# A PER Semantics for Programming Languages

John Mitchell
Ramesh Viswanathan

## Types as PERs

A *partial equivalence relation* over $\mathcal{N}$ is an equivalence relation over some subset of $\mathcal{N}$.

### Intuitive Appeal

$$
\begin{array}{rcl}
\text{Meaning of Term} & = & \text{“Compiled Code”} \\
 & = & \text{bit string} \\
 & & \\
\text{Meaning of Type} & = & \text{Membership Predicate} \\
 & & + \\
 & & \text{Equivalence Relation}
\end{array}
$$

## Example

Represent booleans as single bytes:
true = bytes with LSB 1
false = bytes with LSB 0

$$n : \text{bool iff } n < 2^8$$

For $m, n$ : bool,

$$m \text{ bool } n \text{ iff } m \bmod 2 = n \bmod 2$$

## Functions

A natural number $n$ represents the numeric function $\varphi_n$

$$n : R \to S \text{ iff } \forall x, y.\ x\ R\ y \supset \varphi_n(x)\ S\ \varphi_n(y)$$

For $m, n : R \to S$,

$$m\ R \to S\ n \text{ iff } \forall x : R.\ \varphi_m(x)\ S\ \varphi_n(y)$$

**Definition**

$R \subseteq \mathcal{N} \times \mathcal{N}$ is a partial equivalence relation if $R$ is symmetric and transitive

$$\begin{array}{rcl} n : R & \text{iff} & n\,R\,n \\ [R] & = & \{\, [n]_R \mid n\,R\,n \,\} \end{array}$$

**Examples**

$$\begin{array}{rcl} N, \mathsf{nat} & = & \{\, \langle n, n \rangle \mid n \in \mathcal{N} \,\} \\ \mathsf{unit} & = & \{ \langle 0, 0 \rangle \} \\ m\,R \to S\,n & \text{iff} & \forall x, y.\ x\ R\ y \supset m \cdot x\ S\ n \cdot y \\ m\,R \rightharpoonup S\,n & \text{iff} & \forall x, y.\ x\ R\ y \supset \\ & & \quad m \cdot x{\uparrow}\ \text{ and }\ n \cdot y{\uparrow},\ \text{ or} \\ & & \quad m \cdot x\ S\ n \cdot y \\ R_\perp & = & \mathsf{unit} \rightharpoonup R \end{array}$$

# Desiderata

Suitable category of pers over $\mathcal{N}$ to model

- nat,bool,...
- $\times, \rightarrow, \rightharpoonup, \perp$
- *Fixed Point* Operators at appropriate types
- Recursive Types
- Polymorphism
- Subtyping

## **Extensional PERs** [Freyd, Mulry, Rossolini, Scott]

**Successes**

- Model of polymorphic $\lambda$-calculus with fixed points
- **Uniform** fixed point operator

**Drawbacks**

- Class of ExPERs not closed under recursive isomorphism
- nat,bool are not ExPERs
- Nonstandard intepretations for $\times, \rightarrow, \rightharpoonup, \perp$
- Origins Unclear?

# Failure of Fixed Points

**Recursion Theorem:**
Consider any $F : (N \rightharpoonup N) \to (N \rightharpoonup N)$. There is an $e : N \rightharpoonup N$ with $e\,(N \rightharpoonup N)\; F \cdot e$.

**Conjecture:** For any $F : (R \rightharpoonup S) \to (R \rightharpoonup S)$, there is an $e : R \rightharpoonup S$ with $e\,(R \rightharpoonup S)\; F \cdot e$.

