Monads and More: Part 4

Tarmo Uustalu, Tallinn

Nottingham, 14-18 May 2007
Coeffectful computation and comonads

For coeffectful notions of computation, we have a comonad 
\((D, \varepsilon, \delta)\) on the base category \(C\) of pure functions such that
the category of impure functions is \(\text{CoKl}(D)\), i.e.,

- an impure function between object \(A\) and \(B\) of \(C\) can be viewed as a map \(A \rightarrow^D B\) of \(\text{CoKl}(D)\), i.e., a map \(DA \rightarrow B\) of \(C\),
- the identity impure functions are \(\text{id}^D =_{\text{df}} \varepsilon\),
- and the composition of impure functions is
  \(\ell \circ^D k =_{\text{df}} \ell \circ k^{\dagger}\).

Pure functions are a special case of impure functions via the inclusion
\(J : C \rightarrow \text{CoKl}(D)\), given by \(Jf =_{\text{df}} f \circ \varepsilon\).

Intuition: \(DA\) – values from \(A\) in a context.

Simplest example: \(DA =_{\text{df}} A \times E\) for dependency on
environment, but \(\text{CoKl}(D) \cong \text{KI}(T)\) for \(TA =_{\text{df}} E \Rightarrow A\).
Dataflow computations

Dataflow computation = discrete-time signal transformations = stream functions.

The output value at a time instant (stream position) is determined by the input value at the same instant (position) plus further input values.

Example dataflow programs

\[
\begin{align*}
pos &= 0 \ fby \ (pos + 1) \\
sum \ x &= x + (0 \ fby \ (sum \ x)) \\
fact &= 1 \ fby \ (fact \ \ast \ (pos + 1)) \\
fibo &= 0 \ fby \ (fibo + (1 \ fby \ fibo))
\end{align*}
\]

<table>
<thead>
<tr>
<th>pos</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>sum pos</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>6</td>
<td>10</td>
<td>15</td>
<td>21</td>
<td>...</td>
</tr>
<tr>
<td>fact</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>24</td>
<td>120</td>
<td>720</td>
<td>...</td>
</tr>
<tr>
<td>fibo</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td>8</td>
<td>...</td>
</tr>
</tbody>
</table>
We want to consider functions \( \text{Str} A \rightarrow \text{Str} B \) as impure functions from \( A \) to \( B \).

Streams are naturally isomorphic to functions from natural numbers: \( \text{Str}A =_{df} \nu X. A \times X \cong \text{Nat} \Rightarrow A \).

General stream functions \( \text{Str}A \rightarrow \text{Str}B \) are thus in natural bijection with maps \( \text{Str}A \times \text{Nat} \rightarrow B \).
Comonad for general stream functions

- **Functor:**
  \[ DA =_{df} (\text{Nat} \Rightarrow A) \times \text{Nat} \cong \text{ListA} \times \text{StrA} \]

- **Input streams with past/present/future:**
  \[ a_0, a_1, \ldots, a_{n-1}, a_n, a_{n+1}, a_{n+2}, \ldots \]

- **Counit:**
  \[ \varepsilon_A : (\text{Nat} \Rightarrow A) \times \text{Nat} \rightarrow A \]
  \[ (a, n) \mapsto a(n) \]

- **Co-Kleisli extension:**
  \[ k : (\text{Nat} \Rightarrow A) \times \text{Nat} \rightarrow B \]
  \[ k^* : (\text{Nat} \Rightarrow A) \times \text{Nat} \rightarrow (\text{Nat} \Rightarrow B) \times \text{Nat} \]
  \[ (a, n) \mapsto (\lambda m k(a, m), n) \]
Comonad for causal stream functions

- Functor: \( DA =_{df} \text{NEList} \cong \text{List}A \times A \)
- Input streams with past and present but no future
- Counit:
  \[ \varepsilon_A : \text{NEList}A \to A \]
  \[ [a_0, \ldots, a_n] \mapsto a_n \]
- Co-Kleisli extension:
  \[ k : \text{NEList}A \to B \]
  \[ k^* : \text{NEList}A \to \text{NEList}B \]
  \[ [a_0, \ldots, a_n] \mapsto [k[a_0], k[a_0, a_1], \ldots, k[a_0, \ldots, a_n]] \]

Comonad for anticausal stream functions

- Input streams with present and future but no past
- Functor: \( DA =_{df} \text{Str}A \cong A \times \text{Str}A \)
Relabelling tree transformations

Let $F : C \to C$. Define $\text{Tree}A =_{df} \mu X. A \times FX$. We are interested in functions $\text{Tree}A \to \text{Tree}B$.

