x \in \mathbb{F} . X \subseteq^{fin} x),$$

$$X \vdash_{\mathbb{F}} a \Leftrightarrow X \in Con_{\mathbb{F}} \ \& \ a \in A_{\mathbb{F}} \ \& \ (\forall x \in \mathbb{F} . X \subseteq x \Rightarrow a \in x).$$

Then $I(\mathbb{F}) = (A_{\mathbb{F}}, Con_{\mathbb{F}}, \vdash_{\mathbb{F}})$ is an information system.
(iii) The maps $\mathbb{A} \mapsto |\mathbb{A}|$ and $\mathbb{F} \mapsto I(\mathbb{F})$ are mutual inverses giving a 1–1 correspondence between information systems and closed families: If $\mathbb{A}$ is an information system then $I(|\mathbb{A}|) = \mathbb{A}$; if $\mathbb{F}$ is a closed family then $|I(\mathbb{F})| = \mathbb{F}$.

Proof.

(i) Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an information system. We show $|\mathbf{A}|$ is a closed family.

As Con is non–null and left–closed with respect to $\subseteq$ the null set $\emptyset$ is consistent. By lemma 1.3, $\overline{\emptyset} = \{a \in A \mid \emptyset \vdash a\}$ is an element of $\mathbf{A}$. By axiom (v) it is $\vdash$–closed. Clearly $\overline{\emptyset} \subseteq x$ for all $x \in |\mathbf{A}|$. Thus $|\mathbf{A}|$ has a (least) member $\overline{\emptyset}$. So certainly $|\mathbf{A}|$ is a non–null family of sets.

Now suppose S is a directed subset of $|\mathbf{A}|$. We show $\bigcup S \in |\mathbf{A}|$. Firstly $\bigcup S$ is consistent. Suppose $X \subseteq^{fin} \bigcup S$. Then, because S is directed and X is finite, $X \subseteq s$ for some $s \in S$. Therefore $X \in \mathrm{Con}$. Secondly $\bigcup S$ is $\vdash$–closed. Suppose $X \in \mathrm{Con}$, $X \vdash a$ and $X \subseteq \bigcup S$. Then as X is finite $X \subseteq s$ for some $s \in S$. However $s \in |\mathbf{A}|$ so $a \in s$. Thus $a \in \bigcup S$. Thus $|\mathbf{A}|$ has unions of directed sets.

Suppose $\emptyset \neq U \subseteq |\mathbf{A}|$. We show $\bigcap U \in |\mathbf{A}|$. Take $u \in U$. We see $\bigcap U$ is consistent as $\bigcap U \subseteq u$. Suppose $X \subseteq \bigcap U$ and $X \vdash a$. Then $X \subseteq u$ for all $u \in U$. Each $u \in U$ is $\vdash$–closed so $a \in u$. Thus $a \in \bigcap U$. Therefore $\bigcap U$ is $\vdash$–closed and consistent so $\bigcap U \in |\mathbf{A}|$.

This proves $|\mathbf{A}|$ is a closed family.

(ii) Let $\mathbf{F}$ be a closed family. It is simple to check that $\mathcal{I}(\mathbf{F})$ is an information system—we leave the details to the reader.

(iii) Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an information system. To show $\mathcal{I}(|\mathbf{A}|) = \mathbf{A}$ we need

$$A = \bigcup |\mathbf{A}|,$$
$$X \in \mathrm{Con} \Leftrightarrow X \subseteq A \ \ \& \ \ (\exists x \in |\mathbf{A}|.\ X \subseteq^{fin} x),$$
$$X \vdash a \Leftrightarrow X \in \mathrm{Con} \ \ \& \ \ a \in A \ \ \& \ \ (\forall x \in |\mathbf{A}|.\ X \subseteq x \Rightarrow a \in x).$$

Obviously $A = \bigcup |\mathbf{A}|$ by condition (ii) on information systems.

Let $X \subseteq^{fin} A$. If $X \in \mathrm{Con}$ then $X \subseteq \overline{X} = \{a \mid X \vdash a\} \in |\mathbf{A}|$. If $X \subseteq x$, where $x \in |\mathbf{A}|$, then by the definition of such elements x we must have $X \in \mathrm{Con}$.

Suppose $X \in \mathrm{Con}$ and $a \in A$. Clearly if $X \vdash a$ then from the definition of elements of $\mathbf{A}$ we must have $X \subseteq x \Rightarrow a \in x$ for any $x \in |\mathbf{A}|$. Suppose $(\forall x \in |\mathbf{A}|.\ X \subseteq x \Rightarrow a \in x)$. Then $\overline{X} = \{b \mid X \vdash b\} \in |\mathbf{A}|$ so $X \vdash a$.

Therefore $\mathcal{I}(|\mathbf{A}|) = \mathbf{A}$.

Let $\mathbf{F}$ be a closed family. We show $|\mathcal{I}(\mathbf{F})| = \mathbf{F}$.

If $x \in \mathbf{F}$ then $x \in |\mathcal{I}(\mathbf{F})|$, directly from the definition of consistency and entailment in $\mathcal{I}(\mathbf{F})$. Thus $\mathbf{F} \subseteq |\mathcal{I}(\mathbf{F})|$.

Now we show the converse inclusion $|I(\mathbb{F})| \subseteq \mathbb{F}$. Write $I(\mathbb{F}) = (A_{\mathbb{F}}, \mathrm{Con}_{\mathbb{F}}, \vdash_{\mathbb{F}})$ as above. Suppose $X \in \mathrm{Con}_{\mathbb{F}}$. Then $U = \{y \in \mathbb{F} \mid X \subseteq y\}$ is a non–null subset of $\mathbb{F}$ from the definition of $\mathrm{Con}_{\mathbb{F}}$ and $\overline{X} = \bigcap U$ from the definition $\vdash_{\mathbb{F}}$. As $\mathbb{F}$ is a closed family $\overline{X} \in \mathbb{F}$. Now if $x \in |I(\mathbb{F})|$ then $S = \{\overline{X} \mid X \subseteq^{fin} x\}$ is a directed subset of $(\mathbb{F}, \subseteq)$. As $\mathbb{F}$ is a closed family $x = \bigcup S \in \mathbb{F}$. Thus $|I(\mathbb{F})| \subseteq \mathbb{F}$.

The two inclusions give $|I(\mathbb{F})| = \mathbb{F}$.

The facts, $I(|\mathbf{A}|) = \mathbf{A}$ for all information systems $\mathbf{A}$ and $|I(\mathbb{F})| = \mathbb{F}$ for all closed families $\mathbb{F}$, provide a 1–1 correspondence between information systems and closed families. ∎

Information systems determine a closed family. Ordered by inclusion these form a domain. In fact all domains can be presented this way as we see later.

1.7 Theorem. *Let $\mathbf{A} = (A, Con, \vdash)$ be an information system. Its elements, $|\mathbf{A}|$, ordered by inclusion form a domain i.e. a consistently complete, algebraic complete partial order. Its finite elements are of the form $\overline{X} = \{a \in A \mid X \vdash a\}$, where $X \in Con$, and the least element of $|\mathbf{A}|$ is $\overline{\emptyset}$.*

Proof. Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an information system with elements $|\mathbf{A}|$. As $|\mathbf{A}|$ is a closed family it is a cpo ordered by inclusion with $\bot = \overline{\emptyset}$.

We require that $|\mathbf{A}|$ is consistently complete *i.e.* if $\forall x \in V.\ x \subseteq y$, for $V \subseteq |\mathbf{A}|$ and $y \in |\mathbf{A}|$, then there is a lub of V in $|\mathbf{A}|$. However if $\forall x \in V. x \subseteq y$ then $U = \{y \mid \forall x \in V. x \subseteq y\}$ is a non–null subset of the closed family $|\mathbf{A}|$. Thus by property (ii) in the definition of closed family we have $\bigcap U \in |\mathbf{A}|$, and $\bigcap U$ is clearly a lub of X.

We show $|\mathbf{A}|$ ordered by inclusion is an algebraic cpo.

Firstly we characterise the finite elements: Suppose x is a finite element of the cpo $|\mathbf{A}|$. Then $x = \bigcup S$ where S is the directed set $S = \{\overline{X} \mid X \subseteq^{fin} x\}$. Because x is a finite element, $x \subseteq \overline{X}$ for some $X \subseteq^{fin} x$. Thus $x = \overline{X}$ for some $X \subseteq^{fin} x$. Conversely, assume x is an element of the form $\overline{X}$ for some $X \in \mathrm{Con}$. Suppose $x \subseteq \bigcup S$ for some directed subset of the cpo $|\mathbf{A}|$. Then $X \subseteq s$ for some $s \in S$, making $x \subseteq s$ too. This argument shows the finite elements of the cpo $|\mathbf{A}|$ are precisely those elements of the form $\overline{X}$ for $X \in \mathrm{Con}$.

Clearly for $x \in |\mathbf{A}|$ we have $x = \bigcup\{\overline{X} \mid X \subseteq^{fin} x\}$. Thus each element of $|\mathbf{A}|$ dominates, and is the lub of, a directed set of finite elements. So $|\mathbf{A}|$ ordered by inclusion forms an algebraic cpo.

We conclude that $(|\mathbf{A}|, \subseteq)$ is a consistently complete algebraic cpo and so a domain. ∎

Thus information systems determine domains of information. Notice how the subtle idea of information introduced by Scott in his theory of domains now has a natural interpretation. By representing a domain as an information system we see the information associated with a computation as the set of tokens that are valid of it and an increase in information as the addition of valid tokens to this set.

An arbitrary domain (consistently complete algebraic cpo) is associated with a natural information system. The intuition is that a finite element is a piece of information that a computation realises—uses or produces—in finite time, so it is natural to take the finite elements as tokens. Then the consistency and entailment relations are induced by the original domain. A finite set of finite elements is consistent if it is compatible and entails an element if its least upper bound dominates the element.

1.8 Definition. Let $(D, \sqsubseteq)$ be a domain (a consistently complete algebraic cpo). Define $ISD = (D^0, \mathrm{Con}, \vdash)$ where D^0 is the set of finite elements of D and Con and $\vdash$ are defined as follows:

$$X \in \mathrm{Con} \Leftrightarrow X \subseteq^{fin} D^0 \ \ \& \ \ X\uparrow,$$
$$X \vdash e \Leftrightarrow X \in \mathrm{Con} \ \ \& \ \ e \sqsubseteq \bigsqcup X.$$

1.9 Proposition. *Let D be a domain. Then ISD is an information system with a domain of elements ordered by inclusion isomorphic to D. The isomorphism pair is*

$$\theta : D \to |ISD| \text{ given by } \theta : d \mapsto \{e \in D^0 \mid e \sqsubseteq d\},$$
$$\phi : |ISD| \to D \text{ given by } \phi : x \mapsto \bigsqcup x.$$

Thus we have shown how an information system determines a domain of elements and vice versa how a domain determines an information system with an isomorphic domain of elements. We are justified in saying information systems represent domains. In the next section we see this expressed as an equivalence of categories.

2. The category of information systems.

Information systems are equipped with morphisms called *approximable mappings* in [S]. Approximable mappings between information systems correspond to continuous functions between the associated domains.

2.1 Definition. Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ and $\mathbf{B} = (B, \mathrm{Con}, \vdash)$ be information systems. An *approximable mapping* $r : \mathbf{A} \to \mathbf{B}$ is a relation $r \subseteq \mathrm{Con}_A \times \mathrm{Con}_B$ such that:

(i) $\emptyset r \emptyset$,

(ii) $XrY \ \& \ XrY' \Rightarrow Xr(Y \cup Y')$

(iii) $X' \vdash_A X \ \& \ XrY \ \& \ Y \vdash_B Y' \Rightarrow X'rY'$

for all $X, X' \in \mathrm{Con}_A$ and $Y, Y' \in \mathrm{Con}_B$.

Intuitively an approximable relation expresses how information in one information system entails information in another. For an approximable mapping $r : \mathbf{A} \to \mathbf{B}$, XrY can be read as information X in $\mathbf{A}$ entails information Y in $\mathbf{B}$. In particular the relation r might be induced by a computation which given information in $\mathbf{A}$ as input delivers information in $\mathbf{B}$ as output—see [S] for further intuitions.

It is easy to see, and is shown in [S], that information systems with approximable mappings form a category in which mappings are composed by the usual composition of relations; the identity mapping on an information system $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ is the relation $\vdash_A \subseteq \mathrm{Con}_A \times \mathrm{Con}_A$.

We have seen how an information system $\mathbf{A}$ represents a domain $|\mathbf{A}|$. In a similar way an approximable mapping r represent a continuous function $|r|$. This operation $|-|$ acting on both information systems and approximable mappings is a functor from the category of information systems to the category of domains. Conversely, we have seen how a domain D can be associated with an information system ISD and this operation extends to continuous functions so that a continuous function between domains is associated with an approximable mapping between such information systems. This operation IS is a functor from domains to information systems. Now when we represent a domain D as an information system ISD, the domain $|ISD|$ is isomorphic to D. And similarly when we take the domain of an an information system $\mathbf{A}$, to obtain $|\mathbf{A}|$, and then form the information system $IS|\mathbf{A}|$ this is isomorphic to the original information system $\mathbf{A}$. In the language of category theory the two functors $|-|$ and IS passing back and forth between the two categories determine an equivalence of the categories of information systems and domains (see [Mac] for a precise definition of equivalence).