**Disproof:** Take

$$\begin{array}{rcl} R & = & \text{unit} \\ m\,S\,n & \text{iff} & \exists u \in \mathcal{N}\; \forall x < u\; (m \cdot x = n \cdot x) \text{ and} \\ & & \qquad\qquad \forall x \geq u\; (m \cdot x\!\uparrow \text{ and } n \cdot x\!\uparrow) \end{array}$$

$$\begin{array}{rcl} F \cdot f \cdot 0 \cdot n & = & \text{compute } g = f \cdot 0 \\ & & \text{if } n = 0 \\ & & \qquad \text{return } 0 \\ & & \text{else return } g \cdot (n - 1) \end{array}$$

*Problem: $S$ is not "complete"*

# Intrinsic Ordering

**Definition:** Given any per $R$, the *intrinsic preorder* $\leq_R$ given by

$$x \leq_R y \ \text{ iff } \ \forall f : R \rightharpoonup \text{unit}.\ f \cdot x{\downarrow} \supset\ f \cdot y{\downarrow}$$

**Examples**

$$\begin{array}{rcl} x \leq_N y & \text{iff} & x = y \\ x \leq_{N \to N} y & \text{iff} & \forall n.\ x \cdot n = y \cdot n \\ x \leq_{N \rightharpoonup N} y & \text{iff} & \forall n.\ x \cdot n{\downarrow} \supset\ x \cdot n = y \cdot n \end{array}$$

**Monotonicity:** Let $R, S$ be any pers and $f : R \rightharpoonup S$. Then $f$ is monotonic, *i.e.*,

$$\forall x, y \cdot\ x \leq_R y\ \supset\ f \cdot x{\downarrow} \supset\ f \cdot x \leq_S f \cdot y$$

## “Effective” CPOs

An *increasing sequence* $s$ in a per $R$ is a function $s : N \to R$ with $s \cdot n \leq_R s \cdot (n + 1)$ for all $n$.

A per $R$ is an *effective cpo* if

**Antisymmetry**

If $x \leq_R y$ and $y \leq_R x$ then $x \, R \, y$

**Computable Completeness**

Computable supremum function $sup_R$ such that if $s$ is an increasing sequence in $R$, then $sup_R \cdot s$ is a least upper bound of $s$ in $R$.

# Continuity

**Continuity:** Let pers $R, S$ be effective cpos. Every $f : R \rightharpoonup S$ is continuous , *i.e.*, if $s$ is an increasing sequence in $R$ then

$$f \cdot (\mathit{sup}_R \cdot s)\ S\ \mathit{sup}_B \cdot (f \circ s)$$

**Lemma:** Let per $R$ be an effective cpo and $s$ any increasing sequence in $R$. For any $f : R \rightharpoonup$ unit,

$$f \cdot (\mathit{sup}_R \cdot s){\downarrow} \quad \text{iff} \quad \exists n.\ f \cdot (s \cdot n){\downarrow}$$

**Lemma:** Let per $R$ be an effective cpo and $s$ any increasing sequence in $R$. For any $f : R \rightharpoonup \text{unit}$,

$$f \cdot (\mathit{sup}_R \cdot s){\downarrow} \quad \text{iff} \quad \exists n.\ f \cdot (s \cdot n){\downarrow}$$

**Proof Sketch:** Suppose that $\forall n.\ f \cdot (s \cdot n){\uparrow}$ and $f \cdot (\mathit{sup}_R \cdot s){\downarrow}$.

$$\overline{K} = \{\, x \mid x \cdot x{\uparrow} \,\}$$

For any $x \in \mathcal{N}$, sequence $t_x$ defined by

$$\begin{aligned} t_x \cdot n = {} & \text{compute } x \cdot x \text{ for } n \text{ steps;} \\ & \text{if } x \cdot x \text{ halts in } i \leq n \text{ steps} \\ & \text{then return } s \cdot i \\ & \text{else return } s \cdot n \end{aligned}$$

$x \in \overline{K} \quad \text{iff} \quad f \cdot (\mathit{sup}_R \cdot t_x){\downarrow}$

## Fixed Points

A per $R$ is *pointed* if it has a least equivalence class under its intrinsic ordering $\leq_R$

**Fixed Points:** Let per $R$ be a pointed effective cpo. There is a computable function $\mathit{fix}_R : (R \to R) \to R$ with $\mathit{fix}_R \cdot f$ a least fixed point of any $f : R \to R$.

