A syllabus for algebraic effects

Ohad Kammar
<ohad.kammar@cl.cam.ac.uk>

Mathematical Foundations of Programming Semantics XXXI
25 June 2015
Basic definitions
Inaccessible literature: filtered colimits, monadicity, locally presentable categories, adjoint functor theorems, Lawvere theories, ... satisfaction in a $\Sigma$-algebra, obtaining the notion of a $(\Sigma, E)$-algebra in $C$. This, with the evident definition of homomorphism of algebras, generates a category $(\Sigma, E)\text{-}Alg$ with a forgetful functor

$$U : (\Sigma, E)\text{-}Alg \longrightarrow C$$

which, if $C$ is locally presentable, has a left adjoint $F$, inducing a monad $T = UF$ on $C$. The category $(\Sigma, E)\text{-}Alg$ is isomorphic to the category $T\text{-}Alg$ of algebras for the monad $T$.

– Plotkin and Power, “Notions of computation determine monads”, 1st paragraph after the introduction

and later: powers and copowers, enrichment, presheaf categories, sketches, Kan extensions, nerve and dense functors, ...
Accessible semantics of algebraic effects

Roadmap: syllabus for graduate students

(Cambridge Computer Science MPhils)
Setting: target audience

Course format
Lecture class (9 lectures $= 2$ per week $\times 4\frac{1}{2}$ weeks)
50 minute lectures
(7 more lectures with Marcelo Fiore on abstract syntax with binding)

Attendees
5 students taking the class
2 students sitting in
5 PhDs and Postdocs

Not in this talk:
Evaluation, course material, pedagogy
(course under development!)
Design decisions

Work within and around \textbf{Set}
Keep (categorical) concepts concrete.
Rich toolkit (e.g., equational logic).

\textbf{Focus on semantics, not categories}
Rich categorical picture.
Maintain a computer science connection.

\textbf{Convey semantic intuition}
Obscured by mathematical apparatuses in literature.
Offer vantage points.
Secret to success: prerequisites

‘Category Theory and Logic’ module:

- categories
- products and equational logic
- exponentials, typed λ-calculus and CCCs
- functors
- naturality
- presheaves
- Yoneda
- pullbacks
- adjunctions

No domain theory!
as not taught everywhere :(
Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Starting point

Simply-typed \( \lambda \)-calculus with sum types

Semantic concepts
- Equational theory
- CBV Felleisen SOS
- Denotational semantics
- Adequacy proof

Rationale
- Mostly familiar
- Align baseline
- Modular logical relations

Categorical concepts
- Distributive categories, bi-CCCs
- A category for logical relations

\[
\begin{array}{cccc}
\text{Log}_F & & R_- & \rightarrow & \text{Pred} \\
\downarrow & & & & \downarrow \text{cod} \\
\mid - \mid & & & & \downarrow \\
C & & F & \rightarrow & \text{Set}
\end{array}
\]
Syllabus

1. Pure λ-calculus
2. Moggi’s λ_c
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Moggi’s $\lambda_c$

Semantic concepts
- Failure of equational theory
- Adequacy and the mono requirement
- Lack of general SOS

Rationale
- Most have heard about Moggi/monads
- First brush against open problems

Categorical concepts
- Strong monads
- Lifting of a monad
- Hermida’s lifting

Ohad Kammar <ohad.kammar@cl.cam.ac.uk>
Bird’s eye

Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Semantic concepts

- Computational models
  - exceptions
  - non-determinism
  - mnemoids

Mathematical concepts

- Review eq. logic
- Universal algebra
- Free model monad
- Unranked monads: Powerset, continuations

Rationale

- Effects as algebraic operations
- Algebraic manipulation of monads
- Limitations (rank)

Ohad Kammar <ohad.kammar@cl.cam.ac.uk>

A syllabus for algebraic effects
Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Abstract device driver interaction

Semantic concepts

- Interface: \(\text{lookup} : |\text{State}|\), \(\text{act}_m : 1\)

- Equations:

\[
\begin{align*}
\text{act}_{m_1} \quad | 
\text{act}_{m_1} \cdot m_2 \\
\text{act}_{m_2} &= \quad | \\
&\quad \quad \quad \quad \quad \quad x
\end{align*}
\]

- How to choose the right monad?