2.2 Proposition.
Let $r : \mathbf{A} \to \mathbf{B}$ be an approximable mapping. Then $|r| : |\mathbf{A}| \to |\mathbf{B}|$ given by

$$|r|(x) = \bigcup \{Y \mid \exists X \subseteq x.\ XrY\}$$

is a continuous function between the domains $|\mathbf{A}|$ and $|\mathbf{B}|$ ordered by inclusion. In fact $|-|$ is a functor from the category **ISys** *of information systems to the category* **Dom** *of domains.*

Let $f : D \to E$ be a continuous function between cpos D and E. Define the relation $ISf \subseteq Con_{ISD} \times Con_{ISE}$ by

$$X(ISf)Y \Leftrightarrow \bigsqcup Y \sqsubseteq f(\bigsqcup X).$$

Then $ISf : ISD \to ISE$ is an approximable mapping. In fact IS is a functor from the category **Dom** *of domains to the category* **ISys** *of information systems.*

The functors $|-| :$ **ISys** $\to$ **Dom** *and $IS :$* **Dom** $\to$ **ISys** *establish an equivalence of the categories* **ISys** *and* **Dom** .

Because the categories **ISys** and **Dom** are equivalent, from a category–theoretic point of view they are essentially the same; a categorical construction on one passes over via the appropriate functor to the same categorical construction on the other. Still we have gained something. The rather abstract category of domains of information and continuous functions is represented by the more concrete category of information systems and approximable mappings. We can work concretely, with basic set theory, to produce constructions on information systems. One bonus, as we shall see, is an elementary treatment of recursive domain equations.

3. A complete partial order of information systems.

Because we work with a concrete representation of domains we can replace the usual inverse limit constructions used in building up solutions to recursive domain equations by a fixed-point construction on a complete partial order of information systems. The order on information systems, $\trianglelefteq$, captures an intuitive notion, that of one information system being a subsystem, or substructure, of another.

3.1 Definition. Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Define $\mathbf{A} \trianglelefteq \mathbf{B}$ iff

(i) $A \subseteq B$

(ii) $X \in \mathrm{Con}_A \Leftrightarrow X \subseteq A \;\&\; X \in \mathrm{Con}_B$

(iii) $X \vdash_A a \Leftrightarrow X \subseteq A \;\&\; a \in A \;\&\; X \vdash_B a$

When $\mathbf{A} \trianglelefteq \mathbf{B}$, for two information systems $\mathbf{A}$ and $\mathbf{B}$, we say $\mathbf{A}$ is a *subsystem* of $\mathbf{B}$.

An information system $\mathbf{A}$ is a subsystem of another $\mathbf{B}$ iff $\mathbf{A}$ is a restriction of $\mathbf{B}$ to the tokens of $\mathbf{A}$. Precisely:

3.2 Definition. Let $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be an information system and let $A \subseteq B$. Define the *restriction* of $\mathbf{B}$ to A to be $\mathbf{B}\lceil A =_{def} (A, \mathrm{Con}', \vdash')$ where

$$X \in \mathrm{Con}' \Leftrightarrow X \subseteq A \;\&\; X \in \mathrm{Con} \text{ and}$$
$$X \vdash' b \Leftrightarrow X \subseteq A \;\&\; b \in A \;\&\; X \vdash b.$$

3.3 Proposition. *Let $\mathbf{B} = (B, Con_B, \vdash_B)$ be an information system and let $A \subseteq B$. Then $\mathbf{B}\lceil A$ is an information system and $\mathbf{B}\lceil A \trianglelefteq \mathbf{B}$.*
Let $\mathbf{A} = (A, Con_A, \vdash_A)$ and $\mathbf{B}$ be information systems. If $\mathbf{A} \trianglelefteq \mathbf{B}$ then $\mathbf{A} = \mathbf{B}\lceil A$.
Let $\mathbf{A} = (A, Con_A, \vdash_A)$ and $\mathbf{B} = (B, Con_B, \vdash_B)$ be information systems. If their token-sets are equal, i.e. $A = B$, and $\mathbf{A} \trianglelefteq \mathbf{B}$ then $\mathbf{A} = \mathbf{B}$.

Proof. Obvious from the definition of $\trianglelefteq$. ∎

This definition of subsystem almost gives a complete partial order (cpo) of information systems. There is a least information system, the unique one with the emptyset as tokens. Each ω-chain of information systems increasing with respect to $\trianglelefteq$ has a least upper bound (lub), with tokens, consistency and entailment relations the union of those in the chain. But of course information systems form a class and not a set and for this reason alone they do not quite form a cpo. We could say they form a *large cpo*. This is all we need. (Very similar approaches to solving domain equations, or equations for structures very like domains, occur in [BC], [W1], [A] and [S1].)

3.4 Theorem. *The relation $\trianglelefteq$ is a partial order with $\bot =_{def} (\emptyset, \{\emptyset\}, \emptyset)$ as least element. Moreover if $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ is an increasing ω-chain of information systems $\mathbf{A}_i = (A_i, Con_i, \vdash_i)$ then there exists a least upper bound given by*

$$\bigcup_i \mathbf{A}_i = (\bigcup_i A_i, \bigcup_i \vdash_i, \bigcup_i Con_i).$$

Proof. Clearly $\trianglelefteq$ is a partial order and $\perp$ is the $\trianglelefteq$-least information structure.

Let $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ be an increasing ω-chain of information systems $\mathbf{A}_i = (A_i, \mathrm{Con}_i, \vdash_i)$. Write $\mathbf{A} = (A, \mathrm{Con}, \vdash) = (\bigcup_i A_i, \bigcup_i \vdash_i, \bigcup_i \mathrm{Con}_i)$. It is routine to check that $\mathbf{A}$ is an information system.

It is an upper bound of the chain: Obviously each A_i is a subset of the tokens A; obviously $\mathrm{Con}_i \subseteq \mathrm{Con}$ while conversely, if $X \subseteq A_i$ and $X \in \mathrm{Con}$ then $X \in \mathrm{Con}_j$ for some $j \geq i$ but then $X \in \mathrm{Con}_i$ as $\mathbf{A}_i \trianglelefteq \mathbf{A}_j$; obviously $\vdash_i \subseteq \vdash$ while conversely if $X \subseteq A_i$, $a \in A_i$ and $X \vdash a$ then $X \vdash_j a$ for some $j \geq i$ but then $X \vdash_i a$ as $\mathbf{A}_i \trianglelefteq \mathbf{A}_j$.

It is a least upper bound of the chain: Assume $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ is an upper bound of the chain. Clearly then $A = \bigcup_i A_i \subseteq B$. Clearly $\mathrm{Con} = \bigcup_i \mathrm{Con}_i \subseteq \mathrm{Con}_B$. Also if $X \subseteq A$ and $X \in \mathrm{Con}_B$ then as X is finite $X \subseteq A_i$ for some i. So $X \in \mathrm{Con}_i \subseteq \mathrm{Con}$ as $\mathbf{A}_i \trianglelefteq \mathbf{B}$. Thus $X \in \mathrm{Con} \Leftrightarrow X \subseteq A \;\&\; X \in \mathrm{Con}_B$. Similarly $X \vdash a \Leftrightarrow X \subseteq A \;\&\; a \in A \;\&\; X \vdash_B a$. Thus $\mathbf{A} \trianglelefteq \mathbf{B}$ making $\mathbf{A}$ the lub of the chain. ∎

We extend the subsystem relation to n-tuples of information systems. They form a large cpo too.

3.5 Definition. For $n \in \omega$ write $\mathbf{ISys}^n$ for all n-tuples $(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1})$. Write Π_j for the *projection* map $\Pi_j(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1}) = \mathbf{A}_j$.

For $(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1})$ and $(\mathbf{B}_0, \ldots, \mathbf{B}_{n-1})$ in $\mathbf{ISys}^n$ write

$$(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1}) \trianglelefteq (\mathbf{B}_0, \ldots, \mathbf{B}_{n-1}) \Leftrightarrow \mathbf{A}_0 \trianglelefteq \mathbf{B}_0 \;\&\; \cdots \;\&\; \mathbf{A}_{n-1} \trianglelefteq \mathbf{B}_{n-1}.$$

3.6 Notation. We shall often use the same notation for n-tuples of information systems as for single ones. Whether $\mathbf{A}$ stands for a single information system or a tuple will always be clear from the context.

3.7 Proposition. *The relation $\trianglelefteq$ is a partial order on $\mathbf{ISys}^n$ with least element $(\perp, \cdots, \perp)$. There are least upper bounds of increasing ω-chains in $\mathbf{ISys}^n$; in each co-ordinate j the least upper bound $\bigcup_i \mathbf{A}_i$ of a chain $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ satisfies $\Pi_j(\bigcup_i \mathbf{A}_i) = \bigcup_i \Pi_j(\mathbf{A}_i)$.*

We shall be concerned with continuous operations on information systems and using them to define information systems recursively.

3.8 Definition. Let $\mathbf{F} : \mathbf{ISys}^m \to \mathbf{ISys}^n$ be an operation on information systems.

The operation $\mathbf{F}$ is said to be *monotonic* (with respect to $\trianglelefteq$) iff $\mathbf{A} \trianglelefteq \mathbf{B} \Rightarrow \mathbf{F}(\mathbf{A}) \trianglelefteq \mathbf{F}(\mathbf{B})$ for all information systems $\mathbf{A} \in \mathbf{ISys}^m$ and $\mathbf{B} \in \mathbf{ISys}^n$.

The operation $\mathbf{F}$ is said to be *continuous* (with respect to $\trianglelefteq$) iff it is monotonic and for any increasing ω-chain of information systems $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ in $\mathbf{ISys}^m$, $\bigcup_i \mathbf{F}(\mathbf{A}_i) = \mathbf{F}(\bigcup_i \mathbf{A}_i)$. (Notice that since $\mathbf{F}$ is monotonic $\bigcup_i \mathbf{F}(\mathbf{A}_i)$ exists.)

Thus, as an example, proposition 3.7 says the projection maps Π_j are continuous on tuples of information systems ordered by $\trianglelefteq$.

Fortunately in reasoning about the monotonicity and continuity of an operation we need only consider one input coordinate and one output coordinate at a time because of the following facts, well–known for cpos.

3.9 Proposition. *Let* $\mathbf{F} : \mathbf{ISys}^m \to \mathbf{ISys}^n$ *be an operation on information systems.*
It is monotonic, respectively continuous, (with respect to $\trianglelefteq$*) iff it is monotonic, respectively continuous, in each argument separately (i.e. considered as a function in any one of its argument, holding the others fixed).*
Similarly it is monotonic, respectively continuous, (with respect to $\trianglelefteq$*) iff it is monotonic, respectively continuous, considered as a function to each output coordinate (i.e. each function* $\Pi_j \circ \mathbf{F}$ *is continuous for* $j < n$*).*

Thus in verifying that an operation is monotonic or continuous we ultimately have to show certain unary operations are continuous with respect to the subsystem relation $\trianglelefteq$. The next lemma will be a great help in proving operations continuous. Generally it is very easy to show that a unary operation is monotonic with respect to $\trianglelefteq$ and continuous on the token sets, a notion we now make precise.

3.10 Definition. Say a unary operation $\mathbf{F}$ on information systems is *continuous on token sets* iff for any ω-chain, $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$, each token of $\mathbf{F}(\bigcup_i \mathbf{A}_i)$ is a token of $\bigcup_i \mathbf{F}(\mathbf{A}_i)$.

3.11 Lemma. *Let* $\mathbf{F}$ *be a unary operation on information systems. Then* $\mathbf{F}$ *is continuous iff* $\mathbf{F}$ *is monotonic with respect to* $\trianglelefteq$ *and continuous on token sets.*

Proof.

only if: obvious.

if: Let $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ be an ω–chain of information systems. Clearly $\bigcup_i \mathbf{F}(\mathbf{A}_i) \trianglelefteq \mathbf{F}(\bigcup_i \mathbf{A}_i)$ since $\mathbf{F}$ is assumed monotonic. Thus from the assumption the tokens of $\bigcup_i \mathbf{F}(\mathbf{A}_i)$ are the same as the tokens of $\mathbf{F}(\bigcup_i \mathbf{A}_i)$. Therefore they are the same information system by proposition 3.3. ∎

Now we relate the subsystem relation on information systems to corresponding relations on families of sets and domains. Recall information systems are in 1–1 correspondence with closed families. The reader will have no difficulty in showing:

3.12 Proposition. *For information systems* $\mathbf{A}$ *and* $\mathbf{B}$

$$\mathbf{A} \trianglelefteq \mathbf{B} \Leftrightarrow |\mathbf{A}| = \{y \cap A \mid y \in |\mathbf{B}|\}.$$

The ordering $\trianglelefteq$ on information systems induces an embedding–projection pair between domains. Recall the definition of embedding–projection pair.

