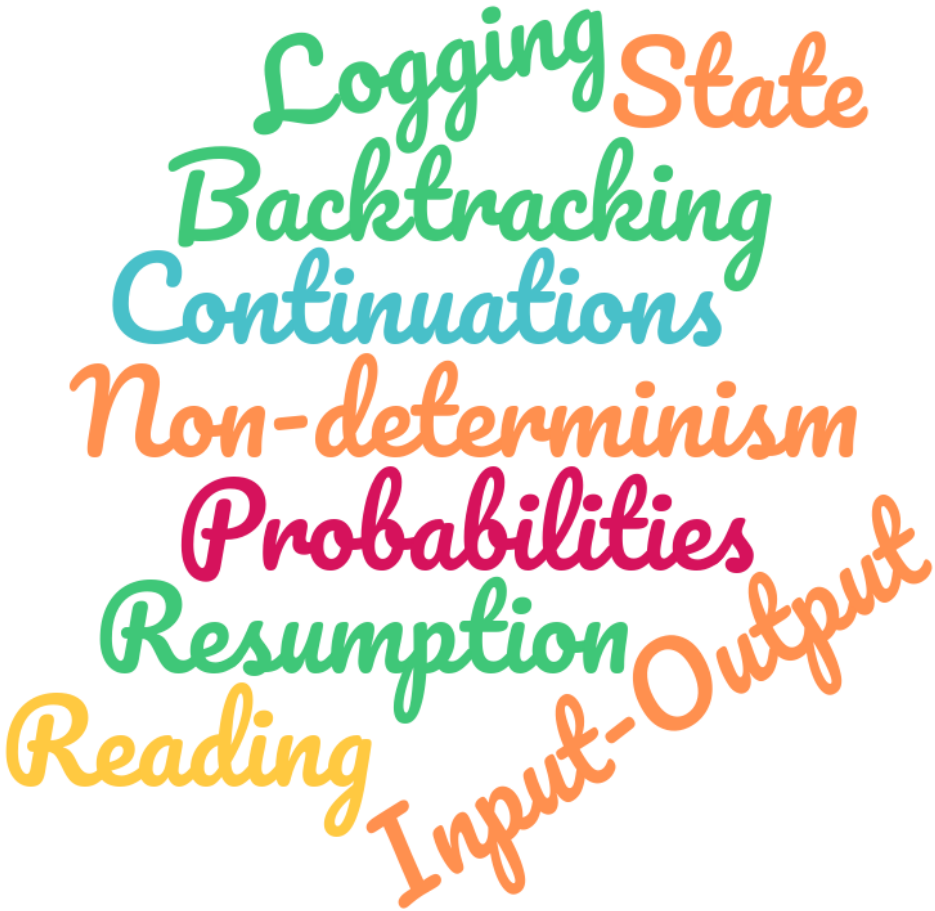
Designing Dijkstra Monads

real title: Dijkstra Monads For All

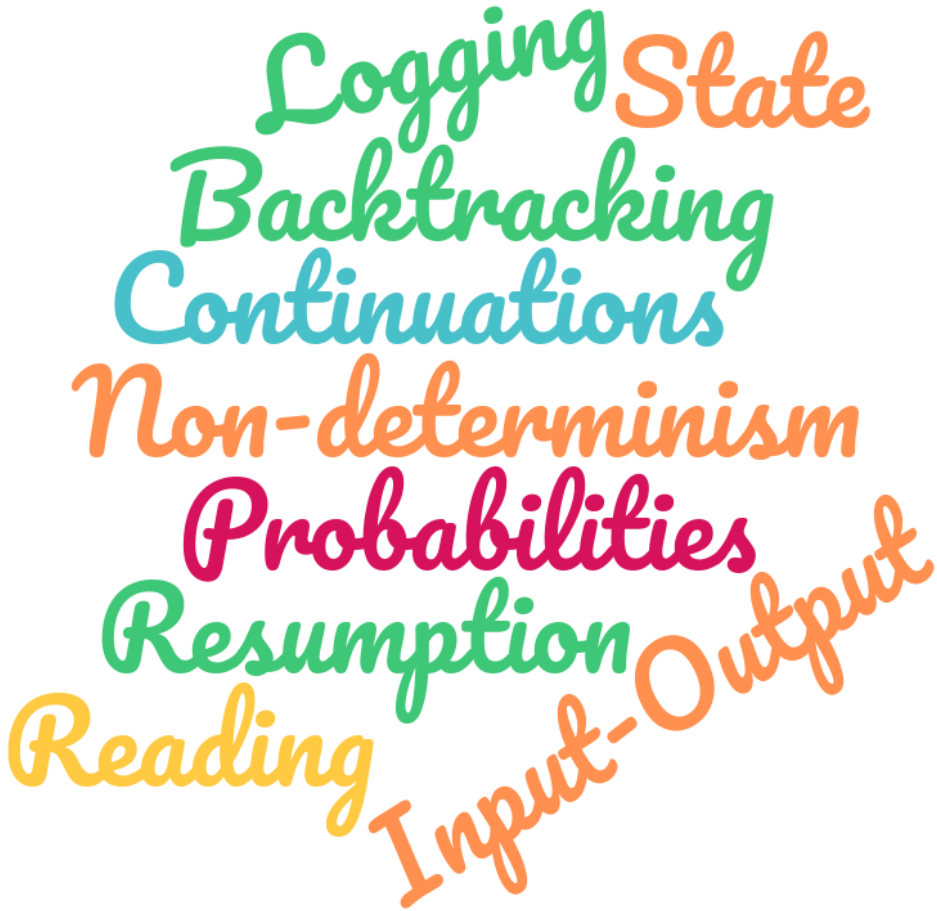
Kenji Maillard
jww D. Ahman, B. Atkey, G.Martinez,
C.Hriţcu, E.Tanter, E. Rivas

April 10, 2019

Programming with side effects



Programming with side effects



We need a model to uniformly account for side-effects !

