Lecture 2: Types and Effects

Gordon Plotkin

Laboratory for the Foundations of Computer Science, School of Informatics,
University of Edinburgh

NII Shonan Meeting No. 146
Programming and Reasoning with Algebraic Effects and Effect Handlers

An example type and effect system

- Have a finite set Loc of boolean locations in memory, divided into finitely many regions: $\mathrm{Loc} = \bigcup_{r \in R} \mathrm{Loc}_r$.
- The set of *effects* is:

$$\mathrm{Eff} =_{\mathrm{def}} \{\mathrm{update}_r \mid r \in R\} \cup \{\mathrm{lookup}_r \mid r \in R\}$$

- Effect typings have the form

$$\Gamma \vdash M : \sigma ! \alpha$$

where M is an effect-annotated term, and $\alpha \subseteq_{\mathrm{fin}} \mathrm{Eff}$

Effect types and effect annotated terms

Raw Syntax

Types $\quad \sigma ::= \text{bool} \mid \sigma \xrightarrow{\alpha} \tau \quad (\alpha \subseteq_{\text{fin}} \text{Eff})$

Terms $\quad M ::= x \mid \lambda x : \sigma.\, M \mid MN \mid$
$\text{true} \mid \text{false} \mid \text{if } L \text{ then } M \text{ else } N \mid$
$\mid l := M \; (l \in \text{Loc}) \mid !l \; (l \in \text{Loc})$

Typing

Environments $\quad \Gamma ::= x_1 : \sigma_1, \ldots, x_n : \sigma_n$

Judgments $\quad \Gamma \vdash M : \sigma ! \alpha$

Effect typing rules

$$\Gamma \vdash x : \sigma ! \emptyset \quad (x : \sigma \in \Gamma)$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau ! \alpha}{\Gamma \vdash \lambda x : \sigma . M : (\sigma \xrightarrow{\alpha} \tau) ! \emptyset}$$

$$\frac{\Gamma \vdash M : (\sigma \xrightarrow{\alpha} \tau) ! \beta \quad \Gamma \vdash N : \sigma ! \gamma}{\Gamma \vdash MN : \tau ! (\alpha \cup \beta \cup \gamma)}$$

$$\frac{\Gamma \vdash M : \mathrm{bool} ! \alpha}{\Gamma \vdash l := M : \mathrm{com} ! (\alpha \cup \{\mathrm{update}_r\})} \quad (l \in \mathrm{Loc}_r)$$

$$\Gamma \vdash ! l : \mathrm{bool} ! \{\mathrm{lookup}_r\} \quad (l \in \mathrm{Loc}_r)$$

Semantics of effect-annotated terms

- Standard (call-by-value) monadic semantics of terms $\Gamma \vdash t : \sigma$, without effect annotations has form:

$$[\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash t:\sigma]\!]} T([\![\sigma]\!])$$

- Wadler's suggestion: semantics of $\Gamma \vdash t : \sigma!\alpha$, with effect annotations should have form:

$$[\![\Gamma]\!] \xrightarrow{[\![\Gamma \vdash t:\sigma!\alpha]\!]} T_\alpha([\![\sigma]\!])$$

  for a collection of monads T_α connected by monad morphisms:

$$T_\alpha(X) \xrightarrow{m_{\alpha,\beta}(X)} T_\beta(X) \quad (\alpha \subseteq \beta)$$

- Where do such collections of monads and monad morphisms come from?

Idea I: Effects are given by (sets of) operations

- Signature for state:

$$\mathrm{update}_{l,b} : 1 \ (l \in \mathrm{Loc}, b \in \mathbb{T}) \quad \mathrm{lookup}_l : 2 \ (l \in \mathrm{Loc})$$

- Effects as (Sets of) operations of algebraic signature:

$$\begin{array}{lcl} \mathit{ops}(\mathrm{update}_r) & = & \{\mathrm{update}_{l,b} \mid l \in \mathrm{Loc}_r, b \in \mathbb{T}\} \\ \mathit{ops}(\mathrm{lookup}_r) & = & \{\mathrm{lookup}_l \mid l \in \mathrm{Loc}_r\} \\ \mathit{ops}(\{e_1, \ldots, e_n\}) & = & \mathit{ops}(e_1) \cup \ldots \cup \mathit{ops}(e_n) \end{array}$$

Below we identify α and $\mathit{ops}(\alpha)$.

