Towards refined notions of computation: multisorted algebras and algebraic effects

Danel Ahman

LFCS, University of Edinburgh

EWSCS, 6 March 2013

work in progress with Gordon Plotkin and Alex Simpson

THE UNIVERSITY of EDINBURGH informatics

lfcs Laboratory for Foundations of Computer Science

Motivation

- We want to study algebraic computational effects in more involved settings (compared to just simple types)
- This work aims to investigate how computational effects can be combined with refinement types, to:
  - use logic to refine existing computational effects
  - and hopefully discover models of useful notions of computations
- Initial directions:
  - adding (computation) refinement types to impure languages, such as Levy's Call-by-Push-Value
  - refinement types + Lawvere theories
  - fibrational semantics for refinement types
  - understanding handlers involving refinement types

Earlier history: Moggi and Monads

- Idea: Use monads to abstractly model impure computations
  - $T : \mathcal{C} \longrightarrow \mathcal{C}$
  - $\eta_X : X \to TX$
  - $(-)^\dagger : (X \to TY) \to (TX \to TY)$

- Example monads proposed by Moggi
  - exceptions - $TX = X + E$
  - global state - $TX = (S \times X)^S$
    - *(stateful computations $S \times X \longrightarrow S \times Y$)*
  - local state - $(TX)_n = (\int^{m \in (n/I)} (S_m \times X_m))^{S_n}$

Later history: Plotkin-Power and Lawvere theories

- Observation: Most of Moggi's monads are actually induced by Lawvere theories and their algebras
  - gives a way to systematically construct these effects
  - gives operationally natural representation
  - notable exception is the continuations monad

- In this talk, we will be mostly looking at algebras of such Lawvere/algebraic/effect theories

  (and hiding the categorical machinery)

Later history: Plotkin-Power and Lawvere theories

- A presentation of a Lawvere theory $\mathcal{L}$ is given by
  - a collection of base types
  - a collection of operations $op : O \longrightarrow I$
  - equations between derived terms
- An algebra in category $\mathcal{C}$ for such a theory $\mathcal{L}$ is given by
  - an object X (i.e., the carrier)
  - a morphism $op : I \times X^O \longrightarrow X$ for every $op : O \longrightarrow I$
  - satisfying the equations in $\mathcal{L}$
- The corresponding monad $TX = UFX$ is induced by

$$\begin{tikzcd} \mathcal{C} \arrow[r, bend right, "F"'] \arrow[r, phantom, "\top"] & Alg(\mathcal{L},\mathcal{C}) \arrow[l, bend right, "U"'] \end{tikzcd}$$

Example: Algebra for global state

- V - set of values , L - set of locations
- Operations:
  - $lookup : L \times X^V \longrightarrow X$
  - $update : (L \times V) \times X \longrightarrow X$
- Equations:
  1. $update_{loc,v}(update_{loc,v'}(x)) = update_{loc,v'}(x)$
  2. $update_{loc,v}(lookup_{loc}(x_{v'})_{v'}) = update_{loc,v}(x_v)$
  3. $update_{loc,v}(update_{loc',v'}(x)) =$
     $update_{loc',v'}(update_{loc,v}(x)) \qquad (loc \neq loc')$
  4. ...
- The free algebra is given by $FX = (S \times X)^S$ together with intuitive operation definitions

Refinement types (à la, Denney)

- One way of allowing more detailed specifications in one's type system
- Well-formedness of a refinement type

$$\frac{\Gamma \vdash_{ref} \phi : \mathit{Ref}(\sigma) \quad \Gamma, x : \phi \vdash_{log} P \ \mathit{wf}}{\Gamma \vdash_{ref} (x : \phi)P : \mathit{Ref}(\sigma)}$$

- Introduction rule for refinement types

$$\frac{\Gamma \vdash M : \phi \quad \Gamma \vdash_{log} P[M/x]}{\Gamma \vdash M : (x : \phi)P}$$

- An example of semantics: sets (denoting underlying types) and environment-indexed relations on them
- Not in this talk: fibrational semantics, computation refinement types in CBPV