Monads: a nifty algebraic tool

$\mathcal{M}X$ represents a computational context producing values in X

Monads: a nifty algebraic tool

$\mathcal{M}X$ represents a computational context producing values in X

A succint syntactic presentation: a type constructor $\mathcal{M}$

$\texttt{ret}^{\mathcal{M}} : X \to \mathcal{M}X \qquad \texttt{bind}^{\mathcal{M}} : \mathcal{M}X \to (X \to \mathcal{M}Y) \to \mathcal{M}Y$

$+$ 3 equations (monad laws)

Examples of computational monads

State

$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Examples of computational monads

State

$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Exceptions

$\mathrm{Exc}\, X = X + \mathcal{E}$

Examples of computational monads

State
$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Exceptions
$\mathrm{Exc}\, X = X + \mathcal{E}$

Partiality
$\mathrm{Div}\, X = X + \{\Omega\}$

Examples of computational monads

State
$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Exceptions
$\mathrm{Exc}\, X = X + \mathcal{E}$

Partiality
$\mathrm{Div}\, X = X + \{\Omega\}$

Input-Output
$\mathrm{IO}\, X = \mu Z.X + (Z \times \mathcal{O}) + (\mathcal{I} \to Z)$

Examples of computational monads

State
$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Exceptions
$\mathrm{Exc}\, X = X + \mathcal{E}$

Partiality
$\mathrm{Div}\, X = X + \{\Omega\}$

Input-Output
$\mathrm{IO}\, X = \mu Z.X + (Z \times \mathcal{O}) + (\mathcal{I} \to Z)$

Continuations
$\mathrm{Cont}_{\mathrm{Ans}}\, X = (X \to \mathrm{Ans}) \to \mathrm{Ans}$

Examples of computational monads

State
$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Exceptions
$\mathrm{Exc}\, X = X + \mathcal{E}$

Partiality
$\mathrm{Div}\, X = X + \{\Omega\}$

Input-Output
$\mathrm{IO}\, X = \mu Z.X + (Z \times \mathcal{O}) + (\mathcal{I} \to Z)$

Continuations
$\mathrm{Cont}_{\mathrm{Ans}}\, X = (X \to \mathrm{Ans}) \to \mathrm{Ans}$

Non-determinism
$\mathrm{NDet}\, X = \mathcal{P}_{\mathit{fin}}(X)$

Examples of computational monads

State
$\mathrm{St}\, X = \mathcal{S} \to X \times \mathcal{S}$

Exceptions
$\mathrm{Exc}\, X = X + \mathcal{E}$

Partiality
$\mathrm{Div}\, X = X + \{\Omega\}$

Input-Output
$\mathrm{IO}\, X = \mu Z.X + (Z \times \mathcal{O}) + (\mathcal{I} \to Z)$

Continuations
$\mathrm{Cont}_{\mathrm{Ans}}\, X = (X \to \mathrm{Ans}) \to \mathrm{Ans}$

Non-determinism
$\mathrm{NDet}\, X = \mathcal{P}_{fin}(X)$

Probabilities
$\mathrm{Prob}\, X = \{\ f : X \to [0;1] \ \mid\ \mathrm{supp}(f)\ \text{finite}, \Sigma_{x \in \mathrm{supp}(f)} f\, x \leq 1\ \}$

Monadic programming

Moggi's computational lambda-calculus (λ_C)(Moggi, 1989)

$$
\begin{array}{lll}
T ::= \mathcal{M}\ T \mid T_1 \to T_2 \mid T_1 \times T_2 \mid T_1 + T_2 & & \text{types} \\
v ::= x \mid \lambda x.\, c \mid \langle c_1, c_2 \rangle \mid \iota_i\ v & & \text{values} \\
c ::= v & & \text{returning values} \\
\quad \mid \texttt{let}\ x = c_1\ \texttt{in}\ c_2 & & \text{sequencing computations} \\
\quad \mid \mathrm{op}(c_1, \ldots, c_n) & & \text{effectful operations} \\
\quad \mid c_1\ c_2 \mid \pi_i\ c \mid \texttt{case}\ c\ x.c_1\ y.c_2 & &
\end{array}
$$

Monadic programming

Moggi's computational lambda-calculus (λ_C)(Moggi, 1989)

$$
\begin{array}{lll}
T ::= \mathcal{M}\ T \mid T_1 \to T_2 \mid T_1 \times T_2 \mid T_1 + T_2 & & \text{types} \\
v ::= x \mid \lambda x.\, c \mid \langle c_1, c_2 \rangle \mid \iota_i\ v & & \text{values} \\
c ::= v & & \text{returning values} \\
\quad \mid \texttt{let}\ x = c_1\ \texttt{in}\ c_2 & & \text{sequencing computations} \\
\quad \mid \mathrm{op}(c_1, \ldots, c_n) & & \text{effectful operations} \\
\quad \mid c_1\ c_2 \mid \pi_i\ c \mid \texttt{case}\ c\ x.c_1\ y.c_2 & &
\end{array}
$$

RET

$$\frac{\Gamma \vdash_{val} v : T}{\Gamma \vdash v : \mathcal{M}\ T}$$

BIND

$$\frac{\Gamma \vdash c_1 : \mathcal{M}\ T_1 \qquad \Gamma, x : T_1 \vdash c_2 : \mathcal{M}\ T_2}{\Gamma \vdash \texttt{let}\ x = c_1\ \texttt{in}\ c_2 : \mathcal{M}\ T_2}$$

Operations associated to monads

a.k.a. **generic effects** (Plotkin and Power, 2003)

State	Input-Output
$\mathtt{get} : \mathbb{1} \to \mathrm{St}\,\mathcal{S}$	$\mathtt{read} : \mathbb{1} \to \mathrm{IO}\,\mathcal{I}$
$\mathtt{put} : \mathcal{S} \to \mathrm{St}\,\mathbb{1}$	$\mathtt{write} : \mathcal{O} \to \mathrm{IO}\,\mathbb{1}$

Operations associated to monads

a.k.a. **generic effects** (Plotkin and Power, 2003)

State

$\texttt{get} : \mathbb{1} \to \mathrm{St}\,\mathcal{S}$

$\texttt{put} : \mathcal{S} \to \mathrm{St}\,\mathbb{1}$

Input-Output

$\texttt{read} : \mathbb{1} \to \mathrm{IO}\,\mathcal{I}$

$\texttt{write} : \mathcal{O} \to \mathrm{IO}\,\mathbb{1}$

Exceptions

$\texttt{throw} : \mathcal{E} \to \mathrm{Exc}\,\mathbb{0}$

Partiality

$\texttt{div} : \mathbb{1} \to \mathrm{Div}\,\mathbb{0}$

Operations associated to monads

a.k.a. **generic effects** (Plotkin and Power, 2003)

State

$\texttt{get} : \mathbb{1} \to \mathrm{St}\,\mathcal{S}$

$\texttt{put} : \mathcal{S} \to \mathrm{St}\,\mathbb{1}$

Input-Output

$\texttt{read} : \mathbb{1} \to \mathrm{IO}\,\mathcal{I}$

$\texttt{write} : \mathcal{O} \to \mathrm{IO}\,\mathbb{1}$

Exceptions

$\texttt{throw} : \mathcal{E} \to \mathrm{Exc}\,\mathbb{0}$

Partiality

$\texttt{div} : \mathbb{1} \to \mathrm{Div}\,\mathbb{0}$

Non-determinism

$\texttt{choose} : \mathbb{1} \to \mathrm{NDet}\,\mathbb{B}$

$\texttt{fail} : \mathbb{1} \to \mathrm{NDet}\,\mathbb{0}$

Probabilities

$\texttt{flip} : [0;1] \to \mathrm{Prob}\,\mathbb{B}$

Starting point

Fix $\mathcal{M}$ a computational monad $\in$

Consider the following monadic program

$$\texttt{let } x = \texttt{read}\,() \texttt{ in}$$
$$\texttt{let } y = \texttt{read}\,() \texttt{ in}$$
$$\texttt{if } y = 0 \texttt{ then } \texttt{throw}\,\texttt{Div\_by\_zero} \texttt{ else } x/y \quad : \quad \mathcal{M}\mathbb{N}$$

