Coalgebras and Higher-Order Computation: a GoI Approach

Ichiro Hasuo
National Institute of Informatics
Tokyo, Japan

Supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST; Grants-in-Aid No. 15KT0012 & 15K11984, JSPS; and JSPS-Inria Bilateral Joint Research Project “CRECOGI”
Outline

* Categorical axiomatization
* Compilation to sequential machines

Coalgebra meets higher-order computation in Geometry of Interaction [Girard, LC’88]

"GoI Animation"

Categorical GoI
[Abramsky, Haghverdi & Scott, MSCS’02]

GoI w/

\[ f : X \times \mathbb{N} \rightarrow T(X \times \mathbb{N}) \]

\[ f : \mathbb{N} \rightarrow TN \]

T-branching
[IH & Hoshino, LICS’11]

Memoryful GoI
[Hoshino, Muroya & IH, CSL-LICS’14 & POPL’16]
References


Geometry of Interaction (GoI)

* J.-Y. Girard, at Logic Colloquium '88

* Provides "denotational" semantics (w/ operational flavor) for linear $\lambda$-term $M$

* As a compilation technique
  [Mackie, POPL’95] [Pinto, TLCA’01] [Ghica et al., POPL’07, POPL’11, ICFP’11, ...]

* Two presentations:
  * (Operator-) Algebraic [Girard]
  * Token machines/ interaction abstract machines
  [Danos & Regnier, TCS’99] [Mackie, POPL’95]
The GoI Animation

\[ [M] = (\mathbb{N} \rightarrow \mathbb{N}, \text{a partial function}) \]

= “piping”

... (countably many)
The GoI Animation

* Function application \([MN]\)

* by “parallel composition + hiding”
\[
[MN] = [M] \parallel [N]
\]

"parallel composition + hiding" (cf. AJM games)
\[ \boxed{MN} \]

\[ \Rightarrow M = \lambda x. x + 1 \quad N = 2 \]
\[ \Rightarrow M = \lambda x. 1 \quad N = 2 \]
\[ \Rightarrow M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
Outline

Coalgebra meets higher-order computation in Geometry of Interaction [Girard, LC’88]

"GoI Animation"

Categorical GoI

[Abramsky, Haghverdi & Scott, MSCS’02]
Categorical GoI

- Axiomatics of GoI in the categorical language
- Our main reference:
  - Especially its technical report version (Oxford CL), since it’s a bit more detailed
- See also:
The Categorical GoI Workflow

Traced monoidal category $\mathcal{C}$ + other constructs $\rightarrow$ "GoI situation" [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of typed calculus

Applicative str. + combinators

Model of untyped calculus

PER, $\omega$-set, assembly, ...

"Programming in untyped $\lambda$"
**GoI situation**

**Defn.** (GoI situation [AHS02])

A *GoI situation* is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \rightarrow \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

  Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)
**GoI situation**

**Defn. (GoI situation [AHS02])**

A *GoI situation* is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \otimes F \to e'\) Comultiplication
  - \(d : \text{id} \otimes F \to d'\) Dereliction
  - \(c : F \otimes F \otimes F \to c'\) Contraction
  - \(w : K_I \otimes F \otimes F \to w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \otimes U \to k\)
  - \(I \otimes U\)
  - \(u : FU \otimes U \to v\)

**Monoidal category** \((C, \otimes, I)\)

**String diagrams**

\[
\begin{array}{ccc}
  A & \xrightarrow{f} & B \\
  \downarrow & & \downarrow \quad \text{Dereliction} \\
  A & \xrightarrow{g} & C \\
\end{array}
\]

\[
\begin{array}{ccc}
  A & \xrightarrow{f} & B \\
  \downarrow & & \downarrow \quad \text{Comultiplication} \\
  A & \xrightarrow{g \circ f} & C \\
\end{array}
\]

\[
\begin{array}{ccc}
  A & \xrightarrow{f} & B \\
  \downarrow & & \downarrow \quad \text{Weakening} \\
  A \otimes C & \xrightarrow{f \otimes g} & B \otimes D \\
\end{array}
\]

\[h \circ (f \otimes g)\]
GoI situation

**Defn.** (GoI situation [AHS02])
A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleleft F : e'\) Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) Dereliction
  - \(c : F \otimes F \triangleleft F : c'\) Contraction
  - \(w : K_I \triangleleft F : w'\) Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)

