The Functional Machine Calculus

Willem Heijltjes
University of Bath

MFPS 2025, Glasgow

The λ-calculus

$$M, N ::= x \mid \lambda x.\, M \mid M\, N$$

$$\sigma, \tau ::= o \mid \sigma \to \tau$$

Good features allow **reasoning** about terms (functional programs):

- Confluent reduction
- Equational theory
- Types give strong normalization
- Minimal syntax
- Intuitive semantics (functions)

How to extend this to imperative programming (**effects**, **control flow**)?

A typical(?) imperative language

$$
\begin{array}{rcll}
M, N & ::= & i \in \mathbb{Z} \mid M + N \mid M \times N \mid \ \dots & \text{arithmetic} \\
& \mid & \top \mid \bot \mid M \wedge N \mid M \leq N \mid \ \dots & \text{booleans} \\
& \mid & a := M \mid\ !a & \text{store} \\
& \mid & \mathit{read} \mid \mathit{print}\ M & \text{input/output} \\
& \mid & M \oplus N & \text{probabilistic choice} \\
& \mid & \mathit{skip} \mid M; N & \text{sequencing} \\
& \mid & \mathit{if}\ M\ N\ P & \text{conditional} \\
& \mid & \mathit{while}\ M\ \mathit{do}\ N \mid \mathit{break} \mid \mathit{return}\ M & \text{loops (with escape)} \\
& \mid & \mathit{throw}\ e \mid \mathit{try}\ M\ \mathit{catch}\ e\ N & \text{exceptions}
\end{array}
$$

Reasoning for effects

- Continuations [Landin 1965] [Reynolds 1993]
- Hoare logic [Hoare 1969]
- Idealized Algol [Reynolds 1978]
- Monads [Moggi 1991]
- Call–by–push–value [Levy 1999]
- Algebraic effects [Plotkin & Power 2002]
- Effect handlers [Plotkin & Pretnar 2009]

The Functional Machine Calculus (FMC)[1]

Idea: λ-calculus as a **stack language**, via the Krivine Machine[2]

Extend with **effects** and **control flow** while preserving:

- Confluence
- Typed termination
- Minimal syntax

[1] [H 2022] [Barrett, H & McCusker 2023] [2] [Krivine 2007]

Sequencing[1]

- Imperative sequencing
- Strategies: CBV[2], computational metalanguage[3], CBPV[4]

Control (new!)

- Exception handling
- Constants & data types (non-recursive)
- Iteration (loops) with escape

Locations[1]

- Mutable store
- Input/output
- Probabilistic/non-deterministic sampling

[1] [H 2022] [2] [Plotkin 1975] [3] [Moggi 1991] [4] [Levy 1999]

The λ-calculus as a stack language

$$M, N \ ::= \ x \ \mid \ M\ N \ \mid \ \lambda x.\, M$$

$$M, N \ ::= \ x \ \mid \ [N].\, M \ \mid \ \langle x \rangle.\, M$$

Stacks: $S, T \ ::= \ \varepsilon \ \mid \ S\, M$

Evaluate M with stack S: $S \xmapsto{M} \twoheadrightarrow$

Application $[N].\, M$ is **push**:

$$S \xmapsto{[N]} S\, N \xmapsto{M} \twoheadrightarrow$$

Abstraction $\langle x \rangle.\, M$ is **pop**:

$$S\, N \xmapsto{\langle x \rangle} S \xmapsto{\{N/x\}M} \twoheadrightarrow$$

Beta-reduction $[N].\, \langle x \rangle.\, M \longrightarrow \{N/x\}M$ shortens evaluation:

$$S \xmapsto{[N]} S\, N \xmapsto{\langle x \rangle} S \xmapsto{\{N/x\}M} \twoheadrightarrow \qquad \longrightarrow \qquad S \xmapsto{\{N/x\}M} \twoheadrightarrow$$

[Krivine 2007]

Sequencing

- Imperative sequencing
- Strategies: CBV, computational metalanguage, CBPV

Control

- Exception handling
- Constants & data types (non-recursive)
- Iteration (loops)