3.13 Definition. For domains D and E we write $D \lhd^{\theta}_{\phi} E$ iff $\theta : D \to E$ and $\phi : E \to D$ are continuous functions such that $\phi \circ \theta = 1_D$ and $\theta \circ \phi \sqsubseteq 1_E$. Then θ is called an *embedding* and ϕ is called a *projection.*

3.14 Theorem. *Let* $\mathbf{A} = (A, Con_A, \vdash_A)$ *and* $\mathbf{B} = (B, Con_B, \vdash_B)$ *be two information systems. Then* $\mathbf{A} \trianglelefteq \mathbf{B} \Rightarrow |\mathbf{A}| \lhd^{\theta}_{\phi} |\mathbf{B}|$ *where* $\theta(x) = \{b \in B | \exists X \subseteq x.\ X \vdash_B b\}$ *and* $\phi(y) = y \cap A$.

Proof. It should be checked that θ and ϕ are well–defined as mappings, that $\theta(x) \in |\mathbf{B}|$ when $x \in |\mathbf{A}|$ and $\phi(y) \in |\mathbf{A}|$ when $y \in |\mathbf{B}|$, and also that the mappings are continuous. This is left to the reader.

The following proves that θ and ϕ form an embedding–projection pair between $|\mathbf{A}|$ and $|\mathbf{B}|$, *i.e.* (i) $\phi \circ \theta = 1_B$ and (ii) $\theta \circ \phi \subseteq 1_A$:

(i) Let $z \in |\mathbf{B}|$. Then

$$\begin{aligned}(\theta \circ \phi)(z) = \theta(\phi(z)) = \theta(z \cap A) &= \{b \in B | \exists X \subseteq (z \cap A).\ X \vdash_B b\} \\ &\subseteq \{b \in B | \exists X \subseteq z.\ X \vdash_B b\} = z.\end{aligned}$$

(ii) Let $x \in |\mathbf{A}|$. Then as $\mathbf{A} \trianglelefteq \mathbf{B}$ we have $\theta(x) \cap A = \{a \in A \mid \exists X \subseteq x. X \vdash_A a\} = x$.

∎

Because the categories of domains and information systems are equivalent, embeddings and projections have their counterparts as approximable mapings. The approximable mappings associated with functions θ and ϕ above are $IS\theta$ and $IS\phi$ given by

$$\begin{aligned}X(IS\theta)Y &\Leftrightarrow X \subseteq A \ \&\ Y \subseteq B \ \&\ X \vdash_B Y \\ Y(IS\phi)X &\Leftrightarrow X \subseteq A \ \&\ Y \subseteq B \ \&\ Y \vdash_B X.\end{aligned}$$

Embedding–projection pairs are central to the inverse–limit method of constructing solutions to recursive domain equations, the original method devised by Scott (see [St, P]). Intuitively an embedding–projection pair tells you how one domain approximates another, and solutions to recursive domain equations are built–up as inverse limits of chains of embeddings. Working at the abstract level of domains, the embeddings are necessary to express how one domain fits inside another. With information systems however, because they are more concrete, the way in which one information system approximates another can be expressed directly by $\trianglelefteq$, based on inclusion of sets. Using the standard theory of least fixed points for cpos we know that any continuous operation, $\mathbf{F}$, on information systems has a least fixed point fix$\mathbf{F}$ given by the least upper bound, $\bigcup_i \mathbf{F}^i(\underline{\perp})$, of the increasing ω-chain

$\underline{\perp} \trianglelefteq \mathbb{F}(\underline{\perp}) \trianglelefteq \mathbb{F}^2(\underline{\perp}) \trianglelefteq \cdots \trianglelefteq \mathbb{F}^n(\underline{\perp}) \trianglelefteq \cdots$. Notice that whereas in constructing the solution to recursive domain equations using inverse limits the solution is generally a solution only to within isomorphism, when we use least fixed points in the cpo of information systems we have the equality $\mathrm{fix}\mathbb{F} = \mathbb{F}(\mathrm{fix}\mathbb{F})$.

In the next section we shall see many examples of operations on information systems and how we can use the large cpos of this section to obtain solutions to recursively defined information systems. Because the machinery works for operations $\mathbf{ISys}^m \to \mathbf{ISys}^n$ we can recursively define several information systems simultaneously.

One thing may be puzzling the reader; why do we build a large cpo from the relation $\trianglelefteq$ rather than the simpler relation based on coordinatewise inclusion of on information in another? This is a partial order and does indeed give another large cpo. It even has the added bonus of having meets of arbitrary sets and forms a consistently complete algebraic domain when restricted to information systems whose tokens all lie in some particular set. (This domain could thus be represented by an information system itself.) However it suffers a major drawback; the function space construction on information systems—defined in the next section—while being continuous in its right argument is not even monotonic in its left argument with respect to this inclusion order.

4. Constructions.

In this section we give constructions of *lifting* $(-)_\uparrow$, *sum* $+$, *separated-sum* $\oplus$, *product* $\times$ and *exponentiation* or *function space* $\rightarrow$ on information systems. They induce the usual lifting, sum, separated-sum, product and function space of constructions on domains. We choose them with a little care so that they are also continuous with respect to $\trianglelefteq$. In this way we can produce solutions to recursive equations for information systems written in terms of these constructions in a swift, uniform and elegant manner. Because each construction extends to a functor on the category of information systems the apparatus transfers to the equivalent category of domains, producing solutions to recursive domain equations there.

Our concrete constructions will rely on these simple operations on sets.

4.1 Notation. For two sets A and B let $A \uplus B$ be the disjoint union of A and B, i.e. $A \uplus B = (\{0\} \times A) \cup (\{1\} \times B)$. Write $inj_0 : A \rightarrow A \uplus B$ and $inj_1 : B \rightarrow A \uplus B$ be the injections taking $inj_0 : a \mapsto (0, a)$ for $a \in A$ and $inj_1 : b \mapsto (1, b)$ for $b \in B$. Define the partial functions $out_0 : A \uplus B \rightarrow A$ and $out_1 : A \uplus B \rightarrow B$ projecting out of the disjoint union to the component parts by $out_0 : (0, a) \mapsto a$ iff $a \in A$ and $out_1 : (1, b) \mapsto b$ iff $b \in B$. We shall often extend functions on sets to functions on subsets in the natural way. For example when $X \subseteq A \uplus B$ we write $out_0 X$ for $\{a \in A \mid \exists c \in X. out_0(c) = a\}$.

So far we have constructed only one trivial information system $\underline{\perp} = (\emptyset, \{\emptyset\}, \emptyset)$ based on the null set of tokens. Our constructions take-off with *lifting*.

4.2 Definition. Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an information system. Define the *lifting* of $\mathbf{A}$ to be $\mathbf{A}_\uparrow = (A', \mathrm{Con}', \vdash')$ where:

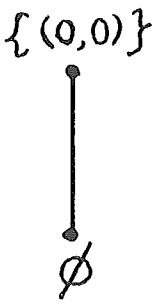
(i) $A' = \{0\} \uplus A$

(ii) $X \in \mathrm{Con}' \Leftrightarrow out_1 X \in \mathrm{Con}$

(iii) $X \vdash' c \Leftrightarrow X \neq \emptyset \ \&\ (out_0(c) = 0$ or $(\exists a \in A. out_1(c) = a \ \&\ out_1 X \vdash a))$.

This construction like those that follow is best motivated by considering its effect on the family of configurations. It prefixes the family by an element consisting of an extra token.

4.3 Example. Define $\mathbf{O} = \underline{\perp}_\uparrow$. Then $\mathbf{O}$ has one token $(0,0)$, consistent sets $\emptyset$ and $\{(0,0)\}$, and entailment relation $\{(0,0)\} \vdash (0,0)$. Its elements ordered by inclusion form the domain:



4.4 Proposition. *Let* $\mathbf{A}$ *be an information system. Then* $\mathbf{A}_\uparrow$ *is an information system with elements of the form*

$$|\mathbf{A}_\uparrow| = \{\emptyset\} \cup \{\{0\} \uplus x \mid x \in |\mathbf{A}|\}.$$

Proof. Firstly we show $\{\emptyset\} \cup \{\{0\} \uplus x \mid x \in |\mathbf{A}|\} \subseteq |\mathbf{A}_\uparrow|$. Obviously $\emptyset$ is consistent and $\emptyset \not\vdash' c$ for any c so $\emptyset \in |\mathbf{A}_\uparrow|$. Assume now that $x \in |\mathbf{A}|$. We show $y = \{0\} \uplus x$ is an element of $|\mathbf{A}_\uparrow|$. Clearly y is consistent. Suppose $X \subseteq y \ \& \ X \vdash' c$. Then $X \neq \emptyset$ and either $out_0(c) = 0$ so $c \in y$ or $out_1 X \vdash a$ where $out_1(c) = a$. In the latter case $out_1 X \subseteq x$ so $a \in x$, making $c \in y$. Thus y is consistent and $\vdash'$–closed and so an element of $\mathbf{A}_\uparrow$.

Now we show the converse inclusion $|\mathbf{A}_\uparrow| \subseteq \{\emptyset\} \cup \{\{0\} \uplus x \mid x \in |\mathbf{A}|\}$. Suppose $y \in |\mathbf{A}_\uparrow|$ and $y \neq \emptyset$. As y is $\vdash'$–closed it contains $inj_0(0)$. Take $x = out_1 y$. Clearly x is consistent in $\mathbf{A}$. Suppose $Z \subseteq x \ \& \ Z \vdash a$. Then $inj_1 Z \vdash' inj_1(a)$ and $inj_1 Z \subseteq y$ which implies $inj_1(a) \in y$ so $a \in x$. Therefore $y = \{0\} \uplus x$ for some element x of $\mathbf{A}$. ∎

4.5 Theorem. *The operation* $\mathbf{A} \mapsto \mathbf{A}_\uparrow$ *is a continuous operation on information systems ordered by* $\trianglelefteq$.

Proof. We use lemma 3.11. We first show lifting is monotonic. Assume $\mathbf{A} \trianglelefteq \mathbf{B}$ for two information systems $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$. Write $\mathbf{A}_\uparrow = (A', \mathrm{Con}_A', \vdash_A')$ and $\mathbf{B}_\uparrow = (B', \mathrm{Con}_B', \vdash_B')$. Let us check $\mathbf{A}_\uparrow \trianglelefteq \mathbf{B}_\uparrow$:

Obviously $A' = \{0\} \uplus A \subseteq \{0\} \uplus B = B'$;

Obviously $X \in \mathrm{Con}_A' \Leftrightarrow X \subseteq B' \ \& \ out_1 X \in \mathrm{Con}_A \Leftrightarrow X \subseteq A' \ \& \ out_1 X \in \mathrm{Con}_B$;

Obviously

$$\begin{aligned} X \vdash_A' c \Leftrightarrow\ & X \subseteq A' \ \& \ c \in A' \ \& \\ & X \neq \emptyset \ \& \ (out_0(c) = 0 \text{ or } (\exists a \in A.\, out_1(c) = a \ \& \ out_1 X \vdash_A a)) \\ \Leftrightarrow\ & X \subseteq A' \ \& \ c \in A' \ \& \\ & X \neq \emptyset \ \& \ (out_0(c) = 0 \text{ or } (\exists a \in B.\, out_1(c) = a \ \& \ out_1 X \vdash_B a)). \end{aligned}$$

Thus $\mathbf{A}_\uparrow \trianglelefteq \mathbf{B}_\uparrow$. Therefore $(-)_\uparrow$ is monotonic. It remains to show that it acts continuously on token-sets. Let $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ be an ω–chain of information systems $\mathbf{A}_i = (A_i, \mathrm{Con}_i, \vdash_i)$. The set of tokens of $(\bigcup_i \mathbf{A}_i)_\uparrow$ is $\{0\} \uplus (\bigcup_i A_i)$ which is clearly equal to $\bigcup_i (\{0\} \uplus A_i)$ the set of tokens of $\bigcup_i (\mathbf{A}_i)_\uparrow$. Thus by lemma 3.11 we know lifting is a continuous operation on information systems ordered by $\trianglelefteq$. ∎

4.6 Example. Because lifting is continuous wrt $\trianglelefteq$ it has a least fixed point $\Omega = \Omega_\uparrow$. As an exercise the reader can work out the set of tokens and show that its domain of elements $|\Omega|$ is isomorphic to the order type $\omega + 1$.

Another useful construction is that of the *sum* of two information systems which is formed by juxtaposing disjoint copies of the two information systems. It has the effect of replacing the two bottom elements of the two sets of configurations by the single configuration $\emptyset$ so joining disjoint copies of the elements at their bottom elements.