**Traced monoidal category**

**“feedback”**

\[
\begin{align*}
A \otimes C & \xrightarrow{f} B \otimes C \\
\quad & \xrightarrow{\text{tr}(f)} B \\
\quad & \xrightarrow{\text{tr}} A
\end{align*}
\]

that is

\[
\begin{array}{ccc}
A & \xrightarrow{f} & C \\
\downarrow & & \downarrow \text{tr} \\
B & \xrightarrow{\text{tr}(f)} & B
\end{array}
\]
String Diagram vs. “Pipe Diagram”

I use two ways of depicting partial functions \( \mathbb{N} \rightarrow \mathbb{N} \).

Pipe diagram

String diagram

In the monoidal category \((\text{Pfn}, +, 0)\)
Traced Sym. Monoidal Category (Pfn, +, 0)

* Category Pfn of partial functions

* Obj. A set $X$

* Arr. A partial function $f : X \to Y$ in Pfn

* is traced symmetric monoidal
Traced Sym. Monoidal Category
(Pfn, +, 0)

\[
\begin{align*}
X + Z & \xrightarrow{f} Y + Z \quad \text{in Pfn} \\
X & \xrightarrow{\text{tr}(f)} Y \quad \text{in Pfn}
\end{align*}
\]

* Trace operator:

\[
\begin{align*}
\text{tr}(f) = \bigcup_{n \in \mathbb{N}} f_{XY} \circ (f_{ZZ})^n \circ f_{XZ}
\end{align*}
\]

* Execution formula (Girard)

* Partiality is essential (infinite

\[
f_{XY}(x) := \begin{cases} f(x) & \text{if } f(x) \in Y \\ \bot & \text{o.w.} \end{cases}
\]

Similar for \(f_{XZ}, f_{ZY}, f_{ZZ}\)
**GoI situation**

* Traced sym. monoidal cat.
* Where one can “feedback”
* Why for GoI?

---

**Defn. (GoI situation [AHS02])**

A GoI situation is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  \[
  \begin{aligned}
  e : FF \otimes F &\rightarrow e' \\
  d : \text{id} \otimes F &\rightarrow d' \\
  c : F \otimes F \otimes F &\rightarrow c' \\
  w : K_I \otimes F &\rightarrow w'
  \end{aligned}
  \]

  - Comultiplication
  - Dereliction
  - Contraction
  - Weakening

  Here \(K_I\) is the constant functor into the monoidal unit \(I\);

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.

  \[
  \begin{aligned}
  j : U \otimes U &\rightarrow U : k \\
  I &\rightarrow U \\
  u : FU &\rightarrow U : v
  \end{aligned}
  \]
\[ [M \ N] = \begin{array}{c} \vdots \\ \vdots \end{array} \]

In string diagram

\[
\begin{array}{c}
\alpha \\
\beta
\end{array}
\]
**GoI situation**

*Traced sym. monoidal cat.*

*Where one can “feedback”*

- **Leading example:** Pfn

---

**Defn. (GoI situation [AHS02])**

A *GoI situation* is a triple \((C, F, U)\) where

- \(C = (C, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : C \to C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  - \(e : FF \otimes F : e'\) \hspace{1cm} Comultiplication
  - \(d : \text{id} \otimes F : d'\) \hspace{1cm} Dereliction
  - \(c : F \otimes F \otimes F : c'\) \hspace{1cm} Contraction
  - \(w : K_I \otimes F : w'\) \hspace{1cm} Weakening

Here \(K_I\) is the constant functor into the monoidal unit \(I\):

- \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
  
  - \(j : U \otimes U \otimes U : k\)
  - \(I \otimes U : v\)

Here \(M\), \(N\) denote morphisms.
GoI situation

**Defn.** (GoI situation [AHS02])
A GoI situation is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \otimes F : e'\) Comultiplication
  - \(d : \text{id} \otimes F : d'\) Dereliction
  - \(c : F \otimes F \otimes F : c'\) Contraction
  - \(w : K_I \otimes F : w'\) Weakening
- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  - \(j : U \otimes U \otimes U : k\)
  - \(I \otimes U : v\)

Here \(K_I\) is the constant functor into the monoidal unit \(I\).