Locations

- Mutable store
- Input/output
- Probabilistic/non-deterministic sampling

Sequencing

$$M, N \;::=\; \underbrace{x \;\mid\; [N].\,M \;\mid\; \langle x\rangle.\,M}_{\lambda\text{-calculus}} \;\mid\; \underbrace{\star \;\mid\; M\,;N}_{\text{sequencing}}$$

Skip/identity $\star$

$$S \xmapsto{\;\star\;}\!\!\!\twoheadrightarrow S$$

Sequence/composition $M\,;N$

$$R \xmapsto{\;M\;}\!\!\!\twoheadrightarrow S \xmapsto{\;N\;}\!\!\!\twoheadrightarrow T$$

Evaluation now **returns** a stack. Example: duplicate $\langle x\rangle.\,[x].\,[x].\,\star$

$$S\,N \xmapsto{\langle x\rangle} S \xmapsto{[N]} S\,N \xmapsto{[N]} S\,N\,N \xmapsto{\;\star\;}\!\!\!\twoheadrightarrow S\,N\,N$$

Example: swap $\langle x\rangle.\,\langle y\rangle.\,[x].\,[y].\,\star$

$$S\,N\,M \xmapsto{\langle x\rangle} S\,N \xmapsto{\langle y\rangle} S \xmapsto{[M]} S\,M \xmapsto{[N]} S\,M\,N \xmapsto{\;\star\;}\!\!\!\twoheadrightarrow S\,M\,N$$

Cf. κ-calculus [Hasegawa 1995] [Power & Thielecke 1999], concatenative programming

Sequencing

$$M, N \;::=\; \underbrace{x \;\mid\; [N].\,M \;\mid\; \langle x\rangle.\,M}_{\lambda\text{-calculus}} \;\mid\; \underbrace{\star \;\mid\; M\,;N}_{\text{sequencing}}$$

Reduction shortens or preserves evaluation:

$$\star\;;M \longrightarrow M \qquad\qquad S \overset{\star}{\longmapsto\!\!\!\!\rightarrow} S \overset{M}{\longmapsto\!\!\!\!\rightarrow} T \;\longrightarrow\; S \overset{M}{\longmapsto\!\!\!\!\rightarrow} T$$

$$(M\,;N)\,;P \longrightarrow M\,;(N\,;P) \qquad\qquad R \overset{M}{\longmapsto\!\!\!\!\rightarrow} S \overset{N}{\longmapsto\!\!\!\!\rightarrow} T \overset{P}{\longmapsto\!\!\!\!\rightarrow} U$$

$$(\langle x\rangle.\,M)\,;N \longrightarrow \langle x\rangle.\,(M\,;N) \quad (x \notin \mathrm{fv}(N)) \qquad\qquad R\,P \overset{\langle x\rangle}{\longmapsto} R \overset{\{P/x\}M}{\longmapsto\!\!\!\!\rightarrow} S \overset{N}{\longmapsto\!\!\!\!\rightarrow} T$$

$$([P].\,M)\,;N \longrightarrow [P].\,(M\,;N) \qquad\qquad R \overset{[P]}{\longmapsto} R\,P \overset{M}{\longmapsto\!\!\!\!\rightarrow} S \overset{N}{\longmapsto\!\!\!\!\rightarrow} T$$

Sequencing: types

$$M, N ::= \overbrace{x \mid [N].M \mid \langle x\rangle.M}^{\lambda\text{-calculus}} \mid \overbrace{\star \mid M;N}^{\text{sequencing}}$$

Types indicate the input stack and return stack

$$\sigma, \tau ::= \sigma_1 \dots \sigma_n \Rightarrow \tau_1 \dots \tau_m$$

Interpretation: (with vector notation $\overline{\tau} = \tau_1 \dots \tau_n$)

$$M : \overline{\sigma} \Rightarrow \overline{\tau} \quad \Longrightarrow \quad \forall S : \overline{\sigma}.\ \exists T : \overline{\tau}.\ S \xmapsto{M} T$$

Semantics: (strict) Cartesian closed category

objects:	type vectors $\overline{\tau}$	products:	$\overline{\sigma} \times \overline{\tau} = \overline{\sigma}\,\overline{\tau}$	$1 = \varepsilon$
morphisms:	closed terms $M : \overline{\sigma} \Rightarrow \overline{\tau}$	closure:	$\overline{\sigma} \to \overline{\tau} = \overline{\sigma} \Rightarrow \overline{\tau}$	

Like [Hasegawa 1995] [Power & Thielecke 1999], different from [Lambek & Scott 1988]!