(Alt. we can define $\text{Tree}^\infty A =_{df} \nu X. A \times FX$ and interest ourselves in functions $\text{Tree}^\infty A \to \text{Tree}^\infty B$.)

Comonad for general relabelling functions:

$$DA =_{df} \text{Path}A \times \text{Tree}A$$

(Huet’s zipper) where $\text{Path}A =_{df} \mu X. 1 + A \times F'(\text{Tree}A) \times X$.

Comonad for bottom-up relabelling functions:

$$DA =_{df} \text{Tree}A$$
Co-Kleisli categories and Cartesian closed structure

Let $D$ be a comonad on a Cartesian closed cat. $\mathcal{C}$.

Since $J$ is right adjoint and preserves limits, $\text{CoKl}(D)$ has products. Explicitly, we can define

\[
\begin{align*}
A \times^D B &= \text{df} \quad A \times B \\
\pi_0^D &= \text{df} \quad \text{fst} \circ \varepsilon \\
\pi_1^D &= \text{df} \quad \text{snd} \circ \varepsilon \\
\langle k_0, k_1 \rangle^D &= \text{df} \quad \langle k_0, k_1 \rangle
\end{align*}
\]
If $D$ is strong/lax symmetric semimonoidal wrt. $(1, \times)$, i.e., comes with a nat. iso./transf. $m : DA \times DB \to D(A \times B)$, then we can also define

$$A \Rightarrow^D B =_{df} DA \Rightarrow B$$

$$\text{ev}^D =_{df} \text{ev} \circ \langle \varepsilon \circ D\text{fst}, D\text{snd} \rangle$$

$$\Lambda^D(k) =_{df} \Lambda(k \circ m)$$

$$D((DA \Rightarrow B) \times A) \xrightarrow{\langle \varepsilon \circ D\text{fst}, D\text{snd} \rangle} (DA \Rightarrow B) \times DA \xrightarrow{\text{ev}} B$$

$$DC \times DA \xrightarrow{m} D(C \times A) \xrightarrow{k} B$$

$$DC \xrightarrow{\Lambda(k \circ m)} DA \Rightarrow B$$
Using a strength (if available) is not a good idea: We have no multiplication

\[ DC \times DA \xrightarrow{sl} D(C \times DA) \xrightarrow{Dsr} DD(C \times A) \xrightarrow{?} D(C \times A) \]

and applying \( \varepsilon \) or \( D\varepsilon \) gives a solution where the order of arguments of a function is important and cofeects do not combine:

\[ DC \times DA \xrightarrow{id \times \varepsilon} DC \times A \xrightarrow{sl} D(C \times A) \]

or

\[ DC \times DA \xrightarrow{\varepsilon \times id} C \times DA \xrightarrow{sr} D(C \times A) \]
If $D$ is strong semimonoidal (in which case it is automatically strong symmetric semimonoidal), then $A \Rightarrow^D -$ is right adjoint to $- \times^D A$ and hence $\Rightarrow^D$ is an exponent functor:

$$
\begin{align*}
D(C \times A) &\to B \\
DC \times DA &\to B \\
DC &\to DA \Rightarrow B
\end{align*}
$$

This is the case, e.g., if $DA =_{df} \nu X. A \times (K \Rightarrow X)$ for some $K$ (e.g., $DA =_{df} \text{Str}A$).
More typically, $D$ is only lax symmetric semimonoidal.

Then it suffices to have $m$ satisfying $m \circ \Delta = D \Delta$, where $\Delta = \langle \text{id}, \text{id} \rangle : A \rightarrow A \times A$ is part of the comonoid structure on the objects of $C$, to get that $m \circ \langle D\text{fst}, D\text{snd} \rangle = \text{id}$ and that $\Rightarrow^D$ is a weak exponent operation on objects. It is not functorial (not even in each argument separately).
Partial uniform parameterized fixpoint operator

Let \( F : C \to C \). Define \( DA =_{df} \nu Z.A \times FZ \).

