Towards a denotational semantics for the ρ-calculus

Germain Faure
LORIA-Nancy

Alexandre Miquel
PPS-Paris 7

Menu of the next 30 minutes

- Presentation of the ρ-calculus (a fragment).
- Scott semantics.
- Discussions (weakness & new insights).
- On the horizon.

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda x \,.\, M \rightsquigarrow \lambda P \,.\, M$$

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda xyz \,.\, zxy \rightsquigarrow \lambda \mathsf{cons}(x, l) \,.\, x$$

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda xyz\,.\,zxy \leadsto \lambda \mathsf{cons}(x,l)\,.\,x$$

  - Matching constraints:

$$[\mathsf{cons}(x,l) \ll \mathsf{nil}]x$$

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda xyz\,.\,zxy \leadsto \lambda \mathsf{cons}(x, l)\,.\,x$$

  - Matching constraints:

$$[\mathsf{cons}(x, l) \ll \mathsf{nil}]x \qquad\qquad \lambda x\,.\,[\mathsf{f}(y) \ll x]y$$

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda xyz\,.\,zxy \leadsto \lambda \mathsf{cons}(x,l)\,.\,x$$

  - Matching constraints:

$$[\mathsf{cons}(x,l) \ll \mathsf{nil}]x \qquad\qquad \lambda x\,.\,[\mathsf{f}(y) \ll x]y$$

  - Structures:

$$1 \quad\quad 2$$

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda xyz \,.\, zxy \leadsto \lambda \mathsf{cons}(x, l) \,.\, x$$

  - Matching constraints:

$$[\mathsf{cons}(x, l) \ll \mathsf{nil}]x \qquad\qquad \lambda x \,.\, [\mathsf{f}(y) \ll x]y$$

  - Structures:

The ρ-calculus as a generalization of λ-calculus

- "Main design concept: to make all the basic ingredients of rewriting explicit objects" (from IGPAL-01)
- Uniform integration of rewriting and λ-calculus.
- Main ideas of the ρ-calculus:
  - Pattern abstractions:

$$\lambda xyz \,.\, zxy \leadsto \lambda \mathsf{cons}(x, l) \,.\, x$$

  - Matching constraints:

$$[\mathsf{cons}(x, l) \ll \mathsf{nil}]x \qquad\qquad \lambda x \,.\, [\mathsf{f}(y) \ll x]y$$

  - Structures:

$$1; 2$$

Presentation of the ρ-calculus

REMINDERS+EXAMPLES EXAMPLES SKIP

Syntax of the ρ-calculus

Terms	M, N	$::=$	x	(Variables)
		$\mid$	c	(Constructors)
		$\mid$	$\lambda P \,.\, M$	(Abstraction)
		$\mid$	$M \bullet N$	(Fonctional application)
		$\mid$	$[P \ll N]M$	(Delayed matching constraints)
		$\mid$	$M ; N$	(Structure)
Patterns	P	$::=$	x	(Variables)
		$\mid$	$cP_1 \ldots P_n$	(Constructors)

Semantics of the ρ-calculus

$$
\begin{array}{llcl}
(\rho) & (\lambda P\,.\,M)\bullet N & \rightarrow & [P \ll N]\bullet M \\[1em]
(\sigma) & [P \ll N]\bullet M & \rightarrow & \sigma M \\
 & & & \text{if } \sigma = P \prec\!\!\!\ll N \\[1em]
(\delta) & (M_1; M_2)\bullet N & \rightarrow & M_1\bullet N; M_2\bullet N
\end{array}
$$

Examples

$$(\lambda\, \mathsf{cons}(x, l)\,.\, x)^{\bullet}\, \mathsf{cons}(1, \mathsf{nil})$$

$$\mapsto_{\rho} \quad [\mathsf{cons}(x, l) \ll \mathsf{cons}(1, \mathsf{nil})]x$$

$$\mapsto_{\sigma} \quad 1$$

Examples

$$
\begin{aligned}
& (\lambda\, \mathsf{cons}(x, l)\, .\, x)^{\bullet}\, \mathsf{cons}(1, \mathsf{nil}) \\
\mapsto_{\rho} \quad & [\mathsf{cons}(x, l) \ll \mathsf{cons}(1, \mathsf{nil})]x \\
\mapsto_{\sigma} \quad & 1
\end{aligned}
$$

$$
\begin{aligned}
& (\lambda\, \mathsf{cons}(x, l)\, .\, x)^{\bullet} \mathsf{nil} \\
\mapsto_{\rho} \quad & [\mathsf{cons}(x, l) \ll \mathsf{nil}]x
\end{aligned}
$$