Refining global state

Refining global state

- Assume we want to model a version of global state where every location/store needs to be "opened/activated" before we can use it
- We also want the static type system to help us to rule out (some) incorrect programs (e.g., update before opening)
- We aim to use refinement types and logic to formalize it
- Therefore, we assume that we now have predicates $Open(L)$ and $Closed(L) = \neg Open(L)$ on the locations L
- Conceptually they denote subsets of L which are currently opened (resp. closed)
- In the type system they would appear as refinement types $\vdash (x : L)(Open(x)) : Ref(L)$ and $\vdash (x : L)(Closed(x)) : Ref(L)$

Refining global state

- Assume we want to model a version of global state where every location/store needs to be "opened/activated" before we can use it
- We also want the static type system to help us to rule out (some) incorrect programs (e.g., update before opening)
- We aim to use refinement types and logic to formalize it
- Therefore, we assume that we now have predicates $Open(L)$ and $Closed(L) = \neg Open(L)$ on the locations L
- Conceptually they denote subsets of L which are currently opened (resp. closed)
- In the type system they would appear as refinement types $\vdash (x : L)(Open(x)) : Ref(L)$ and $\vdash (x : L)(Closed(x)) : Ref(L)$

Refining global state

- We should only be able to read from and write to locations that are open
  - $lookup : X^V \longrightarrow X^{Open(L)}$
  - $update : X \longrightarrow X^{Open(L) \times V}$

- However, notice that this requires an a priori given collection of open locations

Refining global state

- We should only be able to read from and write to locations that are open
  - $lookup : X^V \longrightarrow X^{Open(L)}$
  - $update : X \longrightarrow X^{Open(L) \times V}$

- However, notice that this requires an a priori given collection of open locations

Refining global state

- To be more dynamic, we should also add operations for opening and closing locations
  - $lookup : X^V \longrightarrow X^{Open(L)}$
  - $update : X \longrightarrow X^{Open(L) \times V}$
  - $open : X \longrightarrow X^{Closed(L)}$
  - $close : X \longrightarrow X^{Open(L)}$
- But we should be able to distinguish between computations able to use different locations
- We could take inspiration from the algebra for local state
  - do the theory and algebra with presheaves Set^W
  - meaning of predicates now depends on worlds
- However, we don't yet know what the appropriate non-discrete world category and the corresponding (monoidal) closed structure should be

Refining global state

- To be more dynamic, we should also add operations for opening and closing locations
  - $lookup : X^V \longrightarrow X^{Open(L)}$
  - $update : X \longrightarrow X^{Open(L) \times V}$
  - $open : X \longrightarrow X^{Closed(L)}$
  - $close : X \longrightarrow X^{Open(L)}$
- But we should be able to distinguish between computations able to use different locations
- We could take inspiration from the algebra for local state
  - do the theory and algebra with presheaves Set^W
  - meaning of predicates now depends on worlds
- However, we don't yet know what the appropriate non-discrete world category and the corresponding (monoidal) closed structure should be

Refining global state

- To be more dynamic, we should also add operations for opening and closing locations
    - $lookup : X^V \longrightarrow X^{Open(L)}$
    - $update : X \longrightarrow X^{Open(L) \times V}$
    - $open : X \longrightarrow X^{Closed(L)}$
    - $close : X \longrightarrow X^{Open(L)}$
- But we should be able to distinguish between computations able to use different locations
- We could take inspiration from the algebra for local state
    - do the theory and algebra with presheaves Set^W
    - meaning of predicates now depends on worlds
- However, we don't yet know what the appropriate non-discrete world category and the corresponding (monoidal) closed structure should be

Refining global state

- To be more dynamic, we should also add operations for opening and closing locations
  - $lookup : X^V \longrightarrow X^{Open(L)}$
  - $update : X \longrightarrow X^{Open(L) \times V}$
  - $open : X \longrightarrow X^{Closed(L)}$
  - $close : X \longrightarrow X^{Open(L)}$
- But we should be able to distinguish between computations able to use different locations
- We could take inspiration from the algebra for local state
  - do the theory and algebra with presheaves Set^W
  - meaning of predicates now depends on worlds
- However, we don't yet know what the appropriate non-discrete world category and the corresponding (monoidal) closed structure should be

Refining global state (W-sorted theories)

