

Semantics of PCF

and the full abstraction problem

Language, Operational Semantics and Models

Luca Paolini

paolini@di.unito.it

Università di Torino

Dipartimento di Informatica

August 28th – September 1st

Language	2
Types	3
λ -calculus	4
Constants	5
Scheme	6
Free Variables.	8
Context	9
Renaming	10
Substitution	11
Mechanism.	12
Syntax of PCF.	13
SOS	14
Evaluation of PCF.	15
Recursion.	16
Operational Theories	17
Syntactic Sugar.	18
Models	19
PreModel.	20
Model	21
Basic Properties	22
Full Abstraction	23
Adequacy.	24
Complete Partial Order	25
FixPoint.	26
CPO	27

Examples	28
Standard Model	29
Continuous function	30
CPO Properties	31
FixPoint.	33
Factorial	34
Approximants	35
Interpretation	36
Computability	37
Correctness	40
Domains	41
Domains	42
Examples	43
Scott-Domains	44
Compact	45
Algebraic	46
Bounded-Completeness	47
Domain	48
Step	49
Arrow-Domain	50
Correctness	51
Coherence Spaces	52
dI-Domain	53
Coherence Space.	54
Basic Properties	55
Stable Function.	56
Trace	57
Interpretation Revisited	60
Correctness	61
PCF+	62
Two Terms.	63
PCF+	64
Full Abstraction	65
Definable Compact	66
Rank.	67
Crisp Set	68
Definability.	69
Constructive Domains	70
Universality	71
StPCF	72
Two Terms.	73
StPCF.	74
A Uniform Notation.	75
Definable Cliques	76
Examples	77
Full Abstraction and Universality	78
Final Remarks	79

Models	80
Sequentiality.	81
Higher-Order Computability	82

References	83
-------------------	-----------

Types

$$\sigma ::= \iota \mid (\sigma \multimap \tau)$$

σ, τ, \dots are used as metavariables ranging over types of PCF

ι is the type of natural numbers

\multimap is the unique type constructor


\multimap associates to right, i.e. $\sigma_1 \multimap \sigma_2 \multimap \sigma_3 = \sigma_1 \multimap (\sigma_2 \multimap \sigma_3)$

Exercise 1

It is easy to see that all types are of the shape

$$\tau_1 \multimap \dots \multimap \tau_n \multimap \iota,$$

for some types τ_1, \dots, τ_n where $n \geq 0$.

 **Types**


The type ι is called the **ground** type and types $\sigma \multimap \tau$ are called **higher-order types** or **higher-types**.

In literature, often a type constant \circ representing booleans is added to type-syntax, for an explicit treatment of truth-values.

An introduction to Simple Typed λ -calculus can be found in [7]. This language forms the theoretical core of our language.

Sometimes, we will write $M : \sigma$ in order to say that σ is the type of M .

A typed λ -calculus

$$M^\sigma ::= x^\sigma \mid \text{const}^\sigma \mid (\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau} \mid (P^{\tau \rightarrow \sigma} Q^\tau)^\sigma$$

$M^\sigma, N^\sigma, P^\sigma, Q^\sigma, \dots$ are metavariables ranging over typed **terms**

Var^σ is the set of **variables** of type σ and $x^\sigma \in \text{Var}^\sigma$
while const^σ is a metavariable for **constant** symbols

$(\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau}$ is an **abstraction**,
and $(P^{\tau \rightarrow \sigma} Q^\tau)^\sigma$ is an **application**

Sometimes parentheses are **omitted**,
by respecting the following disambiguating conventions

- application associates to the left
- application binds more tightly than abstraction

Types of variables/constants/subterms/terms will be **omitted** when they are clear from the context or uninteresting

Exercise 2

Given types of all variables of a term M ,
there is a unique σ such that M^σ .

A typed λ -calculus

If $\lambda x^\sigma. M^\tau$ is an abstraction then the variable x^σ is the **formal parameter**
and M^τ is the **body** of the abstraction.

A term of the shape $(\lambda x^\sigma. M^\tau) N^\sigma$ is a **redex**, informally it is a “program” such that N^σ is the argument of the function $\lambda x^\sigma. M^\tau$.

The language without constants is said **pure**.

Terms of the language $M ::= x \mid \text{const} \mid (\lambda x. N) \mid (PQ)$ are called **untyped**.

By forgetting all types of our terms we obtain an untyped term,
is the converse true?

Example 3

PQR should be parsed as $(PQ)R$, while $\lambda x^\sigma. MN$ should be parsed as $\lambda x^\sigma. (MN)$.

Silently, we will avoid the use of the same name for two variables with different type.

Moreover, $\lambda x^\sigma y^\tau z^\mu. M$ will be used as an abbreviation for $\lambda x^\sigma. \lambda y^\tau. \lambda z^\mu. M$.

Examples of Constants

- $\tilde{0}, \tilde{1}, \tilde{2}, \dots$ are examples of constants that we can add to our language, all with type ι
- succ, pred having type $\iota \rightarrow \iota$ are further examples
- or having type $\iota \rightarrow \iota \rightarrow \iota$ is a further example
- if having type $\iota \rightarrow \iota \rightarrow \iota \rightarrow \iota$ is a further example

Note that the **meaning** of constant is not defined!

We can **informally** suggest the evaluation of the language by a evaluation relation \Downarrow_e between terms, for example:

$$\begin{array}{c}
 \frac{}{\tilde{n} \Downarrow_e \tilde{n}} \text{ (num)} \\
 \\
 \frac{M \Downarrow_e \tilde{n}}{\text{succ } M \Downarrow_e \tilde{n}+1} \text{ (succ)} \\
 \\
 \frac{M \Downarrow_e \tilde{n}+1}{\text{pred } M \Downarrow_e \tilde{n}} \text{ (pred)} \qquad \frac{M \Downarrow_e \tilde{0}}{\text{pred } M \Downarrow_e ???} \text{ (pred)} \\
 \frac{M_0 \Downarrow_e \tilde{0} \quad M_1 \Downarrow_e \tilde{n}}{\text{or } M_0 M_1 \Downarrow_e \tilde{0}} \text{ (or)} \qquad \frac{M_0 \Downarrow_e \tilde{n} \quad M_1 \Downarrow_e \tilde{0}}{\text{or } M_0 M_1 \Downarrow_e \tilde{0}} \text{ (or)} \\
 \\
 \frac{M_0 \Downarrow_e \tilde{n}+1 \quad M_1 \Downarrow_e \tilde{m}+1}{\text{or } M_0 M_1 \Downarrow_e \tilde{1}} \text{ (or)} \\
 \\
 \frac{M_0 \Downarrow_e \tilde{0} \quad M_1 \Downarrow_e \tilde{n}}{\text{if } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (oif)} \qquad \frac{M_0 \Downarrow_e \tilde{k}+1 \quad M_2 \Downarrow_e \tilde{n}}{\text{if } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (lif)}
 \end{array}$$

Examples of Constants



Three acceptable solutions for the meaning of pred :

1. No rule for the case $M \Downarrow_e \tilde{0}$
2. An equivalent more explicit solution is the use of the rule

$$\frac{M \Downarrow_e \tilde{0} \quad \text{pred } M \Downarrow_e N}{\text{pred } M \Downarrow_e N} \text{ (pred)}$$

3. A last, different solution is the use of the rule

$$\frac{M \Downarrow_e \tilde{0}}{\text{pred } M \Downarrow_e \tilde{0}} \text{ (pred)}$$

The first two choices make “partial” the behaviour of pred , while the third makes it “total”. **For laziness, we choose the first solution!**

Programming Examples

A (useless) program computing the sum of natural numbers can be wrote in Scheme as follows:

```
(define (sum x y)
  (+ x y))
```

where `+` is a built-in operator. The previous code can be **curryfied** (elimination of n-ary argument functions) in the following way

```
(define sum
  (lambda (x)
    (lambda (y)
      (+ x y))))
```

Note that **sum** is simply a name for the following function

```
(lambda (x)
  (lambda (y)
    (+ x y)))
```

Luca Paolini: The Full Abstraction Problem

Summer School Chambéry-Torino, 2006 – 6 / 88

Programming Examples

Note also that the formal parameter of the program can be **renamed** without changing its meaning

```
(lambda (n)
  (lambda (m)
    (+ n m)))
```

But **clashes** (name collisions) must be avoided, in fact the following program is different from the previous one

```
(lambda (n)
  (lambda (n)
    (+ n n)))
```

Luca Paolini: The Full Abstraction Problem

Summer School Chambéry-Torino, 2006 – 7 / 88

A Programming Examples

A beautiful introduction to Scheme (a dialect of LISP) and functional programming can be found in [1]:

Structure and Interpretation of Computer Programs

by Abelson, Sussman, and Sussman

(<http://mitpress.mit.edu/sicp/full-text/book/book.html>).

Luca Paolini: The Full Abstraction Problem

Summer School Chambéry-Torino, 2006 – **NOTE**

Free Variables

Definition 4

The set of **free variables** of a term M , denoted by $FV(M)$, is inductively defined as follows:

- $M = x^\sigma$ implies $FV(M) = \{x^\sigma\}$,
- $M = \text{const}^\sigma$ implies $FV(M) = \emptyset$,
- $M = \lambda x^\sigma.M'$ implies $FV(M) = FV(M') - \{x^\sigma\}$,
- $M = PQ$ implies $FV(M) = FV(P) \cup FV(Q)$.

A variable is **bound** in M when it is not free in M .

Note that the λ -abstraction is the only **binder** of our language.

A term M is **closed** if and only if $FV(M) = \emptyset$,
otherwise M is said to be **open**.

Free Variables

In computer programming, mathematics, logics and in other disciplines involving formal languages a free variable is a notation for a place or places in an expression, into which some definite substitution may take place, or with respect to which some operation (summation or quantification, to give two examples) may take place. The idea is related to, but somewhat deeper and more complex than, that of a **placeholder** (a symbol that will later be replaced by some literal string), or a wildcard character that stands for an unspecified symbol.

Example 5

- $\sum_{x=1}^5 \log x = \sum_{y=1}^5 \log y$
- $\int e^x dx = \int e^y dy$
- $\forall x \exists z. P(x) = \forall y \exists z. P(y) \neq \forall z \exists z. P(z)$.

Context-Substitution

An auxiliary notion of replacement that does not take in account collision of variable names is presented before the renaming.

Definition 6

The **context-substitution** is defined inductively as follows:

- $x^\sigma \{M^\sigma / x^\sigma\} = M^\sigma$
- $y^\tau \{M^\sigma / x^\sigma\} = y^\tau$ if $y^\tau \neq x^\sigma$
- $\text{const}^\tau \{M^\sigma / x^\sigma\} = \text{const}^\tau$
- $PQ \{M^\sigma / x^\sigma\} = P \{M^\sigma / x^\sigma\} Q \{M^\sigma / x^\sigma\}$
- $(\lambda y^\tau . P) \{M^\sigma / x^\sigma\} = \begin{cases} \lambda x^\sigma . P & \text{if } y^\tau = x^\sigma \\ \lambda y^\tau . P \{M^\sigma / x^\sigma\} & \text{if } y^\tau \neq x^\sigma \end{cases}$

Context-Substitution



The Context-Substitution of M to x is defined only when M and x have the same type.

Example 7

- $(\lambda x^\sigma y^\tau . uxy) \{z^\tau / x^\tau\} = \lambda x^\sigma y^\tau . uxy$
- $(\lambda x^\sigma y^\tau . uxz) \{y^\tau / z^\tau\} = \lambda x^\sigma y^\tau . uxy$
- $((\lambda x^\sigma y^\tau . uxy)x) \{z^\tau / x^\tau\} = (\lambda x^\sigma y^\tau . uxy)z$

Renaming

Terms are considered up to α -equivalence \equiv , namely a bound variable can be renamed provided no free variable is captured. More formally,

Definition 8

The **α -equivalence** \equiv is the least equivalence relation on term s.t.

- $x^\sigma \equiv x^\sigma$
- $\text{const}^\sigma \equiv \text{const}^\sigma$
- $PQ \equiv P'Q'$ if $P \equiv P'$ and $Q \equiv Q'$
- $\lambda x^\sigma . M \equiv \lambda y^\sigma . N$
if $M \{z^\sigma / x^\sigma\} \equiv N \{z^\sigma / y^\sigma\}$ for a **fresh** variable z

Renaming



It is well-known that

the name of formal parameter of a procedure is meaningless!

To say: a name can be changed by taking care to avoid collision/clash with other names.

By using a fresh (not already used) variable-name,
the context-substitution does not produce collision between names.

Note that the renaming can be applied only to bound variables!

Exercise 9

Show that there are terms M, N such that $M \equiv N$ and

$$M\{x^\sigma/y^\sigma\} \not\equiv N\{x^\sigma/y^\sigma\}.$$

Substitution

$M^T[N^\sigma/x^\sigma]$ denotes the **capture-free substitution** of all free occurrences of x^σ in M^T by N^σ . More formally,

Definition 10

The substitution $M[N/x]$ is a term α -equivalent to the the context-substitution $P\{N/x\}$ term where P is a term α -equivalent to M such that no bound variable of P is in $FV(N)$

The notion of substitution is crucial in order to formalize the main evaluation rule of our language, the β -reduction.

Substitution



The Substitution of M to x is **defined only when** M and x have the same type.

Example 11

- $(\lambda x^\sigma y^\tau. uxy)[z^\tau/x^\tau] = \lambda x^\sigma y^\tau. uxy$
- $(\lambda x^\sigma y^\tau. uxz)[y^\tau/z^\tau] = \lambda x^\sigma w^\tau. uxy$
- $((\lambda x^\sigma y^\tau. uxy)x)[z^\tau/x^\tau] = (\lambda w^\sigma y^\tau. uwy)z$

Parameter Passing Mechanism

Exercise 12

It is easy to check that every term M^σ has the following shape:

$$\lambda x_1 \dots x_n. \zeta M_1 \dots M_m \quad (n, m \geq 0),$$

where $M_i \in \Lambda$ are the **arguments** of M^σ ($1 \leq i \leq m$)
and ζ is the **head** of M^σ .

Note that ζ is

- either a variable,
- or a constant,
- or an application of the shape $(\lambda z.P)Q$ called **head-redex**.

The main **evaluation relation** \Downarrow_e of our language will be

$$\frac{P[Q/x]M_1 \dots M_m \Downarrow_e \tilde{n}}{(\lambda x^\sigma.P)QM_1 \dots M_m \Downarrow_e \tilde{n}} \text{ (head)}$$

The rule above implements a **call-by-name** parameter passing mechanism, since the arguments of abstractions are substituted without being evaluated.

Parameter Passing Mechanism

The rule (*head*) presented before is often replaced by

$$\frac{M \Downarrow_e \lambda x.P \quad P[N/x] \Downarrow_e \tilde{n}}{MN \Downarrow_e \tilde{n}} \text{ (cbn)}$$

and some further rules for the evaluation of non-ground terms.

Usually PCF is presented by giving a type assignment system for an untyped language. The two presentations are equivalent in our perspective.

A different parameter passing mechanism is the call-by-value one [10].

Syntax of PCF

PCF formalize the core of
a typed sequential functional programming language.

- **Types** $\sigma ::= \iota \mid (\sigma \multimap \tau)$
- **Terms** $M^\sigma ::= \mathbf{x}^\sigma \mid (\lambda \mathbf{x}^\mu . N^\tau)^{\mu \multimap \tau} \mid (P^{\tau \multimap \sigma} Q^\tau)^\sigma \mid \mathbf{Y}_\sigma$
 $\mid \mathbf{if} \mid \mathbf{succ} \mid \mathbf{pred} \mid \mathbf{\tilde{n}}$

For each type $(\sigma \multimap \sigma) \multimap \sigma$, the constant \mathbf{Y}_σ is added,
endowed with the evaluation rule:

$$\frac{P(\mathbf{Y}_\sigma P)M_1 \dots M_m \Downarrow_e \tilde{n}}{\mathbf{Y}_\sigma P M_1 \dots M_m \Downarrow_e \tilde{n}} \text{ (Y)}$$

A closed term of ground type is called **program**.

Syntax of PCF

PCF has been introduced by Plotkin in [11] inspired by the language LCF of Scott [12].

Note that types of terms/subterms are always superscript (as exponents). Often constants are wrote without its types, since it is implicit. However, be careful to the index-type of \mathbf{Y}_σ : there are infinite constants \mathbf{Y}_σ and σ is the minimum information needed to recovery the the type of a specific instance, namely $(\sigma \multimap \sigma) \multimap \sigma$.

For example, $(\mathbf{Y}_\sigma M^{\sigma \multimap \sigma})$ has type σ , for all σ .

Operational Evaluation of PCF

Let \Downarrow_e be the **evaluation relation** associating a program M to a numeral \tilde{n} whenever a judgment of the shape $M \Downarrow_e \tilde{n}$ can be proved by rules of the formal system:

$$\frac{P[Q/x]M_1 \dots M_m \Downarrow_e \tilde{n}}{(\lambda x^\sigma.P)QM_1 \dots M_m \Downarrow_e \tilde{n}} \text{ (head)} \qquad \frac{P(Y_\sigma P)M_1 \dots M_m \Downarrow_e \tilde{n}}{Y_\sigma PM_1 \dots M_m \Downarrow_e \tilde{n}} \text{ (Y)}$$

$$\frac{M_0 \Downarrow_e \tilde{0} \quad M_1 \Downarrow_e \tilde{n}}{\text{if } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (Oif)} \qquad \frac{M_0 \Downarrow_e \widetilde{k+1} \quad M_2 \Downarrow_e \tilde{n}}{\text{if } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (Iif)}$$

$$\frac{M \Downarrow_e \widetilde{n+1}}{\text{pred } M \Downarrow_e \tilde{n}} \text{ (pred)} \qquad \frac{M \Downarrow_e \tilde{n}}{\text{succ } M \Downarrow_e \widetilde{n+1}} \text{ (succ)} \qquad \frac{}{\tilde{n} \Downarrow_e \tilde{n}} \text{ (num)}$$

If there is a numeral \tilde{n} such that $M \Downarrow_e \tilde{n}$ then we write $M \Downarrow_e$, otherwise we write $M \Uparrow_e$.