Mathematical concepts

- Hilbert-Post completeness

- Monad calculation

\[\prod_{s \in \text{State}} sM \times -\]

- Monoid actions + orbits as abstract automata

Rationale

- Non-obvious monad

- Open problem: model construction
Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Algebraic language design

\[ \lambda_{alg} : \text{Algebraic lambda calculus} \]

**Semantics concepts**

- \[ \lambda_c + \text{Kleisli arrows} \]
  \[ a \rightarrow Tb \]
- A closed language
- No SOS still

**Categorical concepts**

- Mention \( \top\top\)-lifting
  
  \[ \text{[Katsumata'05,'11]} \]
- Algebraic lifting
  
  \[ \text{[Kammar’14]} \]
- Generic effects and alg. operations \( (TX)^b \rightarrow (TX)^a \)

**Rationale**

- Semantically motivate (continuation-based) alg. operations
- General metalanguage for effects

Ohad Kammar <ohad.kammar@cl.cam.ac.uk>
Syllabus

1. Pure \(\lambda\)-calculus
2. Moggi’s \(\lambda_c\)
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Algebraic effect combination

Sum and tensor

Semantic concepts
- Modular model/program construction
- Monad transformers composition order
- Graph tool

Mathematical concepts
- Monads don’t compose, e.g.:

\[ \left( (1 + 1) \times (-) \right) \circ (X \mapsto 1) \]

is NaM (ta Conor)

Rationale
- Still an open problem
- Haskell-relevant

Ohad Kammar <ohad.kammar@cl.cam.ac.uk>
Bird’s eye

Syllabus

1. Pure \(\lambda\)-calculus
2. Moggi’s \(\lambda_c\)
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers

Ohad Kammar <ohad.kammar@cl.cam.ac.uk>

A syllabus for algebraic effects
Model analysis

Type-and-effect systems

Semantic concepts
- Syntax and semantics
- Model generation
- Compiler transformation validation (soundness and completeness)

Mathematical concepts
- Monad morphisms
- Conservative extension/restriction
- Application to algebraic lifting

Rationale
- Solve an open problem
- Application area outside den. sem.
- For programmable handlers
Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Semantics for effect handlers

Semantic concepts

- 'handle' is not an alg. op.
- $\lambda_{alg} +$ fixed set of handlers
- equational laws for handlers
  [Plotkin & Pretnar’09]

Categorical concepts

- Algebras and homomorphisms for a monad

Rationale

- Incorporate exception handlers
- Handle non-free effects
- Possible for unranked monads
Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Tying it all up

Programmable handlers

- $\lambda_{\text{eff}}$: user-defined alg. effects and handlers
- operational and denotational semantics
- programming examples

Rationale

- Synthesises:
  - (free) theories
  - effect systems
  - effect handlers,
  - algebraic lifting (for adequacy)
- “Hot” and active research topic
Conclusion

▶ A graduate-level syllabus
▶ Gateway to more advanced mathematical concepts
▶ Fits in half a lecture course (9 lectures), can co-exist with broader context (e.g., recursive domain equations).
▶ Inconclusive success (still under development)

Further work

▶ Course material, e.g.: lecture notes exercises
▶ Pedagogy
Syllabus

1. Pure $\lambda$-calculus
2. Moggi’s $\lambda_c$
3. Equational logic, universal algebra, and monads
4. Model construction
5. Language design
6. Effect combination
7. Type-and-effect systems
8. Effect handlers
9. Programmable handlers
Images

- http://cmseducation.org/syllabus/images/syllabus.gif