Coeffect Systems and Typing (Preliminary talk)

Tomas Petricek and Dominic Orchard

University of Cambridge

[tp322,dao29]@cam.ac.uk

October 11, 2011

Introducing effect and **co**effect systems

Effect systems

Typing judgement for effect systems

$$\Gamma \vdash e : \tau, F$$

- Expression e has type τ
- And performs effects F

Example: Accessing reference cells

Construct with effects

- Information about effects is lost

$$\frac{r : \mathsf{ref}_p \tau \in \Gamma \qquad \Gamma \vdash e : \tau}{\Gamma \vdash r \mathtt{:=} e : \mathtt{unit}}$$

Composing expressions

$$\frac{\Gamma \vdash e_1 : \tau_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau_2}{\Gamma \vdash \mathtt{let}\ x = e_1\ \mathtt{in}\ e_2 : \tau_2}$$

Example: Accessing reference cells

Construct with effects

- Typing rules track effects too!

$$\frac{r : \mathrm{ref}_p\tau \in \Gamma \qquad \Gamma \vdash e : \tau, \color{red}{F}}{\Gamma \vdash r\,\texttt{:=}e : \texttt{unit}, \color{red}{F \cup \{!p\}}}$$

Example: Accessing reference cells

Construct with effects

- Typing rules track effects too!

$$\frac{r : \mathsf{ref}_p \tau \in \Gamma \qquad \Gamma \vdash e : \tau, F}{\Gamma \vdash r \texttt{:=} e : \texttt{unit}, F \cup \{!p\}}$$

Composing expressions

$$\frac{\Gamma \vdash e_1 : \tau_1, F_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau_2, F_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \tau_2, F_1 \cup F_2}$$

Coeffect systems

Typing judgement has "capabilities" or "constraints"

$$\Gamma, C \vdash e : \tau$$

- Expression e has type τ
- In context specified by C

Example: Distributed programming

Modalities: `server`, `browser`, `phone`

$$\Gamma, \{\mathit{phone}, \mathit{server}\} \vdash \texttt{access}\ \mathit{network} : \mathit{int}$$
$$\Gamma, \{\mathit{browser}, \mathit{phone}\} \vdash \texttt{access}\ \mathit{input} : \mathit{int}$$

Composing expressions

$$\frac{\Gamma, C_1 \vdash e_1 : \tau_1 \qquad \Gamma, x, C_2 : \tau_1 \vdash e_2 : \tau_2}{\Gamma, C_1 \cap C_2 \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \tau_2}$$

Effects and coeffects

Effect systems

- Tracked as part of the result
- Propagate forward
- Reflected in the overall result

Coeffect systems

- Tracked as part of the context
- Propagate backward
- Reflected in the overall input

Tracking effects with monads

Monads

- Functions with structure on result: $\tau_1 \longrightarrow M\tau_2$
- Type $M\tau$ captures the effects

Monads

- Functions with structure on result: $\tau_1 \rightarrow M\tau_2$
- Type $M\tau$ captures the effects

Composition of functions

Compose functions and combine effects

$$(\tau_1 \rightarrow M\tau_2) \rightarrow (\tau_2 \rightarrow M\tau_3) \rightarrow (\tau_1 \rightarrow M\tau_3)$$

Monads

- Functions with structure on result: $\tau_1 \rightarrow M\tau_2$
- Type $M\tau$ captures the effects

Composition of functions

Compose functions and combine effects

$$(\tau_1 \rightarrow M\tau_2) \rightarrow (\tau_2 \rightarrow M\tau_3) \rightarrow (\tau_1 \rightarrow M\tau_3)$$

Unit of a monad

Add empty effect to a value

$$(\tau \rightarrow M\tau)$$

Tagged monads

- Functions with structure on result: $\tau_1 \rightarrow M^r \tau_2$
- Type $M^r \tau$ captures effects r

Composition of functions

Compose functions and combine effects

$$(\tau_1 \rightarrow M^r \tau_2) \rightarrow (\tau_2 \rightarrow M^s \tau_3) \rightarrow (\tau_1 \rightarrow M^{r \cup s} \tau_3)$$