Operational Semantics of PCF



Summarizing, the relation \Downarrow_e implements a **call-by-name** parameter passing mechanism, since the arguments of abstractions are substituted without being evaluated. It implements a **weak** (or **lazy**) evaluation strategy, since no evaluation is done under λ -abstractions.

\Downarrow_e is the abstract evaluation machine of our programming language.

Clearly, $\widetilde{n+1}$ is simple metanotation for a integer constants different from $\tilde{0}$.

Exercise 13

- Show that $(\lambda x^t.y^t.\text{succ } x^t)\tilde{0} \Downarrow_e \tilde{1}$.
- Show that $(\lambda x^t.\text{pred } x^t)\tilde{0} \Uparrow_e$.
- Show that $Y_\iota(\lambda x^t.x^t) \Uparrow_e$.

Recursive Programming

For instance, in order to calculate the **factorial**, we can start by writing its recursive definition:

$$\begin{aligned}
 f^{\ell \rightarrow \ell} x^\ell &= \text{if } x^\ell \tilde{1} \text{ (mult } x^\ell (f^{\ell \rightarrow \ell} (\text{pred } x^\ell))) \\
 f^{\ell \rightarrow \ell} &= \lambda x^\ell. \text{if } x^\ell \tilde{1} \text{ (mult } x^\ell (f^{\ell \rightarrow \ell} (\text{pred } x^\ell))) \\
 f^{\ell \rightarrow \ell} &= (\lambda t. \lambda x^\ell. \text{if } x^\ell \tilde{1} \text{ (mult } x^\ell (t(\text{pred } x^\ell)))) f^{\ell \rightarrow \ell}
 \end{aligned}$$

Note that $F^{(\ell \rightarrow \ell) \rightarrow \ell \rightarrow \ell}$ is not recursive.

The fixpoint of $F^{(\ell \rightarrow \ell) \rightarrow \ell \rightarrow \ell}$ is the desired program, thus $Y_{\ell \rightarrow \ell} F^{(\ell \rightarrow \ell)}$ is a program calculating the factorial

Exercise 14

Prove that $(Y_{\ell \rightarrow \ell} F^{(\ell \rightarrow \ell) \rightarrow \ell \rightarrow \ell}) \tilde{n} \Downarrow_e \tilde{m}$ whenever $m \in \mathbb{N}$ is the factorial of $n \in \mathbb{N}$.

A program computing the factorial of a natural number can be wrote in Scheme as follows:

```

(define fact (lambda (x)
  (if (n=0)
      1
      (* n fact (- n 1)))))

```

Recursive Programming



Exercise 15

Write a PCF-term M calculating the sum. Yet, give operational evaluations of a constant calculating the sum.

Exercise 16

Write a PCF-term M calculating the multiplication. Yet, give operational evaluations of a constant calculating the multiplication, with both “sequential” and “parallel” flavour! (Hint: a multiplication for zero give back zero?). Can both be simulated in PCF?

Exercise 17

Write a PCF-term M calculating the Fibonacci of an integer. Recall that $\text{Fib}(0) = \text{Fib}(1) = 1$ and if $n > 1$ then $\text{Fib}(n) = \text{Fib}(n - 1) + \text{Fib}(n - 2)$.

Operational Theories

Definition 18

Let M^τ be a term with a unique free variable, i.e. $FV(M^\tau) = \{x^\tau\}$. The term M^τ is a τ -context and noted $C[\tau]$.

Moreover $C[N^\tau]$ denotes the term $M^\tau\{N^\tau/x^\tau\}$ obtained by context-substitution of N^τ to all free occurrences of x^τ .

Definition 19

Suppose M^σ, N^σ be terms.

1. $M \lesssim_\sigma N$ whenever
if $C[M] \Downarrow_e \tilde{n}$ for some numeral \tilde{n} then $C[N] \Downarrow_e \tilde{n}$,
for all contexts $C[\sigma]$.
2. $M \approx_\sigma N$ if and only if $M \lesssim_\sigma N$ and $N \lesssim_\sigma M$.

Operational Theories

Usually, contexts are introduced in different way equivalent to that above, by presenting a grammar for them (essentially the same grammar of the language extended with a new constant denoting a “hole” in a term).

Contexts are not closed under α -equivalence (see Exercise 9), so they are not considered up to α -equivalence.

Exercise 20

Check that, if $C[\sigma]$ is a context then $C[M]$ is a program (i.e. a ground closed term).

\lesssim_σ (\approx_σ) is a preorder relation between terms having the same type σ of terms, so sometimes we will write simply \lesssim (\approx).

It is easy to check that \approx is a congruence relation, i.e. an equivalence relation closed under contexts. Sometimes \approx is called **observational** or **contextual equivalence**.

Syntactic Sugar

It will be useful to name some terms. In particular, Ω_σ will denote the term defined by induction on σ as follows:

$$\Omega_\iota \equiv Y_\iota(\lambda x^t . x), \quad \Omega_{\mu \rightarrow \tau} \equiv \lambda x^\mu . \Omega_\tau.$$

By using Ω_σ , it is possible to define terms Y_σ^k ($k \in \mathbb{N}$) in the following way:

$$Y_\sigma^0 \equiv \Omega_{(\sigma \rightarrow \sigma) \rightarrow \sigma}, \quad Y_\sigma^{k+1} \equiv \lambda x^{\sigma \rightarrow \sigma} . x(Y_\sigma^k x).$$


Theorem 21

Let M_0, \dots, M_m be a sequence of terms ($m \geq 0$).

1. If $\Omega_\sigma M_0 \dots M_m$ is a program then $\Omega_\sigma M_0 \dots M_m \uparrow_e$.
2. Let $Y_\sigma M_0 \dots M_m$ be a program.
 $Y_\sigma M_0 \dots M_m \downarrow_e \tilde{n}$ if and only if $Y_\sigma^k M_0 \dots M_m \downarrow_e \tilde{n}$, for some $k \in \mathbb{N}$.

Syntactic Sugar

Proof of Theorem 21.

1. In case $m = 0$ then $\sigma = \iota$ and $\Omega_\iota \equiv Y_\iota(\lambda x^t . x)$. Assume that \mathcal{D} is the derivation proving $Y_\iota(\lambda x^t . x) \downarrow_e$ with the minimum number of evaluation's rules. This is an absurdum, since after the application of the rules **Y** and (*head*) the remaining subderivation must conclude again $Y_\iota(\lambda x^t . x) \downarrow_e$. By induction on $m \geq 1$, it is easy to see that the $\Omega_\sigma M_0 \dots M_m \downarrow_e$ would imply $Y_\iota(\lambda x^t . x) \downarrow_e$, absurdly.
2. Both implications can be proved by induction on derivations proving the hypothesis. Details are left as an exercise. 

PreModel**Definition 22**

A **premodel** is a pair (\mathcal{A}^{type}, A) such that $\mathcal{A}^{type}[\cdot]$ is a function on types and A is a function on pairs of types respectively such that

- $\mathcal{A}^{type}[\sigma]$ is a non-empty set, which we view as the interpretation of type σ ,
- and $A^{\sigma, \tau} : \mathcal{A}^{type}[\sigma \rightarrow \tau] \times \mathcal{A}^{type}[\sigma] \rightarrow \mathcal{A}^{type}[\tau]$ is a function that we view as the interpretation of the application of an element of $\mathcal{A}^{type}[\sigma \rightarrow \tau]$ to an element of $\mathcal{A}^{type}[\sigma]$,

and such that the **extensionality property** holds, namely whenever $f, g \in \mathcal{A}^{type}[\sigma \rightarrow \tau]$:

$f = g$ if and only if $A^{\sigma, \tau}(f, x) = A^{\sigma, \tau}(g, x)$, for every $x \in \mathcal{A}^{type}[\sigma]$.

 *PreModel*


Denotational semantics is an approach for formalizing the semantics of computer languages by mathematical structures (where programs can be represented in a meaningful way) which express the semantics (meaning) of these programs. As originally developed by Strachey and Scott in the 1960s, denotational semantics interprets a program as a function on some mathematical structure [13].

Model

An **environment** is a function ρ that associates values to variables in a type-respecting way, namely $\rho(x^\sigma) \in \llbracket \sigma \rrbracket$ must hold.

We will denote $\mathcal{A}^{\tau_0, \dots, \tau_n, \iota}$ the function of

$$\mathcal{A}^{type}[\tau_0 \multimap \dots \multimap \tau_n \multimap \iota] \times \mathcal{A}^{type}[\tau_0] \dots \times \mathcal{A}^{type}[\tau_n] \rightarrow \mathcal{A}^{type}[\iota]$$

that extend, recursively, the premodel interpretation of the application of an element of $\mathcal{A}^{type}[\tau_0 \multimap \dots \multimap \tau_n \multimap \iota]$ to a sequence of elements having expected types.

Definition 23

A **model** is a premodel (\mathcal{A}^{type}, A) together with a function $\mathcal{A}^{term}[\cdot]$ defined on terms such that $\mathcal{A}^{term}[\llbracket M^\sigma \rrbracket]$ is a function from environments into $\mathcal{A}^{type}[\sigma]$.

The function $\mathcal{A}^{term}[\cdot]$ is required to satisfy the following equations:

- $\mathcal{A}^{term}[\llbracket x^\sigma \rrbracket] \rho = \rho(x)$
- $\mathcal{A}^{term}[\llbracket (P^{\tau \rightarrow \sigma} Q^\tau)^\sigma \rrbracket] \rho = A^{\sigma, \tau}(\mathcal{A}^{term}[\llbracket P^{\tau \rightarrow \sigma} \rrbracket] \rho, \mathcal{A}^{term}[\llbracket Q^\tau \rrbracket] \rho)$
- $A^{\sigma, \tau}(\mathcal{A}^{term}[\llbracket (\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau} \rrbracket] \rho, d) = \mathcal{A}^{term}[\llbracket N^\tau \rrbracket] \rho[d/x]$ s.t. $d \in \llbracket \mu \rrbracket$
- $\mathcal{A}^{term}[\llbracket \text{const}^\sigma \rrbracket] \rho = d_{\text{const}}$,
if $\sigma = \tau_0 \multimap \dots \multimap \tau_n \multimap \iota$ and $\text{const } M_0 \dots M_m \Downarrow_e \tilde{n}$ then
 $\mathcal{A}^{\tau_0, \dots, \tau_n, \iota}(d_{\text{const}}, \llbracket M_0 \rrbracket \rho, \dots, \llbracket M_m \rrbracket \rho) = \llbracket \tilde{n} \rrbracket \rho$

Model



If a premodel has an extension to a model, then the extension is unique.

The syntax of the calculus is ordinarily enclosed between **emphatic brackets** $\llbracket \cdot \rrbracket$ in order to put a denotational barrier between syntax and mathematics concepts surrounding the semantics.

In general, it is convenient to use the same notation \mathcal{A} for both \mathcal{A}^{type} and \mathcal{A}^{term} and, when no ambiguity arises, we write simply $\llbracket \cdot \rrbracket$ in place of $\mathcal{A}[\cdot]$. Moreover, often A will be denoted simply by \circ in infix position and associating to right.

If ρ is an environment and $d \in \llbracket \sigma \rrbracket$ then

$$\rho[d/x^\sigma](y^\tau) = \begin{cases} d & \text{if } x^\sigma \equiv y^\tau \\ \rho(y^\tau) & \text{otherwise.} \end{cases}$$

A notion of set-theoretical model has been introduced along the previous slides, however a categorical characterization of model of PCF can be given through the notion of **Cartesian Closed Category** (see [4]).

Basic Properties

Lemma 24

Let \mathcal{A} be a model and ρ, ρ' be environments.

1. If $\rho(x) = \rho'(x)$ for all $x \in \text{FV}(M)$ then $\llbracket M \rrbracket_\rho = \llbracket M \rrbracket_{\rho'}$.
2. $\llbracket M^\sigma[N^\tau/x^\tau] \rrbracket_\rho = \llbracket M^\sigma \rrbracket_{\rho[\llbracket N^\tau \rrbracket_\rho/x^\tau]}$.
3. If $\llbracket M^\sigma \rrbracket_\rho = \llbracket N^\sigma \rrbracket_\rho$ and $C[\sigma]$ is a context then $\llbracket C[M] \rrbracket = \llbracket C[N] \rrbracket$.

Proof.

1. By induction on M .
2. By induction on M .
3. By induction on $C[\sigma]$.

Hint: in all proofs the most hard case is that of the a λ -abstraction that needs the extensionality condition. ⊙

The first point imply that, in case $\text{FV}(M^\sigma) = \emptyset$, without loss of generality we can omit the environment after the emphatic brackets. In fact $\llbracket M^\sigma \rrbracket_\rho = \llbracket M^\sigma \rrbracket'_\rho$ for all be environments ρ, ρ' .

Full Abstraction

Definition 25

Let \mathcal{A} be a model and let M^σ, N^σ be terms.

We write $M \sim_\sigma N$ if and only if $\llbracket M \rrbracket_\rho = \llbracket N \rrbracket_\rho$, for each $\rho \in \text{Env}_B$.

Correctness:	$\sim \subseteq \approx$
Completeness:	$\approx \subseteq \sim$
Full abstraction:	$\sim = \approx$

The notion of Full abstraction has been presented first in [8].

Adequacy

Lemma 26 : Weak Adequacy

Given a model and M be a program. If $M \Downarrow_e \tilde{n}$ then $\llbracket M \rrbracket = \llbracket \tilde{n} \rrbracket$.

Proof. The proof is done by induction on the derivation proving $M \Downarrow_e \tilde{n}$.

- If the last applied rule is (Y) , (0if) , (1if) , (pred) , (succ) or (num) then the proof is trivial, by interpretation of constants.
- If the last applied rule is (head) then the proof follows by Lemma 24. ◻

Definition 27 : Adequacy

Let \mathcal{A} be a model and M be a program.

\mathcal{A} is **adequate** whenever,

$M \Downarrow_e \tilde{n}$ and $\llbracket M \rrbracket = \llbracket \tilde{n} \rrbracket$ are logically equivalents.

Theorem 28


Adequacy implies Correctness.

Proof. We must to prove that $M^\sigma \sim N^\sigma$ implies $M^\sigma \approx N^\sigma$.

Let $\llbracket M^\sigma \rrbracket_\rho = \llbracket N^\sigma \rrbracket_\rho$, for each environment ρ .

If $C[\sigma]$ is a context such that both $C[M]$ and $C[N]$ are programs and $C[M] \Downarrow_e \tilde{n}$ for some value \tilde{n} , then $\llbracket C[M] \rrbracket = \llbracket \tilde{n} \rrbracket$ by adequacy.

Since $\llbracket C[N] \rrbracket = \llbracket C[M] \rrbracket = \llbracket \tilde{n} \rrbracket$ by Lemma 24, it follows that $C[N] \Downarrow_e \tilde{n}$ by adequacy. By definition of operational equivalence the proof is done. ◻

 **FixPoint**



A **preorder** is a binary relation R over a set S which is reflexive and transitive.

A **partial order** is a antisymmetric preorder.

A **poset** is a set endowed with a partial order.

If D is a poset and $f : D \rightarrow D$ is a endofunction then an element $d \in D$ such that $f(d) = d$ is called a **fixpoint** of f .

FixPoint

Let \mathcal{A} be a model thus by Definition 23,
 if $\mathcal{A}^{term} \llbracket Y_\sigma \rrbracket \rho = d_{Y_\sigma}$ and $\sigma = \tau_1 \rightsquigarrow \dots \rightsquigarrow \tau_m \rightsquigarrow \iota$ then
 $(Y_\sigma M_0^{\sigma \rightsquigarrow \sigma})^\sigma M_1^{\tau_1} \dots M_m^{\tau_m} \Downarrow_e \tilde{n}$ must imply

$$d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho \circ \dots \circ \llbracket M_m \rrbracket \rho = \llbracket \tilde{n} \rrbracket \rho.$$

Weak adequacy and the evaluation rule

$$\frac{M_0(Y_\sigma M_0)M_1 \dots M_m \Downarrow_e \tilde{n}}{Y_\sigma M_0 M_1 \dots M_m \Downarrow_e \tilde{n}} \text{ (Y)}$$

imply that

$$\llbracket M_0 \rrbracket \rho \circ (d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho) \circ \llbracket M_1 \rrbracket \rho \circ \dots \circ \llbracket M_m \rrbracket \rho = d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho \circ \llbracket M_1 \rrbracket \rho \circ \dots \circ \llbracket M_m \rrbracket \rho$$

Thereby, by extensionality

$$\llbracket M_0 \rrbracket \rho \circ (d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho) = d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho$$

Namely, $(d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho)$ is a fixpoint of $\llbracket M_0 \rrbracket \rho$.

Since our starting purpose were to understand how to define d_{Y_σ} ,
 this reasoning says us that $(d_{Y_\sigma} \circ \llbracket M_0 \rrbracket \rho)$ must given back a fixpoint of $\llbracket M_0 \rrbracket \rho$, for all M_0 .

How do we know that there is any element d_{Y_σ} of $\llbracket (\sigma \rightsquigarrow \sigma) \rightsquigarrow \sigma \rrbracket$ having the above property?