**Defn.** (Retraction)
A retraction from \(X\) to \(Y\),

\[
f : X \triangleright Y : g\ ,
\]

is a pair of arrows

\[
\begin{array}{ccc}
\text{id} & \circlearrowleft & X \\
\circlearrowright & & Y \\
g & & f
\end{array}
\]

such that \(g \circ f = \text{id}_X\).

\* **Functor** \(F\)

\* **For obtaining** \(! : A \to A\)
**GoI situation**

* The reflexive object \( U \)

* Retr. \( U \otimes U \xrightarrow{j} U \)

\[
\begin{align*}
\triangleleft j & \quad \text{Comultiplication} \\
\triangleleft d & \quad \text{Dereliction} \\
\triangleleft c & \quad \text{Contraction} \\
\triangleleft w & \quad \text{Weakening}
\end{align*}
\]

With

\[
\begin{align*}
\triangleleft j & \quad \text{with} \\
\triangleleft k & \quad = \text{id}
\end{align*}
\]

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple \((C, F, U)\) where

- \( C = (C, \otimes, I) \) is a traced symmetric monoidal category (TSMC);
- \( F : C \rightarrow C \) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  \[
  \begin{align*}
  e & : FF \otimes F \xrightarrow{e'} F \\
  d & : \text{id} \otimes F \xrightarrow{d'} F \\
  c & : F \otimes F \otimes F \xrightarrow{c'} F \\
  w & : K_I \otimes F \xrightarrow{w'} F
  \end{align*}
  \]

  Here \( K_I \) is the constant functor into the monoidal unit \( I \);

- \( U \in C \) is an object (called reflexive object), equipped with the following retractions.
  
  \[
  \begin{align*}
  j & : U \otimes U \otimes U \xrightarrow{k} U \\
  I & \otimes U \\
  u & : FU \otimes U \xrightarrow{v}
  \end{align*}
  \]
GoI situation

Defn. (GoI situation [AHS02])
A GoI situation is a triple \((\mathcal{C}, F, U)\) where

- \(\mathcal{C} = (\mathcal{C}, \otimes, I)\) is a traced symmetric monoidal category (TSMC);
- \(F : \mathcal{C} \to \mathcal{C}\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  \[
  \begin{align*}
  e : FF \otimes F &\to F' \\
  d : \text{id} \otimes F &\to d'
  \end{align*}
  \]
  Comultiplication

  \[
  \begin{align*}
  c : F \otimes F \otimes F &\to c' \\
  w : K_I \otimes F &\to w'
  \end{align*}
  \]
  Dereliction

  Here \(K_I\) is the constant functor.

- \(U \in \mathcal{C}\) is an object (called reflexive object), equipped with the following retractions.
  
  \[
  \begin{align*}
  j : U \otimes U \otimes U &\to k \\
  I &\otimes U \\
  u : FU \otimes U &\to v
  \end{align*}
  \]

Why for GoI?

Example in Pfn:

\(N \in \text{Pfn}, \) with
\(N + N \cong N,\)
\(N \cdot N \cong N\)
Categorical axiomatics of the "GoI animation"
Categorical GoI: Constr. of an LCA

**Thm.** ([AHS02])
Given a GoI situation \((\mathcal{C}, F, U)\), the homset \(\mathcal{C}(U, U)\) carries a canonical LCA structure.

- **Applicative str.** \(\cdot\)
- **! operator**
- **Combinators** B, C, I, ...

\[
\begin{align*}
\text{\(g \cdot f\)} & \quad := \text{tr(\((U \otimes f) \circ k \circ g \circ j\))} \\
\end{align*}
\]
Summary: Categorical GoI

**Defn.** (GoI situation [AHS02])

A GoI situation is a triple $(\mathbb{C}, F, U)$ where

- $\mathbb{C} = (\mathbb{C}, \otimes, I)$ is a traced symmetric monoidal category (TSMC);

- $F : \mathbb{C} \to \mathbb{C}$ is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  
  - $e : FF \otimes F : e'$ Comultiplication
  - $d : \text{id} \otimes F : d'$ Dereliction
  - $c : F \otimes F \otimes F : c'$ Contraction
  - $w : K_I \otimes F : w'$ Weakening

  Here $K_I$ is the constant functor into the monoidal unit $I$;

- $U \in \mathbb{C}$ is an object (called reflexive object), equipped with the following retractions.
  