Idea II: T_α is a restriction of T

- Conservative Restriction

$$\mathrm{Ax}^c_\alpha = \{t = u \mid \mathrm{Ax} \vdash t = u \text{ and } t, u \ \alpha\text{-terms}\}$$

- Axiomatic Restriction

$$\mathrm{Ax}^a_\alpha = \{t = u \mid t = u \in \mathrm{Ax} \text{ and } t, u \ \alpha\text{-terms}\}$$

- For $\alpha \subseteq \beta$, get theory inclusions

$$\mathrm{Ax}^c_\alpha \subseteq \mathrm{Ax}^c_\beta \subseteq \mathrm{Ax} \quad \text{and} \quad \mathrm{Ax}^a_\alpha \subseteq \mathrm{Ax}^a_\beta \subseteq \mathrm{Ax}$$

and so, as we will see:

- for $\alpha \subseteq \beta$, get monad morphisms

$$T_{\mathrm{Ax}^c_\alpha}(X) \to T_{\mathrm{Ax}^c_\beta}(X) \to T \quad \text{and} \quad T_{\mathrm{Ax}^a_\alpha}(X) \to T_{\mathrm{Ax}^a_\beta}(X) \to T$$

Axioms and Monad for Read-Only State

Axioms: Ax_r

$$\mathrm{lookup}_l(x, x) = x$$

$$\mathrm{lookup}_l(\mathrm{lookup}_l(w, x), \mathrm{lookup}_l(y, z)) = \mathrm{lookup}_l(w, z)$$

$$\mathrm{lookup}_l(\mathrm{lookup}_{l'}(w, x), \mathrm{lookup}_{l'}(y, z)) = \mathrm{lookup}_{l'}(\mathrm{lookup}_l(w, y), \mathrm{lookup}_l(x, z))$$

Monad

$$T_r(X) = X^S$$

where $S = \mathbb{T}^{\mathrm{Loc}}$

Axioms and Monad for Write-Only State:

Axioms: Ax_w

$$\mathrm{update}_{l,b}(\mathrm{update}_{l,b'}(x)) = \mathrm{update}_{l,b'}(x)$$

$$\mathrm{update}_{l,b}(\mathrm{update}_{l',b'}(x)) = \mathrm{update}_{l',b'}(\mathrm{update}_{l,b}(x)) \quad (l \neq l')$$

Monad

$$T_w(X) = S_w \times X$$

where $S_w = (\mathbb{1} + \mathbb{T})^{\mathrm{Loc}}$

Axioms and Monad for State

Axioms: Ax_{rw} These are $\mathrm{Ax}_r \cup \mathrm{Ax}_w$ plus:

$$\mathrm{lookup}_l(\mathrm{update}_{l,\mathrm{true}}(x), \mathrm{update}_{l,\mathrm{false}}(y)) = \mathrm{lookup}_l(x, y)$$

$$\mathrm{update}_{l,\mathrm{true}}(\mathrm{lookup}_l(x, y)) = \mathrm{update}_{l,\mathrm{true}}(x)$$

$$\mathrm{update}_{l,\mathrm{false}}(\mathrm{lookup}_l(x, y)) = \mathrm{update}_{l,\mathrm{false}}(y)$$

$$\mathrm{update}_{l,b}(\mathrm{lookup}_{l'}(x, y)) = \mathrm{lookup}_{l'}(\mathrm{update}_{l,b}(x), \mathrm{update}_{l,b}(y)) \quad (l \neq l')$$

Monad

$$T_{rw}(X) = (S \times X)^S$$

where $S = \mathbb{T}^{\mathrm{Loc}}$

Example: Conservative restrictions of Ax_{rw}.