If there are several, perhaps we could pick one of them!

If there is none, perhaps we want the meaning of the recursion to be “undefined” in some sense, but how does this fit with the equational theory?

An **upper bound** of a subset S of some poset is an element which is greater than or equal to every element of S . The term **lower bound** is defined dually.

The **least upper bound** (lub, supremum, join) of an poset S is the least element that is greater than or equal to each element of S . The **greatest lower bound** (glb, infimum, meet) is defined dually.

The **maximum** or **greatest element** of a subset S of a poset is an element of S which is greater than or equal to any other element of S . The **minimum** or **least element** is defined dually.

The **maximal elements** of a set S are the elements that are not smaller than any other element. The **minimal elements** are defined dually.

If a maximal element is unique then is the maximum. Dually, a unique minimal element is the minimum.

Complete Partial Order (CPO)

A **partial order** or **poset** is a pair (D, \sqsubseteq) where D is a set and \sqsubseteq is an order relation, often the poset is noted simply by D .

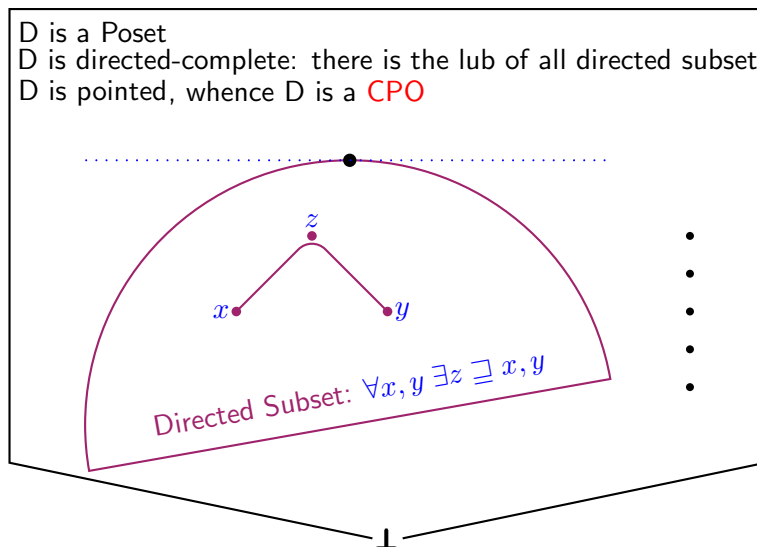
A subset X of D is said **consistent** or **bounded** when there is $z \in D$ such that $y \sqsubseteq z$ for each $y \in X$, in this case z is an upper bound of X .

The lub of a subset X , if it exists, is denoted by $\sqcup X$.

A subset X of D is **directed** if every finite subset $u \subseteq X$ has an upper bound $z \in X$.

An element of D is called **bottom** and denoted \perp if and only if $\perp \sqsubseteq d$ for each $d \in D$.

A **complete partial order (cpo)** is a poset D with a bottom element such that every directed subset $X \subseteq D$ has a least upper bound. Graphically,



Exercise 29

Check that the supremum of a set is unique, if it exists.

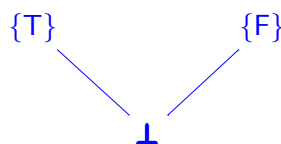
Exercise 30

A nonempty subset X of D is directed if $\forall x, x' \in X \exists x'' \in X$ such that $x \sqsubseteq x''$ and $x' \sqsubseteq x''$, namely for each pair of elements of X there is an upper bound in X . (Hint: A directed set X is always non-empty because $\emptyset \subseteq M$ is a finite subset that must have a bound.)

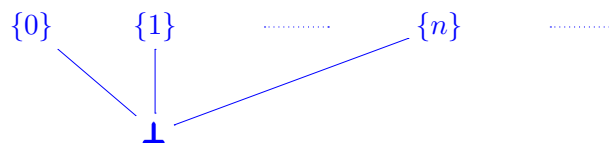
If X is a finite set then $\|X\|$ is the number of elements of X .

Examples

- Let B_{\perp} denote the cpo of boolean values, namely



- Let N_{\perp} denote the cpo of natural numbers, namely



- The most illustrative example of cpo is the set of partial functions $S \rightsquigarrow T$ between the set S, T ordered pointwise.

The natural numbers \mathbb{N} (that is, the non-negative integers) form a poset called ω with their usual ordering \leq . It is not a cpo because it has a directed subset that has no least upper bound (namely itself).

A partial order D is flat when, for all $x, y \in D$, if $x \sqsubseteq z$ then $x = \perp$ or $x = y$.

Let $\wp(S)$, the powerset of S , be the set of all subsets of a set S . Clearly $\wp(S)$ with ordinary set-inclusion order is a cpo.

Standard Model

A model is **standard** when each ground type (ι for PCF) is interpreted on a flat domain.

A model is **extensional** when $\forall f, g \in \llbracket \sigma \multimap \tau \rrbracket \forall x \in \llbracket \sigma \rrbracket$

$$f = g \text{ if and only if } f \circ x = g \circ x.$$

A model is **order-extensional** when $\forall f, g \in \llbracket \sigma \multimap \tau \rrbracket \forall x \in \llbracket \sigma \rrbracket$

$$f \sqsubseteq g \text{ if and only if } f \circ x \sqsubseteq g \circ x.$$

Exercise 31

Check that order-extensional implies extensional,
but the vice versa does not holds.

Continuous function

A function f between ordered sets is **monotone** (or monotonic, or even isotone) whenever it preserves the order,

namely $x \sqsubseteq y$ implies $f(x) \sqsubseteq f(y)$

A monotone function $f : A \rightarrow B$ is **continuous** whenever it preserves limit of directed set,

namely if $X \subseteq D$ is directed then $f(\sqcup X) = \sqcup f(X)$.

In the follows, $[D \rightarrow E] = \{f : D \rightarrow E \mid f \text{ is continuous}\}$.

Continuous function

Cpos and continuous functions form a category [4], in fact

- the identity function is continuous
- the composition of continuous functions is continuous,

$$(f \circ g)(\sqcup X) = f(g(\sqcup X)) = f(\sqcup g(X)) = \sqcup f(g(X)) = \sqcup (f \circ g)(X)$$

Exercise 32

If $f : A \rightarrow B$ is monotone and A is finite then f is continuous.

CPO Properties

Lemma 33

Let D, E be cpos.

If $X \subseteq [D \rightarrow E]$ has lub and $Y \subseteq D$ is directed then

$$\bigsqcup_{f \in X} \bigsqcup_{y \in Y} f(y) = \bigsqcup_{y \in Y} \bigsqcup_{f \in X} f(y).$$

Proof.

Clearly $f'(y') \sqsubseteq \bigsqcup_{f \in X} \bigsqcup_{y \in Y} f(y)$ for all $f' \in X$ and $y' \in Y$.

Moreover $\bigsqcup_{f \in X} f(y') \sqsubseteq \bigsqcup_{f \in X} f(\bigsqcup_{y \in Y} y) \sqsubseteq \bigsqcup_{f \in X} \bigsqcup_{y \in Y} f(y)$ for all $y' \in Y$.

Thus $\bigsqcup_{y \in Y} \bigsqcup_{f \in X} f(y) \sqsubseteq \bigsqcup_{f \in X} \bigsqcup_{y \in Y} f(y)$

and the other direction is similar. ⊞

CPO Properties

Theorem 34

If D, E are cpos then the **continuous function space**

$$[D \rightarrow E] = \{f : D \rightarrow E \mid f \text{ is continuous}\}$$

is a cpo under the pointwise order. ⊞

Proof. We show that the least upper bound $\bigsqcup_{f \in X} f$ of a directed set $X \subseteq [D \rightarrow E]$ is the function $g : x \mapsto \bigsqcup_{f \in X} f(x)$.

This makes sense as a function because the directedness of X means $\{f(x) \mid f \in X\}$ is directed for each x . It is easy to check that g is monotone. To see that it is continuous, suppose $Y \subseteq D$ is directed. Then

$$g(\bigsqcup_{y \in Y} y) = \bigsqcup_{f \in X} f(\bigsqcup_{y \in Y} y) = \bigsqcup_{f \in X} \bigsqcup_{y \in Y} f(y) = \bigsqcup_{y \in Y} \bigsqcup_{f \in X} f(y) = \bigsqcup_{y \in Y} g(y)$$

Clearly $f(x) \sqsubseteq g(x)$ for each $f \in X$. If $f \sqsubseteq g'$ for all $f \in X$, then $\bigsqcup_{f \in X} f(x) \sqsubseteq g'(x)$ for each $x \in D$ so $g \sqsubseteq g'$. ⊞

Lemma 35

If D, E are cpos then $D \times E$ is also a CPO under the coordinatewise order: $(x, y) \sqsubseteq (x', y')$ iff $x \sqsubseteq x'$ and $y \sqsubseteq y'$.

Proof. If $X \subseteq D \times E$ is directed, then

$$\begin{aligned} X_1 &= \{x \mid (x, y) \in X \text{ for some } y\} \\ X_2 &= \{y \mid (x, y) \in X \text{ for some } x\} \end{aligned}$$

are directed. Moreover $\sqcup X = (\sqcup X_1, \sqcup X_2)$.

FixPoint

Theorem 36

If D is a cpo and $f : D \rightarrow D$ is a continuous function then there exists the **minimum fixed point** of f .

Proof. $\perp \sqsubseteq f(\perp) \sqsubseteq f(f(\perp)) \sqsubseteq \dots$ by monotonicity

$f^0(\perp) \sqsubseteq f^1(\perp) \sqsubseteq f^2(\perp) \sqsubseteq \dots$ by using a common notation But $\{f^n(\perp) \mid n \in \mathbb{N}\}$ is directed, so $\sqcup\{f^n(\perp) \mid n \in \mathbb{N}\}$ exists. $\sqcup\{f^n(\perp) \mid n \in \mathbb{N}\}$ is a fix-point element of f , in fact

$$\begin{aligned} f(\sqcup\{f^n(\perp) \mid n \in \mathbb{N}\}) &= && \text{by continuity of } f \\ \sqcup\{f^{n+1}(\perp) \mid n \in \mathbb{N}\} &= && \text{by minimality of } \perp \\ \sqcup\{f^{n+1}(\perp) \mid n \in \mathbb{N}\} \sqcup \{\perp\} &= && \text{trivially} \\ \sqcup\{f^n(\perp) \mid n \in \mathbb{N}\} & & & \end{aligned}$$

$\sqcup\{f^n(\perp) \mid n \in \mathbb{N}\}$ is minimum, in fact

let $d \in D$ be a fix-point of f , namely $f(d) = d$

clearly $\perp \sqsubseteq d$, thus by monotonicity

$$\forall n \in \mathbb{N} \quad f^n(\perp) \sqsubseteq f^n(d) = d$$

immediately,

$$\sqcup\{f^n(\perp) \mid n \in \mathbb{N}\} \sqsubseteq \sqcup\{f^n(d) \mid n \in \mathbb{N}\} = d$$



We define $fix(f) = \sqcup_{n \in \omega} f^n(\perp)$.

Theorem 36 and fix will make us able to give a meaningful interpretation to Y_σ .

Factorial on CPO

The factorial function is the unique function that satisfies the following recursive equation:

$$\mathit{fact}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * \mathit{fact}(n - 1) & \text{if } n > 0 \end{cases}$$

Define a functional $F : [\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp] \rightarrow [\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp]$ by setting

$$F(f)(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * f(n - 1) & \text{if } n > 0 \end{cases}$$

The definition of F is not recursive, so well given.

But F is continuous, so by FixPoint Theorem F has a least fixpoint. Hence,

$$\mathit{fix}(F) = \bigsqcup_{n \in \omega} F^n(\perp_{[\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp]}).$$

Exercise 37

What is the domain of definition of $F^k(\perp_{[\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp]})$, for a given $k \in \mathbb{N}$? Write it's definition.

FixPoint Approximants

Theorem 38

1. $\llbracket \mathbf{Y}_\sigma^{(n)} \rrbracket(x) = F^n(x)$, for all $n \in \mathbb{N}$ and type σ .
2. $\llbracket \mathbf{Y}_\sigma \rrbracket(x) = \bigcup_{n \leq 0} \llbracket \mathbf{Y}_\sigma^{(n)} \rrbracket(x)$, for all $n \in \mathbb{N}$ and type σ .

Proof.

1. By induction on n .
2. By the previous point.



Interpretation of PCF

As seen the model interpretation must respect:

- $\llbracket x^\sigma \rrbracket \rho = \rho(x)$
- $\llbracket (P^{\tau \rightarrow \sigma} Q^\tau)^\sigma \rrbracket \rho = \llbracket P^{\tau \rightarrow \sigma} \rrbracket \rho \circ \llbracket Q^\tau \rrbracket \rho$
- $\llbracket (\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau} \rrbracket \rho \circ d = \llbracket N^\tau \rrbracket \rho[d/x]$

Together with a constraint for every constant

$$\llbracket \text{const}^{\tau_0 \rightarrow \dots \rightarrow \tau_n \rightarrow \iota} \rrbracket \rho = d_{\text{const}},$$

if $\text{const } M_0 \dots M_m \Downarrow_e \tilde{n}$ then $d_{\text{const}} \circ \llbracket M_0 \rrbracket \rho, \dots, \llbracket M_m \rrbracket \rho = \llbracket \tilde{n} \rrbracket \rho$

we interpret constants as follows,

- $\llbracket \tilde{n} \rrbracket \rho = n$
- $\llbracket \text{succ } M^\iota \rrbracket \rho = \begin{cases} n + 1 & \text{if } \llbracket M^\iota \rrbracket \rho = n \\ \perp & \text{otherwise} \end{cases}$
- $\llbracket \text{pred } M^\iota \rrbracket \rho = \begin{cases} n & \text{if } \llbracket M^\iota \rrbracket \rho = n + 1 \\ \perp & \text{otherwise} \end{cases}$
- $\llbracket \text{if } P^\iota M'_0 M'_1 \rrbracket \rho = \begin{cases} \llbracket M'_0 \rrbracket \rho & \text{if } \llbracket P^\iota \rrbracket \rho = 0 \\ \llbracket M'_1 \rrbracket \rho & \text{if } \llbracket P^\iota \rrbracket \rho = 1 \\ \perp & \text{otherwise} \end{cases}$
- $\llbracket Y_\tau M^{\tau \rightarrow \tau} \rrbracket \rho(x) = \text{fix}(\llbracket M^{\tau \rightarrow \tau} \rrbracket \rho)$

Interpretation of PCF

Exercise 39

The given interpretation use all and only continuous functions.

It would be clear that the interpretation of closed terms as constants is invariant with respect to environments, thus in such cases the environment indexing the interpretation mapping can be omitted.

Exercise 40

Write the interpretation of some constants between these which has been defined on slide 5.

Computability Predicate

Definition 41

We write $\text{Comp}(M^\sigma)$ whenever,

1. if $\sigma = \iota$ and $\text{FV}(M^\iota) = \emptyset$ then $\llbracket M \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ implies $M \Downarrow_e \tilde{n}$
2. if $\sigma = \mu \rightarrow \tau$ and $\text{FV}(M^{\mu \rightarrow \tau}) = \emptyset$ then $\text{Comp}(M^{\mu \rightarrow \tau})$ if and only if $\text{Comp}(M^{\mu \rightarrow \tau} N^\mu)$ for all closed N^μ s.t. $\text{Comp}(N^\mu)$
3. if $\text{FV}(M) = \{x_1^{\iota_1}, \dots, x_n^{\iota_n}\}$ for some $n \geq 1$ then $\text{Comp}(M^\sigma)$ if and only if $\text{Comp}(M[N_1/x_1, \dots, N_n/x_n])$ for all closed N^μ such that $\text{Comp}(N_1^{\iota_1})$

Note that $\text{Comp}(M^{\sigma \rightarrow \tau})$ and $\text{Comp}(N^\sigma)$ imply $\text{Comp}(MN^\tau)$,
for all closed terms M, N .

Property 42

$\text{Comp}(M^{\tau_1 \rightarrow \dots \rightarrow \tau_m \rightarrow \iota})$ and $\text{FV}(M) = \{x_0^{\iota_0}, \dots, x_n^{\iota_n}\}$ if and only if
for all closed N_i and P_j such that $\text{Comp}(N_i^{\iota_i})$ and $\text{Comp}(P_j^{\tau_j})$ ($i \leq n, j \leq m$),
 $\llbracket M[N_0/x_0, \dots, N_n/x_n]P_1 \dots P_m \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ implies $M[N_0/x_0, \dots, N_n/x_n]P_1 \dots P_m \Downarrow_e \tilde{n}$.

🔗 Computability Predicate



The Computability Predicate is the crucial tool for proving correctness.
It is used in the proof of Theorem 43.

The proof of the Theorem 43 arise from an adaptation of Plotkin [10] of a proof technique of Tait [14]. The argument was originally used for a different result about evaluation for the simply typed λ -calculus concerning a property know as **strong normalisation**.

Partial Computability Proof

Lemma 43

$\text{Comp}(M^\sigma)$ for each term M^σ s.t. $\tau = \iota$ holds for all Y_τ in M^σ .

Proof. The proof is given by induction on the **untyped** structure of M^σ .

M^σ is a **variable** and $\sigma = \tau_1 \rightsquigarrow \dots \rightsquigarrow \tau_m \rightsquigarrow \iota$ ($m \in \mathbb{N}$).

Let $\text{Comp}(P^\sigma)$ with P^σ closed and $\text{Comp}(N_i^\tau)$ with N_i^τ closed ($1 \leq i \leq m$).