Starting point

Fix $\mathcal{M}$ a computational monad $\in$

Consider the following monadic program

$$\texttt{let } x = \texttt{read}\,() \texttt{ in}$$
$$\texttt{let } y = \texttt{read}\,() \texttt{ in}$$
$$\texttt{if } y = 0 \texttt{ then } \texttt{throwDiv\_by\_zero} \texttt{ else } x/y \quad : \quad \mathcal{M}\mathbb{N}$$

How can we specify and verify such monadic programs ?

Hoare logic Primer

$$\{\ \mathit{pre}\ \}\ \mathit{code}\ \{\ \mathit{post}\ \}$$

$$\{\ \top\ \}\ v\ \{\ x = v\ \}$$

$$\frac{\{\ P\ \}\ c_1\ \{\ Q\ \} \qquad \forall x, \{\ Q(x)\ \}\ c_2\ \{\ R\ \}}{\{\ P\ \}\ \texttt{let}\ x = c_1\ \texttt{in}\ c_2\ \{\ R\ \}}$$

Hoare logic Primer

$$\{\, pre \,\}\ code\ \{\, post \,\}$$

$$\{\, \top \,\}\ v\ \{\, x = v \,\}$$

$$\frac{\{\, P \,\}\ c_1\ \{\, Q \,\} \qquad \forall x, \{\, Q(x) \,\}\ c_2\ \{\, R \,\}}{\{\, P \,\}\ \texttt{let}\ x = c_1\ \texttt{in}\ c_2\ \{\, R \,\}}$$

Pure:	$pre : \mathbb{P}$	$post : X \to \mathbb{P}$
Stateful:	$pre : S \to \mathbb{P}$	$post : X \times S \to \mathbb{P}$
With exceptions:	$pre : \mathbb{P}$	$post : X + E \to \mathbb{P}$

Weakest precondition calculi

Dijkstra's insight: there is a **weakest** precondition that can be computed inductively from a program and a postcondition

$$\vdash \{\, P \,\}\ c\ \{\, Q \,\} \qquad \Longleftrightarrow \qquad \vdash P \Rightarrow \mathtt{wp}[c](Q)$$

$$\mathtt{wp}[v](Q) = Q(v) \qquad \mathtt{wp}[\mathtt{let}\, x = c_1\, \mathtt{in}\, c_2](Q) = \mathtt{wp}[c_1](\lambda x.\, \mathtt{wp}[c_2](Q))$$

Weakest precondition calculi

Dijkstra's insight: there is a **weakest** precondition that can be computed inductively from a program and a postcondition

$$\vdash \{\, P \,\}\ c\ \{\, Q \,\} \qquad \Longleftrightarrow \qquad \vdash P \Rightarrow \mathtt{wp}[c](Q)$$

$$\mathtt{wp}[v](Q) = Q(v) \qquad \mathtt{wp}[\mathtt{let}\, x = c_1\, \mathtt{in}\, c_2](Q) = \mathtt{wp}[c_1](\lambda x.\, \mathtt{wp}[c_2](Q))$$

Pure:	$\mathtt{wp}[c](\cdot) : (X \to \mathbb{P}) \to \mathbb{P}$
Stateful:	$\mathtt{wp}[c](\cdot) : (X \times S \to \mathbb{P}) \to S \to \mathbb{P}$
With exceptions:	$\mathtt{wp}[c](\cdot) : (X + E \to \mathbb{P}) \to \mathbb{P}$

Wait... That's a monad !

Pure: $\mathrm{W}^{\mathrm{Id}} X = \mathrm{Cont}_{\mathbb{P}}(X) = (X \to \mathbb{P}) \to \mathbb{P}$

Continuation monad with answer type $\mathbb{P}$.

Wait. . . That's a monad !

Pure: $\quad \mathrm{W}^{\mathrm{Id}} X = \mathrm{Cont}_{\mathbb{P}}(X) = (X \to \mathbb{P}) \to \mathbb{P}$

Continuation monad with answer type $\mathbb{P}$.

$$\mathtt{ret}^{\mathrm{W}^{\mathrm{Id}}} : X \to \mathrm{W}^{\mathrm{Id}} X \qquad \mathtt{bind}^{\mathrm{W}^{\mathrm{Id}}} : \mathrm{W}^{\mathrm{Id}} X \to (X \to \mathrm{W}^{\mathrm{Id}} Y) \to \mathrm{W}^{\mathrm{Id}} Y$$

$$\mathtt{ret}^{\mathrm{W}^{\mathrm{Id}}}\ x\ Q = Q(x) \qquad \mathtt{bind}^{\mathrm{W}^{\mathrm{Id}}}\ w_1\ w_2\ Q = w_1(\lambda x.\, w_2(x)(Q))$$

Wait... That's a monad !

Pure: $\quad \mathrm{W}^{\mathrm{Id}} X = \mathrm{Cont}_{\mathbb{P}}(X) = (X \to \mathbb{P}) \to \mathbb{P}$

Continuation monad with answer type $\mathbb{P}$.

$$\mathtt{ret}^{\mathrm{W}^{\mathrm{Id}}} : X \to \mathrm{W}^{\mathrm{Id}} X \qquad \mathtt{bind}^{\mathrm{W}^{\mathrm{Id}}} : \mathrm{W}^{\mathrm{Id}} X \to (X \to \mathrm{W}^{\mathrm{Id}} Y) \to \mathrm{W}^{\mathrm{Id}} Y$$

$$\mathtt{ret}^{\mathrm{W}^{\mathrm{Id}}}\ x\ Q = Q(x) \qquad \mathtt{bind}^{\mathrm{W}^{\mathrm{Id}}}\ w_1\ w_2\ Q = w_1(\lambda x.\, w_2(x)(Q))$$

Also monads:

Stateful: $\quad \mathrm{W}^{\mathrm{St}}\, X = (X \times S \to \mathbb{P}) \to S \to \mathbb{P}$

With exceptions: $\quad \mathrm{W}^{\mathrm{Exc}}\, X = (X + E \to \mathbb{P}) \to \mathbb{P}$

Specification monads

Key idea: **specifications** can also be uniformly captured by monads!