In case of state for finitely many boolean locations, have Ax^{a}_{α} generates Ax^{c}_{α} for $\alpha \subseteq_{\mathrm{fin}} \mathrm{Op}$, so monads are the same.

- not generally true of course
- true in all cases at hand not involving nondeterminism.

Translations between presentations

- A signature translation $\Sigma \xrightarrow{\tau} \Sigma'$ is an assignment

$$\mathrm{op} : n \in \Sigma \mapsto \tau(\mathrm{op}) \in \Sigma'\text{-terms}$$

  where $\mathrm{Var}(\tau(\mathrm{op})) = \{z_0, \ldots, z_{n-1}\}$.
- Translating Σ-terms to Σ'-terms:

$$\begin{array}{lcl} x^\tau & = & x \\ \mathrm{op}(t_0, \ldots, t_{n-1})^\tau & = & \tau(\mathrm{op})[t_0^\tau / z_0, \ldots, t_{n-1}^\tau / z_{n-1}] \end{array}$$

- A presentation translation $(\Sigma, \mathrm{Ax}) \xrightarrow{\tau} (\Sigma', \mathrm{Ax}')$ is a signature translation $\Sigma \xrightarrow{\tau} \Sigma'$ such that:

$$\mathrm{Ax} \vdash t = u \quad \Rightarrow \quad \mathrm{Ax}' \vdash t^\tau = u^\tau$$

  It is conservative if

$$\mathrm{Ax} \vdash t = u \quad \Leftrightarrow \quad \mathrm{Ax}' \vdash t^\tau = u^\tau$$

From translations to monad morphisms

- Recall that, given a presentation (Σ, Ax), we get a monad

$$T_{\mathrm{Ax}}(X) =_{\mathrm{def}} \{[t]_{\mathrm{Ax}} \mid t \text{ is a term with variables in } X\}$$

- Then, given a translation $(\Sigma, \mathrm{Ax}) \xrightarrow{\tau} (\Sigma', \mathrm{Ax}')$, we get a monad morphism

$$T_{\mathrm{Ax}}(X) \xrightarrow{\rho X} T_{\mathrm{Ax}'}(X) \quad (X \in \mathbf{Set})$$

where

$$\rho^{\tau}_X([t]_{\mathrm{Ax}}) = [t^{\tau}]_{\mathrm{Ax}'}$$

- Further, ρ^{τ}_X is 1-1 for all sets X iff τ is conservative

Special case for effect systems

- The signatures are $\alpha \subseteq \beta$
- Get inclusion signature translation $\alpha \xrightarrow{\iota} \beta$ where

$$\iota = \mathrm{op} : n \in \alpha \mapsto \mathrm{op}(z_1, \ldots, z_n) \in \beta\text{-terms}$$

- Translating α-terms to β-terms:

$$t^{\iota} = t$$

- Axiomatic case Here $\mathrm{Ax}^a_\beta \subseteq \mathrm{Ax}^a_\beta$ and ι is a presentation translation, as

$$\mathrm{Ax}^a_\alpha \vdash t = u \quad \Rightarrow \quad \mathrm{Ax}^a_\beta \vdash t = u$$

- Conservative case Here $\mathrm{Ax}^c_\beta \subseteq \mathrm{Ax}^c_\beta$ and ι is a conservative translation as

$$\mathrm{Ax}^a_\alpha \vdash t = u \quad \Leftrightarrow \quad (\mathrm{Ax} \vdash t = u) \quad \Leftrightarrow \quad \mathrm{Ax}^a_\beta \vdash t = u$$

Special case for effect systems (cntnd)

- The monad morphism $T_{\mathrm{Ax}_\alpha}(X) \xrightarrow{\rho X} T_{\mathrm{Ax}_\beta}(X)$ is:

$$\rho X([t]_{\mathrm{Ax}_\alpha}) = [t]_{\mathrm{Ax}_\beta}$$

where Ax_ρ is Ax^a_ρ or Ax^c_ρ.