  - $j : U \otimes U \otimes U : k$
  - $I \otimes U$
  - $u : FU \otimes U : v$

**Thm.** ([AHS02])

Given a GoI situation $(\mathbb{C}, F, U)$, the homset $\mathbb{C}(U, U)$ carries a canonical LCA structure.
Outline

Coalgebra meets higher-order computation in Geometry of Interaction [Girard, LC’88]

"GoI Animation"

Categorical GoI

[ Abramsky, Haghverdi & Scott, MSCS’02]

GoI w/ $T$-branching

[ IH & Hoshino, LICS’11]
**Why Categorical Generalization?**

**Examples Other Than Pfn**

- **Pfn (partial functions)**
  
  \[
  X \rightarrow Y \text{ in } \text{Pfn} \\
  X \rightharpoonup Y, \text{ partial function} \quad \text{where } \mathcal{LY} = \{\bot\} + Y \\
  X \rightarrow \mathcal{LY} \text{ in } \text{Sets}
  \]

- **Rel (relations)**
  
  \[
  X \rightarrow Y \text{ in } \text{Rel} \\
  R \subseteq X \times Y, \text{ relation} \quad \text{where } \mathcal{P} \text{ is the powerset monad} \\
  X \rightarrow \mathcal{PY} \text{ in } \text{Sets}
  \]

- **DSRel**
  
  \[
  X \rightarrow Y \text{ in } \text{DSRel} \\
  X \rightarrow \mathcal{DY} \text{ in } \text{Sets} \quad \text{where } \mathcal{DY} = \{d : Y \rightarrow [0, 1] | \sum_y d(y) \leq 1\}
  \]

\(*KL(T)* for different branching monads *T*

(Potential) non-termination

Non-determinism

Probabilistic branching
Different Branching in The GoI Animation

- **Pfn** (partial functions)
  - Pipes can be stuck

- **Rel** (relations)
  - Pipes can branch

- **DSRel**
  - Pipes can branch probabilistically
Branching Monad: Source of Particle-Style GoI Situations

**Thm.** ([Jacobs,CMCS10])
Given a “branching monad” $\mathbf{T}$ on $\mathbf{Sets}$, the monoidal category

$$(\mathcal{Kl}(\mathbf{T}), +, 0)$$

is

- a *unique decomposition category* [Haghverdi,PhD00], hence is
- a traced symmetric monoidal category.

**Cor.**

$$( (\mathcal{Kl}(\mathbf{T}), +, 0), \mathbb{N} \cdot \_ , \mathbb{N} )$$ is a GoI situation.

**Monads in** [Hasuo,Jacobs&Sokolova07]

- $\mathcal{Kl}(\mathbf{T})$ is $\mathbf{Cpo}_\bot$-enriched

**Particle-style: trace via the execution formula**

$$\text{tr}(f) = 
\bigcup \bigg( \prod_{n \in \mathbb{N}} f_{XY} \circ (f_{ZZ})^n \circ f_{XZ} \bigg)$$
The Categorical GoI Workflow

Branching monad B

Coalgebraic trace semantics

Traced monoidal category \( \mathbb{C} \)

+ other constructs \( \rightarrow \) “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Fancy monad

Fancy TSMC

Fancy LCA

Model of fancy language
Model for (a variant of) the Selinger–Valiron quantum $\lambda$-calculus
(linear $\lambda +$ prep./Unitary/meas.)
[Hasuo & Hoshino, LICS’11 & APAL’16]

via the quantum branching monad
... with considerable complication :

\[
[\Gamma \vdash M : \tau] : \Gamma \longrightarrow ([\tau] \rightarrow R) \rightarrow R
\]

where
\[
R = \left\{ p_0 \quad q_0 \quad p_1 \quad q_1 \mid p_\alpha, q_\alpha \in [0, 1] \right\}
\]

Records measurement outcomes

$R$ as a suitable final coalgebra in the realizability category

Fancy monad

Fancy LCA

Realizability

Model of fancy language
Challenge: Memorizing Effects

Already w/ nondeterminism!
Challenge: Memorizing Effects

\[(\lambda x. x + x)(3 \sqcup 5)\]

Already w/ nondeterminism!

