Dijkstra Monads in Monadic Computation

Bart Jacobs

Institute for Computing and Information Sciences – Digital Security
Radboud University Nijmegen

CMCS, Grenoble, 5 & 6 April 2014

Outline

Introduction & overview

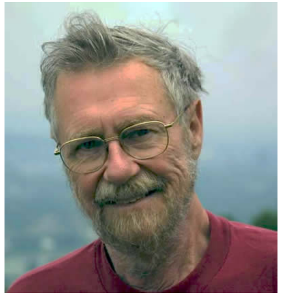
Program logics via examples

Weakest precondition computation as map of monads

Towards a general construction

Conclusions

Edsger Dijkstra 1930 - 2002



Obituary "Portrait of a Genius" by Krzysztof Apt in FACS 2002, see also: http://homepages.cwi.nl/~apt/ps/dijkstra.pdf

Dijkstra monad I

- Introduced within the setting of program verification
  - Swamy, Weinberger, Schlesinger, Chen, Livshits. *Verifying higher-order programs with the Dijkstra monad*. In: PLDI 2013.
- Usually monads capture some form of computation
  - partial, non-deterministic, probabilistic, etc
- Dijkstra monad captures weakest precondition computation
  - it describes a program via its weakest precondition calculation (going backwards)
- There is a similar Hoare monad that captures programs as (forward) maps from (extends of) pre- to post-conditions
  - it does not play a role here

Dijkstra monad II

- The PLDI'13 paper uses the language of the theorem prover Coq
- `DST a wp` is an abbreviation for the type

$$\forall p.h : heap\ \{wp\ p\ h\} \to (x : a * h : heap\ \{p\ x\ h\})$$

> *"That is, in order for the output heap h to satisfy p x h, for any predicate p, one needs to prove wp p h of the input heap h."*

- Own naive translation into monad $\mathfrak{D}$ on **Sets**,

$$\mathfrak{D}(X) = \mathcal{P}(S)^{\mathcal{P}(S\times X)} \qquad \text{for fixed set of states } S$$

$w \in \mathfrak{D}(X)$ transforms a postcondition $Q \subseteq S \times X$ into a precondition $P \subseteq S$ — where X is the type of the output

Dijkstra monad III

- Unit $\eta\colon X \to \mathfrak{D}(X) = \mathcal{P}(S)^{\mathcal{P}(S\times X)}$ of the Dijkstra monad is:

$$\eta(x)(Q) = \{s \in S \mid \langle s, x\rangle \in Q\}$$

- There is a similarity with the state monad $\mathfrak{S}(X) = (S \times X)^S$.
- For instance,

$$\eta^{\mathfrak{D}}(x) = \left(\eta^{\mathfrak{S}}(x)\right)^{-1} : \mathcal{P}(S\times X) \longrightarrow \mathcal{P}(S)$$

where $(-)^{-1}$ is inverse image, i.e. substitution in logic!

What is going on? What logic is behind this?

Dijkstra monad IV

- It turns out that there is a map of monads $\mathfrak{S} \Rightarrow \mathfrak{D}$
  - from the state monad $\mathfrak{S}$ to the Dijkstra monad $\mathfrak{D}$
  - this map is inverse image / substitution / weakest precondition
- Explicitly

$$\begin{array}{ccc} \mathfrak{S}(X) = (S \times X)^S & \longrightarrow & \mathcal{P}(S)^{\mathcal{P}(S\times X)} = \mathfrak{D}(X) \\ f & \longmapsto & f^{-1} = \mathrm{wp}(f) \end{array}$$

- This will be described more generally:
  - in a general set-up for program semantics & logic
  - leading to more examples
  - and to more general (and precise) Dijkstra monads
  - Note: there are different "Dijkstra monads" depending on the monad and on the logic involved.

General picture: "state-and-effect triangles"

$$\mathbf{Log}^{\mathrm{op}} = \Big(\text{predicate transformers}\Big) \underset{\longleftarrow}{\overset{\longrightarrow}{\top}} \Big(\text{state transformers}\Big)$$

$$\text{Pred}: \Big(\text{computations}\Big) \to \Big(\text{predicate transformers}\Big), \qquad \text{Stat}: \Big(\text{computations}\Big) \to \Big(\text{state transformers}\Big)$$

It involves:

- a contravariant adjunction (sometimes equivalence) between predicate- and state-transformers
- In the quantum world this is the duality between states and effects
  - Schrödinger computed on states, Heisenberg on effects
  - this is very close to traditional program logic (in CS)