Clearly $\llbracket x[P/x]N_1 \dots N_m \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ implies $\text{PN}_1 \dots N_m \Downarrow_e \tilde{n}$,
since $\text{Comp}(P^\sigma)$ hold by hypothesis.

Thus $\text{Comp}(x^\sigma)$, by Property 42.

M^σ is an **integer**, so $\sigma = \iota$.

The proof is trivial.

$M^\sigma = \text{if}$ and $\sigma = \iota \rightsquigarrow \iota \rightsquigarrow \iota \rightsquigarrow \iota$.

Let $\text{Comp}(N_i^\tau)$ with N_i^τ closed ($1 \leq i \leq 3$).

If $\llbracket \text{if } N_1 N_2 N_3 \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ then either $\llbracket N_1 \rrbracket_\rho = \llbracket \tilde{0} \rrbracket_\rho$ or $\llbracket N_1 \rrbracket_\rho = \llbracket \tilde{m+1} \rrbracket_\rho$,
by interpretation of **if**.

In the first case, clearly $\llbracket N_2 \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ for some \tilde{n} .

Thus, both $N_1 \Downarrow_e \tilde{0}$ and $N_2 \Downarrow_e \tilde{n}$ by hypotheses $\text{Comp}(N_1^\tau)$ and $\text{Comp}(N_2^\tau)$, and the proof follows by applying the operational evaluation.

The case $\llbracket N_1 \rrbracket_\rho = \llbracket \tilde{m+1} \rrbracket_\rho$ is similar.

$M^\sigma = \text{succ}$ or $M^\sigma = \text{pred}$ and $\sigma = \iota \rightsquigarrow \iota$.

Proofs are easier than that of the previous case.

$M^\sigma = (P^{\mu \rightsquigarrow \sigma} Q^\mu)^\sigma$.

By inductive hypothesis $\text{Comp}(P^\tau)$ and $\text{Comp}(Q^\tau)$ for all τ , so the proof follows.

$M^\sigma = \lambda x^\mu . P^\tau$, $\text{FV}(M^\sigma) = \{x_1^{\nu_1}, \dots, x_h^{\nu_h}\}$ for some $h \in \mathbb{N}$, so $\sigma = \mu \rightsquigarrow \tau$.

Let $\text{Comp}(N_i^{\tau_i})$ with $N_i^{\tau_i}$ closed ($1 \leq i \leq h$) and let $\text{Comp}(Q^\mu)$.

Let $\tau = \tau_1 \rightsquigarrow \dots \rightsquigarrow \tau_m \rightsquigarrow \iota$ for some $m \in \mathbb{N}$ and $\text{Comp}(R_i^{\tau_i})$ with $R_i^{\tau_i}$ closed ($1 \leq i \leq m$).

By inductive hypothesis $\text{Comp}(P^\tau)$ holds, thus

$\text{Comp}((P[Q/x, N_1/x_1, \dots, N_h/x_h]R_1 \dots R_n)^\iota)$.

If $\llbracket (\lambda x^\sigma . P)[N_1/x_1, \dots, N_h/x_h]QR_1 \dots R_n \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ then

$\llbracket (\lambda x^\sigma . P)[N_1/x_1, \dots, N_h/x_h]QR_1 \dots R_n \rrbracket_\rho = \llbracket P[Q/x, N_1/x_1, \dots, N_h/x_h]R_1 \dots R_n \rrbracket_\rho$ by Lemma 24.

Hence it follows that $P[Q/x, N_1/x_1, \dots, N_h/x_h]R_1 \dots R_n \Downarrow_e \tilde{n}$ by Definition 41.

The proof follows by rule (**head**). $M^\sigma = Y_\iota$, so $\sigma = (\iota \rightsquigarrow \iota) \rightsquigarrow \iota$. The proof is easy ⊙

Full Computability Proof

Theorem 44

$\text{Comp}(M^\sigma)$ for each term of PCF.

Proof. Note that Ω_σ and Y_σ^k are defined using only Y_ι .

We will prove that, if $Y_\sigma QR_1 \dots R_n$ is a program and $\llbracket Y_\sigma QR_1 \dots R_n \rrbracket_\rho = \llbracket \tilde{n} \rrbracket_\rho$ then $Y_\sigma QR_1 \dots R_n \Downarrow_e \tilde{n}$.

Note that there exists $k \in \mathbb{N}$ such that $\llbracket Y_\sigma^k QR_1 \dots R_n \rrbracket_\rho = \llbracket Y_\sigma QR_1 \dots R_n \rrbracket_\rho$ by Theorem 38. Thus $Y_\sigma^k QR_1 \dots R_n \Downarrow_e \tilde{n}$ by Lemma 43.

The proof follows by Theorem 21. ⊙

Correctness

Corollary 45 : Adequacy

Standard models based on CPO are adequate for PCF.

Proof. Theorem 44 (together with Definition 41) and Lemma 26 imply that $\llbracket M \rrbracket = \llbracket \tilde{n} \rrbracket$ if and only if $M \Downarrow_e \tilde{n}$, for any program M , numeral \tilde{n} . ⊙

Theorem 46 : Correctness

Standard models based on CPO are correct for PCF.

Proof. The adequacy implies the **Correctness**, by Theorem 28, thereby the proof is done. ⊙

Introduction to Domains

So far, our purpose were to find some necessary abstract conditions on structures inducing models, essentially in order to assure correctness. CPOs reach this goal as we have seen, thus in the following we call them: **domains**.

However, we don't have again dealt with the completeness question. In order to face up this problem we need some deeper analysis of domains.

Since we want to reason on standard extensional models of PCF, no choice is possible for domains being interpretation of ground types.

But many choices are possible for domains being the **interpretation of arrow-types**, thus we will concentrate our attention on them. For instance, more than one order between arrows can be choose in our domains.

Examples

We take into account the two possible CPO-constructions of $\mathbf{B}_{\perp} \rightarrow \mathbf{B}_{\perp}$.

They are described by using an informal but straightforward notation, using sets of pair of elements.

The application of the set $T = \{(x_0, y_0), \dots, (x_1, y_1)\}$ to z is defined as follows

$$T(z) = \begin{cases} \sqcup_{i \in I} y_i & \text{if } I \neq \emptyset \\ \perp & \text{otherwise.} \end{cases} \quad \text{where } I = \{i \mid (x_i, y_i) \in T \wedge x_i \sqsubseteq z\}$$

Note that \emptyset is $\perp_{\mathbf{B}_{\perp} \rightarrow \mathbf{B}_{\perp}}$ and it denotes the function constantly $\perp_{\mathbf{B}_{\perp}}$.

The full set of functions in $\mathbf{B}_{\perp} \rightarrow \mathbf{B}_{\perp}$ is described in the following table

\emptyset	$\{(\perp, T)\}$	$\{(\perp, F)\}$	
$\{(F, T)\}$	$\{(T, T)\}$	$\{(F, F)\}$	$\{(T, F)\}$
$\{(T, T), (F, T)\}$	$\{(T, F), (F, T)\}$	$\{(T, T), (F, F)\}$	$\{(T, F), (F, F)\}$

T, F are respectively the semantic denotations for “true” and “false”.

Note that $\{(\perp, T)\}$ corresponds to a function like $\lambda x. \mathbf{tt}$, where \mathbf{tt} is a ground constant representing syntactically “true”.

On the other hand, $\{(T, T), (F, T)\}$ corresponds to a function like $\lambda x. \mathbf{if } x \mathbf{ tt } \mathbf{ tt}$.

We must choose an order between them inducing an extensional cpo, if we want to use them for a model.

The **order-extensional** cpo is:

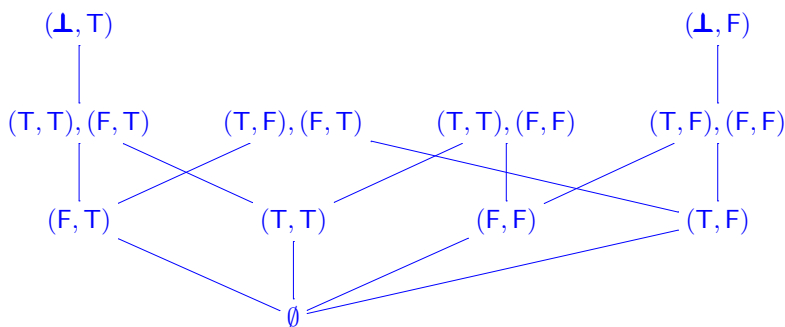


Figure A

A different order called **stable** produce:

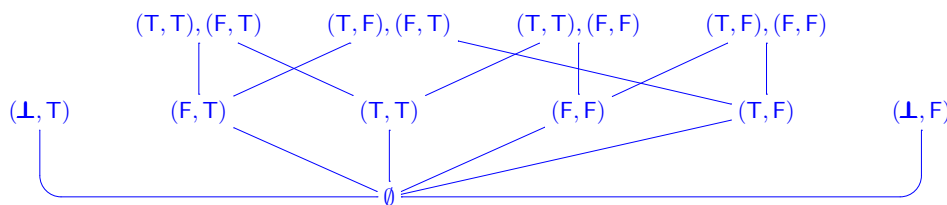


Figure B

Exercise 47

Check that the above structures depict actually cpo of function satisfying extensionality.

Note that for only one of the above depicted domains the order correspond exactly to set-theoretical inclusion!

Definition 48

If D, E are domains and $f, g \in D \rightarrow E$ then $f \sqsubseteq_s g$ whenever $x \sqsubseteq y$ implies $f(x) = f(y) \sqcap g(x)$, for all $x, y \in D$.

Exercise 49

Between domains depicted in Figures A and B ([Click Here](#), slide 43), what is endowed with a stable order?

Exercise 50

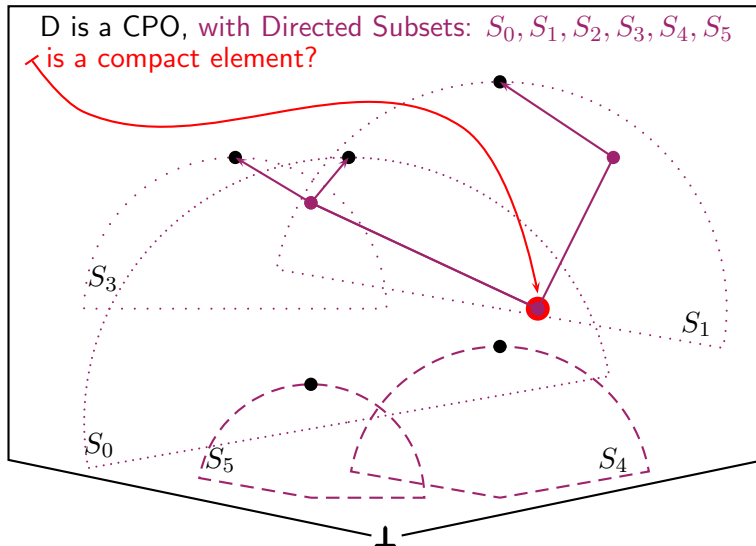
Imagine how we must design the order-extensional domain $\mathbf{N}_\perp \rightarrow \mathbf{N}_\perp$, $\mathbf{B}_\perp \rightarrow (\mathbf{B}_\perp \rightarrow \mathbf{B}_\perp)$ and $(\mathbf{B}_\perp \rightarrow \mathbf{B}_\perp) \rightarrow \mathbf{B}_\perp$.

Compact Element

Definition 51

Let D be a cpo. An element $z \in D$ is **compact** if, for every directed $X \subseteq D$ such that $z \sqsubseteq \sqcup X$, there is some $x \in X$ such that $z \sqsubseteq x$. Let $\mathbb{k}(D)$ be the set of compact element of D .

Graphically,



Algebraic

Definition 52

A cpo D is **algebraic** if, for every $x \in D$ the set $X = \{e \in \mathbb{k}(D) \mid e \sqsubseteq x\}$ is directed and $\sqcup X = x$.

Example 53

- The cpo of partial function $\mathbb{N} \rightsquigarrow \mathbb{N}$ is algebraic and the compact elements are the partial functions with finite graphs.
- Let S be a set, so $\mathcal{P}(S)$ is an algebraic cpo under the set-inclusion order. A subset of S is compact if and only if it is finite.

Unfortunately,

$D \rightarrow E$ may **not** be algebraic for some algebraic cpo D, E so a further constraint will be asked.

Exercise 54

Check that flat cpo are Scott-Domains. (What are their compact element?)

If D, E are algebraic cpo, then so is $D \times E$ and $\mathbb{k}(D) \times \mathbb{k}(E)$.

Exercise 55

Show that \perp is compact in all Scott-Domain.

Definition 56

A poset D enjoys of the **ascending chain condition** (briefly, ACC) whenever no infinite sequence $x_0 \sqsubseteq x_1 \sqsubseteq x_2 \sqsubseteq x_3 \dots$ of its **distinct elements** exists.

Exercise 57

Let D be a poset satisfying the ascending chain condition.

- D is a cpo.
- D is algebraic.

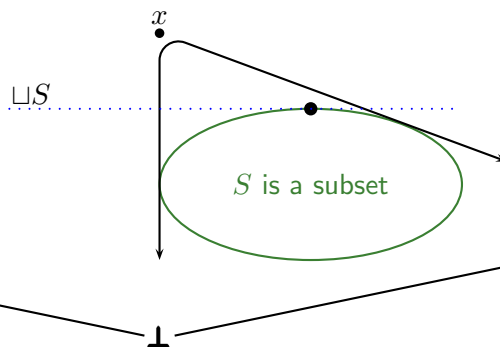
Bounded-Completeness

Definition 58

A non-empty cpo D is **bounded-complete** if every bounded subset $S \subseteq D$ has a least upper bound $\sqcup S$ in D .

Graphically,

D is a poset
 Let x be a bound for all $y \in S$
 D is **bounded-complete**:
 there is the lub of all bounded subset



Bounded-complete sets are sometimes called consistently-complete.

Exercise 59

Say $\omega^{op} = \{0, -1, -2, \dots\}$ the set of negative integers with their usual ordering. Check that ω^{op} is bounded complete but not directed-complete.

Exercise 60



Check that the poset in the beside picture is a cpo, but it's not bounded-complete.

Exercise 61

If D is a bounded complete cpo and a, b are consistent compact elements of D then $a \sqcup b$ is compact in D .

Domain

Definition 62

An Algebraic Bounded-Complete CPO is know, in literature, as **Scott-Domains**.

As a mnemonic joke, a Scott-Domain will be called also **abcpo**:

- a \mapsto Algebraic
- b \mapsto Bounded-Complete
- c \mapsto (Directed-)Complete

Exercise 63

A pointed cpo D is bounded complete if and only if every consistent pair $x, y \in D$ has a least upper bound $x \sqcup y$.
(Hint: use the directed completeness!).

Step Function

In order to deeply understand how domains for the interpretation of arrow-types are done, we have restricted ourselves to consider algebraic cpos.

Definition 64

Let D, E be algebraic cpos, $d \in \mathbb{k}(D)$ and $e \in \mathbb{k}(E)$.

$(d \searrow e) : D \rightarrow E$ is a **step function** defined by

$$(d \searrow e)(x) = \begin{cases} e & \text{if } d \sqsubseteq x \\ \perp_E & \text{otherwise.} \end{cases}$$

Note that $(d \searrow e)$ is simply a pair of compact, representing an element of the graph of a function.

Exercise 65

Check that functions in the examples of $\mathbf{B}_\perp \rightarrow \mathbf{B}_\perp$ ([Click Here](#), slide 43) are actually step function.

Lemma 66

A step-function $(d \searrow e) : D \rightarrow E$ is continuous and compact.

Proof. To see continuity, suppose $S \subseteq D$ be directed. There are two cases.

- If $d \sqsubseteq \sqcup S$ then $d \sqsubseteq x$ for some $x \in S$ since d is compact. Whence,

$$(d \searrow e)(\sqcup S) = e = (d \searrow e)(x) = \sqcup_{x \in S} ((d \searrow e)(x))$$

- If $d \not\sqsubseteq \sqcup S$ then no element of S is greater than d , so

$$(d \searrow e)(\sqcup S) = \perp = \sqcup_{x \in S} ((d \searrow e)(x))$$

To show that $(d \searrow e)$ is compact, suppose $T \subseteq [D \rightarrow E]$ be directed and $(d \searrow e) \sqsubseteq \sqcup T$.

Thereby,

$$e = (d \searrow e)(d) \sqsubseteq (\sqcup T)(d) = \sqcup_{f \in T} f(d).$$

Since e is compact, $e \sqsubseteq f(d)$ for some $f \in T$.

Hence $d \searrow e \sqsubseteq f$. ⊙

Step Function



Definition 67

Given a cpo D , a set D_0 of compact elements of D forms a **basis** if, for every $x \in D$, the set $S = \{d \in D_0 \mid d \sqsubseteq x\}$ is directed and $\sqcup S = x$.

The following lemma is often a convenient way to show that a cpo is algebraic.

Exercise 68

If D_0 forms a basis of D then D is algebraic and $\mathbb{k}(D) = D_0$.

Arrow-Domain

Theorem 69

If D, E are abcpos then the continuous function space

$$[D \rightarrow E] = \{f : D \rightarrow E \mid f \text{ is continuous}\}$$

is a Scott-Domain (abcpo) under the pointwise order.

Proof. $[D \rightarrow E]$ is a CPO by Theorem 34.

Too see that $[D \rightarrow E]$ is bounded-complete,
let $S \subseteq [D \rightarrow E]$ and f be a bound for S .

Given $x \in D$, the set $\{g(x) \mid g \in S\}$ is bounded by $f(x)$.
It has least upper bound, since E is bounded complete.