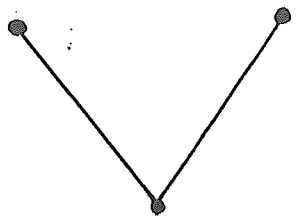
4.7 Definition. Let $\mathbf{A}_0 = (A, \mathrm{Con}_0, \vdash_0)$ and $\mathbf{A}_1 = (A, \mathrm{Con}_1, \vdash_1)$ be information systems. Define their *sum*, $\mathbf{A}_0 + \mathbf{A}_1$, to be $\mathbf{C} = (C, \mathrm{Con}, \vdash)$ where:

(i) $C = (A_0 \setminus \{a \mid \emptyset \vdash_0 a\}) \uplus (A_1 \setminus \{a \mid \emptyset \vdash_1 a\})$

(ii) $X \in \mathrm{Con} \Leftrightarrow \exists X_0 \in \mathrm{Con}_0 . X = inj_0 X_0$ or $\exists X_1 \in \mathrm{Con}_1 . X = inj_1 X_1$,

(iii) $X \vdash a \Leftrightarrow \exists X_0, a_0 . X_0 \vdash_0 a_0 \ \& \ X = inj_0 X_0 \ \& \ a = inj_0(a_0)$ or
$\exists X_1, a_1 . X_1 \vdash_1 a_1 \ \& \ X = inj_1 X_1 \ \& \ a = inj_1(a_1)$.

4.8 Example. We can define the information system of truth values by $\mathbf{T} = \mathbf{O} + \mathbf{O}$. Its domain of configurations looks like:



4.9 Theorem. *The operation $+$ is a continuous operation on information systems ordered by $\trianglelefteq$.*

Proof. It is necessary to verify that if $\mathbf{A}$ and $\mathbf{B}$ are information systems then so is their sum $\mathbf{A} + \mathbf{B}$. That $\mathbf{A} + \mathbf{B}$ satisfies the properties (i) to (v) follows, property for property, from the fact that $\mathbf{A}$ and $\mathbf{B}$ satisfy (i) to (v).

We show that $+$ is continuous with respect to $\trianglelefteq$. By definition of continuity we must show that $+$ is continuous in each argument. We prove $+$ continuous in its first argument. Then, by symmetry, it is easy to see that $+$ will be continuous in its second argument too.

First we show $+$ is monotonic in its first argument. Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$, $\mathbf{A}' = (A', \mathrm{Con}_{A'}, \vdash_{A'})$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems with $\mathbf{A} \trianglelefteq \mathbf{A}'$. Write $\mathbf{C} = (C, \mathrm{Con}, \vdash) = \mathbf{A} + \mathbf{B}$ and $\mathbf{C}' = (C', \mathrm{Con}', \vdash') = \mathbf{A}' + \mathbf{B}$. We require $\mathbf{C} \trianglelefteq \mathbf{C}'$ *i.e.*

(i) $C \subseteq C'$

(ii) $X \in \mathrm{Con} \Leftrightarrow X \subseteq C \ \& \ X \in \mathrm{Con}'$

(iii) $X \vdash a \Leftrightarrow X \subseteq C \ \& \ a \in C \ \& \ X \vdash' a$

(i) From the definition of $+$ and the assumption $\mathbf{A} \trianglelefteq \mathbf{A}'$ we get $C \subseteq C'$.

(ii) "$\Rightarrow$". Let $X \in \mathrm{Con}$. Then $X = \{0\} \times X_0$ for some $X_0 \in \mathrm{Con}_A$ or $X = \{1\} \times X_1$ for some $X_1 \in \mathrm{Con}_B$. Assume $X = \{0\} \times X_0$. Then clearly $X \subseteq C$ and $X_0 \in \mathrm{Con}_{A'}$ since

$\mathbf{A} \trianglelefteq \mathbf{A}'$. Therefore by the definition of $+$, $X \in \mathrm{Con}'$. Now assume $X = \{1\} \times X_1$ where $X_1 \in \mathrm{Con}_1$. Then directly from the definition of $+$ we have $X \in \mathrm{Con}_{C'}$.

(ii) "$\Leftarrow$". Suppose $X \in \mathrm{Con}'$ and $X \subseteq C$. Then either $X = \{0\} \times X_0$ for some $X_0 \in \mathrm{Con}_{A'}$ or $X = \{1\} \times X_1$ for some $X_1 \in \mathrm{Con}_B$. In the former case $X_0 \subseteq A$ so, as $\mathbf{A} \trianglelefteq \mathbf{A}'$, we obtain $X_0 \in \mathrm{Con}_A$. In the latter case $X \in \mathrm{Con}_C$ trivially.

(iii) is very similar to (ii).

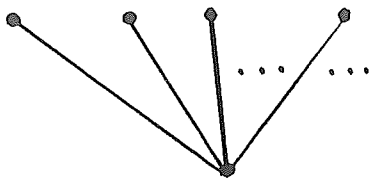
This shows $+$ monotonic in its first argument. It remains to show that $+$ acts continuously on the token-sets. Let $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ be an ω-chain of information systems $\mathbf{A}_i = (A_i, \mathrm{Con}_i, \vdash_i)$. The set of tokens of $(\bigcup_i \mathbf{A}_i) + \mathbf{B}$ is $((\bigcup_{i\in\omega} A_i) \setminus \{a \mid \exists i.\ \emptyset \vdash_i a\}) \uplus (B \setminus \{b \mid \emptyset \vdash_B b\})$ which is equal to $\bigcup_{i\in\omega}((A_i \setminus \{a \mid \emptyset \vdash_i a\}) \uplus (B \setminus \{b \mid \emptyset \vdash_B b\}))$ the set of tokens of $\bigcup_i(\mathbf{A}_i + \mathbf{B})$.

Thus $+$ is continuous in its first and, by symmetry, its second argument, and is therefore continuous. ∎

4.10 Proposition. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be information systems. Then*

$$x \in |\mathbf{A} + \mathbf{B}| \Leftrightarrow (\exists y \in |\mathbf{A}|.x = inj_0(y \setminus \overline{\emptyset})) \quad or \quad (\exists y \in |\mathbf{B}|.x = inj_1(y \setminus \overline{\emptyset})).$$

4.11 Example. Because $+$ is continuous we can construct the least information system $\mathbf{N}$ such that $\mathbf{N} = \mathbf{O} + \mathbf{N}$. Its elements form the flat (or discrete) domain of integers:



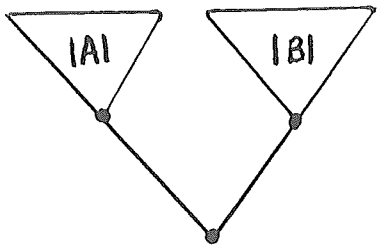
Another form of sum is common in the literature on cpos. It is the so-called separated sum. It can be obtained by composing $+$ with lifting.

4.12 Definition. Let $\mathbf{A}$ and $\mathbf{B}$ be information systems. Define their *separated sum* by $\mathbf{A} \oplus \mathbf{B} = \mathbf{A}_\uparrow + \mathbf{B}_\uparrow$.

4.13 Theorem. *The operation* $\oplus$ *is a continuous operation on information systems with respect to* $\trianglelefteq$.

Proof. Clear as separated sum is derived by composing the two continuous operations of lifting and sum. ∎

If $\mathbf{A}$ and $\mathbf{B}$ are information systems the effect of separated sum on their domains of elements can be illustrated by:



The separated sum forms new bottom elements below the domains and fuses them together.

The product construction on cpos is well–known; it is the coordinatewise order on the cartesian product. The desired effect is obtained on information systems by forming disjoint copies of the token sets and taking finite sets to be consistent if their projections are consistent and a consistent set to entail a token if it does so when projected into the appropriate component.

4.14 Definition. Let $\mathbf{A}_0 = (A, \mathrm{Con}_0, \vdash_0)$ and $\mathbf{A}_1 = (A, \mathrm{Con}_1, \vdash_1)$ be information systems. Define their *product*, $\mathbf{A}_0 \times \mathbf{A}_1$, to be the information system $\mathbf{C} = (C, \mathrm{Con}, \vdash)$ where:

(i) $C = A_0 \uplus A_1$

(ii) $X \in \mathrm{Con} \Leftrightarrow out_0 X \in \mathrm{Con}_0 \ \ \& \ \ out_1 X \in \mathrm{Con}_1$

(iii) $X \vdash c \Leftrightarrow (\forall a \in A_0.\ a = out_0(c) \Rightarrow out_0 X \vdash_0 a) \ \ \&$
$(\forall a \in A_1.\ a = out_1(c) \Rightarrow out_1 X \vdash_1 a).$

4.15 Theorem. *The operation $\times$ is a continuous operation on information systems ordered by $\trianglelefteq$.*

Proof. It is routine to check that the product of two information systems is an information system. We show that the product operation is monotonic and continuous on token–sets. Then by lemma 3.11 we know it is continuous with respect to $\trianglelefteq$.

Monotonic. Let $\mathbf{A} \trianglelefteq \mathbf{A}'$ and $\mathbf{B}$ be information systems. The tokens of $\mathbf{A} \times \mathbf{B}$ obviously form a subset of the tokens of $\mathbf{A}' \times \mathbf{B}$. Suppose X is a subset of the tokens of $\mathbf{A} \times \mathbf{B}$. Then X is consistent in $\mathbf{A} \times \mathbf{B}$ iff $\{a \mid (0,a) \in X\}$ and $\{b \mid (1,b) \in X\}$ are both consistent in $\mathbf{A}$ and $\mathbf{B}$ respectively. Because $\mathbf{A} \trianglelefteq \mathbf{A}'$ this is equivalent to X being consistent in $\mathbf{A}' \times \mathbf{B}$. Suppose X is a finite set of tokens of $\mathbf{A} \times \mathbf{B}$ and c is a token of $\mathbf{A} \times \mathbf{B}$. Then $X \vdash c$ in $\mathbf{A} \times \mathbf{B}$ iff $(c = (0, a_0) \ \ \& \ \ \{a \mid (0,a) \in X\})$ or $(c = (1, b_1) \ \ \& \ \ \{b \mid (1,b) \in X\})$. Because $\mathbf{A} \trianglelefteq \mathbf{A}'$ this is equivalent to $X \vdash c$ in $\mathbf{A}' \times B$. Thus $\mathbf{A} \times \mathbf{B} \trianglelefteq \mathbf{A}' \times \mathbf{B}$. Thus $\times$ is monotonic in its first argument.

Continuous on token–sets. Now let $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ be an ω–chain of information systems. A token of $(\bigcup_i A_i) \times B$ is clearly a token of $A_i \times B$ for some $i \in \omega$, and so a token of $\bigcup_i (A_i \times B)$.

Thus by lemma 3.11, $\times$ is continuous in its first argument. Similarly it is continuous in its second argument. Thus $\times$ is a continuous operation on information systems with respect to $\trianglelefteq$. ∎

As expected the elements of the product of two information systems have two components corresponding to an element from each information system.

4.16 Proposition. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be information systems. Then*

$$x \in |\mathbf{A} \times \mathbf{B}| \Leftrightarrow out_0 x \in |\mathbf{A}| \ \ \& \ \ out_1 x \in |\mathbf{B}|.$$

4.17 Example. Now we know that the product construction $\times$ of information systems is continuous wrt $\trianglelefteq$ we can form information systems like $\mathbf{T}^{\infty}$ as the least fixed point of the continuous operation $X \mapsto \mathbf{T} \times X$, where $\mathbf{T}$ is the information for truth values. The domain of its elements is isomorphic to infinite sequences of elements of $\mathbf{T}$ ordered coordinatewise.

From Scott's work it is well-known that the continuous functions between domains themselves form a domain when ordered pointwise. Using information systems the function space is represented by the following construction which has the approximable mappings between two information systems as its elements.

4.18 Definition. Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B)$ be information systems. Define their *function space*, $\mathbf{A} \to \mathbf{B}$, to be the information system $\mathbf{C} = (C, \mathrm{Con}, \vdash)$ where:

(i) $C = \mathrm{Con}_A \times \mathrm{Con}_B$

(ii) $\{(X_0, Y_0), \ldots, (X_{n-1}, Y_{n-1})\} \in \mathrm{Con} \Leftrightarrow$
$\forall I \subseteq \{0, \ldots, (n-1)\}. \bigcup\{X_i \mid i \in I\} \in \mathrm{Con}_A \Rightarrow \bigcup\{Y_i \mid i \in I\} \in \mathrm{Con}_B$

(iii) $\{(X_0, Y_0), \ldots, (X_{n-1}, Y_{n-1}\} \vdash (X, Y) \Leftrightarrow \bigcup\{Y_i \mid X \vdash_A X_i\} \vdash_B Y.$

4.19 Theorem. *The operation* $\to$ *is a continuous operation on information systems ordered by* $\trianglelefteq$.

Proof. Firstly we require that the $\to$–operation produces an information system. Let $\mathbf{A}$ and $\mathbf{B}$ be information systems. We should check that $\mathbf{A} \to \mathbf{B}$ is an information system. The more difficult conditions are (iii) and (v) which we verify, leaving the others to the reader:

(iii) Suppose $\{(X_0, Y_0), \ldots, (X_{n-1}, Y_{n-1})\} \vdash (X, Y)$. We require

$$\{(X_0, Y_0), \ldots, (X_{n-1}, Y_{n-1}), (X, Y)\} \in \mathrm{Con}.$$

Thus we require that if $J \subseteq \{0, \ldots, n-1\}$ and $\bigcup\{X_j \mid j \in J\} \cup X \in \mathrm{Con}_A$ then $\bigcup\{Y_j \mid j \in J\} \cup Y \in \mathrm{Con}_B$. Assume $\bigcup\{X_j \mid j \in J\} \cup X \in \mathrm{Con}_A$. Then $\bigcup\{X_j \mid j \in J\} \cup \bigcup\{X_i \mid X \vdash_A X_i\} \in \mathrm{Con}_A$. Now as $\{(X_0, Y_0), \ldots, (X_{n-1}, Y_{n-1})\} \in \mathrm{Con}$ this makes $\bigcup\{Y_j \mid j \in J\} \cup \bigcup\{Y_i \mid X \vdash_A X_i\} \in \mathrm{Con}_B$. But $\bigcup\{Y_i \mid X \vdash_A X_i\} \vdash_B Y$, because

$\{(X_0,Y_0),\ldots,(X_{n-1},Y_{n-1})\} \vdash (X,Y)$. Consequently $\bigcup\{Y_j \mid j \in J\} \cup \bigcup\{Y_i \mid X \vdash_A X_i\} \vdash_B Y$ so $\bigcup\{Y_j \mid j \in J\} \cup Y \in \mathrm{Con}_B$, as required to verify (iii).