A bird's eye view on non-deterministic computation I

- Semantics of a non-determinsitic program is given by:
  - relations $R \subseteq X \times Y$, or, more categorically:
  - functions $X \to \mathcal{P}(Y)$, ie. maps in the Kleisli category $\mathcal{K}\ell(\mathcal{P})$
- Full & faithful functor "from Kleisli to Eilenberg-Moore"
  - here: $\mathcal{K}\ell(\mathcal{P}) \to \mathcal{EM}(\mathcal{P}) = \mathbf{CL}_\bigvee$
  - where $\mathbf{CL}_\bigvee$ is complete lattices with join-preserving maps
- According to Dijkstra, each program $s\colon X \to \mathcal{P}(Y)$ gives weakest precondition operation $\mathrm{wp}(s)\colon \mathcal{P}(Y) \to \mathcal{P}(X)$
  - Explicitly, $\mathrm{wp}(s)(Q) = \{x \mid s(x) \subseteq Q\}$
  - $\mathrm{wp}(s)$ preserves meets, so is map in $\mathbf{CL}_\bigwedge$

A bird's eye view on non-deterministic computation II

There are bijective correspondences:

$$\begin{array}{ll} X \xrightarrow{\;s\;} \mathcal{P}(Y) & \\ \hline\hline \mathcal{P}(X) \longrightarrow \mathcal{P}(Y) & \bigvee\text{-preserving} \\ \hline\hline \mathcal{P}(Y) \xrightarrow[\mathrm{wp}(s)]{} \mathcal{P}(X) & \bigwedge\text{-preserving} \end{array}$$

More categorically, there is a commuting diagram:

$$(\mathbf{CL}_\bigwedge)^{\mathrm{op}} \underset{\longleftarrow}{\overset{\longrightarrow}{\cong}} \mathbf{CL}_\bigvee = \mathcal{EM}(\mathcal{P}), \qquad \mathrm{Pred}: \mathcal{K}\ell(\mathcal{P}) \to (\mathbf{CL}_\bigwedge)^{\mathrm{op}}, \quad \mathrm{Stat}: \mathcal{K}\ell(\mathcal{P}) \to \mathbf{CL}_\bigvee$$

- The "predicate" and "state" functors $\mathrm{Pred}, \mathrm{Stat}$ are f&f
- $\mathrm{Pred}(s) = \mathrm{wp}(s) =$ "substitution", for Kleisli maps $X \xrightarrow{s} \mathcal{P}(Y)$

A bird's eye view on non-deterministic computation III

$$(\mathbf{CL}_\bigwedge)^{\mathrm{op}} \underset{\longleftarrow}{\overset{\longrightarrow}{\cong}} \mathbf{CL}_\bigvee = \mathcal{EM}(\mathcal{P}), \qquad \mathrm{Pred}: \mathcal{K}\ell(\mathcal{P}) \to (\mathbf{CL}_\bigwedge)^{\mathrm{op}}, \quad \mathrm{Stat}: \mathcal{K}\ell(\mathcal{P}) \to \mathbf{CL}_\bigvee$$

- In this setting, re-define / refine the Dijkstra as homsets:

$$\begin{aligned} \mathfrak{D}(X) &= (\mathbf{CL}_\bigwedge)\Big(\mathrm{Pred}(S \times X), \mathrm{Pred}(S)\Big) \\ &= (\mathbf{CL}_\bigwedge)^{\mathrm{op}}\Big(\mathrm{Pred}(S), \mathrm{Pred}(S \times X)\Big) \end{aligned}$$

- Recall the state monad $\mathfrak{S}(X) = (S \times X)^S = \mathbf{Sets}(S, S \times X)$
  - It looks like this monad is "lifted to the logic" $\mathbf{CL}_\bigwedge$

A bird's eye view on probabilistic computation I

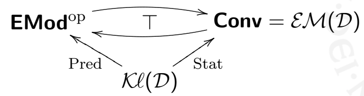
- Semantics of a probabilistic program is given by:
  - a stochastic matrix M on $X \times Y$, or, more categorically:
  - a function $X \to \mathcal{D}(Y)$, ie. a map in the Kleisli category $\mathcal{K}\ell(\mathcal{D})$
  - where $\mathcal{D}$ is the distribution monad on **Sets**
- Full & faithful functor "from Kleisli to Eilenberg-Moore"
  - here: $\mathcal{K}\ell(\mathcal{D}) \to \mathcal{EM}(\mathcal{D}) = \mathbf{Conv}$, category of convex sets
- We now use fuzzy predicates $[0,1]^X$ on X
  - they have the structure of an effect module
  - partial sum $\ovee$, orthocomplement $(-)^\perp$, scalar multiplication
- Again each program $s\colon X \to \mathcal{D}(Y)$ gives weakest precondition operation $\mathrm{wp}(s)\colon [0,1]^Y \to [0,1]^X$
  - Explicitly, $\mathrm{wp}(s)(q)(x) = \sum_y s(x)(y) \cdot q(y)$
  - $\mathrm{wp}(s)$ preserves effect module structure