Define $f'(x) = \sqcup\{g(x) \mid g \in S\}$ for each x .

Note that f' is continuous by Lemma 33 and it is obviously the lub for S .

Too see that $[D \rightarrow E]$ is algebraic, let $f \in [D \rightarrow E]$.

It suffices to show that $f = \sqcup S$ where $S = \{d \searrow e \mid e \sqsubseteq f(d)\}$.

(By Definition 67 and Exercises 68, 61) Since S is bounded by f , we know that $\sqcup S$ exists.

Let $x \in D$ and suppose $e' \sqsubseteq f(x)$ for some $e' \in \mathbb{k}(E)$.

By continuity of f and algebraicity of D

$$e' \sqsubseteq f\left(\sqcup\{d^* \in \mathbb{k}(D) \mid d^* \sqsubseteq x\}\right) = \sqcup f\left(\{d^* \in \mathbb{k}(D) \mid d^* \sqsubseteq x\}\right)$$

so $e' \sqsubseteq f(d')$ for some compact $d' \sqsubseteq x$ (by algebraicity of E).

But $d' \searrow e'$ is compact and less than f ,
so $d' \searrow e' \in S$ by definition of S .

Clearly, $e' \sqsubseteq (\sqcup S)(x)$.

Since e' was arbitrary,

this show that every compact $e \sqsubseteq f(x)$ satisfies $e \sqsubseteq (\sqcup S)(x)$.

Thus $f(x) \sqsubseteq (\sqcup S)(x)$. Since x was arbitrary, $f \sqsubseteq \sqcup S$.

Since $\sqcup S \sqsubseteq f$ too, the proof is done. ⊙

Exercise 70

Let D, E be algebraic cpos and $f : D \rightarrow E$.

- f is continuous if and only if it is monotone and for each $e \in \mathbb{k}(E)$ and $x \in D$ such that $e \sqsubseteq f(x)$, there exists $d \sqsubseteq x$ such that $d \in \mathbb{k}(D)$ and $e \sqsubseteq f(d)$.
- The graph of f , namely $\{(d, e) \in \mathbb{k}(D) \times \mathbb{k}(E) \mid e \sqsubseteq f(d)\}$, determines f entirely.

Definition 71

If D, E are algebraic cpos then an **approximate relation** is a relation $R \subseteq \mathbb{k}(D) \times \mathbb{k}(E)$ satisfying:

- $(d, e_0), (d, e_1) \in R$ implies $\exists e \in \mathbb{k}(E)$ such that $e_0, e_1 \sqsubseteq e$ and $(d, e) \in R$,
- $(d, e) \in R, d \sqsubseteq d_1, e_1 \sqsubseteq e$ imply $(d_1, e_1) \in R$.

Exercise 72

The approximate relations are exactly the graphs of continuous functions.

Correctness

Theorem 73

Standard Model based on Scott-Domains is correct for PCF.

Proof. Clearly Scott-Domains form a pre-model. They are cpo, so the proof as already been given. 📄

We will use \mathcal{E} (from **extensional**) in order to identify uniquely the standard model based on Scott-Domain when, we will compare it with different models.


dl-Domain

Compact element in Scott-Domains can be greater than an infinite number of other compact elements.

Berry [2] has shown that it is possible to build model based on domains such that each compact dominates at most a finite number of other compacts. Those new domains are called **dl-domains** and are essentially Scott-Domain satisfying two further conditions endowed with the stable order.

Instead to present dl-domains, we will present the **Coherence Spaces** [5, 6] of Girard that give a more “direct” representation of compact elements.

A coherence space is always a dl-domain, although the converse is false.

 *dl-Domain*



Exercise 74

Show a compact element of a Scott-domain being greater than an infinite number of other compact elements. (Hint: Reason on $\mathbb{N}_\perp \rightarrow \mathbb{N}_\perp$).

Be careful to the fact that, given a Scott-Domain X and a dl-domain Y being the interpretations of the same arrow-type $\sigma \multimap \tau$ there is no immediate relation between them. In particular, X does not need to be a “substructure” of Y .

Coherence Space

Definition 75

A **coherence space** X is a pair $(|X|, \circ_X)$ where $|X|$ is a set called the **web**, its elements are called **tokens** and \circ_X is called **coherence relation** on X .

\circ_X is a binary reflexive and symmetric relation between tokens.


Definition 76

The set of **cliques** of X is

$$Cl(X) = \{x \subseteq |X| \mid \forall a, b \in x \ a \circ_X b\};$$

moreover, $Cl_{fin}(X)$ denotes the set of finite cliques of $Cl(X)$.

If X is a coherence space then $Cl(X)$ is a poset with respect to the relation \subseteq .

 *Coherence Space*



The strict incoherence \smile_X is the complementary relation of \circ_X ; the incoherence \asymp_X is the union of relations \smile_X and $=$; the strict coherence \frown_X is the complementary relation of \asymp_X .

Basic Properties

Exercise 77

Let X be a coherence space.

1. $\emptyset \in Cl(X)$.
2. $\{a\} \in Cl(X)$, for each $a \in |X|$.
3. If $y \subseteq x$ and $x \in Cl(X)$ then $y \in Cl(X)$.
4. If $D \subseteq Cl(X)$ is directed then $\cup D \in Cl(X)$.

So cliques of a coherence space with set-inclusion form a cpo.

Let x, x' be sets; $x \subseteq_{fin} x'$ means that $x \subseteq x'$ and x is finite.

Stable Function

Definition 78

Let X and Y be coherence spaces and $f : Cl(X) \rightarrow Cl(Y)$ be a monotone function.

- f is **continuous** whenever $\forall x \in Cl(X) \forall b \in f(x) \exists x_0 \subseteq_{fin} x$ such that $b \in f(x_0)$.
- f is **stable** whenever $\forall x \in Cl(X) \forall b \in f(x) \exists x_0 \subseteq_{fin} x$ such that $b \in f(x_0)$ and $\forall x' \subseteq x$, if $b \in f(x')$ then $x_0 \subseteq x'$.

Continuity asks for the existence of a finite amount of input for which some amount of output is produced, while **stability** asks for a minimum finite amount input for which some amount of output is produced.

Stable Function



Exercise 79

1. Let X and Y be coherence spaces and $f : Cl(X) \rightarrow Cl(Y)$ be a monotone function. Then f is continuous if and only if $f(\cup D) = \cup \{f(x) \mid x \in D\}$, for each $D \subseteq Cl(X)$ directed.
2. Let X and Y be coherence spaces and $f : Cl(X) \rightarrow Cl(Y)$ be a continuous function. Then f is stable if and only if $\forall x, x' \in Cl(X)$, $x \cup x' \in Cl(X)$ implies $f(x \cap x') = f(x) \cap f(x')$.

Definition 80

Let X_1 and X_2 be coherence spaces.

$X_1 \& X_2$ is the coherence space having $|X \& Y| = (\{1\} \times |X_1|) \cup (\{2\} \times |X_2|)$ as web; while, $\forall (i, a), (j, b) \in |X_1 \& X_2|$ $(i, a) \circ_{X_1 \& X_2} (j, b)$ if and only if $i \neq j$ or, $i = j$ and $a \circ_{X_i} b$.

It is easy to check that $X_1 \& X_2$ is a categorical product for coherence spaces for the spaces X_1, X_2 .

Trace

Stable functions can be represented as cliques.

Definition 81

Let X and Y be coherence spaces.

The **trace** $\text{tr}(f)$ of the stable function $f : Cl(X) \rightarrow Cl(Y)$ is the set of pairs $(x_0, b) \in Cl_{fin}(X) \times |Y|$ such that $b \in f(x_0)$ and $\forall x \subseteq x_0, b \in f(x)$ implies $x = x_0$.

Trace

Stable functions can be represented as cliques of a coherence space.

Definition 82

Let X and Y be coherence spaces.

$X \Rightarrow Y$ is the coherence space having $|X \Rightarrow Y| = Cl_{fin}(X) \times |Y|$ as web, while if $(x_0, b_0), (x_1, b_1) \in |X \Rightarrow Y|$, then $(x_0, b_0) \circ_{X \Rightarrow Y} (x_1, b_1)$ under the following conditions:

1. $x_0 \cup x_1 \in Cl(X)$ implies $b_0 \circ_Y b_1$;
2. $x_0 \cup x_1 \in Cl(X)$ and $b_0 = b_1$ imply $x_0 = x_1$.

Trace

The bridge between stable functions and cliques follows.

Lemma 83

If $f : Cl(X) \rightarrow Cl(Y)$ is a stable function then $\text{tr}(f) \in Cl(X \Rightarrow Y)$.

Let X, Y be coherence spaces and $t \in Cl(X \Rightarrow Y)$ and $x \in Cl(X)$. Let us define $\mathcal{F}(t) : Cl(X) \rightarrow Cl(Y)$ be the function such that

$$\mathcal{F}(t)(x) = \{b \in |Y| \mid \exists x_0 \in Cl(X) \ (x_0, b) \in t \wedge x_0 \subseteq x\}.$$

Lemma 84

If $t \in Cl(X \Rightarrow Y)$ then $\mathcal{F}(t) : Cl(X) \rightarrow Cl(Y)$ is a stable function.

Trace

Coherence spaces and stable functions form a cartesian closed category which is a full subcategory of the categories of qualitative domains and dl-domains endowed with stable functions.

All these categories contain objects and morphisms in the range of the standard interpretation of PCF, so without ambiguity they will be called **Stable Models**.

Interpretation Revisited

An interesting example is a “very concrete” reformulation of the interpretation of model, directly on cliques:

$$\llbracket \tilde{n} \rrbracket = \{n\}, \text{ for each } n \in \mathbb{N}$$

$$\llbracket \text{if} \rrbracket = \{ (\{0\}; \{n\}; \emptyset \searrow n) \mid n \in \mathbb{N} \} \\ \cup \{ (\{m\}; \emptyset; \{n\} \searrow n) \mid n \in \mathbb{N} \text{ and } m \neq 0 \}$$

$(a_1; a_2; a_3 \searrow b)$ is an abbreviation for $(a_1, (a_2, (a_3, b)))$.

$$\llbracket \lambda x^\mu. P^\tau \rrbracket_\rho = \left\{ (x_0, b) \in Cl(\llbracket \mu \rrbracket) \times \llbracket \tau \rrbracket \mid \begin{array}{l} b \in \llbracket P \rrbracket \rho[x_0/x] \text{ and} \\ \forall y \subseteq x_0 \quad b \in \llbracket P \rrbracket \rho[y/x] \text{ implies } y = x_0 \end{array} \right\}$$

Exercise 85

Show how to give the interpretation of **succ** and **pred**.

Correctness

Theorem 86

Standard Model based on Coherence Spaces is correct for PCF.

Proof. Clearly Coherence Spaces form a premodel. They are cpo, so the proof as already been given. ☹

We will use \mathcal{S} (from **stable**) in order to identify uniquely the standard model based on Coherence Space when, we will compare it with different models.

Two Terms

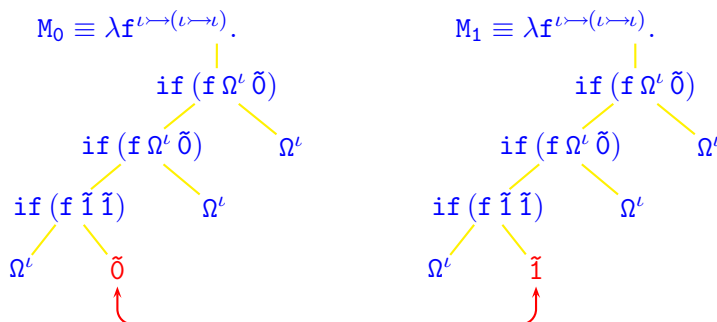
To prove that Scott-Domains are not fully abstract for PCF, we need to show two terms that have the same operational behaviour in all ground contexts but fail to be equal in the model \mathcal{E} .

To this aim, let M_0, M_1 be the following PCF terms:

$$M_0 \equiv \lambda f^{\iota \rightarrow (\iota \rightarrow \iota)}. \text{if } (f \tilde{0} \Omega^\iota) \left(\overline{\text{if } (f \Omega^\iota \tilde{0}) \left(\overline{\text{if } (f \tilde{1} \tilde{1}) \Omega^\iota \tilde{0}} \right) \Omega^\iota} \right) \Omega^\iota$$

$$M_1 \equiv \lambda f^{\iota \rightarrow (\iota \rightarrow \iota)}. \text{if } (f \tilde{0} \Omega^\iota) \left(\text{if } (f \Omega^\iota \tilde{0}) \left(\text{if } (f \tilde{1} \tilde{1}) \Omega^\iota \tilde{1} \right) \Omega^\iota \right) \Omega^\iota$$

having type $(\iota \rightarrow (\iota \rightarrow \iota)) \rightarrow \iota$.



The given programs are similar except for a ground value. By using an informal notation and unspecified subsuming premodel,

it is easy to see that (recall that M_0, M_1 have type $(\iota \rightarrow (\iota \rightarrow \iota)) \rightarrow \iota$):

$$\llbracket M_i \rrbracket = \begin{cases} (d, i) & \text{if } (k+1; h+1 \searrow j+1), (0; \perp \searrow 0), (\perp; 0 \searrow 0) \sqsubseteq d \\ \perp & \text{otherwise} \end{cases}$$

It is easy to check that there is a compact element of the Scott Model \mathcal{E} satisfying the first constraint of the interpretation of M_0, M_1 .

For example the “parallel or” (the “sequential or” stands aside),

por	$\tilde{0}$	$\tilde{1}$	\perp
$\tilde{0}$	$\tilde{0}$	$\tilde{0}$	$\tilde{0}$
$\tilde{1}$	$\tilde{0}$	$\tilde{1}$	\perp
\perp	$\tilde{0}$	\perp	\perp

or	$\tilde{0}$	$\tilde{1}$	\perp
$\tilde{0}$	$\tilde{0}$	$\tilde{0}$	\perp
$\tilde{1}$	$\tilde{0}$	$\tilde{1}$	\perp
\perp	\perp	\perp	\perp

But M_0, M_1 have the same operational behaviour in all ground contexts.

To see this, it suffices to show that $\mathcal{S}\llbracket M_0 \rrbracket = \mathcal{S}\llbracket M_1 \rrbracket$ by correctness.

In fact, $(0; \perp \searrow 0), (\perp; 0 \searrow 0)$ are incoherent, since the lack of minimality.

We can substitute $(0; \perp \searrow 0), (\perp; 0 \searrow 0)$ with $(0; \perp \searrow 0) \sqcap (\perp; 0 \searrow 0) = (\perp; \perp \searrow 0)$, but $(\perp; \perp \searrow 0)$ is a constant function which is incoherent with $(k+1; h+1 \searrow j+1)$.

Thus no stable trace d can satisfy the non-trivial constraint of the interpretation of M_0, M_1 , that are both equal to \perp (the empty-set).

Exercise 87

The following “or” are called RIGHT OR and LEFT OR.

ror	ō	ĩ	⊥
ō	ō	ō	⊥
ĩ	ō	ĩ	⊥
⊥	ō	⊥	⊥

lor	ō	ĩ	⊥
ō	ō	ō	ō
ĩ	ō	ĩ	⊥
⊥	⊥	⊥	⊥

Can they be simulated in PCF or in PCF+?

PCF+

PCF+ extend PCF with a parallel operator.

$$M^\sigma ::= \begin{array}{l} x^\sigma \quad | \quad (\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau} \quad | \quad (P^{\tau \rightarrow \sigma} Q^\tau)^\sigma \quad | \quad Y_\sigma \\ \text{if} \quad | \quad \text{succ} \quad | \quad \text{pred} \quad | \quad \tilde{n} \\ \text{pif} \end{array}$$

The new constant **pif** has type $\iota \rightarrow \iota \rightarrow \iota \rightarrow \iota$ and it is endowed with the evaluation rules:

$$\frac{M_0 \Downarrow_e \tilde{o} \quad M_1 \Downarrow_e \tilde{n}}{\text{pif } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (Opif)} \qquad \frac{M_0 \Downarrow_e \widetilde{k+1} \quad M_2 \Downarrow_e \tilde{n}}{\text{pif } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (lpif)}$$

$$\frac{M_1 \Downarrow_e \tilde{n} \quad M_2 \Downarrow_e \tilde{n}}{\text{pif } M_0 M_1 M_2 \Downarrow_e \tilde{n}} \text{ (*pif)}$$

Exercise 88

Show that **if** is redundant PCF+, since it can be defined.

Exercise 89

Extend PCF with a **por** operator.

Exercise 90

Show that in PCF+ you can write a program simulating **por**.

Conversely, show that in PCF extended with a **por** operator there is a program simulating **pif**.

Full Abstraction

PCF+ is fully abstract for Scott-Domains!

The interpretation of `pif` is the expected one, namely

$$\llbracket \text{pif } P^t M_0^t M_1^t \rrbracket \rho = \begin{cases} \llbracket M_0^t \rrbracket \rho & \text{if } \llbracket P^t \rrbracket \rho = 0 \\ \llbracket M_1^t \rrbracket \rho & \text{if } \llbracket P^t \rrbracket \rho = 1 \\ \llbracket M_1^t \rrbracket \rho & \text{if } \llbracket P^t \rrbracket \rho = \perp \text{ and } \llbracket M_0^t \rrbracket \rho = \llbracket M_1^t \rrbracket \rho \\ \perp & \text{otherwise} \end{cases}$$

It is easy to check that all properties shown for PCF hold also for PCF+. The proof of full abstraction follows from the definability of all compact element that will be shown in the next slide.

Theorem 91

Scott-Model is fully abstract for PCF+.