(v) Suppose $\{(X_0,Y_0),\ldots,(X_{n-1},Y_{n-1})\} \vdash \{(Z_0,V_0),\ldots,(Z_{m-1},V_{m-1})\} \vdash (U,W)$. We require $\{(X_0,Y_0),\ldots,(X_{n-1},Y_{n-1})\} \vdash (U,W)$ *i.e.*

$$\bigcup\{Y_i \mid U \vdash_A X_i\} \vdash W.$$

Suppose $U \vdash_A Z_j$. Then because $\bigcup\{Y_i \mid Z_j \vdash_A X_i\} \vdash_B V_j$ we have $\bigcup\{Y_i \mid U \vdash_A X_i\} \vdash_B V_j$. Therefore $\bigcup\{Y_i \mid U \vdash_A X_i\} \vdash_B \bigcup\{V_j \mid U \vdash_A Z_j\} \vdash_B W$. By the transitivity of $\vdash_B$ we obtain the required result, and have verified (v).

Satisfied that $\to$ is an operation on information systems we now show it is continuous with respect to $\trianglelefteq$ in each argument separately. We use lemma 3.11.

First we show $\to$ is monotonic in its first argument. Suppose $\mathbf{A} \trianglelefteq \mathbf{A}'$ and $\mathbf{B}$ are information systems. Write $\mathbf{C} = (C, \mathrm{Con}, \vdash) = \mathbf{A} \to \mathbf{B}$ and $\mathbf{C}' = (C', \mathrm{Con}', \vdash') = \mathbf{A}' \to \mathbf{B}$. We require $\mathbf{C} \trianglelefteq \mathbf{C}'$ so we check conditions (i), (ii), (iii) of 3.1 hold:

(i) Clearly $C = \mathrm{Con}_A \times \mathrm{Con}_B \subseteq \mathrm{Con}_{A'} \times \mathrm{Con}_B = C'$.

(ii) Let $(X_0,Y_0),\ldots,(X_{n-1},Y_{n-1})$ be tokens of $\mathbf{C}$. Because $\mathbf{A} \trianglelefteq \mathbf{A}'$ we have $\bigcup_{i\in I} X_i \in \mathrm{Con}_A$ iff $\bigcup_{i\in I} X_i \in \mathrm{Con}'_A$, for a subset $I \subseteq \{0,\ldots,(n-1)\}$. So inspecting the definition of the consistency predicate for the $\to$ construction we see that $\{(X_0,Y_0),\ldots(X_{n-1},Y_{n-1})\} \in \mathrm{Con}$ iff $\{(X_0,Y_0),\ldots(X_{n-1},Y_{n-1})\} \in \mathrm{Con}'$.

(iii) Suppose $(X_0,Y_0),\ldots(X_{n-1},Y_{n-1})$ and (X,Y) are tokens of $\mathbf{C}$. Because $\mathbf{A} \trianglelefteq \mathbf{A}'$ we have $X \vdash_A X_i$ iff $X \vdash_{A'} X_i$. So inspecting the definition of the entailment relation for the $\to$ construction we see that $\{(X_0,Y_0),\ldots(X_{n-1},Y_{n-1})\} \vdash (X,Y)$ iff $\{(X_0,Y_0),\ldots(X_{n-1},Y_{n-1})\} \vdash' (X,Y)$.

Thus $\mathbf{C} \trianglelefteq \mathbf{C}'$ so $\to$ is monotonic in its first argument.

Now we show $\to$ is continuous on token-sets in its first argument. Let $\mathbf{A}_0 \trianglelefteq \mathbf{A}_1 \trianglelefteq \ldots \trianglelefteq \mathbf{A}_i \trianglelefteq \ldots$ be an ω-chain of information systems $\mathbf{A}_i = (A_i, \mathrm{Con}_i, \vdash_i)$. Let (X,Y) be a token of $(\bigcup_i \mathbf{A}_i) \to \mathbf{B}$. Then $X \subseteq^{fin} \bigcup_i A_i$ is a consistent set of tokens from $\bigcup_i \mathbf{A}_i$. Being finite $X \subseteq A_i$ for some $i \in \omega$. But then $X \in \mathrm{Con}_i$ so (X,Y) is a token of $\mathbf{A}_i \to \mathbf{B}$. Thus as required (X,Y) is a token of $\bigcup_i(\mathbf{A}_i \to \mathbf{B}_i)$.

By lemma 3.11 we deduce that $\to$ is continuous in its first argument. A similar but even simpler argument shows that $\to$ is continuous in its second argument too, and therefore it is continuous. ∎

4.20 Proposition. *Let* $\mathbf{A}$ *and* $\mathbf{B}$ *be information systems. The elements of* $\mathbf{A} \to \mathbf{B}$ *are precisely the approximable mappings from* $\mathbf{A}$ *to* $\mathbf{B}$.

Remark. We could have constructed an isomorphic information system for the function space $\mathbb{A} \to \mathbb{B}$ using tokens $\mathrm{Con}_A \times B$ instead of $\mathrm{Con}_A \times \mathrm{Con}_B$. For this alternative construction the elements would only be in 1–1 correspondence, and not coincide precisely, with the approximable mappings. But of course that is not a problem.

As remarked in [S] it is the fact that products and function spaces exist, along with some mappings natural to them, that makes information systems and approximable mappings form a cartesian closed category. But we have other constructions too. We can now give definitions of information systems by composing the operations $(-)_\uparrow$, $+$, $\oplus$, $\times$, $\to$, tupling and projection starting from the information system $\underline{\perp}$. Because these operations are all continuous with respect to $\trianglelefteq$ the definitions can be recursive. Clearly these constructions can be used to give a semantics to a language with which to define datatypes. There are some well-known operations we have not mentioned yet like the Hoare and Smyth powerdomains, strict function space and smash product. Their definition is left to the reader (see [S]). Any operation on information systems, provided it is continuous with respect to $\trianglelefteq$, can be used in the recursive definition of information systems. One is free to define and use them.

4.21 Example. Let $\mathbb{A}$ be some fixed information system. The operation $X \mapsto \mathbb{A} \oplus (X \to X)$ is a continuous operation on information systems. It has a least fixed point $\mathbb{D} = \mathbb{A} \oplus (\mathbb{D} \to \mathbb{D})$. This information system, and its associated domain, gives a nontrivial model for the λ–calculus with atoms (see *e.g.* [St] or [P]).

5. Effectively given information systems.

At present information systems do not support a notion of computable element. This in turn rests on what it means for an information system to be given effectively. In this section we define effectively given information systems, their computable elements, and effective morphisms between effectively given information systems and so begin to make the work of the previous sections effective.

We restrict ourselves to information systems with a countable set of tokens. We shall code the tokens of an information system as non-negative integers. We can code pairs via the well-known 1–1 correspondence

$$\langle n, m\rangle = \frac{1}{2}(n+m)(n+m+1)+m$$

between pairs of integers (n, m) and integers $\langle n, m\rangle$. We code triples by $\langle p, \langle q, r\rangle\rangle$ which we abbreviate to $\langle p, q, r\rangle$. Similarly we can code an arbitrary finite sequence of non-negative integers $(p_0, \ldots, p_{n-1})$ as $\langle p_0, \ldots, p_{n-1}\rangle$. Regarding a finite set of integers as a binary numeral we can code finite sets via the 1–1 correspondence

$$\lceil X\rceil = \sum_{i\in X} 2^i$$

between finite sets of integers X and integers $\lceil X\rceil$. Note when X is null $\lceil X\rceil = 0$.

Throughout we will refer to fixed enumeration $\{\phi_n \mid n \in \omega\}$ of the partial recursive functions and an accompanying enumeration $\{W_n \mid n \in \omega\}$ of the recursively enumerable (r.e.) sets such that $i \in W_n$ iff $\phi_n(i) = 1$. We say n is the index of the partial recursive function ϕ_n and of the r.e. set W_n.

An effectively given information system is an information system with a function coding tokens as integers for which there are partial recursive functions which determine if an integer codes a token, whether or not a finite set of tokens is consistent and whether or not a consistent set entails a token, via their codes.

6.1 Definition. Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an information system. A *coding map* for $\mathbf{A}$ is a 1–1 function $\mathrm{cd} : A \to \omega$.

Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an information system, cd a coding map for $\mathbf{A}$ and p, q, r non-negative integers. Say $(\mathbf{A}, \mathrm{cd})$ is *effectively given* by *index* $\langle p, q, r\rangle$ iff

(i) $a \in A \Leftrightarrow \phi_p(\mathrm{cd}(a)) = 1$,

(ii) For $X \subseteq^{fin} A$

$$X \in \mathrm{Con} \Leftrightarrow \phi_q(\lceil \mathrm{cd}X\rceil) = 1$$
$$X \notin \mathrm{Con} \Leftrightarrow \phi_q(\lceil \mathrm{cd}X\rceil) = 0,$$

(iii) For $X \subseteq^{fin} A$, $a \in A$

$$X \vdash a \Leftrightarrow \phi_r(\langle \lceil \mathrm{cd}X\rceil, \mathrm{cd}(a)\rangle) = 1$$
$$X \nvdash a \Leftrightarrow \phi_r(\langle \lceil \mathrm{cd}X\rceil, \mathrm{cd}(a)\rangle) = 0.$$

We say $(\mathbf{A}, \mathrm{cd})$ is an *effectively given information system.*

It is easy to see that the set of tokens, the consistency predicate and the entailment relation correspond to r.e. sets via the coding map.

5.2 Proposition. *Let $(A, Con, \vdash, cd)$ be an effectively given information system. The sets $\{cd(a) \mid a \in A\}$, $\{\lceil cdX \rceil \mid X \in Con\}$ and $\{(\lceil cdX \rceil, a) \mid X \vdash a\}$ are each r.e.*

5.3 Notation. We write $\mathbf{ISys}_e$ for the class of effectively given information systems. Write $\mathbf{ISys}_e{}^n$ for the class of n–tuples $(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1})$ of effectively given information systems; say $(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1})$ has index k if each $\mathbf{A}_i$ has index k_i and $k = \langle k_0, \ldots, k_i, \ldots, k_{n-1} \rangle$. We shall often abbreviate $(\mathbf{A}, \mathrm{cd})$ to $\mathbf{A}$, when $(\mathbf{A}, \mathrm{cd})$ is an effectively given information system, and as before often write an n–tuple of effectively given information systems simply as $\mathbf{A}$ when the context makes it clear.

Remark. The reader familiar with other approaches to effectively given domains may be a little surprised by the way we code tokens. Usually the finite elements A of an effectively given domain are enumerated by a map $\xi : \omega \to A$ and the effective structure is described with respect to this enumeration. While one can define effectively given information systems in a similar way (see [CDL]) the codings become messy—essentially because they do not respect inclusions of sets—and the advantages of the information system approach over that of dealing directly with effectively given domains is lost. Our way preserves the simplicity of the information–systems treatment of the previous sections, and can be proved to induce the usual notion of effectively given domains. A domain D is associated with the information system ISD. So of course we can say a domain D is effectively given if ISD is effectively given. But here we have to be a little careful; there is an established definition of effectively given domains and we must check our definition agrees. In the appendix we present the usual definition of effectively given domains and show how to pass from that notion to ours and back again.

We now define the computable elements of an effectively given information system. An element of an information system is thought of as the set of tokens which hold of a computation. Of course we expect the set of tokens describing a computation to be generated effectively. Thus here where tokens are coded by integers it is natural to take computable elements as those elements with associated sets of codes which are r.e.

5.4 Definition. Let $\mathbf{A} = (\mathbf{A}, \mathrm{cd})$ be an effectively given information system. An element x of $\mathbf{A}$ is *computable*, with index k, iff $\{\mathrm{cd}(a) \mid a \in x\}$ is r.e., with index k. We write $|\mathbf{A}|_c$ for the set of computable elements.

Naturally we expect every finite element to be computable and they are. There are several different ways of defining computability of an element. One way is to demand the finite approximations of the element be r.e. (as is done for example in [CDL]). Another is to insist a computable element is the limit of some an effective chain of finite elements (as in [P] for example). Fortunately they are easily proved equivalent to the definition we have given.