Examples

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^{\bullet}\, \mathsf{cons}(1,\mathsf{nil})$$
$$\mapsto_{\rho} \quad [\mathsf{cons}(x,l) \ll \mathsf{cons}(1,\mathsf{nil})]x$$
$$\mapsto_{\sigma} \quad 1$$

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^{\bullet}\mathsf{nil}$$
$$\mapsto_{\rho} \quad [\mathsf{cons}(x,l) \ll \mathsf{nil}]x$$

$$(\lambda \circledcirc\,.\,\mathsf{stop};\, \lambda \circledcirc\,.\,\mathsf{go};\, \lambda \circledcirc\,.\,\mathsf{go};\, \lambda \circledcirc\,.\,\mathsf{stop})^{\bullet} \circledcirc$$

Examples

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^\bullet\, \mathsf{cons}(1,\mathsf{nil})$$
$$\mapsto_\rho \quad [\mathsf{cons}(x,l) \ll \mathsf{cons}(1,\mathsf{nil})]x$$
$$\mapsto_\sigma \quad 1$$

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^\bullet \mathsf{nil}$$
$$\mapsto_\rho \quad [\mathsf{cons}(x,l) \ll \mathsf{nil}]x$$

$$(\lambda \circledcirc\,.\,\mathsf{stop};\, \lambda \circledcirc\,.\,\mathsf{go};\, \lambda \circledcirc\,.\,\mathsf{go};\, \lambda \circledcirc\,.\,\mathsf{stop})^\bullet \circledcirc$$
$$\mapsto\!\!\!\!\twoheadrightarrow_\delta \quad (\lambda \circledcirc\,.\,\mathsf{stop})^\bullet \circledcirc;\, (\lambda \circledcirc\,.\,\mathsf{go})^\bullet \circledcirc;\, (\lambda \circledcirc\,.\,\mathsf{go})^\bullet \circledcirc;\, (\lambda \circledcirc\,.\,\mathsf{stop})^\bullet \circledcirc$$
$$\mapsto\!\!\!\!\twoheadrightarrow_\rho \quad [\circledcirc \ll \circledcirc]\mathsf{stop};\, [\circledcirc \ll \circledcirc]\mathsf{go};\, [\circledcirc \ll \circledcirc]\mathsf{go};\, [\circledcirc \ll \circledcirc]\mathsf{stop}$$
$$\mapsto\!\!\!\!\twoheadrightarrow_\sigma \quad [\circledcirc \ll \circledcirc]\mathsf{stop};\, [\circledcirc \ll \circledcirc]\mathsf{go};\, \mathsf{go};\, \mathsf{stop}$$

Examples

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^{\bullet}\, \mathsf{cons}(1,\mathsf{nil})$$
$$\mapsto_\rho \quad [\mathsf{cons}(x,l) \ll \mathsf{cons}(1,\mathsf{nil})]x$$
$$\mapsto_\sigma \quad 1$$

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^{\bullet}\mathsf{nil}$$
$$\mapsto_\rho \quad [\mathsf{cons}(x,l) \ll \mathsf{nil}]x$$

$$(\lambda{\circledcirc}\,.\,\mathsf{stop}; \lambda{\circledcirc}\,.\,\mathsf{go}; \lambda{\circledcirc}\,.\,\mathsf{go}; \lambda{\circledcirc}\,.\,\mathsf{stop})^{\bullet}{\circledcirc}$$
$$\mapsto\!\!\!\rightarrow_\delta \quad (\lambda{\circledcirc}\,.\,\mathsf{stop})^{\bullet}{\circledcirc}; (\lambda{\circledcirc}\,.\,\mathsf{go})^{\bullet}{\circledcirc}; (\lambda{\circledcirc}\,.\,\mathsf{go})^{\bullet}{\circledcirc}; (\lambda{\circledcirc}\,.\,\mathsf{stop})^{\bullet}{\circledcirc}$$
$$\mapsto\!\!\!\rightarrow_\rho \quad [{\circledcirc} \ll {\circledcirc}]\mathsf{stop}; [{\circledcirc} \ll {\circledcirc}]\mathsf{go}; [{\circledcirc} \ll {\circledcirc}]\mathsf{go}; [{\circledcirc} \ll {\circledcirc}]\mathsf{stop}$$
$$\mapsto\!\!\!\rightarrow_{\sigma + \mathsf{GC}} \quad \mathsf{go};\ \mathsf{stop}$$