A bird's eye view on probabilistic computation II

There are bijective correspondences (for Y finite):

$$X \xrightarrow{s} \mathcal{D}(Y)$$
$$\overline{\overline{\mathcal{D}(X) \longrightarrow \mathcal{D}(Y)}} \quad \text{preserving convex sums}$$
$$\overline{\overline{[0,1]^Y \xrightarrow[\mathrm{wp}(s)]{} [0,1]^X}} \quad \text{preserving effect module structure}$$

More categorically, there is a triangle:



- In this setting, we can also define a Dijkstra monad, as:

$$\mathfrak{D}(X) = \mathbf{EMod}^{\mathrm{op}}\big(\mathrm{Pred}(S), \mathrm{Pred}(S \times X)\big) = \mathbf{EMod}\big([0,1]^{S \times X}, [0,1]^S\big)$$

More triangles ...

- Many more forms of computation give rise to such state-and-effect triangles
- See proceedings paper for more illustrations
  - most of them with Kleisli category as base category
  - but also with C^*-algebras, for quantum computation
- More about a general construction towards the end

State monad transformer

- So far we have used $X \mapsto (S \times X)^S$ as a monad itself
- However, it is also a monad transformer
  - given a monad T, we can form a new "state" version of T
  - written as: T yields $\mathfrak{S}_T$
- Explicit definition:

$$\mathfrak{S}_T(X) = T(S \times X)^S$$

- Pattern that exists in examples: weakest precondition forms a map of monads:

$$\mathfrak{S}_T \xRightarrow{\mathrm{wp}} \mathfrak{D}_T$$

from state monad for T to Dijkstra monad for T
(Categorically this is very nice!)

Non-deterministic & probabilistic wp as monad-map

$$\mathfrak{S}_{\mathcal{P}}(X) = \mathcal{P}(S \times X)^S \xrightarrow{\mathrm{wp}} \mathbf{CL}_{\bigwedge}\big(\mathcal{P}(S \times X), \mathcal{P}(S)\big) = \mathfrak{D}_{\mathcal{P}}(X)$$
$$f \longmapsto \mathrm{wp}(f) = \mathrm{Pred}(f) = \text{substitution}$$

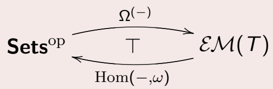
$$\mathfrak{S}_{\mathcal{D}}(X) = \mathcal{D}(S \times X)^S \xrightarrow{\mathrm{wp}} \mathbf{EMod}\big([0,1]^{S \times X}, [0,1]^S\big) = \mathfrak{D}_{\mathcal{D}}(X)$$
$$f \longmapsto \mathrm{wp}(f) = \mathrm{Pred}(f) = \text{substitution}$$

- These wp's commute with the monads' unit & multiplication
- What is behind this? How general is this?
  - the logic $\mathbf{CL}_{\bigwedge}$, $\mathbf{EMod}$ involved is specific for the monads $\mathcal{P}, \mathcal{D}$.

A basic adjunction for Eilenberg-Moore categories

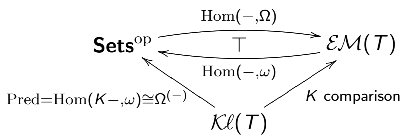
Theorem (folklore?)

Let T be a monad on **Sets**, and $\omega\colon T(\Omega) \to \Omega$ an Eilenberg-Moore algebra. Then there is an adjunction:



- This generalises to strong monads T on a symmetric monoidal closed category $\mathbb{B}$ with equalisers
- The adjunction can be used as starting point for a state-and-effect triangle.

From the adjunction to a triangle



Further remarks

- One can try to restrict the adjunction to a "logically sensible" subcategory of **Sets**. This is ongoing work.
- By composition with the adjunction $\mathbf{Sets} \leftrightarrows \mathcal{EM}(T)$ one gets a second monad on **Sets**, namely Lawvere's double dual:

$$T_\omega(X) = \Omega^{(\Omega^X)} \qquad \text{with monad map} \qquad T \Longrightarrow T_\omega$$

Concluding remarks

- The paper contains:
  - a categorical version of the type-theoretic Dijkstra monad
  - a refined version using the logic involved
  - an extension to other examples
  - weakest precondition as map of monads
- State-and-effect triangles as useful conceptual framework
  - question remains: what is the right logic for which kind of computation?
  - (other question: how to combine the triangle with operational semantics?)
- Other remaining question: what is the Hoare monad?
- Not discussed here, but mentioned in the paper: many triangles are enriched giving wp-rules, like $\mathrm{wp}(s_1 \cup s_2) = \mathrm{wp}(s_1) \wedge \mathrm{wp}(s_2)$.