Proof. Scott-Model is correct for PCF+.

Suppose M^σ, N^σ are terms of PCF+ such that $M \not\sim_\sigma N$ and $FV(M) \cup FV(N) \subseteq \{x_1^{\tau_1}, \dots, x_n^{\tau_n}\}$ for some $n \in \mathbb{N}$.

By extensionality $\lambda x_1 \dots x_n. M \not\sim_\tau \lambda x_1 \dots x_n. N$ for some τ , thus without loss of generality only closed terms will be considered.

Suppose M^σ, N^σ are closed terms of PCF+.

Let $\sigma = \tau_1 \multimap \dots \multimap \tau_m \multimap \iota$ for some $m \geq 0$ and

without loss of generality assume that there is $a = (x_1; \dots; x_m \searrow b)$

where $x_j \in \llbracket \tau_j \rrbracket$ for all j , such that $a \sqsubseteq \llbracket M \rrbracket$ but $a \not\sqsubseteq \llbracket N \rrbracket$.

$\llbracket M \rrbracket \circ (x_1) \circ \dots \circ (x_m) = b$ while, on the other hand, $\llbracket N \rrbracket \circ (x_1) \circ \dots \circ (x_m) = \emptyset \neq b$, for some $b \in \mathbb{N}$.

By definability (see Lemma 95) there is a term $[x_i]$ s.t. $\llbracket [x_i] \rrbracket = x_i$, for all i .

Therefore $M \not\approx N$, since by Corollary 45, both $M[x_1] \dots [x_m] \Downarrow_e [b]$ and $N[x_1] \dots [x_m] \Uparrow_e$, and the proof is done. ⊙

Definable Compact

Definition 92

Let u be a crisp set of a Scott-Domain in the range of the interpretation of PCF-types. The class of closed terms having $\sqcup u$ as interpretation is denoted by $\lceil u \rceil$, namely $\lceil u \rceil = \{M \mid \llbracket M \rrbracket = x\}$.

$\lceil a_1, \dots, a_k \rceil$ is used as an abbreviation for $\lceil \{a_1, \dots, a_k\} \rceil$ and $\lceil x \rceil = M$ is used as an abbreviation for $M \in \lceil x \rceil$.

If M and N are programs then $M \dot{=} N$ is an abbreviation for the application of the following term to M and N :

$$Y_{\iota \rightarrow \iota \rightarrow \iota} \left(\lambda F^{\iota \rightarrow \iota \rightarrow \iota} \lambda x^{\iota} \lambda y^{\iota} . \text{if } x \text{ (if } y \tilde{0} \tilde{1}) \text{ (if } y \tilde{1} \text{ (F(pred } x \text{) (pred } y \text{)))} \right).$$

It is easy to check that

$$\llbracket M \dot{=} N \rrbracket = \begin{cases} 0 & \llbracket M \rrbracket = m = \llbracket N \rrbracket, \\ 1 & \llbracket M \rrbracket = m \neq n = \llbracket N \rrbracket, \\ \emptyset & \text{otherwise.} \end{cases}$$

Let $N_0 \text{or} N_1$ be an abbreviation for the term $\text{if } N_0 \text{ (if } N_1 \tilde{0} \tilde{0}) N_1$ (being equivalent to $\text{if } N_0 \tilde{0} N_1$, under the hypothesis that both $N_0 \Downarrow_e$ and $N_1 \Downarrow_e$).

Let $N_0 \text{and} N_1$ be an abbreviation for the term $\text{if } N_0 \text{ (if } N_1 \tilde{0} \tilde{1}) \text{ (if } N_1 \tilde{1} \tilde{1})$.

Let $\text{not } N_0$ be an abbreviation for the term $\text{if } N_0 \tilde{1} \tilde{0}$.

It is easy to check that the operational behaviour of **and**, **or** and **not** is the expected one.

Note that **and**, **or** and **not** are strict operators, in the sense that if one of their parameters diverges then their evaluation diverges.

Rank

A non standard measure on types will be useful in the proof of the Lemma 95.

Definition 93

The **RANK** of a type is defined inductively as follows:

- $\text{RANK}(\iota) = 0$
- $\text{RANK}(\sigma \rightarrow \tau) = 1 + \text{RANK}(\sigma) + \text{RANK}(\tau)$.

It is easy to check that

$$\text{RANK}(\mu_1 \rightarrow \dots \rightarrow \mu_m \rightarrow \iota) = m + \sum_{j=1}^m \text{RANK}(\mu_j).$$

Let S be a finite set, we denote $\|S\| \in \mathbb{N}$ the number of elements in S . In particular, recall that if S is a crisp set then $\|S\|$ is defined.

Crisp Set

Every compact element of the continuous function space between Scott-Domains is a finite join of step functions $(a \searrow b)$ where a, b are compact. By the proof of Theorem 69 and Exercise 61.

Moreover, $a \searrow (b \sqcup b')$ is equal to $(a \searrow b) \sqcup (a \searrow b')$ whenever the joins exist.

Exercise 94

Let $\sigma \equiv \tau_1 \multimap \dots \multimap \tau_n \multimap \iota$ for some type τ_1, \dots, τ_n .

Each compact element of the interpretation of σ is the join of a finite set S of compact elements satisfying the following two conditions:

1. Each element of S has the form $(a_1; \dots; a_n \searrow b)$ where $a_i \in \mathbb{k}(\llbracket \tau_i \rrbracket)$ for each i and where $b \in \mathbf{N}_\perp$, $b \neq \perp$
2. If $(a_1; \dots; a_n \searrow b)$ and $(a'_1; \dots; a'_n \searrow b')$ are in S and $\exists a_i \sqcup a'_i$ for each i , then $b = b'$

We use $(a_1; \dots; a_n \searrow b)$ as an abbreviation for

$$(a_1 \searrow (a_2 \dots (a_n \searrow b) \dots)).$$

Indeed a set S satisfying the first condition also satisfies the second one exactly when it is consistent. Let us say that a finite set of compact elements satisfying the previous condition is **crisp**.

Crisp Set



The proof of definability will be developed reasoning on

crisp sets in place of compact elements,
since each compact is the lub of a such finite set.

Note that many crisp sets define the same compact, in particular the empty-set can be considered also as a crisp set such that $\sqcup \emptyset = \perp$.

Definability

Lemma 95

If $\sigma = \tau_1 \rightsquigarrow \dots \rightsquigarrow \tau_k \rightsquigarrow \iota$ for some $k \geq 0$ and $d \in \mathbb{k}(\llbracket \sigma \rrbracket)$ then d is definable.

Proof. We prove that if all compact elements of $\llbracket \tau_i \rrbracket$ are definable then, given a crisp set u in $\llbracket \sigma \rrbracket$, the compact $\sqcup u$ is definable.

The proof is given by induction on the pair $\langle \text{RANK}(\sigma), \|u\| \rangle$ ordered in a lexicographic way, where u is a crisp set of $\llbracket \sigma \rrbracket$.

♣ If $\text{RANK}(\sigma) = 0$ then $\llbracket \sigma \rrbracket = \mathbf{N}_\perp$ and $\sigma = \iota$. Thus Ω_ι and numerals define all possible compact elements.

♣ If $\text{RANK}(\sigma) = 1$ then $\sigma = \iota \rightsquigarrow \iota$ and $\llbracket \sigma \rrbracket = \mathbf{N}_\perp \Rightarrow \mathbf{N}_\perp$.

□ If $\|u\| = 0$ then $u = \emptyset$ is defined by $\Omega_{\iota \rightsquigarrow \iota}$.

□ Let $\|u\| = 1$ and $u = \{(y^0, d^0)\}$ such that $y^0, d^0 \in \mathbb{k}(\mathbf{N}_\perp) = \mathbb{N} \cup \{\perp\}$.

If $y^0 = \perp$ then $\lceil u \rceil = \lambda z^\iota. \lceil d^0 \rceil$.

Let $y^0 \neq \emptyset$ and $y^0 \subseteq \mathbb{k}(\mathbf{N}_\perp) - \{\perp\}$, so $y^0 = \{n\}$ then the considered compact element is defined by $\lambda z^\iota. \text{if } (z \doteq \tilde{n}) \lceil d^0 \rceil \Omega_\iota$.

□ Let $\|u\| > 1$. The case $(\perp, d^0) \in u$ is trivial.

So, let $(y^0, d^0) \in u$ such that $y^0, d^0 \in \mathbb{k}(\mathbf{N}_\perp) = \mathbb{N} \cup \{\perp\}$.

If $u' = u - \{(y^0, d^0)\}$ and $y^0 = \{n\}$ then the considered compact is defined by

$\lambda z^\iota. (z \doteq \lceil n \rceil) \lceil d^0 \rceil (\lceil u' \rceil z)$ where $\lceil u' \rceil$ is well defined by induction, since $\|u'\| < \|u\|$.

♣ Suppose $\text{RANK}(\sigma) \geq 2$, so $\text{RANK}(\sigma) = 1 + m + \sum_{j=1}^m \text{RANK}(\mu_j)$.

1. Suppose there are functions

$$f = (a_1; \dots; a_k \searrow b) \text{ and } f' = (a'_1; \dots; a'_k \searrow b') \in u$$

such that, for some $i \leq k$ the elements a_i, a'_i are inconsistent. Then $(a_i \searrow 0) \sqcup (a_i \searrow 1)$ is the sup of a crisp set, so let $T^{T_i \rightsquigarrow \iota}$ to define it by induction. But also the sets $u - \{f\}, u - \{f'\}$ are crisp, so let M and M' to define it by induction. Thus

$$\lceil \sqcup u \rceil = \lambda x_1^{T_1} \dots x_k^{T_k}. \text{if } (T x_i^{T_i}) (M x_1^{T_1} \dots x_k^{T_k}) (M' x_1^{T_1} \dots x_k^{T_k})$$

2. Suppose that for all $i \leq k$ the elements a_i, a'_i are consistent in all

$$f = (a_1; \dots; a_k \searrow b) \text{ and } f' = (a'_1; \dots; a'_k \searrow b') \in u.$$

In this case $b = b'$ by crisp constraints and there is \tilde{m} that define b .

By induction there are terms $T_i^{T_i}$ defining $(a_i \searrow 0)$ for all $i \leq k$.

But also $\sqcup(u - \{f\})$ is crisp and definable by induction. Thus

$$\lceil \sqcup u \rceil = \lambda x_1^{T_1} \dots x_k^{T_k}. \text{if } (T_1 x_1^{T_1} \text{ and } \dots \text{ and } T_k x_k^{T_k}) \tilde{m} (\lceil \sqcup(u - \{f\}) \rceil x_1^{T_1} \dots x_k^{T_k}).$$

In this case it is possible to appreciate the role of the parallel conditional. Since the values of the two branches of the defining term are the same, the parallel conditional has the same value as each of the branches despite the fact that the test $T_1 x_1^{T_1} \text{ and } \dots \text{ and } T_k x_k^{T_k}$ has a bottom as its meaning. In particular, the sequential **if** would not work because the value of the test could be \perp . ⊗

Constructive Domains

A Scott-Domain is **effectively given (ED)** when:

1. we have an enumeration of its compact elements
2. given two compact, whether or not they have lub and what it is, if it exists, is decidable

An element d of an ED D is **computable** iff the set $\{e \in \mathbb{k}(D) \mid e \sqsubseteq d\}$ is recursively enumerable.

A **constructive domain (CD)** is the collection of all computable elements in an effectively given domain.

CD can be used as model for PCF+. It is fully abstract!

It is a natural question, if all elements (also the non-compact) can be defined in PCF+.

The answer is no!

PCF++ and Universality

PCF ++ extend PCF+ with an existential operator.

$$\begin{array}{l}
 M^\sigma ::= x^\sigma \quad | \quad (\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau} \quad | \quad (P^{\tau \rightarrow \sigma} Q^\tau)^\sigma \quad | \quad Y_\sigma \\
 \quad \quad | \quad \text{if} \quad | \quad \text{succ} \quad | \quad \text{pred} \quad | \quad \tilde{n} \\
 \quad \quad | \quad \text{pif} \quad | \quad \textcircled{\exists}
 \end{array}$$

The new constant \exists has type $(\iota \rightarrow \iota) \rightarrow \iota$ and it is endowed with the evaluation rules:

$$\frac{M \Omega^\iota \Downarrow_e \tilde{k+1}}{\exists M \Downarrow_e \tilde{0}} \quad (\Omega \exists) \qquad \frac{M \tilde{n} \Downarrow_e \tilde{0}}{\exists M \Downarrow_e \tilde{1}} \quad (n \exists)$$

PCF ++ is universal for CD, since all its elements are definable!

Two Terms

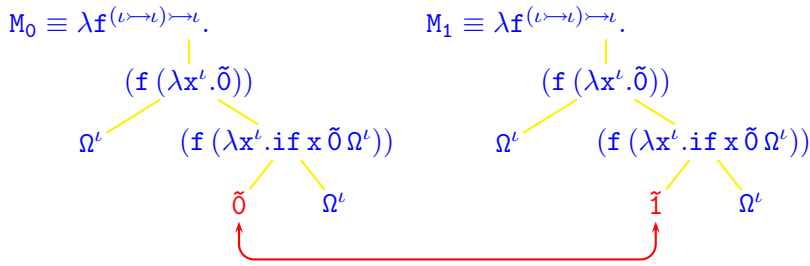
To prove that full abstraction fails for Stable-Domain, we need to show two terms that have the same operational behaviour in all ground contexts but fail to be equal in the model \mathcal{S} .

To this aim, let M_0, M_1 be the following PCF terms:

$$M_0 \equiv \lambda f^{(\iota \multimap \iota) \multimap \iota}. \text{if } (f (\lambda x^\iota. \tilde{0})) \Omega^\iota \overline{(\text{if } (f (\lambda x^\iota. \text{if } x \tilde{0} \Omega^\iota)) \tilde{0} \Omega^\iota)}$$

$$M_1 \equiv \lambda f^{(\iota \multimap \iota) \multimap \iota}. \text{if } (f (\lambda x^\iota. \tilde{0})) \Omega^\iota \overline{(\text{if } (f (\lambda x^\iota. \text{if } x \tilde{0} \Omega^\iota)) \tilde{1} \Omega^\iota)}$$

having type $((\iota \multimap \iota) \multimap \iota) \multimap \iota$.



The given programs are similar except for a ground value. By using an informal notation and unspecified subsuming premodel,

it is easy to see that (recall that M_0, M_1 have type $((\iota \multimap \iota) \multimap \iota) \multimap \iota$)

$$\llbracket M_i \rrbracket = \begin{cases} (d, i) & \text{if } ((\perp \searrow 0) \searrow k + 1), ((0 \searrow 0) \searrow 0) \sqsubseteq d \\ \perp & \text{otherwise} \end{cases}$$

It is easy to check that there is a finite clique of the Stable Model \mathcal{S} satisfying the first constraint of the interpretation of M_0, M_1 .

For example the “strict?”,

strict?	
$(\perp; \tilde{n})$	$\tilde{1}$
$(\tilde{0}; \tilde{n})$	$\tilde{0}$

But M_0, M_1 have the same operational behaviour in all ground contexts.

To see this, it suffices to show that $\mathcal{E}\llbracket M_0 \rrbracket = \mathcal{E}\llbracket M_1 \rrbracket$ by correctness.

In fact, $((\perp \searrow 0) \searrow k + 1), ((0 \searrow 0) \searrow 0)$ does not describe a compact element, since the lack of monotonicity w.r.t. extensional order.

Thus **no continuous graph d that can satisfy the non-trivial constraint of the interpretations of M_0, M_1 , so the interpretations are both equal to \perp .**

StPCF

StPCF extend PCF with two operators.

$$\begin{array}{l}
 M^\sigma ::= x^\sigma \quad | \quad (\lambda x^\mu. N^\tau)^{\mu \rightarrow \tau} \quad | \quad (P^{\tau \rightarrow \sigma} Q^\tau)^\sigma \quad | \quad Y_\sigma \\
 \quad \quad | \quad \text{if} \quad | \quad \text{succ} \quad | \quad \text{pred} \quad | \quad \tilde{n} \\
 \quad \quad | \quad \text{gor} \quad | \quad \text{strict?}
 \end{array}$$

The new constant **gor** has type $\iota \rightarrow \iota \rightarrow \iota \rightarrow \iota$ and it is endowed with the evaluation rules:

$$\begin{array}{c}
 \frac{P_0 \Downarrow_e \tilde{0} \quad P_1 \Downarrow_e \widetilde{k+1}}{\text{gor } P_0 P_1 P_2 \Downarrow_e \tilde{0}} \text{ (0gor)} \qquad \frac{P_1 \Downarrow_e \tilde{0} \quad P_2 \Downarrow_e \widetilde{k+1}}{\text{gor } P_0 P_1 P_2 \Downarrow_e \tilde{1}} \text{ (1gor)} \\
 \\
 \frac{P_2 \Downarrow_e \tilde{0} \quad P_0 \Downarrow_e \widetilde{k+1}}{\text{gor } P_0 P_1 P_2 \Downarrow_e \tilde{2}} \text{ (2gor)}
 \end{array}$$

The new constant **strict?** has type $(\iota \rightarrow \iota) \rightarrow \iota$ and it is endowed with the (non-effective) evaluation rules:

$$\frac{M\tilde{0} \Downarrow_e \quad M\Omega^\iota \Uparrow_e}{\text{strict? } M^{\iota \rightarrow \iota} \Downarrow_e \tilde{0}} \text{ (strict!)} \qquad \frac{M\tilde{0} \Downarrow_e \quad M\Omega^\iota \Downarrow_e}{\text{strict? } M^{\iota \rightarrow \iota} \Downarrow_e \tilde{1}} \text{ (const!)}$$

However, an effective description of the evaluation is presented in the notes!