Embedded calculi

CBN λ-calculus

$$\sigma_1 \to \ldots \to \sigma_n \to o \;=\; \sigma_1 \ldots \sigma_n \Rightarrow \varepsilon$$

CBV λ-calculus

$$\begin{aligned} x_v &= x \\ (\lambda x.M)_v &= \langle x \rangle .\, M_c \\ V_c &= [V_v] .\, \star \\ (M\,N)_c &= N_c \;;\; M_c \;;\; \langle x \rangle .\, x \end{aligned}$$

$$\begin{aligned} o_v &= \varepsilon \Rightarrow \varepsilon \\ (\sigma \to \tau)_v &= \sigma_v \Rightarrow \tau_v \\ \tau_c &= \varepsilon \Rightarrow \tau_v \end{aligned}$$

Computational metalanguage/CBPV

$$\begin{aligned} \text{return } M &= [M] .\, \star \\ \text{let } x = M \text{ in } N &= M \;;\; \langle x \rangle .\, N \end{aligned}$$

$$\sigma_1 \to \cdots \to \sigma_n \to T\tau \;=\; \sigma_1 \ldots \sigma_n \Rightarrow \tau$$

[Douence & Fradet 1998] [Power & Thielecke 1999]

Sequencing

- Imperative sequencing
- Strategies: CBV, computational metalanguage, CBPV

Control

- Exception handling
- Constants & data types (non-recursive)
- Iteration (loops) with escape

Locations

- Mutable store
- Input/output
- Probabilistic/non-deterministic sampling

Branching and looping: coproducts

Branching computation is modeled by a sum of return values:

- Constant types are sums; e.g. Booleans: $\mathbb{B} = 1 + 1$
- Algebraic data types are sums–of–products
- The exception monad $TX = E + X$ (for a set of exceptions E) is a sum

Iteration is also modelled by sums[1] (dual to recursion[2])

$$\frac{f \;:\; A \to B + A}{\mathrm{iter}\, f \;:\; A \to B}$$

Idea:

- **Skip** ($\star$) signifies successful termination
- Generalize to multiple **injections** or **choices** $\star, i, j, k, \ldots$ indicating different branches or modes of termination

[1] [Bloom & Esik 1993] [2] [Simpson & Plotkin 2000]

The Control extension

$$\overbrace{\star \mid M\,;N}^{\text{sequencing}}$$

$$M, N \;::=\; \underbrace{x \;\mid\; [N].\,M \;\mid\; \langle x\rangle.\,M}_{\lambda\text{-calculus}} \;\mid\; \underbrace{i \;\mid\; M\,;i{\to}N \;\mid\; M^i}_{\text{control}}$$

Evaluation now returns a stack plus a **choice** (injection/exit mode) i:

$$S \overset{M}{\longmapsto\!\!\!\!\twoheadrightarrow} T, i$$

A choice i: $\quad S \overset{i}{\longmapsto\!\!\!\!\twoheadrightarrow} S, i$

A case $M\,;i{\to}N$: $\quad R \overset{M}{\longmapsto\!\!\!\!\twoheadrightarrow} S, i \overset{N}{\longmapsto\!\!\!\!\twoheadrightarrow} T, k \qquad R \overset{M}{\longmapsto\!\!\!\!\twoheadrightarrow} S, j \qquad (i \neq j)$

A loop M^i: $\quad R \overset{M}{\longmapsto\!\!\!\!\twoheadrightarrow} S, i \overset{M^i}{\longmapsto\!\!\!\!\twoheadrightarrow} T, k \qquad R \overset{M}{\longmapsto\!\!\!\!\twoheadrightarrow} S, j \qquad (i \neq j)$