**Proof:** For any $f : R \to R$, define sequence $s_f : N \to R$ by

$$\begin{aligned} s_f \cdot 0 &= \bot_R \\ s_f \cdot (n+1) &= f \cdot (s_f \cdot n) \end{aligned}$$

And,

$$\mathit{fix}_R \cdot f \simeq \mathit{sup}_R \cdot s_f$$

## Independence of Axioms

- There is a per that is not antisymmetric

- There is a per that is antisymmetric and in which all ascending chains have a least upper bound but whose supremum operator cannot be computable

- Both the above pers cannot admit a computable least fixed point operator

## Closure under Type Constructors

nat, unit, bool are effective cpos.

Let pers $R, S$ be effective cpos.

- $R \to S$ is an effective cpo with

  $$f \leq_{R \to S} g \quad \text{iff} \quad \forall x : R.\ f \cdot x \leq_S g \cdot x$$

- $R \rightharpoonup S$ is an effective cpo with

  $$f \leq_{R \rightharpoonup S} g \quad \text{iff} \quad \forall x : R.\ f \cdot x{\downarrow} \supset\ f \cdot x \leq_S g \cdot x$$

- $R_\perp$ is an effective cpo

- $R \times S$ is an effective cpo

## Recursive Types

**ECpo** — the full subcategory of all pers that are effective cpos.

A functor $F$ : **ECpo** $\to$ **ECpo** *realized* by computable functions $f^o, f^{\to}$ such that

- For any effective cpo $R$, if $n \vdash$ *sup*$_R$ then $f^0(n) \vdash$ *sup*$_{F(R)}$
- If $n \vdash f : R \to S$ then $f^{\to}(n) \vdash F(f)$.

Usual type constructors all realizable.

Let $F$ be a realizable functor mapping pointed pers to pointed pers. Then there is an effective cpo $S$ with $S \cong F(S)$. Moreover, $sup_S$ and the isomorphism is uniformly computable from $F$ and the least element of $F$(unit).

## Polymorphism

The realizer for an effective cpo is its supremum function

$$n : \forall t.F(t) \text{ iff } n \cdot \mathit{sup}_R : F(R)$$

for all effective cpo's $R$.

Thus,

$$\mathit{fix} : \forall s.\forall t.((s \rightharpoonup t) \to (s \rightharpoonup t)) \to (s \rightharpoonup t)$$

in the model

## Original Goals

Suitable category of pers over $\mathcal{N}$ to model

$\surd$ nat,bool,...

$\surd$ $\times, \rightarrow, \rightharpoonup, \perp$

$\surd$ Fixed Point Operators at appropriate types

$\surd$ Recursive Types

$\surd$ Polymorphism

$*$ Subtyping

## Original Intuition

Model where meaning of term is equivalence class of “compiled code”

Typical compiler does not use types in producing code

## Expectation

Model with *uniform* (type-independent) interpretation of terms

In particular, *uniform fixed point operator*.

## Uniform Fixed Points

Suppose $M : \sigma$

$$[\![M]\!] = [[\![\mathit{Erase}(M)]\!]]_\sigma$$

- Compiled code oblivous to type information
- PERs over $\mathcal{N}$ give such a model for simply-typed lambda caculus *without fixed points*, *e.g.*, HEO.
  Seek a similar quotient structure *with* fixed points.

*Lambda Calculus = Recursion Theory +*
*Extensionality ?*

# Failure of Uniform Suprema

No uniform computable supremum function for $N$ and $N \rightharpoonup N$, *i.e.*, there is no function $f$ such that for any $s : N \to N$

**(a)** If $s$ is an increasing chain in $N$ then $f \cdot s$ is a least upper bound of $s$ under $\leq_N$

**(b)** If $s$ is an increasing chain in $N \rightharpoonup N$ then $f \cdot s$ is a least upper bound of $s$ under $\leq_{N \rightharpoonup N}$.

## Failure of Untyped Fixed Point Operator

$Y$ — untyped call-by-value fixed point operator

For any pers $R, S$ such that $\exists n \notin R \rightharpoonup S$, there is an $f : (R \rightharpoonup S) \to (R \rightharpoonup S)$ with
$Y \cdot f \notin R \rightharpoonup S$, *i.e.*,
$Y$ *is not a uniform fixed point operator*.

No $F : N \to N$ can be a uniform fixed point operator.