Weakest precondition:	$\mathrm{Cont}_{\mathbb{P}}\, X = (X \to \mathbb{P}) \to \mathbb{P}$
Strongest postcondition:	$\mathrm{StrPost}\, X = \mathbb{P} \to X \to \mathbb{P}$
Pre/Postconditions:	$\mathrm{PrePost}\, X = \mathbb{P} \times (X \to \mathbb{P})$

Specification monads

Key idea: **specifications** can also be uniformly captured by monads!

Weakest precondition:	$\mathrm{Cont}_{\mathbb{P}}\, X = (X \to \mathbb{P}) \to \mathbb{P}$
Strongest postcondition:	$\mathrm{StrPost}\, X = \mathbb{P} \to X \to \mathbb{P}$
Pre/Postconditions:	$\mathrm{PrePost}\, X = \mathbb{P} \times (X \to \mathbb{P})$

What's in a specification monad W ?

- ▷ specifications can be compared by strength

$$w_1 \leq^{\mathrm{Cont}_{\mathbb{P}}\, X} w_2 \quad \Leftrightarrow \quad \forall p : X \to \mathbb{P}, w_1\, p \implies w_2\, p$$

- ▷ $\texttt{bind}^{\mathrm{W}}$ is monotonic in both its arguments

$\rightsquigarrow$ restriction to monotonic elements in $\mathrm{Cont}_{\mathbb{P}}, \mathrm{StrPost}$

Predicate transformers from monad transformers

Examples of **predicate transformer** monads:

Pure:	$\mathrm{W}^{\mathrm{Id}} : (X \to \mathbb{P}) \to \mathbb{P}$
Stateful:	$\mathrm{W}^{\mathrm{St}} : (X \times \mathcal{S} \to \mathbb{P}) \to \mathcal{S} \to \mathbb{P}$
With exceptions:	$\mathrm{W}^{\mathrm{Exc}} : (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$

Predicate transformers from monad transformers

Examples of **predicate transformer** monads:

Pure: $\mathrm{W}^{\mathrm{Id}} : (X \to \mathbb{P}) \to \mathbb{P}$

Stateful: $\mathrm{W}^{\mathrm{St}} : (X \times \mathcal{S} \to \mathbb{P}) \to \mathcal{S} \to \mathbb{P}$

With exceptions: $\mathrm{W}^{\mathrm{Exc}} : (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$

$$\mathrm{W}^{\mathrm{Id}} = \mathcal{T}^{\mathrm{Id}}(\mathrm{Cont}_{\mathbb{P}}) \qquad \mathcal{T}^{\mathrm{Id}}(\mathcal{M}) = \mathcal{M}$$

$$\mathrm{W}^{\mathrm{St}} = \mathcal{T}^{\mathrm{St}}(\mathrm{Cont}_{\mathbb{P}}) \qquad \mathcal{T}^{\mathrm{St}}(\mathcal{M}) = \mathcal{S} \to \mathcal{M}(- \times \mathcal{S})$$

$$\mathrm{W}^{\mathrm{Exc}} = \mathcal{T}^{\mathrm{Exc}}(\mathrm{Cont}_{\mathbb{P}}) \qquad \mathcal{T}^{\mathrm{Exc}}(\mathcal{M}) = \mathcal{M}(- + \mathcal{E})$$

Predicate transformers from monad transformers

Examples of **predicate transformer** monads:

Pure: $\mathrm{W}^{\mathrm{Id}} : (X \to \mathbb{P}) \to \mathbb{P}$

Stateful: $\mathrm{W}^{\mathrm{St}} : (X \times \mathcal{S} \to \mathbb{P}) \to \mathcal{S} \to \mathbb{P}$

With exceptions: $\mathrm{W}^{\mathrm{Exc}} : (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$

$$\mathrm{W}^{\mathrm{Id}} = \mathcal{T}^{\mathrm{Id}}(\mathrm{Cont}_{\mathbb{P}}) \qquad \mathcal{T}^{\mathrm{Id}}(\mathcal{M}) = \mathcal{M}$$

$$\mathrm{W}^{\mathrm{St}} = \mathcal{T}^{\mathrm{St}}(\mathrm{Cont}_{\mathbb{P}}) \qquad \mathcal{T}^{\mathrm{St}}(\mathcal{M}) = \mathcal{S} \to \mathcal{M}(- \times \mathcal{S})$$

$$\mathrm{W}^{\mathrm{Exc}} = \mathcal{T}^{\mathrm{Exc}}(\mathrm{Cont}_{\mathbb{P}}) \qquad \mathcal{T}^{\mathrm{Exc}}(\mathcal{M}) = \mathcal{M}(- + \mathcal{E})$$

Monad transformer $\mathcal{T}$: $\begin{cases} \text{map a monad } \mathcal{M} \text{ to a monad } \mathcal{T}\mathcal{M} \\ \texttt{lift}^{\mathcal{T}} : \mathcal{M} \to \mathcal{T}\mathcal{M} \end{cases}$

Specifying programs with exceptions

$$\theta^{\mathrm{Exc}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Exc}}\, X = (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Exc}}(m) = \lambda p.\, p\, m$$

Specifying programs with exceptions

$$\theta^{\mathrm{Exc}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Exc}}\, X = (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Exc}}(m) = \lambda p.\, p\, m$$

$$\theta^{\mathrm{Exc}}(\mathtt{ret}^{\mathrm{Exc}}\ v) = \mathtt{ret}^{\mathrm{W}^{\mathrm{Exc}}}\ v$$

$$\theta^{\mathrm{Exc}}(\mathtt{bind}^{\mathrm{Exc}}\ m\ f) = \mathtt{bind}^{\mathrm{W}^{\mathrm{Exc}}}\ (\theta^{\mathrm{Exc}}\ m)\ (\theta^{\mathrm{Exc}} \circ f)$$

Specifying programs with exceptions

$$\theta^{\mathrm{Exc}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Exc}}\, X = (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Exc}}(m) = \lambda p.\, p\, m$$

$$\theta^{\mathrm{Tot}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Id}}\, X = (X \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Tot}}(\mathtt{inr}\, v) = \lambda p.\, p\, v \qquad \theta^{\mathrm{Tot}}(\mathtt{inl}\, e) = \lambda p.\, \bot$$