A slight digression: equivalence of presentations

Composing translations

Given $(\Sigma, \mathrm{Ax}) \xrightarrow{\tau} (\Sigma', \mathrm{Ax}') \xrightarrow{\tau'} (\Sigma, \mathrm{Ax}'')$

Define by: $(\Sigma, \mathrm{Ax}) \xrightarrow{\tau' \circ \tau} (\Sigma, \mathrm{Ax}'')$

$$\tau' \circ \tau(\mathrm{op}) = \tau(\mathrm{op})^{\tau'}$$

Equivalence of presentations

(Σ', Ax') and (Σ', Ax') are equivalent if there are translations

$$(\Sigma, \mathrm{Ax}) \xrightarrow{\tau} (\Sigma', \mathrm{Ax}') \xrightarrow{\tau'} (\Sigma, \mathrm{Ax})$$

such that

$$\mathrm{Ax} \vdash \tau' \circ \tau(\mathrm{op}) = \mathrm{op}(z_1, \ldots, z_n) \quad (\mathrm{op} \in \Sigma)$$

$$\mathrm{Ax}' \vdash \tau \circ \tau(\mathrm{op}') = \mathrm{op}'(z_1, \ldots, z_n) \quad (\mathrm{op}' \in \Sigma')$$

A somewhat general formulation of effect systems

Fix an algebraic presentation (Σ, Ax) and set $\mathrm{Eff} = \mathcal{F}(\mathrm{Op})$

Types $\quad \sigma ::= \mathrm{b} \mid \sigma \xrightarrow{\alpha} \tau \quad (\alpha \subseteq_{\mathrm{fin}} \mathrm{Eff})$

Terms $\quad M ::= x \mid \lambda x : \sigma. M \mid MN \mid \mathrm{coerce}_{\alpha,\beta}(M) \mid \mathrm{op}(M_1, \ldots, M_n) \quad (\mathrm{op} : n) \mid \ldots$

Form of effect typing rules

$$\Gamma \vdash M : \sigma ! \alpha$$

Form of semantics

$$\frac{\Gamma \vdash M : \sigma ! \alpha}{[\![\Gamma]\!] \xrightarrow{[\![M]\!]} T_{\mathrm{Ax}_\alpha}([\![\sigma]\!])}$$

Effect Typing & Algebraic semantics: Variables and Abstraction

Typing

$$x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash x_i : \sigma_i ! \emptyset$$

Semantics

$$[\![x]\!](a_1, \ldots, a_n) = [a_i]_{Ax_\emptyset}$$

Typing

$$\frac{\Gamma, x : \sigma \vdash M : \tau ! \alpha}{\Gamma \vdash \lambda x : \sigma. M : (\sigma \xrightarrow{\alpha} \tau) ! \emptyset}$$

Semantics

$$[\![\lambda x : \sigma. M]\!](\gamma) = [a \in [\![\sigma]\!] \mapsto [\![M]\!](\gamma, a)]_{Ax_\emptyset}$$

Algebraic semantics: Application

Typing

$$\frac{\Gamma \vdash M : (\sigma \xrightarrow{\alpha} \tau)!\beta \quad \Gamma \vdash N : \sigma!\gamma}{\Gamma \vdash MN : \tau!(\alpha \cup \beta \cup \gamma)}$$

Semantics

Suppose

$$[\![M]\!](\gamma) = [t(f_1, \ldots, f_m)]_{\mathrm{Ax}_\beta}$$
$$[\![N]\!](\gamma) = [u(a_1, \ldots, a_n)]_{\mathrm{Ax}_\gamma}$$
$$f_i(a_j) = [v_{ij}]_{\mathrm{Ax}_\alpha}$$

Then

$$[\![MN]\!](\gamma) = [t(u(v_{11}, \ldots, v_{1n}), \ldots, u(v_{m1}, \ldots, v_{mn}))]_{\mathrm{Ax}_{\alpha\cup\beta\cup\gamma}}$$