Standard sequencing is retained by syntactic sugar: $\quad M\,;N \;=\; M\,;\,\star \to N$

Constants and exceptions

$$M, N \;::=\; \overbrace{x \;\mid\; [N].M \;\mid\; \langle x\rangle.M}^{\lambda\text{-calculus}} \;\mid\; \overbrace{i \;\mid\; M\,;i{\to}N \;\mid\; M^i}^{\text{control}}$$

Exceptions are choices as **computations**:

$$\text{throw e} \;=\; \text{e}$$

$$\text{try } \{M\} \text{ catch e } \{N\} \;=\; M \;;\; \text{e}{\to}N$$

Booleans and other **constants** are choices as **values**:

$$\top, \bot \;=\; [\top].\star \;,\; [\bot].\star$$

$$\text{if } B \text{ then } M \text{ else } N \;=\; B\,;\langle x\rangle.x\;;\;\top{\to}M\;;\;\bot{\to}N$$

$$c \;=\; [c].\star$$

$$\text{case } M \text{ of } \{c_1 \to N_1, \ldots, c_n \to N_n\} \;=\; M\,;\langle x\rangle.x\,;c_1{\to}N_1\;;\;\ldots\;;c_n{\to}N_n$$

Data types

$$M, N \;::=\; \overbrace{x \;\mid\; [N].M \;\mid\; \langle x\rangle.M}^{\lambda\text{-calculus}} \;\mid\; \overbrace{i \;\mid\; M\,;i\to N \;\mid\; M^{i}}^{\text{control}}$$

Data constructors are choices:

$$c\,M_1\ \ldots\ M_n \;=\; [M_n]\ldots[M_1].\,c$$

Pattern-matching becomes unnecessary: arguments are passed on the stack

$$\text{case } M \text{ of } \{c_1\,\overline{x}_1 \to N_1, \ldots, c_n\,\overline{x}_n \to N_n\}$$

$$=$$

$$M \;;\; c_1 \to \langle\overline{x}_1\rangle.\,N_1 \;;\; \ldots \;;\; c_n \to \langle\overline{x}_n\rangle.\,N_n$$

Loops

$$M, N \;::=\; \overbrace{x \;\mid\; [N].\,M \;\mid\; \langle x\rangle.\,M}^{\lambda\text{-calculus}} \;\mid\; \overbrace{i \;\mid\; M\,;i\to N \;\mid\; M^{i}}^{\text{control}}$$

Do–while loops:

$$\text{do } M \text{ while } B \;=\; (M\,;B\,;\langle x\rangle.\,x)^{\top}\,;\bot\to\star$$

While–do loops:

$$\text{while } B \text{ do } M \;=\; (B\,;\langle x\rangle.\,x\,;\top\to M)^{\star}\,;\bot\to\star$$

Breaks and **returns** are choices:

$$\text{break} \;=\; \text{brk}$$

$$\text{return } M \;=\; [M].\,\text{ret}$$

$$\text{while true do } M \;=\; M^{\star}\,;\text{brk}\to\star\;;\text{ret}\to\star$$

Reduction

$$M, N \ ::= \ \overbrace{x \ \mid \ [N].\,M \ \mid \ \langle x\rangle.\,M}^{\lambda\text{-calculus}} \ \mid \ \overbrace{i \ \mid \ M\,;i{\to}N \ \mid \ M^{i}}^{\text{control}}$$

Reduction shortens or preserves evaluation: (let $i \neq j$)

$$i\,;i{\to}M \ \longrightarrow \ M \qquad S \xmapsto{\ i\ } S, i \xmapsto{\ M\ }\!\!\!\twoheadrightarrow T, k \ \longrightarrow \ S \xmapsto{\ M\ }\!\!\!\twoheadrightarrow T, k$$

$$i\,;j{\to}M \ \longrightarrow \ i \qquad S \xmapsto{\ i\ } S, i$$

$$M^{i} \ \longrightarrow \ M\,;i{\to}M^{i} \qquad R \xmapsto{\ M\ }\!\!\!\twoheadrightarrow S, i \xmapsto{\ M^{i}\ }\!\!\!\twoheadrightarrow T, k$$

$$R \xmapsto{\ M\ }\!\!\!\twoheadrightarrow S, j$$

Types

$$M, N \ ::= \ \overbrace{x \ \mid \ [N].\,M \ \mid \ \langle x\rangle.\,M}^{\lambda\text{-calculus}} \ \mid \ \overbrace{i \ \mid \ M\,;i\!\to\! N \ \mid \ M^{i}}^{\text{control}}$$