☯ StPCF



Exercise 96

Show that **gor** can be defined in PCF+. Explain why the converse does not hold.

Exercise 97

Check that the interpretation of **gor** and **strict?** are

$$\begin{array}{l}
 \llbracket \text{gor} \rrbracket = \left\{ \left(\begin{array}{ccc} \{0\}; & \{n+1\}; & \emptyset \searrow 0 \end{array} \right) \mid n \in \mathbb{N} \right\} \\
 \cup \left\{ \left(\begin{array}{ccc} \emptyset; & \{0\}; & \{n+1\} \searrow 1 \end{array} \right) \mid n \in \mathbb{N} \right\} \\
 \cup \left\{ \left(\begin{array}{ccc} \{n+1\}; & \emptyset; & \{0\} \searrow 2 \end{array} \right) \mid n \in \mathbb{N} \right\}
 \end{array}$$

$$\llbracket \text{strict?} \rrbracket = \left\{ \left(\{ \{0\}, n \} \searrow 0 \right) \mid n \in |\mathbb{N}_\perp| \right\} \cup \left\{ \left(\{ \{\emptyset, n\} \} \searrow 1 \right) \mid n \in |\mathbb{N}_\perp| \right\}.$$

Exercise 98

Check that **gor** and **strict?** are stable functions.

$$\begin{array}{c}
 \frac{}{\text{strict?}(\lambda x^t.\tilde{n}) \Downarrow_e \tilde{1}} \quad (\lambda?num) \qquad \frac{}{\text{strict?}(\lambda x^t.x) \Downarrow_e \tilde{0}} \quad (\lambda?x) \\
 \\
 \frac{\text{strict?}(P[Q/x]M_1\dots M_m) \Downarrow_e \tilde{n}}{\text{strict?}((\lambda x^\sigma.P)QM_1\dots M_m) \Downarrow_e \tilde{n}} \quad (?h) \qquad \frac{\text{strict?}(\lambda x^t.P[Q/z]M_1\dots M_m) \Downarrow_e \tilde{n}}{\text{strict?}(\lambda x^t.(\lambda z^\sigma.P)QM_1\dots M_m) \Downarrow_e \tilde{n}} \quad (\lambda?h) \\
 \\
 \frac{\text{strict?}(P(Y_\sigma P)M_1\dots M_m) \Downarrow_e \tilde{n}}{\text{strict?}(Y_\sigma PM_1\dots M_m) \Downarrow_e \tilde{n}} \quad (?Y) \qquad \frac{\text{strict?}(\lambda x^t.P(Y_\sigma P)M_1\dots M_m) \Downarrow_e \tilde{n}}{\text{strict?}(\lambda x^t.(Y_\sigma P)M_1\dots M_m) \Downarrow_e \tilde{n}} \quad (\lambda?Y) \\
 \\
 \frac{M[\tilde{0}/x] \Downarrow_e \widetilde{m+1} \quad \text{strict?}(\lambda x^t.M) \Downarrow_e \tilde{n}}{\text{strict?}(\lambda x^t.\text{pred}M) \Downarrow_e \tilde{n}} \quad (\lambda?pred) \qquad \frac{}{\text{strict? succ} \Downarrow_e \tilde{0}} \quad (?succ) \\
 \\
 \frac{\text{strict?}(\lambda x^t.(M\tilde{0})) \Downarrow_e \tilde{n}}{\text{strict?}(\lambda x^t.\text{strict?}M) \Downarrow_e \tilde{n}} \quad (\lambda??) \qquad \frac{\text{strict?}(\lambda x^t.M) \Downarrow_e \tilde{n}}{\text{strict?}(\lambda x^t.\text{succ}M) \Downarrow_e \tilde{n}} \quad (\lambda?succ) \\
 \\
 \frac{M_0 \Downarrow_e \tilde{0} \quad M_1 \Downarrow_e \tilde{n}}{\text{strict?}(\text{if } M_0 M_1) \Downarrow_e \tilde{1}} \quad (?0if) \qquad \frac{M_0 \Downarrow_e \widetilde{k+1}}{\text{strict?}(\text{if } M_0 M_1) \Downarrow_e \tilde{0}} \quad (?1if) \\
 \\
 \frac{M_0[\tilde{0}/x] \Downarrow_e \tilde{0} \quad \text{strict?}(\lambda x^t.M_0) \Downarrow_e \tilde{n}_0 \quad \text{strict?}(\lambda x^t.M_1) \Downarrow_e \tilde{n}_1}{\text{strict?}(\lambda x^t.\text{if } M_0 M_1 M_2) \Downarrow_e \tilde{n}_0 \text{ or } \tilde{n}_1} \quad (\lambda?0if) \quad (\ddagger) \\
 \\
 \frac{M_0[\tilde{0}/x] \Downarrow_e \widetilde{k+1} \quad \text{strict?}(\lambda x^t.M_0) \Downarrow_e \tilde{n}_0 \quad \text{strict?}(\lambda x^t.M_2) \Downarrow_e \tilde{n}_2}{\text{strict?}(\lambda x^t.\text{if } M_0 M_1 M_2) \Downarrow_e \tilde{n}_0 \text{ or } \tilde{n}_2} \quad (\lambda?1if) \quad (\ddagger) \\
 \\
 \frac{P_0 \Downarrow_e \tilde{0} \quad P_1 \Downarrow_e \widetilde{k+1}}{\text{strict?}(\text{gor } P_0 P_1) \Downarrow_e \tilde{1}} \quad (?0gor) \qquad \frac{P_0 \Downarrow_e \widetilde{k+1}}{\text{strict?}(\text{gor } P_0 P_1) \Downarrow_e \tilde{0}} \quad (?2gor) \\
 \\
 \frac{\frac{P_0[\tilde{0}/x] \Downarrow_e \tilde{0} \quad P_1[\tilde{0}/x] \Downarrow_e \widetilde{k+1}}{\text{strict?}(\lambda x^t.\text{gor } P_0 P_1 P_2) \Downarrow_e \tilde{n}_0 \text{ or } \tilde{n}_1} \quad (\lambda?0gor) \quad (\ddagger)}{\text{strict?}(\lambda x^t.P_0) \Downarrow_e \tilde{n}_0 \quad \text{strict?}(\lambda x^t.P_1) \Downarrow_e \tilde{n}_1} \\
 \\
 \frac{\frac{P_1[\tilde{0}/x] \Downarrow_e \tilde{0} \quad P_2[\tilde{0}/x] \Downarrow_e \widetilde{k+1}}{\text{strict?}(\lambda x^t.\text{gor } P_0 P_1 P_2) \Downarrow_e \tilde{n}_1 \text{ or } \tilde{n}_2} \quad (\lambda?1gor) \quad (\ddagger)}{\text{strict?}(\lambda x^t.P_1) \Downarrow_e \tilde{n}_1 \quad \text{strict?}(\lambda x^t.P_2) \Downarrow_e \tilde{n}_2} \\
 \\
 \frac{\frac{P_2[\tilde{0}/x] \Downarrow_e \tilde{0} \quad P_0[\tilde{0}/x] \Downarrow_e \widetilde{k+1}}{\text{strict?}(\lambda x^t.\text{gor } P_0 P_1 P_2) \Downarrow_e \tilde{n}_0 \text{ or } \tilde{n}_2} \quad (\lambda?2gor) \quad (\ddagger)}{\text{strict?}(\lambda x^t.P_2) \Downarrow_e \tilde{n}_2 \quad \text{strict?}(\lambda x^t.P_0) \Downarrow_e \tilde{n}_0}
 \end{array}$$

‡ Note that $\tilde{n}_0 \text{ or } \tilde{n}_1$ is an abbreviation for the numerals \tilde{k} such that $\text{if } \tilde{n}_0 \tilde{0} \tilde{n}_1 \Downarrow_e \tilde{k}$.

A Uniform Notation

Lemma 99

Let $E = X_1 \Rightarrow \dots \Rightarrow X_m \Rightarrow \mathbf{N}_1$ be a coherence space ($m \geq 1$) and let $(x_1; \dots; x_m \searrow b_x), (y_1; \dots; y_m \searrow b_y)$ be distinct tokens of $|E|$.

$(x_1; \dots; x_m \searrow b_x) \frown_E (y_1; \dots; y_m \searrow b_y)$ iff $\exists k \leq m$ s.t. $x_k \cup y_k \notin Cl(X_k)$.

Proof. Both directions are proved by induction on m .

(\Leftarrow) If $m = 1$ then $x_1 \cup y_1 \notin Cl(X_1)$, by hypotheses. Thus the proof is immediate, by coherence conditions.

If $m \geq 2$ then there are two cases.

If $x_1 \cup y_1 \notin Cl(X_1)$ then again the proof is immediate.

Otherwise, $x_1 \cup y_1 \in Cl(X_1)$ implies $(x_2; \dots; x_m \searrow b_x) \neq (y_2; \dots; y_m \searrow b_y)$, since $\exists k \leq m$ such that $x_k \cup y_k \notin Cl(X_k)$ by hypothesis.

So $(x_2; \dots; x_m \searrow b_x) \frown_E (y_2; \dots; y_m \searrow b_y)$ by induction, and the proof follows by coherence conditions.

(\Rightarrow) Let $m = 1$ and $(x_1, b_x) \frown_E (y_1, b_y)$. There are two cases, since $Cl(\mathbf{N}_1)$ is flat.

The case $b_x = b_y$ implies $x_1 \neq y_1$, since $(x_1, b_x) \neq (y_1, b_y)$ by hypothesis; therefore $x_1 \cup y_1 \notin Cl(X_1)$, by Definition 82.2. In the second case $b_x \sim_{\mathbf{N}_1} b_y$, therefore $x_1 \cup y_1 \notin Cl(X_1)$ by Definition 82.1.

Let $(x_1; \dots; x_m \searrow b_x) \frown_E (y_1; \dots; y_m \searrow b_y)$. If $x_1 \cup y_1 \notin Cl(X_1)$ then the proof is trivial.

If $x_1 \cup y_1 \in Cl(X_1)$ then $(x_2; \dots; x_m \searrow b_x) \circ_E (y_2; \dots; y_m \searrow b_y)$ by coherence conditions, thus there are two cases.

- $(x_2; \dots; x_m \searrow b_x) = (y_2; \dots; y_m \searrow b_y)$ would imply $x_1 = y_1$ by coherence conditions, and therefore $(x_1; \dots; x_m \searrow b_x) = (y_1; \dots; y_m \searrow b_y)$ against the hypothesis.
- The case $(x_2; \dots; x_m \searrow b_x) \frown_E (y_2; \dots; y_m \searrow b_y)$ follows by induction. ⊙

🌀 Uniform Notation i

Ambiguously, we use $(a_1; (a_2; \dots (a_n \searrow b) \dots))$ also in the context of Coherence Spaces, as an abbreviation for the token

$$(a_1, (a_2, \dots (a_n, b) \dots))$$

in the web of the interpretation of a type $\tau_1 \multimap \dots \multimap \tau_n \multimap \iota$ for some $n \in \mathbb{N}$.

Definable Cliques

If P_i, M_i are programs of StPCF ($i \leq 2$) then $\text{gif } P_0 P_1 P_2 M_0 M_1 M_2$ is used as an abbreviation for the term $\text{if } (\text{gor } P_0 P_1 P_2) M_0 \left(\text{if } (\text{pred } (\text{gor } P_0 P_1 P_2)) M_1 M_2 \right)$.

Clearly $\llbracket \text{gif } P_0 P_1 P_2 M_0 M_1 M_2 \rrbracket_\rho = \begin{cases} \llbracket M_0 \rrbracket_\rho & \text{if } \llbracket P_0 \rrbracket_\rho = \{0\}, \llbracket P_1 \rrbracket_\rho = \{n+1\}, \\ \llbracket M_1 \rrbracket_\rho & \text{if } \llbracket P_1 \rrbracket_\rho = \{0\}, \llbracket P_2 \rrbracket_\rho = \{n+1\}, \\ \llbracket M_2 \rrbracket_\rho & \text{if } \llbracket P_2 \rrbracket_\rho = \{0\}, \llbracket P_0 \rrbracket_\rho = \{n+1\}, \\ \emptyset & \text{otherwise.} \end{cases}$

Last, let $\tilde{k}\text{-succ } M$ be an abbreviation for $(\underbrace{\text{succ } \dots \text{succ}}_k M) \dots$ where $k \in \mathbb{N}$ and M is a term (possibly open) having type ι .

Definability Examples

1. Consider $(\{\{3\}, 4\} \in \llbracket \iota \mapsto \iota \rrbracket$; clearly $\lceil \{\{3\}, 4\} \rceil = \lambda x'. \text{if } (x \doteq \tilde{3}) \tilde{4} \Omega_\iota$.

2. Consider $(\{\{\{\{3\}, 4\}\}, 5\} \in \llbracket (\iota \mapsto \iota) \mapsto \iota \rrbracket$.

At a first sight, the term $M \equiv \lambda f^{\iota \mapsto \iota}. \text{if } (f \tilde{3} \doteq \tilde{4}) \tilde{5} \Omega_\iota$ is a natural candidate for $\lceil \{\{\{\{3\}, 4\}\}, 5\} \rceil$ but unfortunately this impression is wrong.

In fact, $\llbracket M \rrbracket = \{\{\{\{\{3\}, 4\}\}, 5\}, \{\{\{\emptyset, 4\}\}, 5\}\}$.

It is easy to check that

$$\lceil \{\{\{\{3\}, 4\}\}, 5\} \rceil = \lambda f^{\iota \mapsto \iota}. \text{if } \left(\begin{array}{l} f \tilde{3} \doteq \tilde{4} \text{ and} \\ \text{strict?}(\lambda z'. f(\tilde{3}\text{-succ } z)) \end{array} \right) \tilde{5} \Omega_\iota.$$

3. Consider $(\{\{\{\{\{3\}, 4\}\}, 5\}\}, 6\} \in \llbracket ((\iota \mapsto \iota) \mapsto \iota) \mapsto \iota \rrbracket$.

Thus $M \equiv \lambda F^{((\iota \mapsto \iota) \mapsto \iota)}. \text{if } (F(\lambda x'. \text{if } (x \doteq \tilde{3}) \tilde{4} \Omega_\iota) \doteq \tilde{5}) \tilde{6} \Omega_\iota$

does not define the given token, in fact

$$\llbracket M \rrbracket = \{\{\{\{\{\{3\}, 4\}\}, 5\}\}, 6\}, \{\{\{\{\emptyset, 4\}\}, 5\}\}, 6\}$$

It is easy to check that

$$\lceil \{\{\{\{\{\{3\}, 4\}\}, 5\}\}, 6\} \rceil = \lambda F^{((\iota \mapsto \iota) \mapsto \iota)}. \text{if } \left(\begin{array}{l} (F(\lambda x'. \text{if } (x \doteq \tilde{3}) \tilde{4} \Omega_\iota) \doteq \tilde{5}) \text{ and} \\ \text{strict?}(\lambda z'. F(\lambda x'. \text{if } (x \doteq \tilde{3}) (\tilde{4}\text{-succ } z) \Omega_\iota)) \end{array} \right) \tilde{6} \Omega_\iota.$$

4. Let $a = (\{\{\{\{\{\{\{3\}, 4\}\}, 5\}\}, 6\}\}, 7\} \in \llbracket (((\iota \mapsto \iota) \mapsto \iota) \mapsto \iota) \mapsto \iota \rrbracket$.

Note that the term

$$M \equiv \lambda F^{(((\iota \mapsto \iota) \mapsto \iota) \mapsto \iota)}. \text{if } (F(\lambda f^{\iota \mapsto \iota}. \text{if } ((f \tilde{3}) \doteq \tilde{4}) \tilde{5} \Omega_\iota) \doteq \tilde{6}) \tilde{7} \Omega_\iota$$

does not define the given token, in fact

$$\llbracket M \rrbracket = \{\{\{\{\{\{\{\{3\}, 4\}\}, 5\}\}, 6\}\}, 7\}, \{\{\{\{\{\{\{\emptyset, 4\}\}, 5\}\}, 6\}\}, 7\}, \{\{\{\{\emptyset, 6\}\}, 7\}\}$$

Let $N \equiv \lambda F^{(((\iota \mapsto \iota) \mapsto \iota) \mapsto \iota)}. \text{if } (F[\lceil \{\{\{3\}, 4\}\}, 5 \rceil] \doteq \tilde{6}) \tilde{7} \Omega_\iota$, where $\lceil \{\{\{3\}, 4\}\}, 5 \rceil$ is defined in (2).

Again, N does not define the considered token. In fact, it is easy to check that

$$\llbracket N \rrbracket = \{\{\{\{\{\{\{\{3\}, 4\}\}, 5\}\}, 6\}\}, 7\}, \{\{\{\{\emptyset, 6\}\}, 7\}\}. \text{ Finally,}$$

$$\lceil a \rceil = \lambda F^{(((\iota \mapsto \iota) \mapsto \iota) \mapsto \iota)}. \text{if } \left(\begin{array}{l} ((F[\lceil \{\{\{3\}, 4\}\}, 5 \rceil] \doteq \tilde{6}) \text{ and} \\ \text{strict?}(\lambda x'. F(\lambda f^{\iota \mapsto \iota}. \text{if } ((f \tilde{3}) \doteq \tilde{4}) (\tilde{5}\text{-succ } z) \Omega_\iota)) \end{array} \right) \tilde{7} \Omega_\iota.$$

5. Let $a = (\underbrace{\{\{10\}, 11\}}_{\iota \mapsto \iota}; \underbrace{\{\{\{\{3\}, 4\}\}, 5\}, \{\{\{\{3\}, 8\}\}, 9\}}_{((\iota \mapsto \iota) \mapsto \iota)}; 6) \in \llbracket (\iota \mapsto \iota) \mapsto ((\iota \mapsto \iota) \mapsto \iota) \mapsto \iota \rrbracket$.