Algebraic semantics: Effects

Typing

$$\frac{\Gamma \vdash M_i : \sigma!\alpha_i}{\Gamma \vdash \mathrm{op}(M_1, \ldots, M_n) : \sigma!(\{\mathrm{op}\} \cup \alpha_1 \cup \ldots \cup \alpha_n)}$$

Semantics

Suppose

$$[\![M_i]\!](\gamma) = [t_i]_{\mathrm{Ax}_{\alpha_i}}$$

Then

$$[\![\mathrm{op}(M_1, \ldots, M_n)]\!] = [\mathrm{op}(t_1, \ldots, t_n)]_{\mathrm{Ax}_{(\{\mathrm{op}\} \cup \alpha_1 \cup \ldots \cup \alpha_n)}}$$

Algebraic Effects: Coercion

Typing

$$\frac{\Gamma \vdash M : \sigma ! \alpha}{\Gamma \vdash \mathrm{coerce}_{\alpha,\beta}(M) : \sigma ! \beta} \quad (\alpha \subseteq \beta)$$

Semantics

Suppose

$$[\![M]\!](\gamma) = [t]_{\mathrm{Ax}_\alpha}$$

Then

$$\mathrm{coerce}_{\alpha,\beta}([\![M]\!])(\gamma) = \rho_{[\![\sigma]\!]}([t]_{\mathrm{Ax}_\alpha}) = [t]_{\mathrm{Ax}_\beta}$$

The let construct

Definition

$$\text{let } x : \sigma \text{ be } M \text{ in } N = (\lambda x : \sigma.\, M)N$$

(also written as: M to $x : \sigma$ in N)

Typing

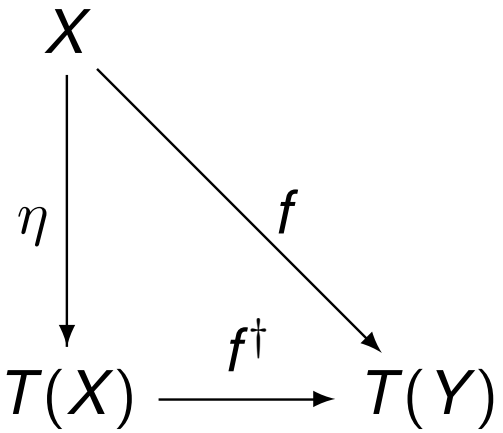
$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma, x : \sigma \vdash N : \tau}{\text{let } x : \sigma \text{ be } M \text{ in } N : \tau}$$

Semantics (Exceptions case)

$$[\![\text{let } x : \sigma \text{ be } M \text{ in } N]\!](\gamma) = \begin{cases} \text{inr}(e) & (\text{if } [\![M]\!](\gamma) = \text{inr}(e)) \\ [\![N]\!](\gamma, a) & (\text{if } [\![M]\!](\gamma) = \text{inl}(a)) \end{cases}$$

Kleisli lifting

Lifting



where $f^\dagger = \mu_{T(Y)} \circ T(f)$.

Kleisli lifting with parameters

$$\frac{X \times Y \xrightarrow{f} X}{X \times T(Y) \xrightarrow{\text{st}} T(X \times Y) \xrightarrow{f^\dagger} T(Z)}$$

Semantics of let

Typing

$$\frac{\Gamma \vdash M : \sigma \qquad \Gamma, x : \sigma \vdash N : \tau}{\Gamma \vdash \text{let } x : \sigma \text{ be } M \text{ in } N : \tau}$$

Categorical semantics

$$[\![\Gamma]\!] \xrightarrow{\Delta} [\![\Gamma]\!] \times [\![\Gamma]\!] \xrightarrow{\text{id} \times [\![M]\!]} [\![\Gamma]\!] \times [\![\sigma]\!] \xrightarrow{[\![N]\!]^\dagger} [\![\tau]\!]$$