Examples

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^\bullet\, \mathsf{cons}(1,\mathsf{nil})$$
$$\mapsto_\rho \quad [\mathsf{cons}(x,l) \ll \mathsf{cons}(1,\mathsf{nil})]x$$
$$\mapsto_\sigma \quad 1$$

$$(\lambda\, \mathsf{cons}(x,l)\,.\,x)^\bullet \mathsf{nil}$$
$$\mapsto_\rho \quad [\mathsf{cons}(x,l) \ll \mathsf{nil}]x$$

$$(\lambda \circledcirc\,.\,\mathsf{stop};\, \lambda \circledcirc\,.\,\mathsf{go};\, \lambda \circledcirc\,.\,\mathsf{go};\, \lambda \circledcirc\,.\,\mathsf{stop})^\bullet \circledcirc$$
$$\mapsto\!\!\!\rightarrow_\delta \quad (\lambda \circledcirc\,.\,\mathsf{stop})^\bullet \circledcirc;\, (\lambda \circledcirc\,.\,\mathsf{go})^\bullet \circledcirc;\, (\lambda \circledcirc\,.\,\mathsf{go})^\bullet \circledcirc;\, (\lambda \circledcirc\,.\,\mathsf{stop})^\bullet \circledcirc$$
$$\mapsto\!\!\!\rightarrow_\rho \quad [\circledcirc \ll \circledcirc]\mathsf{stop};\, [\circledcirc \ll \circledcirc]\mathsf{go};\, [\circledcirc \ll \circledcirc]\mathsf{go};\, [\circledcirc \ll \circledcirc]\mathsf{stop}$$
$$\mapsto\!\!\!\rightarrow_{\sigma + \text{ GC}} \quad \mathsf{go};\ \mathsf{stop}$$

Theory on structures?

Why a denotational semantics

- **Operational semantics** focuses on computation

  Computation equivalence not always clear without the Church-Rosser property

  $\Rightarrow$ Typical difficulty: *Prove that terms* `foo` *and* `bar` *are not convertible*

Why a denotational semantics

- **Operational semantics** focuses on computation

  Computation equivalence not always clear without the Church-Rosser property

  $\Rightarrow$ Typical difficulty: *Prove that terms* `foo` *and* `bar` *are not convertible*

- **Denotational semantics** focuses on the input-output behaviour

  – Computations become transparent (convertible terms have the same denotation)

  – A clear notion of value $\Rightarrow$ connect with mathematical intuitions

Why a denotational semantics

- **Operational semantics** focuses on computation

  Computation equivalence not always clear without the Church-Rosser property

  $\Rightarrow$ Typical difficulty: *Prove that terms* `foo` *and* `bar` *are not convertible*

- **Denotational semantics** focuses on the input-output behaviour

  – Computations become transparent (convertible terms have the same denotation)

  – A clear notion of value $\Rightarrow$ connect with mathematical intuitions

- **How it works:**

  – Find a suitable model D for the given notion of computation

  – Define the interpretation function $[\![\ ]\!]$: Terms $\rightarrow D$

Why a denotational semantics

- **Operational semantics** focuses on computation

  Computation equivalence not always clear without the Church-Rosser property

  $\Rightarrow$ Typical difficulty: *Prove that terms* `foo` *and* `bar` *are not convertible*

- **Denotational semantics** focuses on the input-output behaviour

  – Computations become transparent (convertible terms have the same denotation)

  – A clear notion of value $\Rightarrow$ connect with mathematical intuitions

- **How it works:**

  – Find a suitable model D for the given notion of computation

  – Define the interpretation function $[\![\ ]\!]$: Terms $\rightarrow D$

  – Prove the soundness property: $M_1 \underset{\text{comput.}}{=} M_2 \quad \Rightarrow \quad [\![M_1]\!] = [\![M_2]\!]$

Why a denotational semantics

- **Operational semantics** focuses on computation

  Computation equivalence not always clear without the Church-Rosser property

  $\Rightarrow$ Typical difficulty: *Prove that terms* `foo` *and* `bar` *are not convertible*

- **Denotational semantics** focuses on the input-output behaviour

  – Computations become transparent (convertible terms have the same denotation)

  – A clear notion of value $\Rightarrow$ connect with mathematical intuitions

- **How it works:**

  – Find a suitable model D for the given notion of computation

  – Define the interpretation function $[\![\ ]\!] \ : \ \text{Terms} \to D$

  – Prove the soundness property: $M_1 \underset{\text{comput.}}{=} M_2 \quad \Rightarrow \quad [\![M_1]\!] = [\![M_2]\!]$