Types: $\rho,\ \sigma,\ \tau \ ::= \ \overline{\sigma} \Rightarrow \overline{\tau}_I$

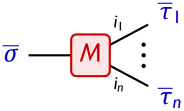
Stack types: $\overline{\tau} \ ::= \ \tau_1 \dots \tau_n$

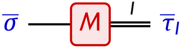
Control types: $\overline{\tau}_I \ ::= \ \overline{\tau}_1.i_1 + \ldots + \overline{\tau}_n.i_n \qquad I = \{i_1, \ldots, i_n\}$

Interpretation (without loops):

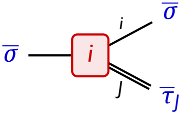
$$M : \overline{\sigma} \Rightarrow \overline{\tau}_I \quad \Longrightarrow \quad \forall S : \overline{\sigma}.\ \exists i \in I.\ \exists T : \overline{\tau}_i.\ \ S \xrightarrow{M} T, i$$

$$M\colon \overline{\sigma} \Rightarrow \overline{\tau}_1.i_1 + \ldots + \overline{\tau}_n.i_n$$



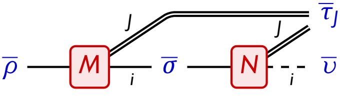


$$\frac{}{\Gamma \vdash i\colon \overline{\sigma} \Rightarrow \overline{\sigma}.i + \overline{\tau}_J}$$



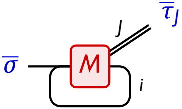
i

$$\frac{\Gamma \vdash M\colon \overline{\rho} \Rightarrow \overline{\tau}_J + \overline{\sigma}_i \qquad \Gamma \vdash N\colon \overline{\sigma} \Rightarrow \overline{\tau}_J\,(+\,\overline{\upsilon}.i)}{\Gamma \vdash M\,;i{\to}N\colon \overline{\rho} \Rightarrow \overline{\tau}_J\,(+\,\overline{\upsilon}.i)}$$



$M\,;i{\to}N$

$$\frac{\Gamma \vdash M\colon \overline{\sigma} \Rightarrow \overline{\tau}_J + \overline{\sigma}.i}{\Gamma \vdash M^i\colon \overline{\sigma} \Rightarrow \overline{\tau}_J}$$



M^i

Example: factorial

$$[1].\,\text{while True do}\left(\langle a\rangle.\,\langle x\rangle.\,\underbrace{\text{if}\ \overbrace{x \le 1}^{B}\ \text{then}\ \overbrace{\text{return}\ a}^{M}\ \text{else}\ \overbrace{a \times x\ ;\ x - 1}^{N}}_{L}\right)$$

Constants (operators via case splits):

$$\mathbb{B} \;=\; \varepsilon \Rightarrow \varepsilon.\top + \varepsilon.\bot \qquad \top,\ \bot : \mathbb{B} \qquad \le\ : \mathbb{N}\,\mathbb{N} \Rightarrow \mathbb{B}.\star$$

$$\mathbb{N} \;=\; \varepsilon \Rightarrow \varepsilon.0 + \varepsilon.1 + \varepsilon.2 + \ldots \qquad 0,\ 1,\ 2,\ \ldots : \mathbb{N} \qquad -\ ,\ \times\ : \mathbb{N}\,\mathbb{N} \Rightarrow \mathbb{N}.\star$$

Example: factorial

$$[1].\ \text{while True do}\ \Big(\langle a\rangle.\ \langle x\rangle.\ \underbrace{\text{if}\ \overbrace{x \leq 1}^{B}\ \text{then}\ \overbrace{\text{return}\ a}^{M}\ \text{else}\ \overbrace{a \times x\ ;\ x - 1}^{N}}_{L}\ \Big)$$