Specifying programs with exceptions

$$\theta^{\mathrm{Exc}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Exc}}\, X = (X + \mathcal{E} \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Exc}}(m) = \lambda p.\, p\, m$$

$$\theta^{\mathrm{Tot}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Id}}\, X = (X \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Tot}}(\mathtt{inr}\, v) = \lambda p.\, p\, v \qquad \theta^{\mathrm{Tot}}(\mathtt{inl}\, e) = \lambda p.\, \bot$$

$$\theta^{\mathrm{Part}} \quad : \quad \mathrm{Exc}\, X \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Id}}\, X = (X \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\mathrm{Part}}(\mathtt{inr}\, v) = \lambda p.\, p\, v \qquad \theta^{\mathrm{Part}}(\mathtt{inl}\, e) = \lambda p.\, \top$$

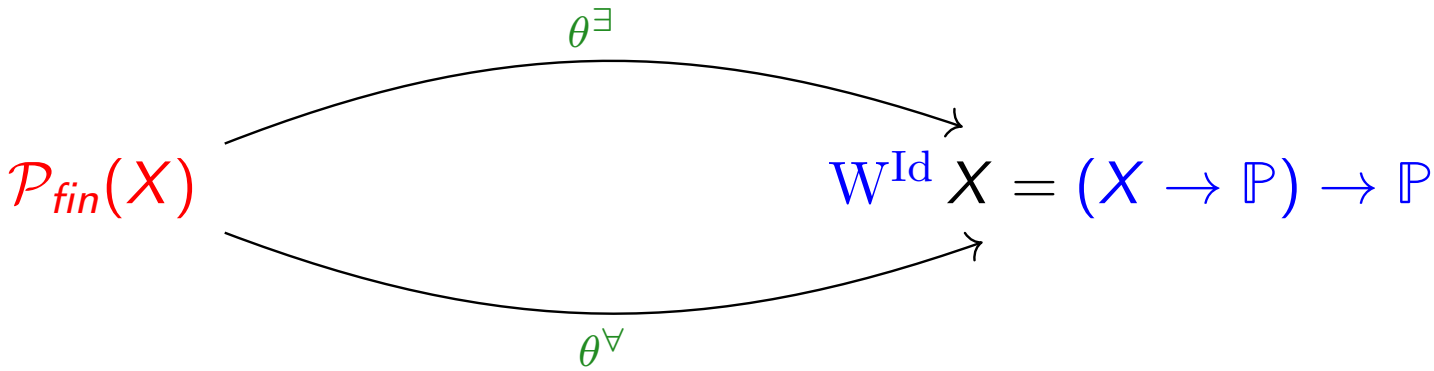
Effect observation

An effect observation θ (Katsumata, 2014)

is a **monad morphism**, that is

$$\theta(\mathtt{ret}^{\mathcal{M}}\, v) = \mathtt{ret}^{\mathrm{W}}\, v \qquad \theta(\mathtt{bind}^{\mathcal{M}}\, m\, f) = \mathtt{bind}^{\mathrm{W}}\ (\theta\, m)\ (\theta \circ f)$$

Verifying non-deterministic programs

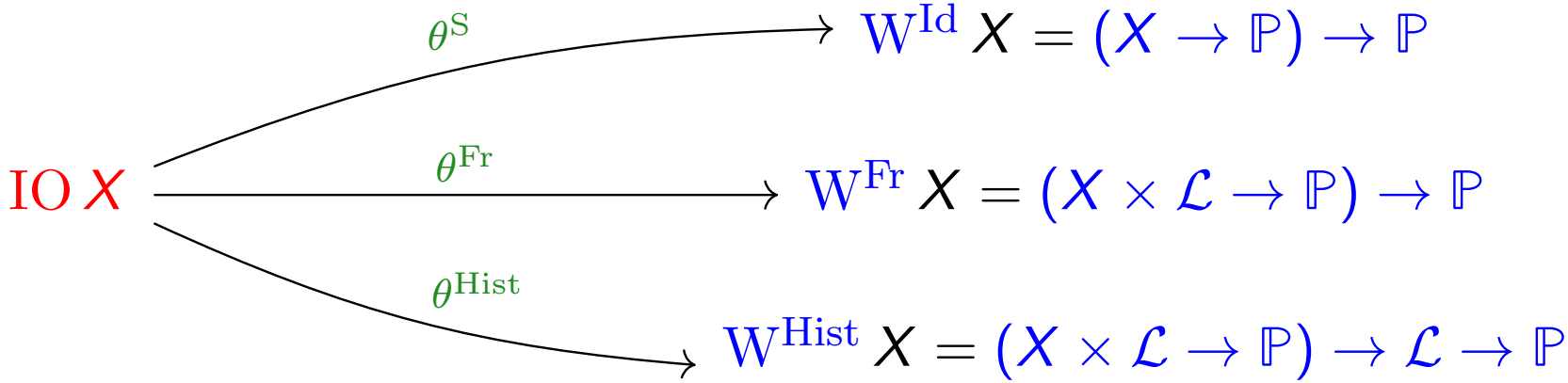


$$\theta^{\exists}(\{v_1, \ldots, v_n\}) \quad = \quad \lambda p.\, p\, v_1 \vee \ldots \vee p\, v_n$$
$$\theta^{\forall}(\{v_1, \ldots, v_n\}) \quad = \quad \lambda p.\, p\, v_1 \wedge \ldots \wedge p\, v_n$$

Angelic non-determinism $\theta^{\exists}$ vs **Demonic** non-determinism $\theta^{\forall}$

Input-Output in context

$$\mathcal{L} = (\mathcal{I} + \mathcal{O})^*$$



$$\theta^{\mathrm{S}}(\mathtt{write}\, o) = \lambda p.\, p\, ()$$

$$\theta^{\mathrm{Fr}}(\mathtt{write}\, o) = \lambda p.\, p\, \langle (),\ [o] \rangle$$

$$\theta^{\mathrm{Hist}}(\mathtt{write}\, o) = \lambda p\, \mathit{log}.\, p\, \langle (),\ (o :: \mathit{log}) \rangle$$

A second bridge between computations and specifications

$\mathcal{M}$

computational monad

W

predicate transformer monad

code

$c : \mathcal{M}\ \mathbb{N}$

specification

$w_c : \mathrm{W}\ \mathbb{N}$

A second bridge between computations and specifications

$\mathcal{M}$
computational monad

W
predicate transformer monad

code
$c : \mathcal{M}\ \mathbb{N}$

specification
$w_c : \mathrm{W}\ \mathbb{N}$

Dijkstra monad
$c : \mathcal{D}^{\mathcal{M}}\ \mathbb{N}\ w_c$

A second bridge between computations and specifications

$\mathcal{M}$
computational monad

W
predicate transformer monad

code
$c : \mathcal{M}\ \mathbb{N}$

specification
$w_c : \mathrm{W}\ \mathbb{N}$

Dijkstra monad
$c : \mathcal{D}^{\mathcal{M}}\ \mathbb{N}\ w_c$

$\mathtt{ret}^{\mathcal{D}^{\mathcal{M}}} : (x : A) \to \mathcal{D}^{\mathcal{M}}\ A\ (\mathtt{ret}^{\mathrm{W}}\ x)$

$$\frac{m : \mathcal{D}^{\mathcal{M}}\ A\ w_1 \qquad f : (x : A) \to \mathcal{D}^{\mathcal{M}}\ B\ w_2(x)}{\mathtt{bind}^{\mathcal{D}^{\mathcal{M}}}\ m\ f : \mathcal{D}^{\mathcal{M}}\ B\ (\mathtt{bind}^{\mathrm{W}}\ w_1\ w_2)}$$