  – Syntactic corollaries (Typically: *terms* `foo` *and* `bar` *are not convertible*)

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

  – Partiality: $x \leq y \equiv x$ is less defined than y

  – In Comp. Sc.:

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

  – Partiality: $x \leq y \;\equiv\; x$ is less defined than y

  – In Comp. Sc.: Lack of (observable) information $\equiv$ Non-termination

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

  – Partiality: $x \leq y \quad \equiv \quad x$ is less defined than y

  – In Comp. Sc.: Lack of (observable) information $\equiv$ Non-termination

  – Bottom el^{t}: $\perp \quad \equiv \quad (\lambda x \,.\, xx)(\lambda x \,.\, xx) \quad \equiv$ `10 GOTO 10`, etc.

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

  – Partiality: $x \leq y \quad \equiv \quad x$ is less defined than y

  – In Comp. Sc.: Lack of (observable) information $\equiv$ Non-termination

  – Bottom el^t^: $\perp \quad \equiv \quad (\lambda x \,.\, xx)(\lambda x \,.\, xx) \quad \equiv$ `10 GOTO 10`, etc.

  – Monotonicity: Cannot extract information from non-termination (halting problem)

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

  – Partiality: $x \leq y \quad \equiv \quad x$ is less defined than y

  – In Comp. Sc.: Lack of (observable) information $\equiv$ Non-termination

  – Bottom el$^{\mathrm{t}}$: $\perp \quad \equiv \quad (\lambda x \,.\, xx)(\lambda x \,.\, xx) \quad \equiv$ `10 GOTO 10`, etc.

  – Monotonicity: Cannot extract information from non-termination (halting problem)

  – Finiteness: A finite piece of output is produced by only a finite piece of input

Scott domains

- A Scott domain is a poset $(D, \leq)$ satisfying particular axioms.

  Namely: CPO + bottom element + bounded completeness + algebraicity

  – Work with continuous functions (induced by Scott topology on D)

  – Domain equations such as $D \approx (D \to D)$ have non-trivial solutions

  $\Rightarrow$ Useful to interpret λ-calculi

- **Intuition:** Each point of D represents some amount of information

  – Partiality: $x \leq y \quad \equiv \quad x$ is less defined than y

  – In Comp. Sc.: Lack of (observable) information $\equiv$ Non-termination

  – Bottom el^{t}: $\perp \quad \equiv \quad (\lambda x\,.\,xx)(\lambda x\,.\,xx) \quad \equiv$ `10 GOTO 10`, etc.

  – Monotonicity: Cannot extract information from non-termination (halting problem)

  – Finiteness: A finite piece of output is produced by only a finite piece of input

  – Continuity: $\equiv$ Monotonicity + Finiteness [i.e. commutation with directed limits]

Rho-models

A ρ-model is a Scott-domain D equipped with:

- **Beta-rule** Two (Scott-continuous) functions:

$$\begin{array}{lcl} \mathsf{lam} & : & (D{\rightarrow}D) \rightarrow D \\ \mathsf{app} & : & D \rightarrow (D{\rightarrow}D) \end{array} \quad \text{s.t.} \quad \begin{array}{lcl} \mathsf{app} \circ \mathsf{lam} & = & \mathsf{id}_{D\rightarrow D} \\ \mathsf{lam} \circ \mathsf{app} & \leq & \mathsf{id}_{D} \end{array}$$

Rho-models

A ρ-model is a Scott-domain D equipped with:

- **Beta-rule** Two (Scott-continuous) functions:

$$\begin{array}{rcl} \mathsf{lam} & : & (D{\to}D) \to D \\ \mathsf{app} & : & D \to (D{\to}D) \end{array} \quad \text{s.t.} \quad \begin{array}{rcl} \mathsf{app} \circ \mathsf{lam} & = & \mathsf{id}_{D\to D} \\ \mathsf{lam} \circ \mathsf{app} & \leq & \mathsf{id}_{D} \end{array}$$

- **Pattern-matching** For each constructor c of arity n, two functions:

$$\begin{array}{rcl} c_* & : & D^n \to D \\ c^* & : & D \to \mathsf{opt}(D^n) \end{array} \quad \text{s.t.} \quad c^* \circ c_* \;=\; (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

Rho-models

A ρ-model is a Scott-domain D equipped with:

- **Beta-rule** Two (Scott-continuous) functions:

$$\begin{array}{lcl} \mathsf{lam} & : & (D{\to}D) \to D \\ \mathsf{app} & : & D \to (D{\to}D) \end{array} \qquad \text{s.t.} \qquad \begin{array}{lcl} \mathsf{app} \circ \mathsf{lam} & = & \mathsf{id}_{D\to D} \\ \mathsf{lam} \circ \mathsf{app} & \leq & \mathsf{id}_D \end{array}$$