Algebraic semantics

$$\frac{[\![M]\!](\gamma) = [t(a_1, \ldots, a_k)] \quad (a_i \in [\![\sigma]\!]) \qquad [\![N]\!](\gamma, a_i) = [u_i] \quad (i = 1, k)}{[\![\text{let } x : \sigma \text{ be } M \text{ in } N]\!](\gamma) = [t(u_1, \ldots, u_k)]}$$

Effect version

Effect typing

$$\frac{\Gamma \vdash M : \sigma ! \alpha \quad \Gamma, x : \sigma \vdash N : \tau ! \beta}{\text{let } x : \sigma \text{ be } M \text{ in } N : \tau ! (\alpha \cup \beta)}$$

Algebraic semantics

Suppose

$$\begin{array}{lcll} [\![M]\!](\gamma) & = & [t(a_1, \ldots, a_k)]_{\mathrm{Ax}_\alpha} & (a_i \in [\![\sigma]\!]) \\ [\![N]\!](\gamma, a_i) & = & [u_i]_{\mathrm{Ax}_\beta} & (i = 1, k) \end{array}$$

Then

$$[\![\text{let } x : \sigma \text{ be } M \text{ in } N]\!](\gamma) \quad = \quad [t(u_1, \ldots, u_k)]_{\mathrm{Ax}_{(\alpha \cup \beta)}}$$

Algebraic optimisations: Discard

Suppose that $\Gamma \vdash M : \sigma!\alpha$ and $\Gamma \vdash N : \tau!\beta$, with $\alpha \subseteq \beta$ then, if α is Ax_α-discardable:

$$\Gamma \models_{T_{\mathrm{Ax}_\alpha}} \text{let } x : \sigma \text{ be } M \text{ in } N = N$$

where discardability is that the only definable unary function is the identity, ie, whenever $\mathrm{Var}(t) = \{x\}$ is an α-term, then:

$$\mathrm{Ax}_\alpha \vdash t(x) = x$$

Equivalently

$$\mathrm{Ax}_\alpha \vdash \mathrm{op}(x, \ldots, x) = x \quad (\mathrm{op} : n)$$

Example discardable theories Reader, both forms of non-determinism.
Example nondiscardable theories Exceptions, writing.

Proof of validity of optimisation

We have

$$\begin{array}{lcll} [\![M]\!](\gamma) & = & [t(a_1,\ldots,a_k)]_{\mathrm{Ax}_\alpha} & (a_i \in [\![\sigma]\!]) \\ [\![N]\!](\gamma, a_i) & = & [u]_{\mathrm{Ax}_\beta} & \end{array}$$

So

$$\begin{array}{lcl} [\![\text{let } x : \sigma \text{ be } M \text{ in } N]\!](\gamma) & = & [t(u,\ldots,u)]_{\mathrm{Ax}_{(\alpha\cup\beta)}} \\ & = & [u]_{\mathrm{Ax}_\beta} \\ & = & [\![N]\!](\gamma) \end{array}$$

Algebraic optimisations: Copy

Suppose that $\Gamma \vdash M : \sigma!\alpha$ and $\Gamma, x : \sigma, y : \sigma \vdash N : \tau!\beta$, with $\alpha \subseteq \beta$ then, if α is Ax_α-copyable:

$$\Gamma \models_{T_{\mathrm{Ax}_\alpha}} \begin{array}{l} \text{let } x : \sigma \text{ be } M \text{ in } (\text{let } y : \sigma \text{ be } M \text{ in } N) = \\ \text{let } x : \sigma \text{ be } M \text{ in } N[x/y] \end{array}$$

where copyability is defined by, whenever $\mathrm{Var}(t) = \{x_1, \ldots, x_n\}$ is an α-term:

$$\mathrm{Ax}_\alpha \vdash t(t(x_{11}, \ldots, x_{1n}), \ldots, t(x_{n1}, ..., x_{nn})) = t(x_{11}, \ldots, x_{nn})$$

Example copyable theories Exceptions, read-only state, write-only state (proof: look at the normal forms)

Example non-copyable theories Nondeterminism, probabilistic nondeterminism, state (with both reading and writing!)