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer 3 or 5
- Query \(x\)
- Answer 3 or 5
- Answer 3 + 3, 3 + 5, 5 + 3 or 5 + 5
Coalgebra meets higher-order computation in Geometry of Interaction [Girard, LC’88]

“GoI Animation”

Categorical GoI [Abramsky, Haghverdi & Scott, MSCS’02]

GoI w/ $T$-branching [IH & Hoshino, LICS’11]

Memoryful GoI [Hoshino, Muroya & IH, CSL-LICS’14 & POPL’16]
Memoryful GoI

* Equip piping with internal states, or memory

* Not

\[ [3 \sqcup 5]: \mathbb{N} \rightarrow \mathcal{P}\mathbb{N}, \quad q \mapsto \{3, 5\} \]

but a transducer (Mealy machine)

\[ [3 \sqcup 5]: X \times \mathbb{N} \rightarrow \mathcal{P}(X \times \mathbb{N}), \quad q/3 \]

* Not a new idea:

* Slices in GoI for additives [Laurent, TLCA’01]

* Resumption GoI [Abramsky, CONCUR’96]
Memoryful GoI

* We introduce memory in a structured manner... → the “traced monoidal category” of transducers

\[
\begin{array}{l}
\text{Trans}(T) \\
\text{Objects: sets } A, B, \ldots \\
\text{Arrows: } A \longrightarrow B \text{ in } \text{Trans}(T) \\
( X, X \times A \xrightarrow{c} T(X \times B), \ x_0 \in X ), \ T\text{-transducer}
\end{array}
\]

* with operations like

\[
\begin{array}{c}
A \\
(X,c,x) \\
\downarrow \\
B \\
(Y,d,y) \\
\downarrow \\
C
\end{array}
\]

\[
\begin{array}{c}
A \\
(X,c,x) \\
\downarrow \\
B \\
(Y,d,y) \\
\downarrow \\
D
\end{array}
\]

\[
\begin{array}{c}
A \\
(X,c,x) \\
\downarrow \\
B \\
\circlearrowright
\end{array}
\]

composition \circ \quad tensor \otimes \quad trace
**Trans($T$) by Coalgebraic Component Calculus**

[Barbosa '03][IH & Jacobs '11]

<table>
<thead>
<tr>
<th>$\text{Trans}(T)$</th>
<th>Objects: sets $A, B, \ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrows:</td>
<td>$A \rightarrow B$ in $\text{Trans}(T)$</td>
</tr>
<tr>
<td></td>
<td>$(X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X)$, $T$-transducer</td>
</tr>
</tbody>
</table>

**Diagram:**

- Objects: $A, B, \ldots$
- Arrows: $A \rightarrow B$ in $\text{Trans}(T)$
- $(X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X)$, $T$-transducer

- **Composition:** $\circ$
- **Tensor:** $\otimes$
- **Trace:** $\mu$
The Memoryful GoI Framework

Given:

* a monad \( T \) on \( \text{Sets} \), s.t. \( \text{Kl}(T) \) is \text{Cpo}-enriched

* an alg. signature \( \Sigma \) with algebraic operations on \( T \) [Plotkin & Power]

\[
\left\{ \alpha_{A,B} : (A \Rightarrow TB)^{|\alpha|} \longrightarrow (A \Rightarrow TB) \right\}_{A \in \text{Sets}, B \in \text{Kl}(T)}
\]

For the calculus: \( \lambda_c + (\text{alg. opr. from } \Sigma) + (\text{co})\text{products} + \text{arith.} \)

We give

\[
\Gamma \vdash M : \tau \quad \vdots \quad \Gamma \vdash M_{|\alpha|} : \tau \quad \Gamma \vdash \alpha(M_1, \ldots, M_{|\alpha|}) : \tau \quad \alpha \in \Sigma
\]

in \( \text{Trans}(T) \)