- **Pattern-matching** For each constructor c of arity n, two functions:

$$\begin{array}{lcl} c_* & : & D^n \to D \\ c^* & : & D \to \mathsf{opt}(D^n) \end{array} \qquad \text{s.t.} \qquad c^* \circ c_* \;=\; (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

- **Errors** For each pattern P of arity n, a function:

$$\mathsf{error}_P \;:\; D \times (D^n \to D) \to D \qquad \text{(no axioms)}$$

Rho-models

A ρ-model is a Scott-domain D equipped with:

- **Beta-rule** Two (Scott-continuous) functions:

$$\begin{array}{rcl} \mathsf{lam} & : & (D{\to}D) \to D \\ \mathsf{app} & : & D \to (D{\to}D) \end{array} \quad \text{s.t.} \quad \begin{array}{rcl} \mathsf{app} \circ \mathsf{lam} & = & \mathsf{id}_{D\to D} \\ \mathsf{lam} \circ \mathsf{app} & \leq & \mathsf{id}_D \end{array}$$

- **Pattern-matching** For each constructor c of arity n, two functions:

$$\begin{array}{rcl} c_* & : & D^n \to D \\ c^* & : & D \to \mathsf{opt}(D^n) \end{array} \quad \text{s.t.} \quad c^* \circ c_* \quad = \quad (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

- **Errors** For each pattern P of arity n, a function:

$$\mathsf{error}_P \ : \ D \times (D^n \to D) \to D \qquad \text{(no axioms)}$$

- **Structures** A function:

$$\mathsf{merge} \ : \ D \times D \to D \quad \text{s.t.} \quad \begin{array}{l} \mathsf{app}(\mathsf{merge}(v_1, v_2), w) = \\ \mathsf{merge}(\mathsf{app}(v_1, w), \mathsf{app}(v_2, w)) \end{array}$$

Additional axioms

Extensions of the basic ρ-theory require extra axioms:

- **Extensionality:**

  (η-rule) $\qquad \mathsf{lam} \circ \mathsf{app} \quad = \quad \mathsf{id}_D$

Additional axioms

Extensions of the basic ρ-theory require extra axioms:

- **Extensionality:**

$$(\eta\text{-rule}) \qquad \mathsf{lam} \circ \mathsf{app} = \mathsf{id}_D$$

- **ACI theory** (or any combination of the axioms **A**, **C**, **I**) :

$$\begin{array}{lrcl}
(\text{Associativity}) & \mathsf{merge}(\mathsf{merge}(v_1, v_2), v_3) & = & \mathsf{merge}(v_1, \mathsf{merge}(v_2, v_3)) \\
(\text{Commutativity}) & \mathsf{merge}(v_1, v_2) & = & \mathsf{merge}(v_2, v_1) \\
(\text{Idempotence}) & \mathsf{merge}(v, v) & = & v
\end{array}$$

Additional axioms

Extensions of the basic ρ-theory require extra axioms:

- **Extensionality:**

  (η-rule) $\qquad \mathsf{lam} \circ \mathsf{app} \;=\; \mathsf{id}_D$

- **ACI theory** (or any combination of the axioms **A**, **C**, **I**) :

  $$\begin{array}{lrcl} (\text{Associativity}) & \mathsf{merge}(\mathsf{merge}(v_1, v_2), v_3) & = & \mathsf{merge}(v_1, \mathsf{merge}(v_2, v_3)) \\ (\text{Commutativity}) & \mathsf{merge}(v_1, v_2) & = & \mathsf{merge}(v_2, v_1) \\ (\text{Idempotence}) & \mathsf{merge}(v, v) & = & v \end{array}$$

- **Constructor discrimination** :

  for all $c \neq c'$ $\qquad c^* \circ c'_* \;=\; (\_ \mapsto \mathsf{none})$

Compiling pattern matching

Compositionality of pattern-matching

The interpretation of constructors (of arity n) provided by any ρ-model

$$c_* : D^n \to D, \qquad c^* : D \to \mathsf{opt}(D^n) \qquad \text{s.t.} \qquad c^* \circ c_* \;=\; (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

is easily extended to all ML-style patterns P (of arity n):

$$P_* : D^n \to D, \qquad P^* : D \to \mathsf{opt}(D^n) \qquad \text{s.t.} \qquad P^* \circ P_* \;=\; (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