Proof of optimisation

Suppose

$$[\![M]\!](\gamma) = [t(a_1, \ldots, a_k)]_{\mathrm{Ax}_\alpha}$$

$$[\![N]\!](\gamma, a_i, a_j) = [u_{ij}]_{\mathrm{Ax}_\beta} \quad (i, j = 1, n))$$

Then

$$[\![\text{let } x : \sigma \text{ be } M \text{ in } (\text{let } y : \sigma \text{ be } M \text{ in } N)]\!](\gamma)$$

$$= [t(t(u_{11}, \ldots, u_{1n}), \ldots, t(u_{n1}, \ldots, u_{nn}))]_{\mathrm{Ax}_\beta}$$

$$= [t(u_{11}, \ldots, u_{nn})]_{\mathrm{Ax}_\beta}$$

$$= [\![\text{let } x : \sigma \text{ be } M \text{ in } N[y/x]]\!](\gamma)$$

Algebraic optimisations: Permutation

Suppose $\Gamma \vdash L : \sigma!\alpha$, $\Gamma \vdash M : \tau!\beta$ and $\Gamma, x : \sigma, y : \sigma \vdash N : \tau!\beta$, with $\alpha, \beta \subseteq \rho$ then, if α, β are $\mathrm{Ax}_\alpha, \mathrm{Ax}_\beta$-permutable:

$$\Gamma \models_{T_{\mathrm{Ax}_\rho}} \begin{array}{l} \text{let } x : \sigma \text{ be } L \text{ in } (\text{let } y : \tau \text{ be } M \text{ in } N) = \\ \text{let } y : \tau \text{ be } M \text{ in } (\text{let } x : \sigma \text{ be } L \text{ in } N) \end{array}$$

where permutability is defined by, whenever $\mathrm{Var}(t) = \{x_1, \ldots, x_m\}$ is an α term and $\mathrm{Var}(u) = \{y_1, \ldots, y_m\}$ is a β term, then:

$$\mathrm{Ax}_{\alpha\cup\beta} \vdash \begin{array}{l} t(u(x_{11}, \ldots, x_{1n}), \ldots, u(x_{m1}, ..., x_{mn})) \\ = u(t(x_{11}, \ldots, x_{m1}), \ldots, t(x_{1n}, ..., x_{mn})) \end{array}$$

Equivalently, just the operations.

Example permutable theories Distinct memory locations; state and nondeterminism.

Example non-permutable theories Reading and writing the same location; state and exceptions.

Justification of optimisations

Theorem

Suppose $\Gamma \vdash M : \mathbf{b}!\alpha$ *and* $\Gamma \vdash N : \mathbf{b}!\alpha$ *where* $\Gamma = x_1 : \mathbf{b}_1, \ldots, x_n : \mathbf{b}_n$ *and* $\mathbf{b}$ *and the* $\mathbf{b}_i$ *are all ground. Then:*

1. $|\Gamma| \models_{\mathrm{Ax}} |M| = |N| \quad \Leftarrow \quad \Gamma \models_{T_{\mathrm{Ax}^a_\alpha}} M = N$
2. $|\Gamma| \models_{\mathrm{Ax}} |M| = |N| \quad \textit{iff} \quad \Gamma \models_{T_{\mathrm{Ax}^c_\alpha}} M = N$

So can use equations between annotated terms that are true in effect models to optimise unannotated programs.