Call a coKleisli map \( k : A \times B \to^D B \) guarded if for some \( k' \) we have

\[
D(A \times B) \xrightarrow{k} B \\
\cong \\
(A \times B) \times FD(A \times B) \xrightarrow{\text{fst} \times \text{id}} A \times FD(A \times B)
\]

For any guarded \( k : A \times B \to^D B \), there is a unique map \( \text{fix}(k) : A \to^D B \) satisfying

\[
A \xrightarrow{\text{fix}(k)} B \\
\langle \text{id}^D, \text{fix}(k) \rangle^D \xrightarrow{k} A \times B
\]
fix is a partial *Conway operator* defined on guarded maps, i.e., besides the *fixpoint property*, for any guarded
\[ k : A \times^D B \rightarrow^D B, \]

\[ \text{fix}(k) = k \circ^D \langle \text{id}^D, \text{fix}(k) \rangle^D \]

it satisfies *naturality* in \( A \), *dinaturality* in \( B \), and the *diagonal property*: for any guarded \( k : A \times^D B \times^D B \rightarrow^D B, \)

\[ \text{fix}(k \circ^D (\text{id}^D \times^D \Delta^D)) = \text{fix}(\text{fix}(k)) \]

Wrt. pure maps, fix is also *uniform* (i.e., strongly dinatural in \( B \) instead of dinatural), i.e., for any guarded
\[ k : A \times^D B \rightarrow^D B, \ell : A \times^D B' \rightarrow^D B' \text{ and } h : B \rightarrow B' \]

\[ Jh \circ^D k = \ell \circ^D (\text{id}^D \times^D Jh) \quad \Longrightarrow \quad Jh \circ^D \text{fix}(k) = \text{fix}(\ell) \]
Comonadic semantics

As in the case of monadic semantics, we interpret the lambda-calculus into $\textsf{CoKI}(D)$ in the standard way, getting

$\llbracket A \times B \rrbracket^D = \text{df} \quad \llbracket A \rrbracket^D \times \llbracket B \rrbracket^D$

$\llbracket A \Rightarrow B \rrbracket^D = \text{df} \quad \llbracket A \rrbracket^D \Rightarrow \llbracket B \rrbracket^D$

$\llbracket (x) \times_i \rrbracket^D = \text{df} \quad \pi_i^D$

$\llbracket (x) \text{fst}(t) \rrbracket^D = \text{df} \quad \pi_0^D \circ \llbracket (x) t \rrbracket^D$

$\llbracket (x) \text{snd}(t) \rrbracket^D = \text{df} \quad \pi_1^D \circ \llbracket (x) t \rrbracket^D$

$\llbracket (x) (t_0, t_1) \rrbracket^D = \text{df} \quad \langle \llbracket (x) t_0 \rrbracket^D, \llbracket (x) t_1 \rrbracket^D \rangle^D$

$\llbracket (x) \lambda x t \rrbracket^D = \text{df} \quad \Lambda^D(\llbracket (x, x) t \rrbracket^D)$

$\llbracket (x) t u \rrbracket^D = \text{df} \quad \text{ev}^D \circ \llbracket (x) t \rrbracket^D \cdot \llbracket (x) u \rrbracket^D \rangle^D$

$\llbracket (x) \text{rec} x t \rrbracket^D = \text{df} \quad \text{fix}^D(\llbracket (x, x) t \rrbracket^D)$

Coeffect-specific constructs are interpreted specifically.

Again, $x : C \vdash t : A$ implies $\llbracket (x) t \rrbracket^D : \llbracket C \rrbracket^D \rightarrow^D \llbracket A \rrbracket^D$, but not all equations of the lambda-calculus are validated.
Closed terms: Soundness of typing for $\vdash t : A$ says that $\llbracket t \rrbracket^D : 1 \rightarrow^D \llbracket A \rrbracket^D$, i.e., $D1 \rightarrow \llbracket A \rrbracket^D$, so closed terms are evaluated relative to a coeffect over 1.

In case of general or causal stream functions, this is a list over 1 (i.e., a natural number), the time elapsed.

If $D$ is properly symmetric monoidal (e.g., Str), we have a canonical choice $e : 1 \rightsquigarrow D1$. 
Comonadic dataflow language semantics: The first-order language agrees perfectly with Lucid and Lustre by its semantics.

The meaning of higher-order dataflow computation has been unclear. We get a neat semantics from mathematical considerations (cf. Colaço, Pouzet’s design with two flavors of function spaces).
Related linear/modal logic work

Strong symmetric monoidal comonads are central in the semantics of intuitionistic linear logic and modal logic to interpret the $!$ and $\Box$ operators.

Linear logic: Benton, Bierman, de Paiva, Hyland; Bierman; Benton; Mellies; Maneggia; etc.

Modal logic: Bierman, da Paiva.

Applications to staged computation and semantics of names: Pfenning, Davies, Nanevski.