Compiling pattern matching

Compositionality of pattern-matching

The interpretation of constructors (of arity n) provided by any ρ-model

$$c_* : D^n \to D, \qquad c^* : D \to \mathsf{opt}(D^n) \qquad \text{s.t.} \qquad c^* \circ c_* \;=\; (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

is easily extended to all ML-style patterns P (of arity n):

$$P_* : D^n \to D, \qquad P^* : D \to \mathsf{opt}(D^n) \qquad \text{s.t.} \qquad P^* \circ P_* \;=\; (\vec{w} \mapsto \mathsf{some}(\vec{w}))$$

The matching function $\mathsf{match}_P : D \times (D^n \to D) \to D$ is defined by

$$\begin{array}{rcll} \mathsf{match}_P(v, f) & = & \mathsf{case\_opt}\ \ P^*(v)\ \ \mathsf{with} & \\ & & \mid\ \ \mathsf{some}(\vec{w}) & \mapsto\ \ f(\vec{w}) \\ & & \mid\ \ \mathsf{none} & \mapsto\ \ \mathsf{error}_P(v, f) \end{array}$$

(where case_opt is the destruction operation of values of type $\mathsf{opt}(D^n)$)

The interpretation function

- **Valuations** A valuation is a function $\rho : \mathcal{V} \to D$ ($\mathcal{V}$ = set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

The interpretation function

- **Valuations** A valuation is a function $\rho : \mathcal{V} \to D$ ($\mathcal{V}$ = set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

- **Interpretation** By induction on M we set:

$$\begin{aligned} [\![x]\!]_\rho &= \rho(x) \\ [\![c]\!]_\rho &= c_* \end{aligned}$$

The interpretation function

- **Valuations** A valuation is a function $\rho : \mathcal{V} \to D$ ($\mathcal{V}$ = set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

- **Interpretation** By induction on M we set:

$$\begin{aligned} [\![x]\!]_\rho &= \rho(x) \\ [\![c]\!]_\rho &= \mathsf{lam}_n(\mathsf{curry}_n(c_*)) \end{aligned}$$

The interpretation function

Valuations A valuation is a function $\rho : \mathcal{V} \to D$ ($\mathcal{V}$ = set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

Interpretation By induction on M we set:

$$
\begin{array}{rcl}
[\![x]\!]_\rho & = & \rho(x) \\
[\![c]\!]_\rho & = & \mathsf{lam}_n(\mathsf{curry}_n(c_*)) \\
[\![[P_{\vec{x}} \ll N]M]\!]_\rho & = & \mathsf{match}_P\big([\![N]\!]_\rho,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big)
\end{array}
$$

The interpretation function

- **Valuations** A valuation is a function $\rho : \mathcal{V} \to D$ $\quad (\mathcal{V} =$ set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

- **Interpretation** By induction on M we set:

$$
\begin{array}{rcl}
[\![x]\!]_\rho & = & \rho(x) \\
[\![c]\!]_\rho & = & \mathsf{lam}_n(\mathsf{curry}_n(c_*)) \\
[\![[P_{\vec{x}} \ll N]M]\!]_\rho & = & \mathsf{match}_P\big([\![N]\!]_\rho,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big) \\
[\![\lambda P_{\vec{x}} \,.\, M]\!]_\rho & = & \mathsf{lam}\Big(v \mapsto \mathsf{match}_P\big(v,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big)\Big)
\end{array}
$$

The interpretation function

- **Valuations** A valuation is a function $\rho : \mathcal{V} \to D$ ($\mathcal{V}$ = set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

- **Interpretation** By induction on M we set:

$$
\begin{array}{rcl}
[\![x]\!]_\rho & = & \rho(x) \\
[\![c]\!]_\rho & = & \mathsf{lam}_n(\mathsf{curry}_n(c_*)) \\
[\![[P_{\vec{x}} \ll N]M]\!]_\rho & = & \mathsf{match}_P\big([\![N]\!]_\rho,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big) \\
[\![\lambda P_{\vec{x}} \,.\, M]\!]_\rho & = & \mathsf{lam}\Big(v \mapsto \mathsf{match}_P\big(v,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big)\Big) \\
[\![MN]\!]_\rho & = & \mathsf{app}\ [\![M]\!]_\rho\ [\![N]\!]_\rho \\
[\![M;N]\!]_\rho & = & \mathsf{merge}([\![M]\!]_\rho, [\![N]\!]_\rho)
\end{array}
$$

The denotation $[\![M]\!]_\rho \in D$ continuously depends on the valuation $\rho \in D^{\mathcal{V}}$.