Monadic program verification in F*

Dijkstra monads are heavily used in F*:

- ▷ Dependently-typed, programs and specifications in the same language

Monadic program verification in F*

Dijkstra monads are heavily used in F*:

- ▷ Dependently-typed, programs and specifications in the same language
- ▷ Multiple primitive effects (from **C**, **OCaml**)

$$\text{Pure}\ A\ (w : W^{\text{Id}}\ A) \qquad \text{Div}\ A\ (w : W^{\text{Id}}\ A) \qquad \text{State}\ A\ (w : W^{\text{St}}\ A)$$

$$\text{Exc}\ A\ (w : W^{\text{Exc}}\ A) \qquad \text{All}\ A\ (w : W^{\text{StExc}}\ A)$$

Monadic program verification in F*

Dijkstra monads are heavily used in F*:

- ▷ Dependently-typed, programs and specifications in the same language
- ▷ Multiple primitive effects (from **C**, **OCaml**)

$$\text{Pure}\ A\ (w : W^{\text{Id}}\ A) \qquad \text{Div}\ A\ (w : W^{\text{Id}}\ A) \qquad \text{State}\ A\ (w : W^{\text{St}}\ A)$$

$$\text{Exc}\ A\ (w : W^{\text{Exc}}\ A) \qquad \text{All}\ A\ (w : W^{\text{StExc}}\ A)$$

```
let rec fib (n:ℕ)
  : PURE ℕ (λ p → ∀ r. r ≥ n ∧ r > 0 ⟹ p r)
  =
  if n ≤ 1 then 1 else fib (n-1) + fib (n-2)
```

Monadic program verification in F*

Dijkstra monads are heavily used in F*:

- ▷ Dependently-typed, programs and specifications in the same language
- ▷ Multiple primitive effects (from **C**, **OCaml**)

$$\text{Pure}\ A\ (w : W^{\text{Id}}\ A) \qquad \text{Div}\ A\ (w : W^{\text{Id}}\ A) \qquad \text{State}\ A\ (w : W^{\text{St}}\ A)$$

$$\text{Exc}\ A\ (w : W^{\text{Exc}}\ A) \qquad \text{All}\ A\ (w : W^{\text{StExc}}\ A)$$

```
let rec fib (n:ℕ)
  : Pure ℕ (requires T)
    (ensures (λ r → r ≥ n ∧ r > 0))
  =
  if n ≤ 1 then 1 else fib (n-1) + fib (n-2)
```

Dijkstra monads in F*

- Region and stack-based low-level model for extraction to **C**, used for implementing cryptographic primitives and protocols: **HACl***, **MiTLS**

Dijkstra monads in F*

- Region and stack-based low-level model for extraction to **C**, used for implementing cryptographic primitives and protocols: **HACl***, **MiTLS**

- Dijkstra Monads For Free (DM4Free) (Ahman et al., 2017)
  User-defined effects: monads in a DSL elaborate to
  - a specification monad W
  - a Dijkstra monad $\mathcal{D}$ indexed by W

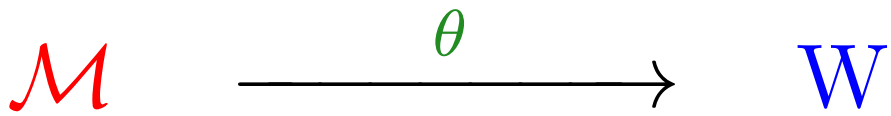
Dijkstra monads in F*

- Region and stack-based low-level model for extraction to **C**, used for implementing cryptographic primitives and protocols: **HACl***, **MiTLS**

- Dijkstra Monads For Free (DM4Free) (Ahman et al., 2017)
  User-defined effects: monads in a DSL elaborate to
  - a specification monad W
  - a Dijkstra monad $\mathcal{D}$ indexed by W

  $\rightsquigarrow$ State, exceptions, State+Exceptions,...
  $\rightsquigarrow$ No Input-output, non-determinism, probabilities...

From effect observation to Dijkstra monad



From effect observation to Dijkstra monad

$$\mathcal{M} \xrightarrow{\quad\theta\quad} \mathrm{W}$$

$$\mathcal{D}^{\mathcal{M}}\, A\, (w : \mathrm{W}\ A) \quad = \quad \{ m : \mathcal{M}\ A \mid w \leq^{\mathrm{W}} \theta(m) \}$$

A Dijkstra monad for demonic non-determinism

Recall that there is an effect observation

$$\theta^{\forall} \quad : \quad \mathrm{NDet}\, A \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Id}}\, A = (A \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\forall}(m) = \lambda p.\, \forall v \in m, p\, v$$

A Dijkstra monad for demonic non-determinism

Recall that there is an effect observation

$$\theta^{\forall} \quad : \quad \mathrm{NDet}\, A \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Id}}\, A = (A \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\forall}(m) = \lambda p.\, \forall v \in m, p\, v$$

We obtain a Dijkstra monad

$$\mathrm{ND} : (A : \mathrm{Type}) \to (w : \mathrm{W}^{\mathrm{Id}}\, A) \to \mathrm{Type}$$
$$\texttt{choose} : (u : \mathbb{1}) \to \mathrm{ND}\ \mathbb{B}\ (\lambda p.\, p\ \texttt{true} \wedge p\ \texttt{false})$$
$$\texttt{fail} : (u : \mathbb{1}) \to \mathrm{ND}\ \mathbb{B}\ (\lambda p.\, \top)$$

A Dijkstra monad for demonic non-determinism

Recall that there is an effect observation

$$\theta^{\forall} \quad : \quad \mathrm{NDet}\, A \quad \longrightarrow \quad \mathrm{W}^{\mathrm{Id}}\, A = (A \to \mathbb{P}) \to \mathbb{P}$$

$$\theta^{\forall}(m) = \lambda p.\, \forall v \in m, p\, v$$

We obtain a Dijkstra monad

$$\mathrm{ND} : (A : \mathrm{Type}) \to (w : \mathrm{W}^{\mathrm{Id}}\, A) \to \mathrm{Type}$$
$$\texttt{choose} : (u : \mathbb{1}) \to \mathrm{ND}\ \mathbb{B}\ (\lambda p.\, p\ \texttt{true} \wedge p\ \texttt{false})$$
$$\texttt{fail} : (u : \mathbb{1}) \to \mathrm{ND}\ \mathbb{B}\ (\lambda p.\, \top)$$

```
let rec pick #a (l : list a)
  : ND a (λ p → ∀ x. List.memP x l ⟹ p x)
  =
  match l with
  | [] → fail ()
  | x::xs → if choose true false then x else pick xs
```