- \text{Exception} \( 1 + E + (\_ \_ ) \)
  - with 0-ary opr. \( \text{raise}_e \) (\( e \in E \))
- \text{Nondeterminism} \( P \)
  - with binary opr. \( \sqcup \)
- \text{Probability} \( D \), where
  \[
  DX = \{ d : X \rightarrow [0, 1] \mid \sum_x d(x) \leq 1 \}
  \]
  - with binary opr. \( \sqcup_p \) (\( p \in [0, 1] \))
- \text{Global state} \( (1 + S \times \_ \_ )^S \)
  - with \( |V|\)-ary \( \text{lookup}_l \) and unary \( \text{update}_{l,v} \)
Missing Ingredient II: Recursion

- Girard style fixed point operator
- Mackie style fixed point operator

- Obviously a fixed point
- Fixed-point induction
- Finitary string diagram

**Theorem** The two coincide. (for any suitable $T$!)
The Memoryful GoI Framework

Theorem (Adequacy)

Let $\vdash M : \text{nat}$. Then, as elem. of $T(N)$,

$$
\left(\begin{array}{c}
N \\
\vdash M : \text{nat}
\end{array}\right) = \left[\left[ M \right] \right].
$$

Interpretation

$[\_ ] : \text{EffVal}_N^\Sigma \to T(N)$

(exploiting free conti. $\Sigma$-alg.)

Opr. sem.: Plotkin-Power

effect-value. E.g.

$\left| 3 \sqcup (5 \sqcup \text{div}) \right| = 3$

feeding a query and observing the outcome

$\left(\begin{array}{c}
N \\
\vdash M : \tau
\end{array}\right)$

$\left| 5 \right| \left| \cdot \right| \left. \text{div} \right| \\
N \quad N \quad \cdots \quad N$

For the calculus:

$\lambda c + \text{(alg. opr. from $\Sigma$)} + \text{(co)products}$

We give

Theorem (Adequacy)

Let $\vdash M : \text{nat}$. Then, as elem. of $T(N)$,

$$
\left(\begin{array}{c}
N \\
\vdash M : \text{nat}
\end{array}\right) = \left[\left[ M \right] \right].
$$

Interpretation

$[\_ ] : \text{EffVal}_N^\Sigma \to T(N)$

(exploiting free conti. $\Sigma$-alg.)

Opr. sem.: Plotkin-Power

effect-value. E.g.

$\left| 3 \sqcup (5 \sqcup \text{div}) \right| = 3$

feeding a query and observing the outcome

$\left(\begin{array}{c}
N \\
\vdash M : \tau
\end{array}\right)$

$\left| 5 \right| \left| \cdot \right| \left. \text{div} \right| \\
N \quad N \quad \cdots \quad N$
Our Tool TtT

Developed by Koko Muroya
http://koko-m.github.io/TtT/

TtT (Terms to Transducers)

Enter a term, or type "ex" to select one from 13 examples. [read documents]

This is a simulation tool of the memoryful Go! framework.
Implemented by Koko Muroya, using Processing.js v1.4.8 and PEG.js v0.8.0.
Summary

Coalgebra meets higher-order computation in Geometry of Interaction [Girard, LC’88]

“GoI Animation”

Categorical GoI [Abramsky, Haghverdi & Scott, MSCS’02]

GoI w/ $T$-branching [IH & Hoshino, LICS’11]

Memoryful GoI [Hoshino, Muroya & IH, CSL-LICS’14 & POPL’16]
Retracing some paths in Process Algebra

Samson Abramsky
Laboratory for the Foundations of Computer Science
University of Edinburgh

1 Introduction

The very existence of the CONCUR conference bears witness to the fact that “concurrency theory” has developed into a subject unto itself, with substantially different emphases and techniques to those prominent elsewhere in the semantics of computation.

Whatever the past merits of this separate development, it seems timely to look for some convergence and unification. In addressing these issues, I have found it instructive to trace some of the received ideas in concurrency back to their origins in the early 1970’s. In particular, I want to focus on a seminal paper by Robin Milner [Mil75]¹, which led in a fairly direct line to his enormously influential work on ccs [Mil80, Mil89]. I will take (to the extreme) the liberty of applying hindsight, and show how some different paths could have been taken, which, it can be argued, lead to a more unified approach to the semantics of computation, and moreover one which may be better suited to modelling today’s concurrent, object-oriented languages, and the type systems and logics required to support such languages.

2 The semantic universe: transducers

Milner’s starting point was the classical automata-theoretic notion of transducers, i.e. structures

\[(Q,X,Y,q_0,\delta)\]