Building up the conditional:

$$\begin{aligned}
B &= x \leq 1 &&= [1].\,[x].\,\leq\ : \varepsilon \Rightarrow \mathbb{B}.\star \\
M &= \text{return}\ a &&= [a].\ \text{ret}\ : \varepsilon \Rightarrow \mathbb{N}.\mathit{ret} \\
N &= a \times x\ ;\ x - 1 &&= [x].\,[a].\,\times\ ;\ [1].\,[x].\,-\ : \varepsilon \Rightarrow (\mathbb{N}\,\mathbb{N}).\star
\end{aligned}$$

$$\begin{aligned}
&\langle x\rangle.\,x && : \mathbb{B} \Rightarrow\ \varepsilon.\top + \varepsilon.\bot \\
&B\ ;\ \langle x\rangle.\,x && : \varepsilon \Rightarrow\ \varepsilon.\top + \varepsilon.\bot \\
&B\ ;\ \langle x\rangle.\,x\ ;\ \top \to M && : \varepsilon \Rightarrow \mathbb{N}.\mathit{ret} + \varepsilon.\bot \\
L = \text{if}\ B\ \text{then}\ M\ \text{else}\ N = {}&B\ ;\ \langle x\rangle.\,x\ ;\ \top \to M\ ;\ \bot \to N && : \varepsilon \Rightarrow \mathbb{N}.\mathit{ret} + (\mathbb{N}\,\mathbb{N}).\star
\end{aligned}$$

Example: factorial

$$[1].\ \text{while True do}\ \Big(\langle a\rangle.\,\langle x\rangle.\ \underbrace{\text{if}\ \overbrace{x \le 1}^{B}\ \text{then}\ \overbrace{\text{return}\ a}^{M}\ \text{else}\ \overbrace{a \times x\ ;\ x - 1}^{N}}_{L}\ \Big)$$

Building up the loop:

$$\begin{array}{rcl}
L & : & \varepsilon \Rightarrow \mathbb{N}.\mathit{ret} + (\mathbb{N}\,\mathbb{N}).\star \\
\langle a\rangle.\,\langle x\rangle.\,L & : & \mathbb{N}\,\mathbb{N} \Rightarrow \mathbb{N}.\mathit{ret} + (\mathbb{N}\,\mathbb{N}).\star \\
(\langle a\rangle.\,\langle x\rangle.\,L)^{\star} & : & \mathbb{N}\,\mathbb{N} \Rightarrow \mathbb{N}.\mathit{ret} \\
\text{while True do}\ \langle a\rangle.\,\langle x\rangle.\,L \ = \ (\langle a\rangle.\,\langle x\rangle.\,L)^{\star}\ ;\mathrm{ret}{\to}\star & : & \mathbb{N}\,\mathbb{N} \Rightarrow \mathbb{N}.\star \\
[1].\,((\langle a\rangle.\,\langle x\rangle.\,L)^{\star}\ ;\mathrm{ret}{\to}\star) & : & \mathbb{N} \Rightarrow \mathbb{N}.\star
\end{array}$$

Typed exceptions

- **Classical continuations**:
  distinct from exceptions [Riecke & Thielecke 1999]
- **Explicit substitutions** (let/letrec): $M[x \to N]$ versus $M\,;\,i{\to}N$
  not coproducts but their continuation encoding

	Dynamic (not static)	Affine (not duplicating)	Fall-through	Single-case	Iteration
Case splits	×	×			
Let/letrec			×	×	×
Exceptions	×	×	×	×	

Typed exceptions

- **Classical continuations**:
  distinct from exceptions [Riecke & Thielecke 1999]
- **Explicit substitutions** (let/letrec): $M[x \to N]$ versus $M\,; i \to N$
  not coproducts but their continuation encoding