Computing pythagorean triples

```
let guard (b:bool)
  : ND unit (λ p → b ⟹ p ())
  =
  if b then () else fail ()

let pyths ()
  : ND (ℤ & ℤ & ℤ)
       (λ p → ∀ x y z. x×x + y×y == z×z ⟹ p (x,y,z))
  =
  let l = [1;2;3;4;5;6;7;8;9;10] in
  let x = pick l in
  let y = pick l in
  let z = pick l in
  guard (x×x + y×y = z×z);
  (x,y,z)
```

Synthesis

- ▷ Start with a **computational monad** $\mathcal{M}$
- ▷ Select a **specification monad** W
- ▷ Define an **effect observation** $\theta : \mathcal{M} \to \mathrm{W}$
- ▷ Obtain a **Dijkstra monad** $\mathcal{D}^{\mathcal{M}}$ indexed by W

$\rightsquigarrow$ Convenient way to verify code in $\mathcal{M}$

Synthesis

- ▷ Start with a **computational monad** $\mathcal{M}$
- ▷ Select a **specification monad** W
- ▷ Define an **effect observation** $\theta : \mathcal{M} \to \mathrm{W}$
- ▷ Obtain a **Dijkstra monad** $\mathcal{D}^{\mathcal{M}}$ indexed by W

$\rightsquigarrow$ Convenient way to verify code in $\mathcal{M}$

Further questions:

- ▷ How far is $\mathcal{D}^{\mathcal{M}}$ from θ ?
- ▷ Can we explain the previous awkward restrictions in DM4Free ?

From Dijkstra monad to effect observation

Given a Dijkstra monad $\mathcal{D}$ over W

From Dijkstra monad to effect observation

Given a Dijkstra monad $\mathcal{D}$ over W

$$\mathcal{M}\,A \quad = \quad (w : \mathrm{W}\,A) \times \mathcal{D}\,A\,w$$

$$\mathcal{M} \xrightarrow{\pi_1} \mathrm{W}$$

From Dijkstra monad to effect observation

Given a Dijkstra monad $\mathcal{D}$ over W

$$\mathcal{M}\,A \quad = \quad (w : \mathrm{W}\,A) \times \mathcal{D}\,A\,w$$

$$\mathcal{M} \xrightarrow{\quad\pi_1\quad} \mathrm{W}$$

Extends to a (categorical) equivalence

$$\begin{array}{ccc} \text{Dijkstra monads} & \cong & \text{Effect observations} \\ (\mathrm{W}, \mathcal{D}) & & \theta : \mathcal{M} \to \mathrm{W} \end{array}$$

A DSL for monad transformers

$$C ::= \mathbb{M}A \mid C_1 \times C_2 \mid (x : A) \to C \mid C_1 \to C_2 \quad A \in \mathit{Type}_{\mathcal{L}}$$

$$t ::= \texttt{ret} \mid \texttt{bind} \mid \langle t_1,\ t_2 \rangle \mid \pi_i\ t \mid x \mid \lambda x.\, t \mid t_1\ t_2 \mid \lambda^{\diamond} x.\ t \mid t\ u$$

A DSL for monad transformers

$$C ::= \mathbb{M}A \mid C_1 \times C_2 \mid (x : A) \to C \mid C_1 \to C_2 \quad A \in \mathit{Type}_{\mathcal{L}}$$

$$t ::= \texttt{ret} \mid \texttt{bind} \mid \langle t_1,\ t_2 \rangle \mid \pi_i\ t \mid x \mid \lambda x.\, t \mid t_1\ t_2 \mid \lambda^{\diamond} x.\ t \mid t\ u$$

Observation 1: if C and $\mathcal{M}$ are monads, $\mathcal{T}^{C}(\mathcal{M}) = C[\mathcal{M}/\mathbb{M}]$ is a monad

A DSL for monad transformers

$$C ::= \mathbb{M}A \mid C_1 \times C_2 \mid (x : A) \to C \mid C_1 \to C_2 \quad A \in Type_{\mathcal{L}}$$

$$t ::= \texttt{ret} \mid \texttt{bind} \mid \langle t_1,\ t_2 \rangle \mid \pi_i\ t \mid x \mid \lambda x.\, t \mid t_1\ t_2 \mid \lambda^{\diamond} x.\ t \mid t\ u$$

Observation 1: if C and $\mathcal{M}$ are monads, $\mathcal{T}^C(\mathcal{M}) = C[\mathcal{M}/\mathbb{M}]$ is a monad

Observation 2: $\mathcal{T}^C(\mathcal{M}) = C[\mathcal{M}/\mathbb{M}]$ comes with an $\mathcal{M}$-algebra structure α

A DSL for monad transformers

$$C ::= \mathbb{M}A \mid C_1 \times C_2 \mid (x : A) \to C \mid C_1 \to C_2 \quad A \in \mathit{Type}_{\mathcal{L}}$$

$$t ::= \mathtt{ret} \mid \mathtt{bind} \mid \langle t_1,\ t_2 \rangle \mid \pi_i\ t \mid x \mid \lambda x.\, t \mid t_1\ t_2 \mid \lambda^{\diamond} x.\ t \mid t\ u$$

Observation 1: if C and $\mathcal{M}$ are monads, $\mathcal{T}^C(\mathcal{M}) = C[\mathcal{M}/\mathbb{M}]$ is a monad

Observation 2: $\mathcal{T}^C(\mathcal{M}) = C[\mathcal{M}/\mathbb{M}]$ comes with an $\mathcal{M}$-algebra structure α

$$\mathtt{lift} \quad : \quad \mathcal{M} \xrightarrow{\mathcal{M}(\mathtt{ret}^{\mathcal{T}^C(\mathcal{M})})} \mathcal{M}(\mathcal{T}^C(\mathcal{M})) \xrightarrow{\alpha} \mathcal{T}^C(\mathcal{M})$$

Reinterpreting DM4Free

Under a few conditions on C: $\mathcal{T}^C$ is actually a **monad transformer**

$\rightsquigarrow$ uses a logical relation to extend $\mathcal{T}^C$ on monad morphisms

Reinterpreting DM4Free

Under a few conditions on C: $\mathcal{T}^C$ is actually a **monad transformer**

$\rightsquigarrow$ uses a logical relation to extend $\mathcal{T}^C$ on monad morphisms

