Dijkstra Monads for All

An Everest All Hands Pitch

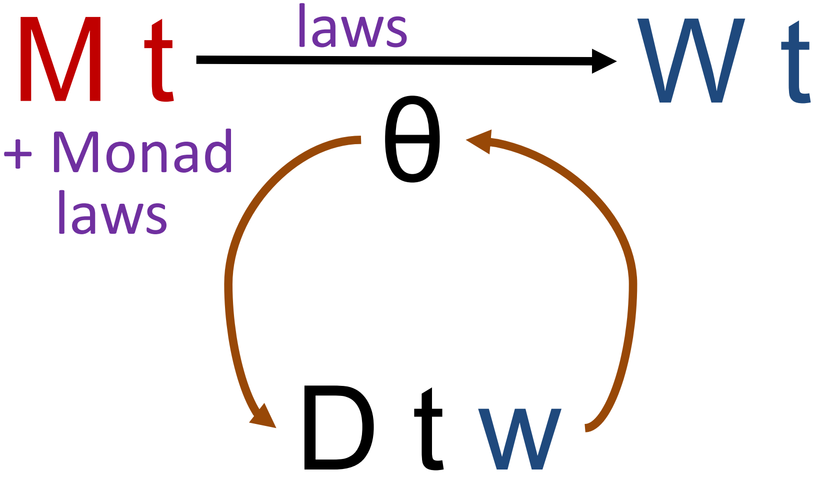
Kenji Maillard
Danel Ahman Robert Atkey Guido Martínez
Cătălin Hriţcu Éric Tanter Exequiel Rivas

ICFP'19 paper @ https://arxiv.org/abs/1903.01237

Dijkstra Monads

ret^M
bind^M
act^M

+ Monad morphism laws



+ Monad laws

$\mathrm{ret}^W : x{:}a \to W\ a$
$\mathrm{bind}^W : W\ a \to (a \to W\ b) \to W\ b$
$\mathrm{act}^W : \quad \ldots \to W\ a$
$(\leq) : w_1{:}W\ a \to w_2{:}W\ a \to \mathrm{Type}_0$

+ Monad laws + ≤ is a preorder
+ bind^W monotonic

$\mathrm{ret}^D : x{:}a \to D\ a\ (\mathrm{ret}^W\ x)$
$\mathrm{bind}^D : \#w{:}W\ a \to \#f{:}(a \to W\ b) \to \ldots \to D\ b\ (\mathrm{bind}^W\ w\ f)$
$\mathrm{act}^D : \quad \ldots \to D\ a\ (\mathrm{act}^W\ \ldots)$
$\mathrm{weaken}^D : w_1{:}W\ a \to w_2{:}W\ a\{w_1 \leq w_2\} \to D\ a\ w_1 \to D\ a\ w_2$

+ Dijkstra monad laws (bind^D-ret^D, ret^D-bind^D, bind^D-bind^D, weaken^D-bind^D, weaken^D-refl, weaken^D-trans)

Short-term benefits for F*

- big step towards **effect definition mechanism** that is **general**, **sound**, and **usable**
  - like DM4Free, aiming for soundness by construction
- **more expressive**, can do more effects than DM4Free:
  - IO (ongoing case study: small web server by Cezar, Exe, ...)
  - nondeterminism (... later probabilities, continuations?)
- **more flexible** than DM4Free:
  - nondeterminism: angelic θ vs demonic θ
  - IO: context-free W vs. history-dependent W (ghost state)
- **ready to merge in F* master soon (Guido)**

Long-term benefits

1. Better understanding of Dijkstra monads

- Formal definition of Dijkstra monads (including laws!)
- **In Coq** we can abstract over Dijkstra monads, which gives us a form of **effect polymorphism**
  - Kenji used the spec. monad laws to verify map and fold
- **In F* effect polymorphism is interesting direction**
  - F* effects are not first class (by design)
  - spec. monad laws might be automatable via SMT or tactics
  - bonded effect polymorphism already interesting
    - e.g. all effects with the same W (Pure, Div, Ghost)

Long-term benefits

2. Better understanding of DM4Free

- **DM4Free is just a special case of DM4All**
  - **for any monad transformer** T:
    M=T(Id), W=T((_→Prop)→Prop), canonical θ
- **SM: lang. for defining correct monad transformers**
  - **subsumes DM language** from DM4Free
  - **currently in Coq, could be ported to F***
- **Make F* effect definitions usable and sound:**
  - Currently F* **ignores** all laws, **let's enforce them!**
  - either manually (with SMT) or get them from SM