5.5 Proposition. *Let* $\mathbf{A}$ *be an effectively given information system. Then every finite element of* $\mathbf{A}$ *is computable.*

Proof. A finite element has the form $\overline{X} = \{a | X \vdash a\}$ for some $X \in \mathrm{Con}$. So $\{\mathrm{cd}(a) \mid a \in \overline{X}\} = \{\mathrm{cd}(a) \mid a \in A \ \& \ \phi_r(\langle \lceil \mathrm{cd}X \rceil, \mathrm{cd}(a)\rangle) = 1\}$ which is clearly r.e. ∎

5.6 Theorem. *Let* $(\mathbf{A}, \mathrm{cd})$ *be an effectively given information system. Then the following conditions on an element* x *of* $\mathbf{A}$ *are all equivalent:*

(i) x *is computable,*

(ii) $\{\lceil cdX \rceil \mid X \subseteq^{fin} x\}$ *is r.e.,*

(iii) there is a recursive function h *and a sequence of finite sets* $X_0, X_1, \ldots, X_i, \ldots$ *such that* $h(i) = \lceil \mathrm{cd}X_i \rceil$ *and* $\overline{X}_0 \subseteq \overline{X}_1 \subseteq \cdots \subseteq \overline{X}_i \cdots$ *with lub* x.

Proof.

$(i) \Rightarrow (ii)$ is obvious.

$(ii) \Rightarrow (iii)$: Let η be a recursive enumeration of $\{\lceil \mathrm{cd}Y \rceil \mid Y \subseteq^{fin} x\}$. Then $\eta_i = \lceil \mathrm{cd}Y_i \rceil$ for some unique Y_i. Inductively define $X_0 = Y_0$ and $X_{i+1} = X_i \cup Y_{i+1}$. Define $h(i) = \lceil \mathrm{cd}X_i \rceil$. Because $\cup$ is effective, h is recursive and is the required enumeration of a chain of finite elements.

$(iii) \Rightarrow (i)$: Assume condition (iii) holds. Clearly $a \in x$ iff a is entailed by some X_i in the chain. This makes the set of codes of x r.e. ∎

What is a reasonable idea of effective mapping between effectively given information systems? Certainly we expect the function space construction to be effective and so the function space of two effectively given information systems should be effectively given. Its computable elements should correspond to effective approximable mappings. This will be so when we define an approximable mapping between two effectively given information systems to be effective when the codes of its constituent pairs is r.e.

5.7 Definition. Let $(\mathbf{A}, \mathrm{cd}_A)$ and $(\mathbf{B}, \mathrm{cd}_B)$ be effectively given domains. An *effective approximable mapping* $r : (\mathbf{A}, \mathrm{cd}_A) \to_e (\mathbf{B}, \mathrm{cd}_B)$, with index k, is an approximable mapping $r : \mathbf{A} \to \mathbf{B}$ such that

$$\{\langle \lceil \mathrm{cd}_A X \rceil, \lceil \mathrm{cd}_B Y \rceil \rangle \mid XrY\}$$

is r.e. with index k.

5.8 Proposition. *Effectively given information systems with effective approximable mappings form a category with composition the usual composition of relations and identities the entailment relation on the consistency predicate.*

The image of a computable element under an effective approximable mapping is computable itself—the proof is left to the reader.

5.9 Proposition. *Let $r : \mathbb{A} \to_e \mathbb{B}$ be an effective mapping between information systems. If $x \in |\mathbb{A}|_c$ then $|r|(x) \in |\mathbb{B}|_c$.*

Remark. There is a weaker notion of effectively given information system, and correspondingly of effectively given domains. Instead of taking consistency of a set of tokens to be recursive, take its complement, inconsistency, to be r.e and similarly take entailment to just be r.e. too. This would allow for example the language of first order arithmetic with the usual logical entailment and consistency to be an effectively given information system. We do not know how a theory based on this idea shapes-up. It would give a theory of effectively given domains different from the usual.

6. Effectively continuous operations on information systems.

We shall show our constructions on information systems induce effectively continuous constructions and how to solve recursive equations for information systems effectively. First we extend the subsystem order on information systems, $\trianglelefteq$, to effectively given information systems.

6.1 Definition. Let $\mathbf{A} = (\mathbf{A}, \mathrm{cd}_A)$ and $\mathbf{B} = (\mathbf{B}, \mathrm{cd}_B)$ be effectively given information systems. Define $(\mathbf{A}, \mathrm{cd}_A) \trianglelefteq_e (\mathbf{B}, \mathrm{cd}_B)$ iff $\mathbf{A} \trianglelefteq \mathbf{B}$ and $\mathrm{cd}_A = \mathrm{cd}_B \lceil A$, the restriction of the coding function of $\mathbf{B}$ to the tokens of $\mathbf{A}$. In this case we say $\mathbf{A}$ is an *effective subsystem* of $\mathbf{B}$.

We extend the effective subsystem relation to n–tuples as follows. For $(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1})$ and $(\mathbf{B}_0, \ldots, \mathbf{B}_{n-1})$ in $\mathbf{ISys}_e{}^n$ write

$$(\mathbf{A}_0, \ldots, \mathbf{A}_{n-1}) \trianglelefteq_e (\mathbf{B}_0, \ldots, \mathbf{B}_{n-1}) \Leftrightarrow \mathbf{A}_0 \trianglelefteq_e \mathbf{B}_0 \ \& \ \cdots \ \& \ \mathbf{A}_{n-1} \trianglelefteq_e \mathbf{B}_{n-1}.$$

The definition of effective subsystem clearly gives a partial order of effectively given information systems in $\mathbf{ISys}_e{}^n$. There is a least element, the n–tuple with each component the null information system with the null coding, and the order is effectively complete in the sense that all effectively given chains of effectively given information systems have a least upper bound, which is itself effectively given. We need a definition to state this formally.

6.2 Definition. A chain $\mathbf{A}_0 \trianglelefteq_e \mathbf{A}_1 \trianglelefteq_e \ldots \trianglelefteq_e \mathbf{A}_i \trianglelefteq_e \ldots$ in $\mathbf{ISys}_e{}^n$ of n–tuples of effectively given information systems is an *effective chain*, with index k, iff for all $i \in \omega$ the n–tuple of effectively given information systems $\mathbf{A}_i$ has index $\phi_k(i)$.

6.3 Theorem. *The relation $\trianglelefteq_e$ is a partial order of effectively given information systems. There is a least element $\perp_e = (\perp, \emptyset)$ with respect to $\trianglelefteq_e$. Let $\mathbf{A}_0 \trianglelefteq_e \mathbf{A}_1 \trianglelefteq_e \ldots \trianglelefteq_e \mathbf{A}_i \trianglelefteq_e \ldots$ be an effective chain of effectively given information systems $(\mathbf{A}_i, cd_i)$. Then $(\bigcup_i \mathbf{A}_i, \bigcup_i cd_i)$ is an effectively given information system which is the least upper bound of the chain with respect to $\trianglelefteq_e$.*

Similarly $\mathbf{ISys}_e{}^n$ has a least element and least upper bound of effective chains; in each coordinate they are defined as above.

Proof. We prove the existence of lubs of effective chains in $\mathbf{ISys}_e$, leaving the remaining proof to the reader.

Let $\mathbf{A}_0 \trianglelefteq_e \mathbf{A}_1 \trianglelefteq_e \ldots \trianglelefteq_e \mathbf{A}_i \trianglelefteq_e \ldots$ be an effective chain with index k. Write $\mathbf{A}_i = (A_i, \mathrm{Con}_i, \vdash_i, \mathrm{cd}_i)$ and $\mathbf{A} = (A, \mathrm{Con}, \vdash, \mathrm{cd}) = (\bigcup_i A_i, \bigcup_i \mathrm{Con}_i, \bigcup_i \vdash_i, \bigcup_i \mathrm{cd}_i)$ for the chain's tentative lub.

By assumption $\phi_k(i) = \langle p_i, q_i, r_i \rangle$ is an index for $\mathbf{A}$. We describe how to obtain an index $\langle p, q, r \rangle$ for $\mathbf{A}$.

Clearly $n \in \mathrm{cd}A$ iff $n \in \mathrm{cd}A_i$ for some i. As $n \in \mathrm{cd}A_i$ is r.e. in n using the machines indexed p_i, the predicate $n \in \mathrm{cd}A$ is r.e.. Take p to be the index of an associated machine.

Let $X \subseteq^{fin} A$. Then $X \subseteq A_i$ for some i and such an i can be determined effectively; run each machine indexed p_i on input $\lceil \mathrm{cd}X \rceil$ until one terminates successfully. Now decide whether or not $X \in \mathrm{Con}_i$ using the machine indexed q_i on input $\lceil \mathrm{cd}X \rceil$. If it is $X \in \mathrm{Con}_i$ and otherwise not. This procedure is clearly effective in $\lceil \mathrm{cd}X \rceil$. Take q to be the index of the corresponding machine.

In a similar way a machine, with index r, can be constructed to decide whether or not $X \vdash a$, for $X \subseteq^{fin} A$ and $a \in A$, by examining the input $\langle \lceil \mathrm{cd}X \rceil, \mathrm{cd}(a) \rangle$. ∎

6.4 Notation. Write $\bigcup \mathbf{X}$ for the lub of a set of n–tuples of information systems $\mathbf{X} \subseteq \mathbf{ISys}_e{}^n$ when it exists.

All the operations we defined previously, like product and function space, extend to operations on effectively given information systems. They are effective operations in the sense that they can be associated with recursive functions on the indices for the effectively given information systems. We have seen that they are all continuous with respect to $\trianglelefteq$. Consequently their effective versions are continuous on effective chains. We state the properties more formally.

6.5 Definition.

Let $\mathbf{F} : \mathbf{ISys}_e{}^m \to \mathbf{ISys}_e{}^n$ be an operation on effectively given information systems.

Say $\mathbf{F}$ is an *effective operation* with index k iff whenever the m–tuple $\mathbf{A}$ is effectively given with index l then $\mathbf{F}(\mathbf{A})$ is an effectively given n–tuple with index $\phi_k(l)$.

Say $\mathbf{F}$ is *monotonic* with respect to $\trianglelefteq_e$ iff $\mathbf{A} \trianglelefteq_e \mathbf{B} \Rightarrow \mathbf{F}(\mathbf{A}) \trianglelefteq_e \mathbf{F}(\mathbf{B})$ for all m–tuples $\mathbf{A}$ and $\mathbf{B}$ of effectively given information systems.

Say $\mathbf{F}$ is *effectively continuous* iff $\mathbf{F}$ is an effective operation which is monotonic wrt $\trianglelefteq_e$ and whenever $\mathbf{A}_0 \trianglelefteq_e \mathbf{A}_1 \trianglelefteq_e \ldots \trianglelefteq_e \mathbf{A}_i \trianglelefteq_e \ldots$ is an effective chain in $\mathbf{ISys}_e{}^m$ then $\mathbf{F}(\bigcup_i \mathbf{A}_i) = \bigcup_i \mathbf{F}(\mathbf{A}_i)$. (Note the rhs exists as it is the lub of an effective chain.)

6.6 Proposition. *Effectively continuous operations are closed under composition.*

We extend the constructions of lifting, sum, product and function space to operations on effectively given information systems simply by acting on the coding functions in a way that reflects the operation on the token sets. (Refer to the constructions of section 4.)

6.7 Definition.

Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A, \mathrm{cd}_A)$ and $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B, \mathrm{cd}_B)$ be effectively given information systems.

Effective lifting: Define $\mathbf{A}_{\uparrow_e}$ to be $((A, \mathrm{Con}_A, \vdash_A)_\uparrow, \mathrm{cd})$ where

$$\mathrm{cd}(a') = \begin{cases} \langle 0,0 \rangle & \text{if } a' = (0,0) \\ \langle 1, \mathrm{cd}_A(a) \rangle & \text{if } a' = (1,a). \end{cases}$$

Effective sum: Define $\mathbf{A} +_e \mathbf{B} = (\mathbf{A} + \mathbf{B}, \mathrm{cd})$ where

$$\mathrm{cd}(c) = \begin{cases} \langle 0, \mathrm{cd}_A(a)\rangle & \text{if } \exists a \in A.\ c = (0,a) \\ \langle 1, \mathrm{cd}_B(b)\rangle & \text{if } \exists b \in B.\ c = (1,b). \end{cases}$$

Effective product: Define $\mathbf{A} \times_e \mathbf{B} = (\mathbf{A} \times \mathbf{B}, \mathrm{cd})$ where

$$\mathrm{cd}(c) = \begin{cases} \langle 0, \mathrm{cd}_A(a)\rangle & \text{if } \exists a \in A.\ c = (0,a) \\ \langle 1, \mathrm{cd}_B(b)\rangle & \text{if } \exists b \in B.\ c = (1,b). \end{cases}$$

Effective function space: Define $\mathbf{A} \to_e \mathbf{B} = (\mathbf{A} \to \mathbf{B}, \mathrm{cd})$ where

$$\mathrm{cd}(X,Y) = \langle \lceil \mathrm{cd}_A X \rceil, \lceil \mathrm{cd}_B Y \rceil \rangle.$$

6.8 Theorem. *The operations $(-)_\uparrow$, $+_e$, and $\to_e$ define effectively given information systems and are themselves effective. Each operation is effectively continuous.*

Proof. We illustrate in a moderate amount of detail how to obtain an index for $\mathbf{A} \to_e \mathbf{B}$ effectively from indices for $\mathbf{A}$ and $\mathbf{B}$, showing that $\to_e$ is effective.