The interpretation function

- **Valuations** A valuation is a function $\rho : \mathcal{V} \to D$ ($\mathcal{V}$ = set of all variables)

The set of all valuations $D^{\mathcal{V}}$ is a Scott-domain (i.e. infinite cartesian product).

- **Interpretation** By induction on M we set:

$$
\begin{aligned}
[\![x]\!]_\rho &= \rho(x) \\
[\![c]\!]_\rho &= \mathsf{lam}_n(\mathsf{curry}_n(c_*)) \\
[\![[P_{\vec{x}} \ll N]M]\!]_\rho &= \mathsf{match}_P\big([\![N]\!]_\rho,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big) \\
[\![\lambda P_{\vec{x}} \,.\, M]\!]_\rho &= \mathsf{lam}\Big(v \mapsto \mathsf{match}_P\big(v,\ (\vec{w} \mapsto [\![M]\!]_{(\rho;\vec{x}\leftarrow\vec{w})})\big)\Big) \\
[\![M N]\!]_\rho &= \mathsf{app}\ [\![M]\!]_\rho\ [\![N]\!]_\rho \\
[\![M ; N]\!]_\rho &= \mathsf{merge}([\![M]\!]_\rho, [\![N]\!]_\rho)
\end{aligned}
$$

The denotation $[\![M]\!]_\rho \in D$ continuously depends on the valuation $\rho \in D^{\mathcal{V}}$.

- **Closed case** If M is closed, we write $[\![M]\!] = [\![M]\!]_\rho$ (does not depend on ρ)

The soundness property

Write: $$D \models M_1 = M_2 \qquad \equiv \qquad \forall \rho \in D^{\mathcal{V}} \quad [\![M_1]\!]^D_\rho = [\![M_2]\!]^D_\rho$$

The soundness property

Write: $D \models M_1 = M_2 \qquad \equiv \qquad \forall \rho \in D^{\mathcal{V}} \quad [\![M_1]\!]^D_\rho = [\![M_2]\!]^D_\rho$

Lemma (Soundness) — *In all the ρ-models D:*

$$M_1 \underset{\rho\sigma\delta}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

The soundness property

Write: $$D \models M_1 = M_2 \qquad \equiv \qquad \forall \rho \in D^{\mathcal{V}} \quad [\![M_1]\!]^D_\rho = [\![M_2]\!]^D_\rho$$

Lemma (Soundness) — *In all the ρ-models D:*

$$M_1 \underset{\rho\sigma\delta}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

Lemma (Soundness w.r.t. ACI) — *In all the ρ-models D where* 'merge' *is* $\boldsymbol{ACI}$*:*

$$M_1 \underset{\substack{\rho\sigma\delta \\ \mathsf{ACI}}}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

The soundness property

Write: $$D \models M_1 = M_2 \qquad \equiv \qquad \forall \rho \in D^{\mathcal{V}} \quad [\![M_1]\!]^D_\rho = [\![M_2]\!]^D_\rho$$

Lemma (Soundness) — *In all the ρ-models D:*

$$M_1 \underset{\rho\sigma\delta}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

Lemma (Soundness w.r.t. ACI) — *In all the ρ-models D where* 'merge' *is* $\boldsymbol{ACI}$*:*

$$M_1 \underset{\substack{\rho\sigma\delta \\ \mathsf{ACI}}}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

Remarks:

– Soundness still holds for any combination of A, C, I and/or the η-reduction rule (provided we restrict to the corresponding notion of ρ-model).

The soundness property

Write: $D \models M_1 = M_2 \quad \equiv \quad \forall \rho \in D^{\mathcal{V}} \quad [\![M_1]\!]_\rho^D = [\![M_2]\!]_\rho^D$

Lemma (Soundness) — *In all the ρ-models D:*

$$M_1 \underset{\rho\sigma\delta}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

Lemma (Soundness w.r.t. ACI) — *In all the ρ-models D where* 'merge' *is* ***ACI****:*

$$M_1 \underset{\substack{\rho\sigma\delta \\ \text{ACI}}}{=} M_2 \qquad \Rightarrow \qquad D \models M_1 = M_2$$

Remarks:

– Soundness still holds for any combination of A, C, I and/or the η-reduction rule (provided we restrict to the corresponding notion of ρ-model).

– Proofs do not depend on any kind of Church-Rosser/confluence property.