	Dynamic (not static)	Affine (not duplicating)	Fall-through	Single-case	Iteration
Case splits	×	×			
Let/letrec			×	×	×
Exceptions	×	×	×	×	
[Benton & Kennedy 2001]	×	×	×		
[Fiore & Staton 2014]		×	×	×	
[Maurer et al. 2017]		×	×	×	×
This work	×	×	×	×	×

Sequencing

- Imperative sequencing
- Strategies: CBV, computational metalanguage, CBPV

Control

- Exception handling
- Constants & data types (non-recursive)
- Iteration (loops)

Locations

- Mutable store
- Input/output
- Probabilistic/non-deterministic sampling

Locations

$$M, N \ ::= \ \underbrace{\overbrace{x \ \mid \ [N]a.\,M \ \mid \ a\langle x\rangle.\,M}^{\lambda\text{-calculus: } x \ \mid\ [N].\,M \ \mid\ \langle x\rangle.\,M}}_{\text{locations}} \ \mid \ \underbrace{i \ \mid \ M\,;i{\to}N \ \mid \ M^i}_{\text{control}}$$

Multiple stacks (and streams) named by **locations** $A = \{\lambda, a, b, c, \dots\}$

Evaluation is on a **memory** $S_A = S_\lambda \cdot S_a \cdot S_b \cdot S_c \dots$ (finitely many non-empty)

$$S_A \overset{M}{\longmapsto\!\!\!\!\rightarrow\!\!\!\!\rightarrow} T_A, i$$

Push $[N]a.\,M$, **pop** $a\langle x\rangle.\,M$ operate on a chosen stack a, with default stack λ

$$[N].\,M \ = \ [N]\lambda.\,M \qquad \langle x\rangle.\,M \ = \ \lambda\langle x\rangle.\,M$$

Reduction implements **independence** of different stacks

$$[N]a.\,a\langle x\rangle.\,M \ \longrightarrow \ \{N/x\}M$$

$$[N]b.\,a\langle x\rangle.\,M \ \longrightarrow \ a\langle x\rangle.\,[N]b.\,M \qquad (a \neq b, x \notin \mathsf{fv}(N))$$

Effects

$$M, N \ ::= \ \overbrace{x \mid [N]a.\,M \mid a\langle x\rangle.\,M}^{\text{locations}} \mid \overbrace{i \mid M\,;i{\to}N \mid M^i}^{\text{control}}$$

Input/output, **probabilities** as dedicated locations stdin, stdout, rnd

read: $\mathsf{stdin}\langle x\rangle.\,x$ print: $[N]\mathsf{stdout}.\,M$ random: $\mathsf{rnd}\langle x\rangle.\,M$

Store (mutable variables) as chosen locations a, b, c, . .

$$\begin{array}{lrcl} \text{update:} & a := N\,;\,M & = & a\langle \_ \rangle.\,[N]a.\,M \\ \text{lookup:} & !a & = & a\langle x\rangle.\,[x]a.\,x \qquad ({}^1) \end{array}$$

Confluence: beta-eta equivalence gives the algebraic theory for store[2]
Typed termination: Landin's Knot[3] cannot be typed

[1] Cf. Haskell MVars [Peyton Jones, Gordon & Finne 1996] [2] [Plotkin & Power 2002] [3] [Landin 1964]

Types

$$M, N \;::=\; \overbrace{x \;\mid\; [N]a.\,M \;\mid\; a\langle x\rangle.\,M}^{\text{locations}} \;\mid\; \overbrace{i \;\mid\; M\,;i\!\to\! N \;\mid\; M^{i}}^{\text{control}}$$

Types:	$\rho,\ \sigma,\ \tau$	$::=$	$\overline{\overline{\sigma}} \Rightarrow \overline{\overline{\tau}}_J$
Stack types:	$\overline{\tau}$	$::=$	$\tau_1 \ldots \tau_n$
Memory types:	$\overline{\overline{\tau}}$	$::=$	$a_1(\overline{\tau}_1)\ldots a_n(\overline{\tau}_n)$
Control types:	$\overline{\overline{\tau}}_I$	$::=$	$\overline{\overline{\tau}}_1.i_1 + \ldots + \overline{\overline{\tau}}_n.i_n \qquad I = \{i_1, \ldots, i_n\}$