Let $\mathbf{A} = (A, \mathrm{Con}_A, \vdash_A, \mathrm{cd}_A)$ range over effectively given information systems with index $k = \langle k_0, k_1, k_2\rangle$. Let $\mathbf{B} = (B, \mathrm{Con}_B, \vdash_B, \mathrm{cd}_B)$ range over effectively given information systems with index $l = \langle l_0, l_1, l_2\rangle$. Write $\mathbf{A} \to_e \mathbf{B} = (C, \mathrm{Con}, \vdash, \mathrm{cd})$.

There is a partial recursive function such that

$$\begin{aligned} f(k,l,m) = 1 \Leftrightarrow \exists X, Y \subseteq^{fin} \omega.\ n = \langle \lceil X \rceil, \lceil Y \rceil \rangle \\ \&\ (\forall n \in X.\ \phi_{k_0}(n) = 1\ \&\ \phi_{k_1}(\lceil X \rceil) = 1) \\ \&\ (\forall n \in Y.\ \phi_{l_0}(n) = 1\ \&\ \phi_{l_1}(\lceil Y \rceil) = 1). \end{aligned}$$

Then by the s–m–n theorem (see [C]) there is a total recursive function p such that $\phi_{p(k,l)}(m) = f(k,l,m)$ so

$$\phi_{p(k,l)}(m) = 1 \Leftrightarrow \exists c \in C.m = \mathrm{cd}(c).$$

There is a partial recursive g such that when m has the form

$$m = \lceil \langle \lceil X_0 \rceil, \lceil Y_0 \rceil \rangle, \cdots, \langle \lceil X_{n-1} \rceil, \lceil Y_{n-1} \rceil \rangle \rceil$$

for $X_0, \ldots, X_{n-1} \subseteq^{fin} \mathrm{cd}_A A$ and $Y_0, \cdots, Y_{n-1} \subseteq^{fin} \mathrm{cd}_B B$ we have:

$$g(k,l,m) = \begin{cases} 1 & \text{if } \forall I \subseteq \{0, \ldots, n-1\}.\phi_{k_1}(\lceil \bigcup_{i \in I} X_i \rceil = 1 \Rightarrow \phi_{l_1}(\lceil \bigcup_{i \in I} Y_i \rceil) = 1 \\ 0 & \text{otherwise.} \end{cases}$$

By the s–m–n theorem there is a recursive function q such that $\phi_{q(k,l)}(m) = g(k,l,m)$. Then

$$\phi_{q(k,l)}(\lceil \mathrm{cd}Z \rceil) = \begin{cases} 1 & \text{if } Z \in \mathrm{Con} \\ 0 & \text{if } Z \subseteq^{fin} C \ \ \& \ \ Z \notin \mathrm{Con}. \end{cases}$$

Similarly there is a recursive function r such that

$$\phi_{r(k,l)}((\lceil \mathrm{cd}Z \rceil, \mathrm{cd}(c))) = \begin{cases} 1 & \text{if } X \vdash c \\ 0 & \text{if } X \nvdash c \end{cases}$$

when $Z \subseteq^{fin} C$ and $c \in C$.

Thus there is total recursive function $h : \langle k,l \rangle \mapsto \langle p(k,l), q(k,l), r(k,l) \rangle$ taking indices of effectively given information systems $\mathbf{A}$, $\mathbf{B}$ to an index of $\mathbf{A} \to_e \mathbf{B}$. Thus $\to_e$ is an effective operation with index any index of h.

Once it is shown the operations are effective, it follows from the fact that they are continuous wrt $\trianglelefteq$ and the simple way codings are defined that the operations are effectively continuous (or see the next theorem). ∎

It is easy to see that an operations like $\to_e$ is effectively continuous because it is effective and $\to$ is continuous. The next theorem shows that an effective operation is continuous provided it is monotonic—a consequence of the Rice–Shapiro theorem.

6.9 Theorem. *Let $\mathbf{F} : \mathbf{ISys}_e{}^m \to \mathbf{ISys}_e{}^n$ be an operation on information systems. Then $\mathbf{F}$ is effectively continuous iff it is effective and monotonic with respect to $\trianglelefteq_e$.*

Proof.

"$\Rightarrow$" follows directly from the definition of effectively continuous.

"$\Leftarrow$" For simplicity assume $\mathbf{F} : \mathbf{ISys}_e \to \mathbf{ISys}_e$; the proof is similar when $\mathbf{F}$ is a more general operation and is left to the reader.

Suppose $\mathbf{F}$ is monotonic wrt $\trianglelefteq_e$ and effective with index k. Let $\mathbf{A}_0 \trianglelefteq_e \mathbf{A}_1 \trianglelefteq_e \ldots \trianglelefteq_e \mathbf{A}_i \trianglelefteq_e \ldots$ be an effective chain of information systems. By monotonicity we know $\bigcup_i \mathbf{F}\mathbf{A}_i \trianglelefteq_e \mathbf{F}(\bigcup_i \mathbf{A}_i)$. We show every token of $\mathbf{F}(\bigcup \mathbf{A}_i)$ is a token of $\bigcup_i \mathbf{F}\mathbf{A}_i$. It then follows that $\bigcup_i \mathbf{F}\mathbf{A}_i = \mathbf{F}(\bigcup_i \mathbf{A}_i)$, thus establishing the continuity of $\mathbf{F}$.

Let c be a token of $\mathbf{F}(\bigcup_i \mathbf{A}_i)$. We show $c \in \mathbf{F}\mathbf{A}_i$, for some i, by using the Rice–Shapiro theorem (see [C]) which says:

If $\mathcal{W}$ is a subset of r.e. sets such that $\{m \mid W_m \in \mathcal{W}\}$ is r.e. then

$$w \in \mathcal{W} \Leftrightarrow \exists w_0 \subseteq^{fin} w.\ w_0 \in \mathcal{W}.$$

However in order to apply the Rice–Shapiro theorem we first show how we can pass effectively from arbitrary r.e. sets to effectively given information systems and, again effectively, from effectively given information systems to r.e. sets.

We associate an r.e. set effectively with an effectively given information system as follows. Let $\mathbf{A} = (A, \mathrm{Con}, \vdash)$ be an effectively given information system. Define

$$S\mathbf{A} = \{\langle 0, n\rangle \mid n \in \mathrm{cd}A\} \cup \{\langle 1, \lceil X\rceil\rangle \mid X \in \mathrm{Con}\} \cup \{\langle 2, \langle \lceil X\rceil, \mathrm{cd}(n)\rangle\rangle \mid X \vdash a\}.$$

Because this is effective there is a recursive function s such that if $\mathbf{A}$ has index l then $S\mathbf{A}$ has index $s(l)$.

Now we sketch how to obtain an effectively given information system from an r.e. set effectively. Let w be an r.e. set. Define Iw to be the following effectively given information system. As its set of tokens take $\{n \mid \langle 0, n\rangle \in w\}$ with codes $\mathrm{cd}(n) = n$. Take its consistency predicate, Con, and its entailment relation, $\vdash$, to be the least sets such that

$$\{X \mid \langle 1, \lceil X\rceil\rangle \in w\} \subseteq \mathrm{Con},$$
$$\{(X, n) \mid \langle 2, \langle \lceil X\rceil, n\rangle\rangle \in w\} \subseteq \vdash,$$

and $(\{n \mid \langle 0, n\rangle \in w\}, \mathrm{Con}, \vdash)$ is an information system; the sets Con and $\vdash$ can be built-up inductively "closing-up" under the conditions (i)—(iv) required of an information system in 1.1. Because this process is effective there is a recursive function ι such that $\iota(m)$ is an index of the effectively given information system $I(W_m)$. Notice that if $\mathbf{A}$ is an information system with index l then $IS\mathbf{A}$ is essentially the information system $\mathbf{A}$ but with its original tokens replaced by their codes. For this reason $IS\mathbf{A}$ has index l (as well as $\iota s(l)$).

Now we can define $\mathcal{W}$ by:

$$\mathcal{W} = \{w \mid w \text{ is an r.e. set such that } \mathbf{F}(Iw) \text{ has } \mathrm{cd}(c) \text{ as a token}\}.$$

We have

$$W_m \in \mathcal{W} \Leftrightarrow \exists p, q, r.\ \phi_k(\iota(m)) = \langle p, q, r\rangle \ \ \& \ \ \phi_p(\mathrm{cd}(c)) = 1$$

which is r.e. in m *i.e.* $\{m \mid W_m \in \mathcal{W}\}$ is r.e.. We have assumed c is a token of $\mathbf{F}(\bigcup_i \mathbf{A}_i)$. Consequently $S(\mathbf{F}(\bigcup_i \mathbf{A}_i)) \in \mathcal{W}$. By the Rice–Shapiro theorem there is a finite set $F \in \mathcal{W}$ such that $F \subseteq S(\mathbf{F}(\bigcup_i \mathbf{A}_i)) \in \mathcal{W}$. But then $F \subseteq S\mathbf{A}_i$ for some i, which implies $\mathbf{A}_i$ has c as a token. Thus $\bigcup_i \mathbf{F}\mathbf{A}_i = \mathbf{F}(\bigcup_i \mathbf{A}_i)$. This completes the proof that $\mathbf{F}$ is effectively continuous. ∎

We show that effectively continuous operations on effectively given information systems have least fixed points which are themselves effectively given.

6.10 Theorem. *Let $\mathbf{F}$ be an effectively continuous operation on effectively given information systems. Then $\bigcup_{i\in\omega} \mathbf{F}^i(\underline{\perp}_e, \cdots, \underline{\perp}_e)$ is an effectively given information system and is the least fixed point of $\mathbf{F}$.*

Proof.

To simplify the presentation assume $\mathbb{F}$ is a unary operation.

Clearly the chain $\underline{\perp}_e \trianglelefteq_e \mathbb{F}(\underline{\perp}_e) \trianglelefteq_e \cdots \trianglelefteq_e \mathbb{F}^i(\underline{\perp}_e) \trianglelefteq_e \cdots$ is effective. Since $\mathbb{F}$ is effectively continuous the lub of the chain is the least fixed point of $\mathbb{F}$ by the standard argument. ∎

Information systems could form the datatypes of a programming language. They would be defined by constructions like lifting, $+$, $\times$, $\rightarrow$ and fix for recursively defined datatypes. The datatypes would possess operations natural to them including least fixed point operators; for example a product would have projections and a function space would have application. These could be extended to a language for defining elements. Because all our constructions are effective, in principle we have an implementation too.

7. Conclusion, related work.

We have presented a simple treatment of recursively defined information systems and through them an elementary treatment of recursive domain equations. And this was made effective too. We turn to the relation with other work and where we think(!) the future lies.

It is a pain working with indices. Recently David McCarty has shown how to do the theory of effectively given domains inside a realisability model of constructive set theory [Mc, Mc1]. One immediate advantage of his approach is that the work on indices is done once and for all in building the realisability model. McCarty's approach seems very general; it appears to work for a much wider category of cpos than just domains. This seems a good area to look for a smoother, more uniform treatment of effective structures.

Although information systems carry a strong intuition, as a category, with approximable mappings, they are equivalent to category of domains, with continuous functions. From the point of view of categorical constructions they say no more than domains; they are just a representation of the old category of domains. (Yes, the cpo of information systems was useful but it did nothing that could not have been done before using the more abstract method of inverse limits or retracts on a universal domain.) But information systems do have potentially more structure than domains and quite likely there are morphisms on them which can respect this greater detail. With new morphisms come new constructions, constructions that might take into account the internal structure of tokens. If we are to take seriously the idea that tokens are assertions, or propositions, they had better have internal structure which is taken account of by the morphisms. There would be a strong point to this if it made closer the connection between denotational semantics and the proof of properties of programs, and could exploit the idea that the denotation of a programming construct can be the set of assertions which are provable of it.

In this paper we have said nothing about universal domains and the use of retractions to solve domain equations. We say a little about that approach and how it relates to what what we have described. The method was introduced by Dana Scott as an alternative to his inverse limit construction, firstly for algebraic lattices using $P\omega$ as the universal ω–algebraic lattice [S3], and later for domains in [S1,S2,S]. In [P2] Gordon Plotkin shows similar results for those domains which are coherent using T^ω. The results for the universal domain in [S1,S2,S] are easily stated in the framework of information systems, which will explain the sense in which the domain is *universal*. The universal domain used there—there are many others—is the domain of theories of the propositional calculus. This naturally comes along with the information system $\mathbb{P} = (P, \mathrm{Con}, \vdash)$ which consists of tokens the propositions built up from a countable set of propositional variables using logical connectives, $\vdash$ the obvious logical entailment, and Con the consistent sets—those which do not simultaneously entail a proposition and its negation. The information system $\mathbb{P}$ is universal in the sense that any countable information system can be found as a subsystem of it. And correspondingly any domain with a countable basis can be embedded in its associated domain, $|\mathbb{P}|$. Thus one can talk about arbitrary domains in terms of projection mappings on $|\mathbb{P}|$. Now because the

domain is universal its own function space embeds in it, so such projections can themselves be regarded as elements of $|\mathbb{P}|$, and functors on domains can be represented as operations on $|\mathbb{P}|$. In fact domain equations can be solved by taking least fixed points of operations on $|\mathbb{P}|$. While it takes a little trouble to set up this machinery one clear advantage is that effective functors are now treated in exactly the same way as effective functions because that is how they are represented in this approach.

