Dijkstra Monads for All

An Everest All Hands Pitch

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

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

\( \text{Dijkstra monad laws} \)

\( \text{ret}^\text{W} : x : a \rightarrow W \, a \)
\( \text{bind}^\text{W} : W \, a \rightarrow (a \rightarrow W \, b) \rightarrow W \, b \)
\( \text{act}^\text{W} : \ldots \rightarrow W \, a \)
\( (\leq) : w_1 : W \, a \rightarrow w_2 : W \, a \rightarrow Type_0 \)

+ Monad laws + \( \leq \) is a preorder
+ \( \text{bind}^\text{W} \) monotonic

\( \text{ret}^\text{D} : x : a \rightarrow \text{D} \, a \) (\( \text{ret}^\text{W} \, x \))
\( \text{bind}^\text{D} : \#w : W \, a \rightarrow \#f : (a \rightarrow W \, b) \rightarrow \ldots \rightarrow \text{D} \, b \) (\( \text{bind}^\text{W} \, w \, f \))
\( \text{act}^\text{D} : \ldots \rightarrow \text{D} \, a \) (\( \text{act}^\text{W} \, \ldots \))
\( \text{weaken}^\text{D} : w_1 : W \, a \rightarrow w_2 : W \, a \{w_1 \leq w_2\} \rightarrow \text{D} \, a \, w_1 \rightarrow \text{D} \, a \, w_2 \)

+ Dijkstra monad laws (\( \text{bind}^\text{D} \)-\( \text{ret}^\text{D} \), \( \text{ret}^\text{D} \)-\( \text{bind}^\text{D} \), \( \text{bind}^\text{D} \)-\( \text{bind}^\text{D} \),

\( \text{weaken}^\text{D} \)-\( \text{bind}^\text{D} \), \( \text{weaken}^\text{D} \)-\( \text{refl} \), \( \text{weaken}^\text{D} \)-\( \text{trans} \))
Short-term benefits for

- 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 $\theta$ vs demonic $\theta$
  - 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 $T$:
    $M = T(Id)$, $W = T((\_ \rightarrow \text{Prop}) \rightarrow \text{Prop})$, canonical $\theta$

- **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)
Immediate benefits
- standard I/O
- towards general definition
- mechanism for effects in languages
- entirely subsumes Dplug
- can define more effects
- IO, monad, ...
- more flexibility
- support for dynamic programming

Long-term benefits
- we need better understanding
- Dplug monads (definition)
- Dplug-free, special case
- strict version (induction)

- Specification monads
  - update, rewrite, etc.
  - no lookup general
  - finds the data structure, specifications

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

Monadic actions
- algebraic operations (get, put, fold)
- one step to lift
- handles more complicated
- (1st: input, 2nd: output, generalization)
- But more worked needed
  - (Daml, etc.)