Interpretation (without loops):

$$M : \overline{\overline{\sigma}} \Rightarrow \overline{\overline{\tau}}_I \quad\Longrightarrow\quad \forall S_A : \overline{\overline{\sigma}}.\ \ \exists i \in I.\ \ \exists T_A : \overline{\overline{\tau}}_i.\ \ S_A \xmapsto{\;M\;} T_A\,, i$$

The machine

$$M, N \;::=\; \overbrace{x \mid [N]a.\,M \mid a\langle x\rangle.\,M}^{\text{locations}} \mid \overbrace{i \mid M\,;i{\to}N \mid M^i}^{\text{control}}$$

Stacks: $S \;::=\; \varepsilon \mid S\,M$ 　　 Memories: $S_A \;::=\; S_a \cdot S_b \cdot S_c \ldots$

States: (S_A, M, K) 　　 Continuation stacks: $K \;::=\; \varepsilon \mid (j{\to}M)\,K$

Transitions:

$$\frac{(\;S_A \cdot S_a \;,\; [N]a.\,M \;,\; K\;)}{(\;S_A \cdot (S\,N)_a \;,\; M \;,\; K\;)} \qquad \frac{(\;S_A \;,\; M\,;i{\to}N \;,\; K\;)}{(\;S_A \;,\; M \;,\; (i{\to}N)\,K\;)}$$

$$\frac{(\;S_A \cdot (S\,N)_a \;,\; a\langle x\rangle.\,M \;,\; K\;)}{(\;S_A \cdot S_a \;,\; \{N/x\}M \;,\; K\;)} \qquad \frac{(\;S_A \;,\; i \;,\; (i{\to}N)\,K\;)}{(\;S_A \;,\; N \;,\; K\;)}$$

$$\frac{(\;S_A \;,\; M^i \;,\; K\;)}{(\;S_A \;,\; M \;,\; (i{\to}M^i)\,K\;)} \qquad \frac{(\;S_A \;,\; i \;,\; (j{\to}N)\,K\;)}{(\;S_A \;,\; i \;,\; K\;)}\;(i{\neq}j)$$

Reduction

$$M, N \;::=\; \overbrace{x \;\mid\; [N]a.\,M \;\mid\; a\langle x\rangle.\,M}^{\text{locations}} \;\mid\; \overbrace{i \;\mid\; M\,;i{\rightarrow}N \;\mid\; M^{i}}^{\text{control}}$$

$$
\begin{array}{rcll}
[N]a.\,a\langle x\rangle.\,M & \longrightarrow & \{N/x\}M & \\
[N]b.\,a\langle x\rangle.\,M & \longrightarrow & a\langle x\rangle.\,[N]b.\,M & (a \neq b, x \notin \mathrm{fv}(N)) \\
i\,;i{\rightarrow}P & \longrightarrow & P & \\
i\,;j{\rightarrow}P & \longrightarrow & i & (i \neq j) \\
(M\,;i{\rightarrow}N)\,;i{\rightarrow}P & \longrightarrow & M\,;i{\rightarrow}(N\,;i{\rightarrow}P) & \\
([N]a.\,M)\,;i{\rightarrow}P & \longrightarrow & [N]a.\,(M\,;i{\rightarrow}P) & \\
(a\langle x\rangle.\,M)\,;i{\rightarrow}P & \longrightarrow & a\langle x\rangle.\,(M\,;i{\rightarrow}P) & (x \notin \mathrm{fv}(P)) \\
M^{i} & \longrightarrow & M\,;i{\rightarrow}M^{i} &
\end{array}
$$

Overview

Technical results

- Confluence
- Typed machine termination and strong normalization (without loops)
- Embedding a complete (toy) imperative language

Design results

- Only six constructors
- Seamless integration of λ-calculus, sequencing, effects, control
- Intuitive abstract machine, using only stacks
- Semantics in sums, products, and function spaces

Future directions

- Algebraic effects & handlers (with Ohad Kammar and Nicolas Wu)
- Concurrency (with Ugo Dal Lago)

Thank you