Unfortunately the category **Dom** , and so **ISys** , does not have all the closure properties one could wish for. The Plotkin powerdomain of a domain need not be a domain. For this reason Plotkin proposed the category of SFP objects, a full subcategory of ω–algebraic cpos which includes ω–**Dom** (see [P1]). The category of SFP objects is closed under the Plotkin powerdomain functor and is cartesian–closed. In fact, recently Mike Smyth has proved a conjecture of Plotkin that it is the largest cartesian–closed full subcategory of the ω–algebraic cpos which affirms the naturalness of the category of SFP objects (see [Sm1]). Because SFP objects are a kind of algebraic cpo they can be represented as completions by ideals of certain kinds of preorder. Whether or not they have quite as natural a representation as information systems remains to be demonstrated. The work of Carl Gunter [G] should help here. There is a characterisation of the three powerdomain constructions, the Hoare, Smyth and the Plotkin powerdomain, in terms of modal assertions that can be made about non–deterministic computations [W2] but the Plotkin powerdomain does not quite fit into the scheme of information systems. (One can add disjunction to information systems and change the definition of element a little so it does, but at present it is clumsy and certainly not ready for the market.)

Information systems express the logical relations between assertions about a computation. On the other hand event structures [NPW, W, W1] express how the computation behaves in time, how for example one event causally depends on others. The techniques used for event structures are very close to those used for information systems. The $\trianglelefteq$ relation on event structures corresponds to Kahn and Plotkin's rigid embeddings [KP] and is used to define event structures recursively. There must surely be a natural synthesis of the two kinds of structure. There are already examples of the sort of thing we mean in the bidomains of Gérard Berry [B, W] and the structures in [Cu] but it would be pleasant to put all this in a general setting. It is also possible that the SFP objects can be represented naturally in this way.

Acknowledgements

Our debt to the work of Dana Scott is clear. We are grateful to C.A.R. Hoare for suggested improvements.

This work was supported in part by the Computer Science Department, Carnegie–Mellon University.

Appendix
The usual definition of effectively given domains related.

As explained the definition of effectively given domain induced by our definition of effectively given information system is not standard so in this appendix we relate our definition to the usual one. Usually an effectively given domain is defined in the following way—see *e.g.* [S1, P].

A1 Definition.

Let $D = (D, \sqsubseteq, \bot)$ be a domain. An *enumeration* of D is a function ξ from the non-negative integers onto the finite elements D^0. We write the finite element $\xi(n)$ as ξ_n.

An *effectively given domain with index* k is an enumeration of a domain (D, ξ) such that, writing $k = \langle i, j \rangle$:

(i) $\xi_0 = \bot$,

(ii) $\xi_m \uparrow \xi_n$ is recursive in m, n, with index i,

(iii) $\xi_m = \xi_n \sqcup \xi_{n'}$ is recursive in m, n, n', with index j.

An *effectively given domain* is an enumeration for which there is some such index k.

A *computable element* of an effectively given domain (D, ξ) is an element $x \in D$ for which there is an r.e. subset of integers W such that $x = \bigsqcup\{\xi_n \mid n \in W\}$. The set of computable elements is written as $(D, \xi)_c$.

A2 Proposition. *Let (D, ξ) be an effectively given domain. The following are equivalent:*

(i) $x \in (D, \xi)_c$,

(ii) *the set* $\{n \mid \xi_n \sqsubseteq x\}$ *is* r.e.,

(iii) *there is a recursive function* h *such that* $\xi_{h(0)} \sqsubseteq \xi_{h(1)} \sqsubseteq \cdots \sqsubseteq \xi_{h(i)} \sqsubseteq \cdots$ *and* $x = \bigsqcup_{i \in \omega} \xi_i$.

It is quite easy to see how an effectively given domain determines an effectively given information system. The proof uses this proposition (to be found in [P]).

A3 Proposition. *Let (D, ξ) be an effectively given domain. Then*

(i) $\{\xi_{n_0}, \ldots, \xi_{n_m}\}\uparrow$ *is recursive in* $\lceil n_0, \ldots, n_m \rceil$ *and*

(ii) $\xi_l \sqsubseteq \bigsqcup\{\xi_{n_0}, \ldots, \xi_{n_m}\}$ *is recursive in* $\langle \lceil n_0, \ldots, n_m \rceil, l \rangle$.

A4 Theorem. *Let (D, ξ) be an effectively given domain. Define* $cd(e) = \min\{n \mid \xi_n = e\}$ *for* $e \in D^0$. *Then (ISD, cd) is an effectively given information system such that*

$$x \in |(ISD, cd)|_c \Leftrightarrow \bigsqcup x \in (D, \xi)_c.$$

Proof.

Write $ISD = (D^0, \mathrm{Con}, \vdash)$.

An integer n is a code $\mathrm{cd}(e)$ of some finite element e iff $\xi_n \neq \xi_m$ for any $m < n$. This is clearly recursive.

By the above proposition, $\{\xi_{n_0}, \ldots \xi_{n_m}\}\uparrow$ is recursive in $\lceil n_0, \ldots, n_m \rceil$. Let q be an index of this recursive function. Then clearly

$$\phi_q(\lceil \mathrm{cd}X \rceil) = \begin{cases} 0 & \text{if } X \in \mathrm{Con} \\ 1 & \text{if } X \notin \mathrm{Con} \end{cases}$$

for $X \subseteq^{fin} D^0$.

By the above proposition, $\xi_l \sqsubseteq \bigsqcup\{\xi_{n_0}, \ldots \xi_{n_m}\}$ is recursive in $\langle \lceil n_0, \ldots, n_m \rceil, l \rangle$. Let r be an index of this recursive function. Then clearly

$$\phi_r(\langle \lceil \mathrm{cd}X \rceil, \mathrm{cd}(e) \rangle) = \begin{cases} 0 & \text{if } X \vdash a \\ 1 & \text{if } X \not\vdash a \end{cases}$$

for $X \subseteq^{fin} D^0$ and $e \in D^0$.

Thus (ISD, cd) is an effectively given information system.

Obviously if x is a computable element of ISD then $\mathrm{cd}x$ is r.e. so $\bigsqcup x$ is a computable element of D. Conversely suppose y is a computable element of D. Then $y = \bigsqcup \xi W$ for some r.e. set W making $\{e \sqsubseteq y \mid e \in D^0\}$ a computable element of ISD. ∎

It is a little harder to show the converse that an effectively given information system determines an equivalent effectively given domain.

A5 Theorem. *Let $(\mathbf{A}, cd)$ be an effectively given information system. There is an enumeration ξ such that $(|\mathbf{A}|, \xi)$ is an effectively given domain with the same computable elements as $(\mathbf{A}, cd)$.*

Proof. Let $(A, \mathrm{Con}, \vdash, \mathrm{cd})$ be an effectively given information system with index $\langle p, q, r \rangle$.

Because $\{\lceil \mathrm{cd}X \rceil \mid X \in \mathrm{Con}\}$ is a non–null r.e. set it has a recursive enumeration ζ. We can ensure $\zeta_0 = 0$. For each n, $\zeta_n = \lceil \mathrm{cd}X_n \rceil$ for a unique $X_n \in \mathrm{Con}$. Define $\xi_n = \overline{X}_n$.

We check that $(|\mathbf{A}|, \xi)$ is effectively given.

Clearly $\xi_0 = \overline{\emptyset} = \bot$.

To decide whether or not $\xi_m \uparrow \xi_n$ is a matter of deciding the consistency of $X_m \cup X_m$. This is achieved by running the machine indexed q on $\lceil \mathrm{cd}X_m \cup \mathrm{cd}X_n \rceil$, the code for their union. Clearly this is effective in m and n.

Similarly, to decide whether or not $\xi_m = \xi_n \sqcup \xi_{n'}$ is to decide whether or not every token in X_m is entailed by $X_n \cup X_{n'}$ and conversely, that every token in $X_n \cup X_{n'}$ is entailed by X_m. This is achieved effectively by running the machine indexed r on the obvious finite set of inputs.

Thus $(|\mathbf{A}|, \xi)$ is effectively given. Suppose x is a computable element of $(|\mathbf{A}|, \xi)$. Then $x = \bigsqcup \xi W$ for some r.e. set W. As ς is recursive, ςW is an r.e. subset of C. Thus x is a computable element of $\mathbf{A}$. Conversely suppose x is a computable element of $\mathbf{A}$. Then x is an r.e. subset of tokens. Thus $A = \{\mathrm{cd}X \mid X \subseteq x\}$ is an r.e. subset of C. But then $\varsigma^{-1}A$ is an r.e. subset of ω and clearly $x = \bigsqcup\{\xi_n \mid n \in \varsigma^{-1}A\}$. ∎

References

[A] Aczel, P., A note on Scott's theory of domains. Unpublished note, Math. Dept., Univ. of Manchester, (1983).

[B] Berry, G., Modeles Completement Adequats et Stables des λ–calculs typés. These de Doctorat dÉtat, Université Paris VII (1979).

[BC] Berry, G., and Curien, P–L., Sequential algorithms on concrete data structures. Report of Ecole Nationale Superieure des Mines de Paris, Centre de Mathematiques Appliquées, Sophia Antipolis (1981).

[C] Cutland, N. J., Computability, an introduction to recursive function theory. Cambridge University Press (1980).

[CDL] Coppo, M., Dezani, M., Longo, G., Applicative Information Systems. Proc. of CAAP '83, Springer Lecture Notes in Computer Science, vol. 159 (1983).

[Cu] Curien, P–L, Algorithmes sequentiels et extensionnalité. Rapport LITP 82–67 Université Paris VII (1982).

[Grä] Grätzer, G., Universal Algebra. Van Nostrand University series in Higher Mathematics (1968).

[G] Gunter, C., Ph.D. thesis. Math. Dept., University of Wisconsin, Madison, to appear (1985).

[KP] Kahn, G., and Plotkin, G., Domaines Concrètes. Rapport IRIA Laboria No. 336 (1978).

[Mac] Maclane, S., Categories for the working mathematician. Springer –Verlag Graduate texts in Math. (1971).

[Mc] McCarty, C., Information Systems, Continuity and Realizability. Notices of AMS, August issue (1983).

[Mc1] McCarty, C., Constructivity, Realizability and Recursive Mathematics. D. Phil. thesis, University of Oxford (to be submitted).

[NPW] Nielsen, M., Plotkin, G. and Winskel, G., Petri nets, event structures and domains. Theor. Comp. Sc. (1981)

[P] Plotkin, G. D., The category of complete partial orders: a tool for making meanings. Lecture notes, Pisa Summer School (1978).

[P1] Plotkin, G. D., A powerdomain construction. SIAM Journal on Computing, vol. 5, no. 3, p. 452-487 (1976)

[P2] Plotkin, G. D., T^{ω} as a universal domain. J. Comp. Sys. Sci., vol.17, pp. 209–236 (1978).

[S] Scott, D. S., Domains for Denotational Semantics. ICALP 1982.

[S1] Scott, D. S., Lectures on a mathematical theory of computation. Oxford University Computing Laboratory Technical Monograph PRG–19 (1981).

[S2] Scott, D. S., A Space of Retracts. Manuscript of a talk given at Bremen in November '79 (April 1980).

[S3] Scott, D. S., Data types as lattices. SIAM J. on Computing, vol.5, pp.522–587 (1976).

[S–D] Saheb–Djahromi, N., Probabilistic Non–Determinism. Report CSR–7–77 of the Dept. of Comp. Sc., University of Edinburgh (1977).

[St] Stoy, J. Denotational semantics: The Scott–Strachey approach to programming language theory. MIT Press (1977).

[Sm] Smyth, M., Effectively given domains. Theoretical Computer Science, vol. 5, pp. 257–274 (1977).

[Sm1] Smyth, M., The largest cartesian–closed category of domains. Theor. Comp. Sc. (1983).

[W] Winskel, G., Events in Computation. Ph.D. thesis, Comp. Sc. Dept., University of Edinburgh (1980).

[W1] Winskel, G., Event structure semantics of CCS and related languages. Report of comp. Sc. Dept., Aarhus University, Ny Munkegarde, 8000 Aarhus C, Denmark, and extended abstract in proc. ICALP 82 in Springer LNCS (1982).

[W2] Winskel, G., A note on powerdomains and modality. In proc. Foundations of Computation Theory, '83, Springer LNCS (1983).