Long-term benefits

3. Better understanding of specification monads

- **they are ordered monads with monotonic bind**
  - + **conjunction** seems to account for recording conditional guards or effect-specific asserts (Guido, Kenji)
- **general recipe for obtaining specification monads**
  - – apply **monad transformers** (from SM) to **various base specification monads:**
  - – not just **weakest-pre** and **pre+post**, but also **strongest-post** (as expressive as weakest-liberal-pre)
- **optimize wps: use strongest-post? wlps? (Guido)**
- **monotonic state: from "Prop" to "S -> Prop"? (Danel, Kenji, ...)**
- **quantitative spec. monads (cost, probabilities -- Kenji)**

Long-term benefits

4. Better understanding Dijkstra monad actions

- **algebraic operations are simple (get, put)**
- **handlers more complicated**
  - experiment 1: exception catching (Danel)
  - experiment 2: fixpoints / general recursion (Bob, Kenji)
    - independent validation for F*'s semantic termination check
  - **more work needed for the general story (Danel, ...)**

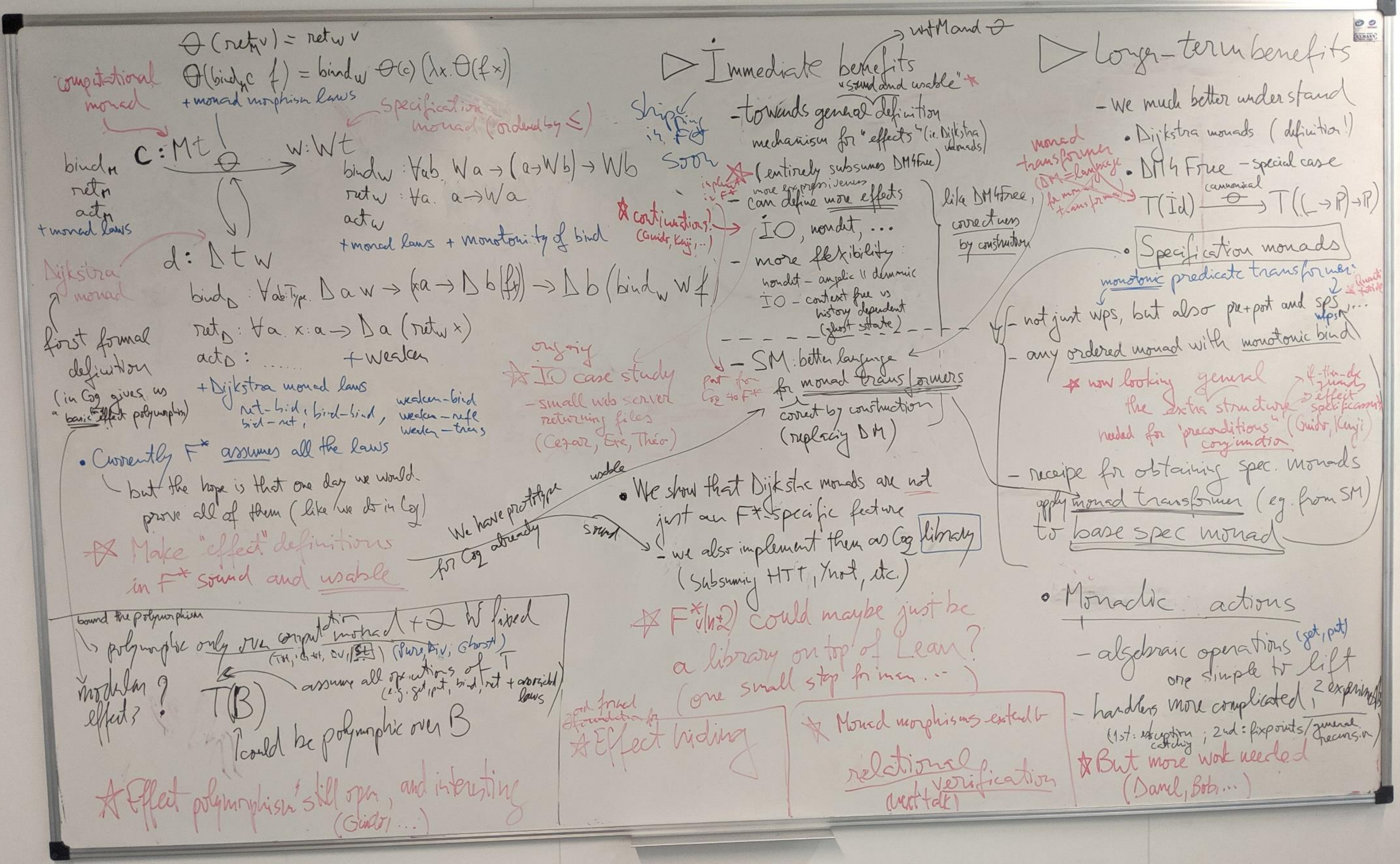
Long-term benefits

5. Showing that Dijkstra Monads not F*-specific

- **we implemented them as just a library in Coq**
  - subsuming Hoare Type Theory, Ynot, etc.
- **maybe F* v(2+n) will be just a library on top of Lean**
  - would be great, many more steps needed though:
    e.g. there's more to F* effects than just Dijkstra monads
    e.g. SMT encoding, extensional equality, ...