A fundamental particular case: D^∞

Let D^∞ be the 'historical' non-trivial solution of: $D^\infty \approx (D^\infty \to D^\infty)$

and set: merge$(v_1, v_2) := \sup\{v_1, v_2\}$ [Works since D^∞ has a top-element]

error$_P(v, f) := \bot$

A fundamental particular case: D^∞

Let D^∞ be the 'historical' non-trivial solution of: $D^\infty \approx (D^\infty \to D^\infty)$

and set: merge$(v_1, v_2) := \sup\{v_1, v_2\}$ [Works since D^∞ has a top-element]
error$_P(v, f) := \bot$

Lemma. — D^∞ *is a* ρ*-model that satisfies the axioms* $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{I}$ *and* η

A fundamental particular case: D^∞

Let D^∞ be the 'historical' non-trivial solution of: $D^\infty \approx (D^\infty \to D^\infty)$

and set: $\quad \text{merge}(v_1, v_2) := \sup\{v_1, v_2\}$ [Works since D^∞ has a top-element]

$\qquad \text{error}_P(v, f) := \bot$

Lemma. — *D^∞ is a ρ-model that satisfies the axioms $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{I}$ and η*

Moreover, the following diagram commutes:

$$\begin{tikzcd}
\rho\text{-calculus} \arrow[r, "[\![\cdot]\!]"] & D^\infty \\
\lambda\text{-calculus} \arrow[u, "\text{syntactic embedding}"] \arrow[ur, "\text{Scott's } [\![\cdot]\!]"'] &
\end{tikzcd}$$

A fundamental particular case: D^∞

Let D^∞ be the 'historical' non-trivial solution of: $D^\infty \approx (D^\infty \to D^\infty)$

and set: $\mathrm{merge}(v_1, v_2) := \sup\{v_1, v_2\}$ [Works since D^∞ has a top-element]
$\mathrm{error}_P(v, f) := \bot$

Lemma. — *D^∞ is a ρ-model that satisfies the axioms $\boldsymbol{A}$, $\boldsymbol{C}$, $\boldsymbol{I}$ and η*
Moreover, the following diagram commutes:

$$
\begin{tikzcd}
\rho\text{-calculus} \arrow[r, "{[\![\cdot]\!]}"] & D^\infty \\
\lambda\text{-calculus} \arrow[u, "\text{syntactic embedding}"] \arrow[ur, "\text{Scott's } {[\![\cdot]\!]}"'] &
\end{tikzcd}
$$

Since Scott's interpretation is faithfull (i.e. injective) on $\beta\eta$-normal forms, we get:
Corollary (Conservativity on λ-normal forms / Weak C.R.) — *The $\rho\sigma\delta\eta$ACI-theory of the ρ-calculus identifies no pair of distinct $\beta\eta$-normal terms of the λ-calculus.*

Discussion

- Compositionality of matching.
- Right distributivity rule.
- Notion of values.
- Weakness of the model
  - Management of errors.
  - Structures (cannot be destructed).

  $\Rightarrow$ Monads.

Another perspective

- The construction $\lambda P_{\vec{x}} \,.\, M_{\vec{x}}$ can be understood as $M \circ P^{-1}$

  The (hidden) dream of ρ-calculists: allow the use of all terms as patterns...
  $\Rightarrow$ Allow the inversion of arbitrary functions: $M^{-1} \;\equiv\; \lambda M x \,.\, x$

Another perspective

- The construction $\lambda P_{\vec{x}} \,.\, M_{\vec{x}}$ can be understood as $M \circ P^{-1}$

  The (hidden) dream of ρ-calculists: allow the use of all terms as patterns...
  $\Rightarrow$ Allow the inversion of arbitrary functions: $M^{-1} \equiv \lambda Mx \,.\, x$

- **Slogans:**

1940's	The λ-calculus:	*'Legalize (arbitrary) application'*
2000's	The ρ-calculus:	*'Legalize (arbitrary) inversion'*

Another perspective

- The construction $\lambda P_{\vec{x}} \,.\, M_{\vec{x}}$ can be understood as $M \circ P^{-1}$

  The (hidden) dream of ρ-calculists: allow the use of all terms as patterns...
  $\Rightarrow$ Allow the inversion of arbitrary functions: $M^{-1} \equiv \lambda M x \,.\, x$

- **Slogans:**

1940's	The λ-calculus:	*'Legalize (arbitrary) application'*
2000's	The ρ-calculus:	*'Legalize (arbitrary) inversion'*

- But inversion is fundamentally anti-monotonic! [Think of $x \mapsto 1/x$, $f \mapsto f^{-1}$]

  $\Rightarrow$ The full dream of ρ-calculists will not be realised with Scott-style semantics

  $\Rightarrow$ But an exciting challenge for denotational semantics!