Unit of a monad

Add empty effect to a value

$$(\tau \rightarrow M^{\emptyset} \tau)$$

Marriage of effects and monads

Use tagged monads to capture effects

$$\frac{r : \mathrm{ref}_p\tau \in \Gamma \qquad \Gamma \vdash e : M^r\tau}{\Gamma \vdash r \mathtt{:=} e : M^{r \cup \{!p\}}\mathtt{unit}}$$

Marriage of effects and monads

Use tagged monads to capture effects

$$\frac{r : \mathrm{ref}_p \tau \in \Gamma \qquad \Gamma \vdash e : M^r \tau}{\Gamma \vdash r := e : M^{r \cup \{!p\}} \texttt{unit}}$$

Using monadic composition

$$\frac{\Gamma \vdash e_1 : \tau_1, F_1 \qquad \Gamma, x : \tau_1 \vdash e_2 : \tau_2, F_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \tau_2, F_1 \cup F_2}$$

$$\frac{\Gamma \vdash e_1 : M^r A \qquad \Gamma, x : A \vdash e_2 : M^s B}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : M^{r \cup s} B}$$

Marriage of effects and monads

Effectful functions become pure functions returning $M^r \tau$

$$\frac{\Gamma, x : \tau_1 \vdash e : \tau_2, F}{\Gamma \vdash \lambda x.e : \tau_1 \xrightarrow{F} \tau_2, \emptyset}$$

$$\frac{\Gamma, x : \tau_1 \vdash e : M^r \tau_2}{\Gamma \vdash \lambda x.e : M^{\emptyset}(\tau_1 \rightarrow M^r \tau_2)}$$

Monoid tagged monads

- Generalize tags to monoid $(X, \otimes, 1)$
- Previously $(\mathcal{P}(\mathcal{E}), \cup, \emptyset)$ monoid

Monoid tagged monads

- Generalize tags to monoid $(X, \otimes, 1)$
- Previously $(\mathcal{P}(\mathcal{E}), \cup, \emptyset)$ monoid

Composition of functions

Compose functions and combine effects

$$(\tau_1 \longrightarrow M^r \tau_2) \longrightarrow (\tau_2 \longrightarrow M^s \tau_3) \longrightarrow (\tau_1 \longrightarrow M^{r \otimes s} \tau_3)$$

Monoid tagged monads

- Generalize tags to monoid $(X, \otimes, 1)$
- Previously $(\mathcal{P}(\mathcal{E}), \cup, \emptyset)$ monoid

Composition of functions

Compose functions and combine effects

$$(\tau_1 \rightarrow M^r \tau_2) \rightarrow (\tau_2 \rightarrow M^s \tau_3) \rightarrow (\tau_1 \rightarrow M^{r \otimes s} \tau_3)$$

Unit of a monad

Add empty effect to a value

$$(\tau \rightarrow M^1 \tau)$$

Tracking context-dependence with comonads

Comonads

Monads: functions with structure on result:

$$\tau_1 \longrightarrow M\tau_2$$

Comonads: functions with structure on argument:

$$C\tau_1 \longrightarrow \tau_2$$

Comonads

Monads: functions with structure on result:

$$\tau_1 \to M\tau_2$$

Comonads: functions with structure on argument:

$$C\tau_1 \to \tau_2$$

$C\tau$ captures "context-dependence"

- Value τ depends on extra information

$$C\tau = (time \times location) \to \tau$$

Comonads

Monads: functions with structure on result:

$$\tau_1 \rightarrow M\tau_2$$

Comonads: functions with structure on argument:

$$C\tau_1 \rightarrow \tau_2$$

$C\tau$ captures "context-dependence"

- Value τ depends on extra information

$$C\tau = (\mathit{time} \times \mathit{location}) \rightarrow \tau$$

- Argument τ paired with "implicit" parameter A:

$$C\tau = \tau \times A$$

Comonads

- Functions with structure on argument: $C\tau_1 \longrightarrow \tau_2$
- $C\tau$ captures "context-dependence"