- Nevertheless, we can look at multi-sorted theories and algebras to see which monad we would get
- Let the worlds be $W = Bool^L$
- We get the algebra in Set^W
  - $lookup_{w \in W, loc \in Open_w(L)} : (X^V)_w \longrightarrow X_w$
  - $update_{w \in W, loc \in Open_w(L), v \in V} : X_w \longrightarrow X_w$
  - $open_{w \in W, loc \in Closed_w(L)} : X_{w[loc \mapsto \top]} \longrightarrow X_w$
  - $close_{w \in W, loc \in Open_w(L)} : X_{w[loc \mapsto \bot]} \longrightarrow X_w$
- Free algebra for this theory induces the following monad

$$TX_w = UFX_w = (\textstyle\sum_{w' \in W} (S_{w'} \times X_{w'}))^{S_w}$$

$$= (\textstyle\sum_{w' \in W} (S \times X_{w'}))^{S}$$

What next?

- The W-sorted approach gave us the monad we were after
- Can we make it work naturally in the singlesorted case?
- Idea, try to give more general form to the operations
  - $op_w : \prod_{o \in O_w} X_{\delta_o(w,o)} \longrightarrow \prod_{i \in I_w} X_{\delta_i(w,i)}$
- But can't always define them uniformly in w, e.g.:

  $lookup_{[l_i \mapsto \bot]} : \prod_{v \in V} X_{\{[l_i \mapsto \bot]\}} \longrightarrow 1$
- Seems to be kind of inherent to the idea that not all operations should be definable in all worlds
- Other ideas:
  - W induces a family of algebras sharing common carrier
  - Lawvere theories with partiality and dependency

What next?

- The W-sorted approach gave us the monad we were after
- Can we make it work naturally in the singlesorted case?
- Idea, try to give more general form to the operations
  - $op_w : \prod_{o \in O_w} X_{\delta_o(w,o)} \longrightarrow \prod_{i \in I_w} X_{\delta_i(w,i)}$
- But can't always define them uniformly in w, e.g.:

$$lookup_{[l_i \mapsto \perp]} : \prod_{v \in V} X_{\{[l_i \mapsto \perp]\}} \longrightarrow 1$$

- Seems to be kind of inherent to the idea that not all operations should be definable in all worlds
- Other ideas:
  - W induces a family of algebras sharing common carrier
  - Lawvere theories with partiality and dependency

What next?

- The W-sorted approach gave us the monad we were after
- Can we make it work naturally in the singlesorted case?
- Idea, try to give more general form to the operations
  - $op_w : \prod_{o \in O_w} X_{\delta_o(w,o)} \longrightarrow \prod_{i \in I_w} X_{\delta_i(w,i)}$
- But can't always define them uniformly in w, e.g.:

$$lookup_{[l_i \mapsto \perp]} : \prod_{v \in V} X_{\{[l_i \mapsto \perp]\}} \longrightarrow 1$$

- Seems to be kind of inherent to the idea that not all operations should be definable in all worlds
- Other ideas:
  - W induces a family of algebras sharing common carrier
  - Lawvere theories with partiality and dependency

What next?

- The W-sorted approach gave us the monad we were after
- Can we make it work naturally in the singlesorted case?
- Idea, try to give more general form to the operations
  - $op_w : \prod_{o \in O_w} X_{\delta_o(w,o)} \longrightarrow \prod_{i \in I_w} X_{\delta_i(w,i)}$
- But can't always define them uniformly in w, e.g.:

$$lookup_{[l_i \mapsto \perp]} : \prod_{v \in V} X_{\{[l_i \mapsto \perp]\}} \longrightarrow 1$$

- Seems to be kind of inherent to the idea that not all operations should be definable in all worlds
- Other ideas:
  - W induces a family of algebras sharing common carrier
  - Lawvere theories with partiality and dependency

Questions?

Another example of a simple theory

- Inspiration from McBride's work on file operations
- Take the simple set of worlds $W = Bool$
- We are interested in axiomatizing logging in to and logging off from some system
- We would model this with the following algebra
  - $LogIn_{p \in Password} : X_{true} \times X_{false} \longrightarrow X_{false}$
  - $DoSomething : X_{true} \longrightarrow X_{true}$

    (e.g, the state operations)
  - $LogOut : X_{false} \longrightarrow X_{true}$