6. Strong foundations for further research

- **effect hiding / observational purity**
- **relational verification (Friday @ 9am)**



$\Theta(\text{ret}_M v) = \text{ret}_W v$

$\Theta(\text{bind}_M c\ f) = \text{bind}_W\ \Theta(c)\ (\lambda x.\ \Theta(f x))$

+monad morphism laws

computational monad

Specification monad (ordered by ≤)

$$\begin{tikzcd} c : M\,t \arrow[r, "\Theta"] & w : W\,t \end{tikzcd}$$

bind_M
ret_M
act_M
+monad laws

$\text{bind}_W : \forall a b.\ W a \to (a \to W b) \to W b$
$\text{ret}_W : \forall a.\ a \to W a$
act_W
+monad laws + monotonicity of bind

Dijkstra monad

$d : \Delta\ t\ w$

$\text{bind}_\Delta : \forall a b{:}\text{Type}.\ \Delta\ a\ w \to (x{:}a \to \Delta\ b\ (f x)) \to \Delta\ b\ (\text{bind}_W\ w\ f)$

$\text{ret}_\Delta : \forall a.\ x{:}a \to \Delta\ a\ (\text{ret}_W\ x)$

$\text{act}_\Delta :$ …… + weaken

+Dijkstra monad laws
ret-bind, bind-bind, bind-ret, weaken-bind, weaken-refl, weaken-trans

first formal definition
(in Coq gives us a basic form effect polymorphism)

- Currently F* assumes all the laws
  - but the hope is that one day we would prove all of them (like we do in Coq)

☆ Make "effect" definitions in F* sound and usable

We have prototype for Coq already — usable — sound

Shipping in F* soon

☆ continuations? (Guido, Kenji, ..)

ongoing
☆ IO case study
- small web server returning files
(Cezar, Eric, Théo)

▷ İmmediate benefits → wrt M and Θ

"sound and usable" *

- towards general definition mechanism for "effects" (i.e. Dijkstra monads)
☆ (entirely subsumes DM4Free)

implement in F*

- can define more effects — more expressiveness
  İO, nondet, …
- more flexibility:
  nondet – angelic ‖ demonic
  İO – context free vs history dependent (ghost state)

like DM4Free, correctness by construction

- SM: better language for monad transformers correct by construction (replacing DM)

port from Coq to F*

- We show that Dijkstra monads are not just an F*-specific feature
  - we also implement them as Coq library (subsuming HTT, Ynot, etc.)

☆ F* v(n+2) could maybe just be a library on top of Lean? (one small step for man…)

▷ Longer-term benefits

- We much better understand
  - Dijkstra monads (definition!)
  - DM4Free – special case

monad transformer (DM = language for monad transformers)

$$T(\text{Id}) \xrightarrow[\Theta]{\text{canonical}} T((\_ \to \mathbb{P}) \to \mathbb{P})$$

- Specification monads

monotonic predicate transformer: Quantitative

- not just wps, but also pre+post and sps, wpsn…
- any ordered monad with monotonic bind

☆ now looking general the extra structure needed for "preconditions" conjunction (Guido, Kenji) → if-then-else guards → effect specifications

- receipe for obtaining spec. monads
  apply monad transformer (eg. from SM) to base spec monad

- Monadic actions
  - algebraic operations (get, put) one simple to lift
  - handlers more complicated, 2 experiments (1st: exception catching; 2nd: fixpoints/general recursion)

☆ But more work needed (Daniel, Bob…)

bound the polymorphism

↳ polymorphic only over computation monad ∆ + Θ W fixed
(TM, CH, ST, DV, ST) (Pure, Div, Ghost)

modular effect? ?

T(B) ← assume all operations of T (e.g. get, put, bind, ret + associated laws)

could be polymorphic over B

☆ Effect polymorphism still open, and interesting (Guido, …)

formal foundation for
☆ Effect hiding

☆ Monad morphisms extend to relational verification (next talk)