Comonads

- Functions with structure on argument: $C\tau_1 \to \tau_2$
- $C\tau$ captures "context-dependence"

Composition of functions

Compose functions, propogating context-dependence

$$(C\tau_1 \to \tau_2) \to (C\tau_2 \to \tau_3) \to (C\tau_1 \to \tau_3)$$

Comonads

- Functions with structure on argument: $C\tau_1 \longrightarrow \tau_2$
- $C\tau$ captures "context-dependence"

Composition of functions

Compose functions, propogating context-dependence

$$(C\tau_1 \longrightarrow \tau_2) \longrightarrow (C\tau_2 \longrightarrow \tau_3) \longrightarrow (C\tau_1 \longrightarrow \tau_3)$$

Unit of a comonad

Extract value at some "current" context

$$(C\tau \longrightarrow \tau)$$

Tagged comonads

- Functions with structure on argument: $C^r \tau_1 \to \tau_2$
- Monoid $(X, \otimes, 1)$ specifies "what" context

Composition of functions

Compose functions, propogating context-dependence

$$(C^r \tau_1 \to \tau_2) \to (C^s \tau_2 \to \tau_3) \to (C^{r \otimes s} \tau_1 \to \tau_3)$$

Unit of a comonad

Extract value at some "current" context

$$(C^1 \tau \to \tau)$$

Tagged comonads vs. tagged monads

Monads propogate effects forwards

$$(\tau_1 \to M^r \tau_2) \to (\tau_2 \to M^s \tau_3) \to (\tau_1 \to M^{r \otimes s} \tau_3)$$

Comonads propogate coeffects backwards

$$(C^r \tau_1 \to \tau_2) \to (C^s \tau_2 \to \tau_3) \to (C^{r \otimes s} \tau_1 \to \tau_3)$$

Example: Distributed programming

Comonad tagged with sets of possible modalities

$$C^{\{phone,server\}}\Gamma \vdash \texttt{access}\ network : int$$
$$C^{\{browser,phone\}}\Gamma \vdash \texttt{access}\ input : int$$

Example: Distributed programming

Comonad tagged with sets of possible modalities

$$C^{\{phone,server\}}\Gamma \vdash \texttt{access}\ network : int$$
$$C^{\{browser,phone\}}\Gamma \vdash \texttt{access}\ input : int$$

Context-dependencies composed using comonads

$$\begin{array}{l} \texttt{let}\ hashInput = \lambda^{\circ}(). \\ \quad \texttt{let}\ i = \texttt{access}\ input \\ \quad \texttt{let}\ n = \texttt{access}\ network \\ \quad hash\ i\ n \end{array}$$

Input captures modality $hashInput : C^{\{phone\}} unit \longrightarrow int$

Marriage of **co**effects and **co**monads

Tagged comonads capture capabilities or context

$$C^{\{phone\}}\Gamma \vdash \texttt{access}\ gps : \mathit{float} \times \mathit{float}$$

Marriage of **co**effects and **co**monads

Tagged comonads capture capabilities or context

$$C^{\{phone\}}\Gamma \vdash \texttt{access}\ gps : float \times float$$

Using comonadic composition

$$\frac{\Gamma, F_1 \vdash e_1 : \tau_1 \qquad (\Gamma, x : \tau_1), F_2 \vdash e_2 : \tau_2}{\Gamma, F_1 \cap F_2 \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : \tau_2}$$

$$\frac{C^r\Gamma \vdash e_1 : A \qquad C^s(\Gamma, x : A) \vdash e_2 : B}{C^{r \otimes s}\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 : B}$$

Comandic coeffect typing

$$\frac{C^r\Gamma \vdash e_1 : (C^s A \rightarrow B) \qquad C^t\Gamma \vdash e_2 : A}{C^{r\otimes t\otimes s}\Gamma \vdash e_1 e_2 : B}$$

$$C^1\Gamma, x : A \vdash x : A$$

$$\frac{C^{r\otimes s}(\Gamma, x : A) \vdash e : B}{C^r\Gamma \vdash \lambda^\circ x.e : C^s A \rightarrow B}$$