Effect observation for DM4Free:

$$\frac{\mathrm{Id} \xrightarrow{\texttt{ret}} \mathrm{Cont}_{\mathbb{P}}}{\theta : \mathcal{T}(\mathrm{Id}) \xrightarrow{\mathcal{T}(\texttt{ret})} \mathcal{T}(\mathrm{Cont}_{\mathbb{P}})}$$

Reinterpreting DM4Free

Under a few conditions on C: $\mathcal{T}^C$ is actually a **monad transformer**

$\rightsquigarrow$ uses a logical relation to extend $\mathcal{T}^C$ on monad morphisms

Effect observation for DM4Free:

$$\frac{\mathrm{Id} \xrightarrow{\texttt{ret}} \mathrm{Cont}_{\mathbb{P}}}{\theta : \mathcal{T}(\mathrm{Id}) \xrightarrow{\mathcal{T}(\texttt{ret})} \mathcal{T}(\mathrm{Cont}_{\mathbb{P}})}$$

State	$C[X] = \mathcal{S} \to \mathbb{M}(X \times \mathcal{S})$
Exceptions	$C[X] = \mathbb{M}(X + \mathcal{E})$
Update	$C[X] = \chi \to \mathbb{M}(X \times U)$
Monotonic State	$C[X] = (s_0 : \mathcal{S}) \to \mathbb{M}(X \times \{s_1 : S \mid s_0 \leq s_1\})$

Further directions

- Implementations: native in F*, quite similar to Hoare Type Theory in Coq (experimental)

Further directions

- ▷ Implementations: native in F*, quite similar to Hoare Type Theory in Coq (experimental)
- ▷ Algebraic effects and handlers (catch for exceptions, general recursion using free monads à la McBride (2015))

Further directions

- ▷ Implementations: native in F*, quite similar to Hoare Type Theory in Coq (experimental)
- ▷ Algebraic effects and handlers (catch for exceptions, general recursion using free monads à la McBride (2015))
- ▷ Partial monad morphism, monadic relations as effect observations

Further directions

- ▷ Implementations: native in F*, quite similar to Hoare Type Theory in Coq (experimental)
- ▷ Algebraic effects and handlers (catch for exceptions, general recursion using free monads à la McBride (2015))
- ▷ Partial monad morphism, monadic relations as effect observations
- ▷ Relational reasoning (RHTT, RF*)

Further directions

- ▷ Implementations: native in F*, quite similar to Hoare Type Theory in Coq (experimental)
- ▷ Algebraic effects and handlers (catch for exceptions, general recursion using free monads à la McBride (2015))
- ▷ Partial monad morphism, monadic relations as effect observations
- ▷ Relational reasoning (RHTT, RF*)

Thank you !

Bibliography

D. Ahman, C. Hriţcu, K. Maillard, G. Martínez, G. Plotkin, J. Protzenko, A. Rastogi, and N. Swamy. Dijkstra monads for free. **POPL**. 2017.

S. Katsumata. Parametric effect monads and semantics of effect systems. **POPL**. 2014.

C. McBride. Turing-completeness totally free. **MPC**. 2015.

E. Moggi. Computational lambda-calculus and monads. **LICS**. 1989.

G. D. Plotkin and J. Power. Algebraic operations and generic effects. **Applied Categorical Structures**, 11(1):69–94, 2003.

General recursion

Fix domain Dom, $\prec \subseteq \mathrm{Dom} \times \mathrm{Dom}$ well-founded, codomain Cod.

GenRec free monad on one operation

$$\texttt{call} \quad : \quad \mathrm{Dom} \quad \rightsquigarrow \quad \mathrm{Cod}$$

Given a recursion invariant $(pre_d, post_d)_d : (d : \mathrm{Dom}) \to \mathbb{P} \times (\mathrm{Cod} \to \mathbb{P})$ and $d_0 : \mathrm{Dom}$, define

$$\theta^{\mathsf{GenRec}} \quad : \quad \mathsf{GenRec} \quad \longrightarrow \quad \mathrm{Cont}_{\mathbb{P}}$$

$$\begin{aligned}
\theta^{\mathsf{GenRec}}(\texttt{ret}\, v) &= \lambda p.\, p\, v \\
\theta^{\mathsf{GenRec}}(\texttt{call}\, d\, k) &= \lambda p.\, pre_d \wedge d \prec d_0 \wedge \\
&\qquad \forall c : \mathrm{Cod}, post_d\, c \Rightarrow \theta^{\mathsf{GenRec}}(k\, c)\, p
\end{aligned}$$

General recursion in Dijkstra monad form

From θ^{GenRec}, derive a dijkstra monad $\mathcal{D}^{\mathsf{GenRec}}_{d_0}$ with

▷ An operation

$$\texttt{call} \quad : \quad (d : \mathrm{Dom}) \quad \longrightarrow \quad \mathcal{D}^{\mathsf{GenRec}}_{d_0}\,\mathrm{Cod}\;(\mathit{pre}_d \wedge d \prec d_0)\;\mathit{post}_d$$

▷ A handler

$$\begin{aligned}\texttt{fix} \quad : \quad & ((d : \mathrm{Dom}) \to \mathcal{D}^{\mathsf{GenRec}}_{d}\,\mathrm{Cod}\;\mathit{pre}_d\;\mathit{post}_d) \to \\ & (d : \mathrm{Dom}) \to \mathrm{Pure}\;\mathrm{Cod}\;\mathit{pre}_d\;\mathit{post}_d\end{aligned}$$

Back to fibonacci

$$\texttt{val}\ \ \mathit{fib}_0 : (n : \mathbb{N}) \to \mathcal{D}_n^{\mathsf{GenRec}}\,\mathbb{N}\ \top\ (\lambda r.\, r \geq 1 \wedge r \geq n))$$

$$\begin{array}{l}
\texttt{let}\ \ \mathit{fib}_0\ n = \\
\qquad \texttt{if}\ \ n \leq 1\ \texttt{then}\ 1 \\
\qquad \texttt{else} \\
\qquad\quad \texttt{let}\ \ r_1 = \texttt{call}\,(n-1)\ \texttt{in} \\
\qquad\quad \texttt{let}\ \ r_2 = \texttt{call}\,(n-2)\ \texttt{in}\ \ r_1 + r_2
\end{array}$$

$$\texttt{val}\ \ \mathit{fib} : (n : \mathbb{N}) \to \mathrm{Pure}\,\mathbb{N}\ \top\ (\lambda r.\, r \geq 1 \wedge r \geq n))$$

$$\texttt{let}\ \ \mathit{fib} = \texttt{fix}\ \mathit{fib}_0$$