Note that the term

$$M \equiv \lambda f^{\iota \mapsto \iota}. \text{if } (f \tilde{10} \doteq \tilde{11} \text{ and } (F[\lceil \{\{\{3\}, 4\}\} \rceil] \doteq \tilde{5}) \text{ and } (F[\lceil \{\{\{3\}, 8\}\} \rceil] \doteq \tilde{9}) \tilde{6} \Omega_\iota$$

does not define the given token a , in fact

$$\llbracket M \rrbracket = \left\{ \begin{array}{l} \left(\{\{10\}, 11\}; \underbrace{\{\{\{\{3\}, 4\}\}, 5\}, \{\{\{\{3\}, 8\}\}, 9\}}_{((\iota \mapsto \iota) \mapsto \iota)}; 6 \right) \\ \left(\{\{\emptyset, 11\}\}; \underbrace{\{\{\{\{3\}, 4\}\}, 5\}, \{\{\{\{3\}, 8\}\}, 9\}}_{((\iota \mapsto \iota) \mapsto \iota)}; 6 \right) \end{array} \right\}$$

It is easy to check that

$$\lceil a \rceil = \lambda f^{\iota \mapsto \iota}. \text{if } \left(\begin{array}{l} f \tilde{10} \doteq \tilde{11} \text{ and } \text{strict?}(\lambda z'. f(\tilde{10}\text{-succ } z)) \\ (F[\lceil \{\{\{3\}, 4\}\} \rceil] \doteq \tilde{5}) \text{ and } (F[\lceil \{\{\{3\}, 8\}\} \rceil] \doteq \tilde{9}) \end{array} \right) \tilde{6} \Omega_\iota.$$

Full Abstraction and Universality



The same technique used for PCF^+ can be used in order to prove that all finite cliques can be defined in StPCF .

Yet, reasoning as done for PCF^+ it is possible to prove that Stable Domains give a fully abstract model of StPCF [9].

I am proving that StPCF is universal for Coherence Spaces.

Luca Paolini: The Full Abstraction Problem

Summer School Chambéry-Torino, 2006 – 78 / 88

 *Incompleteness of the Presentation* 

Note that this presentation, final remarks and references don't have completeness velleity!

Luca Paolini: The Full Abstraction Problem

Summer School Chambery-Torino, 2006 – **NOTE****Models**

Scott has presented a model based on lattices and continuous functions [12].

Plotkin has proposed has a model based on Scott-Domains and continuous functions [11].

Milner has presented the first fully abstract model of PCF which is built by using a sophisticated inverse limit construction [8]. Milner has also proved that, under some very basic conditions, there is a unique fully abstract model of PCF (up to isomorphism).

Many approaches has been developed in order to model PCF.

Some models are based on syntactical tools.

Mulmuley's model [20] takes the original lattice model of Scott and, using syntactic closure, collapses it syntactically to the fully abstract model.

Stoughton's model [22] applies a syntactically defined preorder to inductively reachable subalgebra in order to build a fully abstract model. The construction was improved by Jung and Stoughton [19].

Abramsky, Jagadeesan, Malacaria [15] and Hyland, Ong [18] have presented game models for PCF, sometimes called AJM games and H2O (since the work of Hyland, Ong is based on that of Hanno Nickau [21]).

Berry has proposed a model for PCF based on dl-Domain endowed with stable functions in [2].

Girard has formalized two interesting subcategories of dl-Domain: Qualitative Domains [17] and Coherence Spaces [5, 6].

Berry has also introduced bidomains that are domains satisfying both extensional and stable order. A nice characterization of bidomains has been obtained by Winskel in [23].

Bucciarelli and Ehrhard [16] refined the model with stronger conditions on functions to arrive at a model that is fully abstract for the first-order fragment of PCF.

Brookes and Geva [3] also achieve full abstraction for a fragment of PCF by using some domain-theoretic ideas.

Some models related to sequentiality are recalled in the next slide.

Luca Paolini: The Full Abstraction Problem

Summer School Chambery-Torino, 2006 – 80 / 88

Sequentiality

Some notions of sequentiality have been considered in [8][31][33].

Kahn and Plotkin in [29] introduced concrete data structures and sequential function that does not form a cartesian closed category.

Then, Berry and Curien [24] have defined sequential algorithms on concrete data structures, by obtaining a cartesian closed category.

Cartwright, Felleisen [28, 27] have contributed to the development of this approach with the introduction of the observably sequential functions.

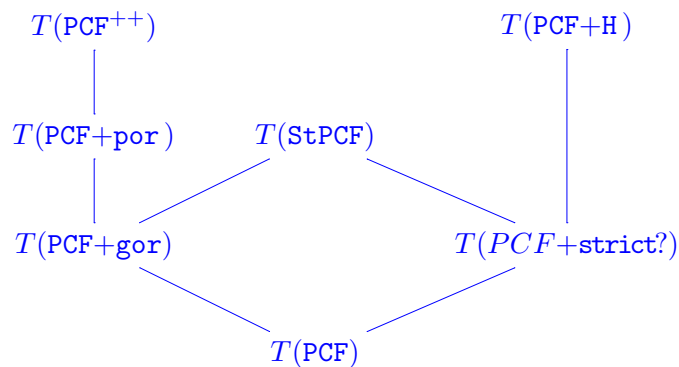
Sieber [32] pioneered a different approach to sequentiality, using logical relations.

O'Hearn and Riecke [30] extending this model have obtained a fully abstract model of PCF.

Bucciarelli has developed some further approaches very interesting to sequentiality, see for instance [25, 26].

Higher-Order Computability

John Longley [41] noted that there are seemingly natural incomparable notions of higher-type computability. In contrast with the Church's thesis, there is no a maximum "higher-type computational formal system".



The research on the higher-type computability has been investigated also by Kleene in a long series of papers [34, 35, 36, 37, 38, 39]. See [40] for a survey.

Bibliography

- [1] H. Abelson, G. J. Sussman, and J. Sussman. *Structure and Interpretation of Computer Programs*. The MIT Press, Cambridge, MA, 1985. <http://mitpress.mit.edu/sicp/full-text/book/book.html>.
- [2] G. Berry. Stable models of typed λ -calculi. In G. Ausiello and C. Böhm, editors, *Fifth International Colloquium Automata, Languages and Programming - - ICALP'78, Udine, Italy, July 17-21, 1978*, volume 62 of *Lecture Notes in Computer Science*, pages 72–89. Springer-Verlag, 1978.
- [3] S. D. Brookes and S. Geva. Sequential functions on indexed domains and full abstraction for a sub-language of pcf. In S. D. Brookes, M. G. Main, A. Melton, M. W. Mislove, and D. A. Schmidt, editors, *Proceedings of Mathematical Foundations of Programming Semantics, 9th International Conference, New Orleans, LA, USA, April 7-10, 1993*, volume 802 of *Lecture Notes in Computer Science*, pages 320–332. Springer-Verlag, 1994.
- [4] R. L. Crole. *Categories for Types*. Cambridge University Press, Cambridge, 1993.
- [5] J.-Y. Girard. Linear logic. *Theoretical Computer Science*, 50:1–102, 1987.
- [6] J.-Y. Girard, Y. Lafont, and P. Taylor. *Proofs and Types*. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1989.
- [7] J. R. Hindley. *Basic Simple Type Theory*, volume 42 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, UK, 1997.
- [8] R. Milner. Fully abstract models of typed lambda-calculus. *Theoretical Computer Science*, 4:1–22, 1977.
- [9] L. Paolini. A stable programming language. *Information and Computation*, 204(3):339–375, 2006.
- [10] G. D. Plotkin. Call-by-name, call-by-value and the λ -calculus. *Theoretical Computer Science*, 1:125–159, 1975.
- [11] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–225, 1977.
- [12] D. S. Scott. A type-theoretical alternative to ISWIM, CUCH, OWHY. *Theoretical Computer Science*, 121(1–2):411–440, 6 Dec. 1993. A Collection of Contributions in Honour of Corrado Böhm on the Occasion of his 70th Birthday. This paper widely circulated in unpublished form since 1969.
- [13] J. E. Stoy. *Denotational Semantics of Programming Languages: The Scott-Strachey Approach to Programming Language Theory*. The MIT Press, Cambridge, USA, 1977.
- [14] W. W. Tait. Intensional interpretation of functionals of finite type. *The Journal of Symbolic Logic*, 32:198–212, 1967.

Some Model References

- [15] S. Abramsky, P. Malacaria, and R. Jagadeesan. Full abstraction for PCF. *Information and Computation*, 163(2):409–470, 2000. An extended abstract can be found in *Theoretical Aspects of Computer Software (Sendai, 1994)*, Lecture Notes in Computer Science, 789:1-15, Springer-Verlag, Berlin, 1994.
- [16] A. Bucciarelli and T. Ehrhard. Extensional embedding of a strongly stable model of PCF. In J. L. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proceedings of the 18th International Colloquium on Automata, Languages and Programming – ICALP*, volume 510 of *Lecture Notes in Computer Science*, pages 35–46, Madrid, Spain, 1991. Springer-Verlag.
- [17] J.-Y. Girard. The system F of variable types, fifteen years later. *Theoretical Computer Science*, 45(2):159–192, 1986.
- [18] J. M. E. Hyland and L. C.-H. Ong. On full abstraction for PCF: I, II, and III. *Information and Computation*, 163(2):285–408, 2000.
- [19] A. Jung and A. Stoughton. Studying the fully abstract model of pcf within its continuous function model. In M. Bezem and J. F. Groote, editors, *Proceedings of International Conference on Typed Lambda Calculi and Applications, TLCA '93, Utrecht, The Netherlands, March 16-18*, volume 664 of *Lecture Notes in Computer Science*, pages 230–244. Springer-Verlag, 1993.
- [20] K. Mulmuley. *Full Abstraction and Semantics Equivalence*. ACM Doctoral Dissertation Award. The MIT Press, 1987.
- [21] H. Nickau. Hereditarily sequential functionals. In A. Nerode and Y. Matiyasevich, editors, *Proceedings of Logical Foundations of Computer Science, Third International Symposium, LFCS'94, St. Petersburg, Russia, July 11-14, 1994*, volume 813 of *Lecture Notes in Computer Science*, pages 253–264. Springer-Verlag, 1994.
- [22] A. Stoughton. *Fully Abstract Models of Programming Languages*. Research Notes in Theoretical Computer Science. Pitman Press and John Wiley and Sons, New York, 1988.
- [23] G. Winskel. Stable bistructure models of PCF. In I. Prívvara, B. Rován, and P. Ruzicka, editors, *19th International Symposium Mathematical Foundations of Computer Science - MFCS'94*, volume 841 of *Lecture Notes in Computer Science*, pages 177–197, Kosice, Slovakia, 22–26 Aug. 1994. Springer-Verlag.

Some Sequentiality References

- [24] G. Berry and P.-L. Curien. Sequential algorithms on concrete data structures. *Theoretical Computer Science*, 20:265–321, 1982.
- [25] A. Bucciarelli. Another approach to sequentiality: Kleene’s unimonotone functions. In S. D. Brookes, M. G. Main, A. Melton, M. W. Mislove, and D. A. Schmidt, editors, *Proceedings of the 9th International Conference of Mathematical Foundations of Programming Semantics, New Orleans, USA, April 7-10*, volume 802 of *Lecture Notes in Computer Science*, pages 333–358. Springer-Verlag, 1993.
- [26] A. Bucciarelli. Degrees of parallelism in the continuous type hierarchy. *Theoretical Computer Science*, 177(1):59–71, 1997.
- [27] R. Cartwright, P.-L. Curien, and M. Felleisen. Fully abstract semantics for observably sequential languages. *Information and Computation*, 111(2):297–401, 1994.
- [28] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. In *Proceedings of 19th Symposium on Principles of Programming Languages - POPL’92*, ACM Sigplan Notices, pages 328–342. ACM Press, 2000.
- [29] G. Kahn and G. Plotkin. Concrete domains. *Theoretical Computer Science*, 121:187–277, 1993. First appeared in French as INRIA-LABORIA technical report, 1978.
- [30] P. W. O’Hearn and J. G. Riecke. Kripke logical relations and PCF. *Information and Computation*, 120(1):107–116, 1995.
- [31] V. Y. Sazonov. Sequentially and parallelly computable functionals. In *Proceedings of Lambda-Calculus and Computer Science Theory*, volume 37 of *Lecture Notes in Computer Science*, pages 312–318, Roma, Italia, 25–27 Mar. 1975. Springer-Verlag.
- [32] K. Sieber. Reasoning about sequential functions via logical relations. In M. P. Fourman, P. T. Johnstone, and A. M. Pitts, editors, *Proc. LMS Symposium on Applications of Categories in Computer Science, Durham 1991*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 258–269. Cambridge University Press, 1992.
- [33] J. Vuillemin. *Proof Techniques for Recursive Programs*. PhD thesis, Computer Science Department, Stanford University, USA, 1973.

Some Higher-Order Computability References

- [34] S. C. Kleene. Recursive functionals and quantifiers of finite types II. *Transactions of the American Mathematical Society*, 108:106–142, 1963.
- [35] S. C. Kleene. Recursive functionals and quantifiers of finite types revisited I. In R. G. J.E. Fenstad and G.E.Sacks, editors, *Symposium on Generalised Recursion Theory II*, pages 185–222. North-Holland, 1978.
- [36] S. C. Kleene. Recursive functionals and quantifiers of finite types revisited II. In H. K. J. Barwise and K. Kunen, editors, *The Kleene Symposium*, pages 1–29. North-Holland, 1980.
- [37] S. C. Kleene. Recursive functionals and quantifiers of finite types revisited III. In G. Metakides, editor, *Patras Logic Symposium*, pages 1–80. North-Holland, 1982.
- [38] S. C. Kleene. Unimonotone functions of finite types (recursive functionals and quantifiers of finite types revisited IV). In A. Nerode and R. Shore, editors, *Recursion Theory, AMS Proceedings of Symposia in Pure Mathematics*, 42, 1985.
- [39] S. C. Kleene. Recursive functionals and quantifiers of finite types revisited V. *Transactions of the American Mathematical Society*, 325:593–630, 1991.
- [40] J. R. Longley. Notions of computability at higher types I. In R. Cori, A. Razborov, S. Todorcevic, and C. Wood, editors, *Proceedings of the Annual European Summer Meeting of the Association for Symbolic Logic – Logic Colloquium 2000*, volume 19 of *Lecture Notes in Logic*, pages 32–142, Paris, France, 2000. Association for Symbolic Logic.
- [41] J. R. Longley. The sequentially realizable functionals. *Annals of Pure and Applied Logic*, 117:1–93, 2002.

Some Further References

- [42] R. Amadio and P.-L. Curien. *Domains and Lambda-Calculi*, volume 46 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, Cambridge, 1998.
- [43] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: the state of the art. In M. Nivat and J. Reynolds, editors, *Algebraic Semantics*, pages 89–132. Cambridge University Press, 1985.
- [44] B. Bloom. Can LCF be topped? Flat lattice models of typed λ -calculus. *Information and Computation*, 87(1-2):263–300, 1990. A preliminary version was appeared in the Proceedings of Third Annual Symposium on Logic in Computer Science, 5-8 July 1988, Edinburgh, UK.
- [45] A. Bucciarelli. *Sequential Models of PCF: some contributions to the domain theoretic approach to full abstraction*. PhD thesis, Dipartimento di Informatica, Università di Pisa, 1993.
- [46] A. Bucciarelli and T. Ehrhard. Extensional embedding of a strongly stable model of PCF. In J. L. Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proceedings of the 18th International Colloquium on Automata, Languages and Programming – ICALP*, volume 510 of *Lecture Notes in Computer Science*, pages 35–46, Madrid, Spain, 1991. Springer-Verlag.
- [47] M. P. Fiore, A. Jung, E. Moggi, P. O’Hearn, J. Riecke, G. Rosolini, and I. Stark. Domains and denotational semantics: History, accomplishments and open problems. *Bulletin of the EATCS*, 59:227–256, June 1996. Also published as Technical Report CSR-96-2, University of Birmingham School of Computer Science.
- [48] C. A. Gunter. *Semantics of Programming Languages: Structures and Techniques*. Foundations of Computing Series. The MIT Press, Cambridge, MA, 1992.
- [49] T. Jim and A. R. Meyer. Full abstraction and context lemma. *SIAM Journal of Computing*, 25(3):663–696, 1996.
- [50] J. C. Mitchell. *Foundations of Programming Languages*. Foundations of Computing Series. The MIT Press, Cambridge, MA, 1996.
- [51] C.-H. L. Ong. Correspondence between operational and denotational semantics: the full abstraction for PCF. In S. Abramsky, D. Gabbay, and T. S. E. Maibaum, editors, *Handbook of Logic in Computer Science*, volume 4, pages 269–356. Oxford University Press, 1995.
- [52] V. Y. Sazonov. Expressibility of functions in D. Scott’s LCF language. *Algebra i Logika*, 15(3):308–330, 1976. Translation from Russian.
- [53] V. Y. Sazonov. Functionals computable in series and in parallel. *Matematicheskii Zhurnal*, 17:648–672, 1976.
- [54] D. S. Scott. Continuous lattices. In F. W. Lawvere, editor, *Toposes, Algebraic Geometry, and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. Springer-Verlag, Berlin, 1972.