Example: Implicit parameters

Configuration problem

- Configure function deep in a call tree
- Add parameter to nearly all functions?

$$\begin{array}{l}\texttt{let}\ \mathit{printString} = \lambda s.\\ \quad \texttt{if}\ \mathit{length}\ s > 78\ \texttt{then} \ldots\end{array}$$

$$\begin{array}{l}\texttt{let}\ \mathit{print} = \lambda \mathit{doc}.\\ \quad \ldots \mathit{printString} \ldots\end{array}$$

Example: Implicit parameters

Add *dynamically scoped* implicit parameters *?p*

$$\begin{array}{l}
\texttt{let}\ \mathit{prints} = \lambda^{\circ} s.\\
\quad \texttt{if}\ \mathit{length}\ s > \mathit{?width}\ \texttt{then} \ldots\\
\qquad \lambda^{\circ} \mathit{str}.\ \mathit{append}\ \mathit{str}\ s\ \mathit{?width}\ \mathit{?size}\\
\qquad\qquad \ldots
\end{array}$$

Example: Implicit parameters

Add *dynamically scoped* implicit parameters $?p$

$$\begin{array}{l} \texttt{let } \mathit{prints} = \lambda^{\circ} s. \\ \quad \texttt{if } \mathit{length}\ s > \mathit{?width}\ \texttt{then} \ldots \\ \quad\quad \lambda^{\circ} \mathit{str}.\ \mathit{append}\ \mathit{str}\ s\ \mathit{?width}\ \mathit{?size} \\ \quad\quad\quad \ldots \end{array}$$

$$\mathit{prints} : C^{\{\mathit{?width}:\mathit{int}\}}\mathit{string} \rightarrow (C^{\{\mathit{?size}:\mathit{int}\}}\mathit{string} \rightarrow \mathit{string})$$

Example: Implicit parameters

Add *dynamically scoped* implicit parameters $?p$

$$
\begin{array}{l}
\texttt{let}\ \mathit{prints} = \lambda^{\circ} s.\\
\quad \texttt{if}\ \mathit{length}\ s > \mathit{?width}\ \texttt{then}\ldots\\
\quad\quad \lambda^{\circ} \mathit{str}.\ \mathit{append}\ \mathit{str}\ s\ \mathit{?width}\ \mathit{?size}\\
\quad\quad\quad \ldots
\end{array}
$$

$$\mathit{prints} : C^{\{?width:int\}}\mathit{string} \rightarrow (C^{\{?size:int\}}\mathit{string} \rightarrow \mathit{string})$$

$$
\begin{array}{l}
\texttt{let}\ \mathit{pf} = \mathit{prints} \diamond \text{``}\mathit{world}\text{``}\ \texttt{with}\ \mathit{?width} = 78\\
\texttt{in}\ \mathit{pf} \diamond \text{``}\mathit{Hello}\text{``}\ \texttt{with}\ \mathit{?size} = 100
\end{array}
$$

Comparing monads and comonads

Comonadic abstraction captures current context

$$\frac{C^{r \otimes s}\Gamma, x : \tau_1 \vdash e : \tau_2}{C^s\Gamma \vdash \lambda x.e : C^r\tau_1 \rightarrow \tau_2}$$

Monadic abstraction is always in a pure context

$$\frac{\Gamma, x : \tau_1 \vdash e : M^r\tau_2}{\Gamma \vdash \lambda x.e : M^{\emptyset}(\tau_1 \rightarrow M^r\tau_2)}$$

- Compare implicit parameters and reader monad!
- Two variants of *comonadic* type system

Conclusion

- View context-dependent computations as *coeffects*
- Based on denotational semantics using comonads
- Related to modal logics for distributed computations
- Examples include: distributed programming, implicit parameters