Note: Theorem is false if, e.g., Γ is allowed to have first, or higher, order variables, as guarantees on function applications are lost on the left, e.g.:

$$f : \mathrm{unit} \xrightarrow{\alpha} \mathrm{unit} \vdash \mathrm{let}\ x : \mathrm{unit}\ \mathrm{be}\ f(*)\ \mathrm{in}\ * = *$$

Interesting re separate compilation

Modularity (examples)

- Idea Build-up axioms/theories/monads for sets of effects by simple operations on axioms/theories/monads for smaller sets of effects.
- Sum of Presentations Given $(\Sigma_1, \mathrm{Ax}_1)$, $(\Sigma_2, \mathrm{Ax}_1)$ their *sum* is the evident disjoint union, $(\Sigma_1 + \Sigma_2, \mathrm{Ax}_1+, \mathrm{Ax}_2)$.
- If Ax_1, Ax_2 are discardable, so is $\mathrm{Ax}_1 + \mathrm{Ax}_2$.
- If both Ax_1 and Ax_2 are permutable with Ax_3, then so is $\mathrm{Ax}_1 + \mathrm{Ax}_2$.
- If $\alpha_1 \subseteq \Sigma_1$ and $\alpha_2 \subseteq \Sigma_2$ then

$$(\mathrm{Ax}_1 + \mathrm{Ax}_2)^a_{\alpha+\beta} = (\mathrm{Ax}_1)^a_\alpha + (\mathrm{Ax}_2)^a_\beta$$

$$(\mathrm{Ax}_1 + \mathrm{Ax}_2)^c_{\alpha+\beta} = ((\mathrm{Ax}_1)^c_\alpha + (\mathrm{Ax}_2)^c_\beta)^*$$

Exercise 8

1. Show that in the case of state for finitely many boolean locations $(\mathrm{Ax}_{rw})^c_{\mathrm{lookup}}$ is the deductive closure of Ax_r, where $\mathrm{lookup} = \{\mathrm{lookup}_l | l \in \mathrm{Loc}\}$.

2. Show that in the case of state for finitely many boolean locations $(\mathrm{Ax}_{rw})^c_{\mathrm{update}}$ is the deductive closure of Ax_w,where $\mathrm{update} = \{\mathrm{update}_{l,b} | l \in \mathrm{Loc}, b \in \mathbb{T}\}$.

Exercise 9: Presentation translations and monad morphisms

1. Show that, as claimed, ρ^{τ}_{X} is 1-1 for all sets X iff τ is conservative
2. Show that ρ^{τ} is actually a bijection between presentation translations and monad morphisms. What is its inverse?
3. (For the particularly categorically minded) Define a category of axiomatic presentations and (equivalence classes of) translations. Show that ρ^{τ} is a fully faithful functor. Perhaps go on to show it is an equivalence of categories with the category of finitary monads. (Both are equivalent to the category of Lawvere theories.)

Exercise 10: Type and effect systems

1. Give a type and effect system for the language of Exercise 1. Give its algebraic semantics.
2. Unique Typing For the “reasonably general" system given above, or for your system of part 1 of this exercise, prove that, given Γ and M there is at most one pair σ, α such that $\Gamma \vdash M : \sigma ! \alpha$.
3. Explicit coercion is annoying. Remove it (e.g. from the “reasonably general" system given above) and give an alternative system with subtyping and subeffecting. This system does not have unique typing. Show that if $\Gamma \vdash M : \sigma ! \alpha$ then there is a term M^+ of the previous system such that $\Gamma \vdash M^+ : \sigma ! \alpha$ and M is obtained from M^+ by removing coercions and certain natural maps corresponding to subtyping. (There should be a better way to say this.)

Exercise 11: Optimisations I

1. Discard Various example theories were claimed to be discardable or non-discardable. Establish these claims (with proofs and counterexamples).
2. Copy Various example theories were claimed to be copyable or non-copycardable. Establish these claims (with proofs and counterexamples).
3. Permutation Various example pairs of theories were claimed to be permutable or non-permutable. Establish these claims (with proofs and counterexamples).

Exercise 12: Optimisations II

1. Discard It is claimed above that discardability can be equivalently formulated in terms of just the operations. Prove this.
2. Copy It is claimed above that copyability cannot be equivalently similarly formulated in terms of just the operations. Prove this. [Hint: look at the counterexample from the previous exercise.]
3. Permutation It is claimed above that permutability can be equivalently formulated in terms of just the operations. Prove this.