Coalgebras and Higher-Order Computation: a GoI Approach

Ichiro Hasuo
University of Tokyo (JP)

FSCD 2016, Porto, 24 Jun 2016
Coalgebra meets higher-order computation in Geometry of Interaction [Girard, LC’88]
Outline

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

“GoI Animation”

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

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

“GoI Animation”

Categorical GoI

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

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

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

Naohiko Hoshino
(Kyoto U)
Collaborators

Koko Muroya
(Tokyo => Birmingham)

Naohiko Hoshino
(Kyoto U)
Collaborators

Koko Muroya
(Tokyo => Birmingham)

Naohiko Hoshino
(Kyoto U)

Bart Jacobs
(Nijmegen)

Hasuo (Tokyo)
Collaborators

Koko Muroya
(Tokyo => Birmingham)

Naohiko Hoshino
(Kyoto U)

Bart Jacobs
(Nijmegen)

Toshiki Kataoka
(Tokyo)

Hasuo (Tokyo)
References


Geometry of Interaction (GoI)

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

Hasuo (Tokyo)

Friday, June 24, 16
Geometry of Interaction (GoI)

- J.-Y. Girard, at Logic Colloquium ’88
- Provides “denotational” semantics (w/ operational flavor) for linear $\lambda$-term $M$
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, …]
Geometry of Interaction (GoI)

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

* Provides “denotational” semantics (w/ operational flavor) for linear λ-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”

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\vdots & \vdots & \vdots & \vdots \\
\end{array} \]

... (countably many)
The GoI Animation

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

= “piping”

0 1 2 3 ...

(countably many)

Hasuo (Tokyo)
The GoI Animation

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

= “piping”

0 1 2 3 ...

(countably many)
The GoI Animation

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

= “piping”

\[
\begin{array}{cccccc}
0 & & & & & 3 \\
\downarrow & & & & & \downarrow \\
& \text{token} & & & & \\
\downarrow & & & & & \downarrow \\
& & & & & \\
\downarrow & & & & & \downarrow \\
& & & & & \\
\downarrow & & & & & \downarrow \\
& & & & & \\
\end{array}
\]

\[ [M] \]

... (countably many)
The GoI Animation

$$\llbracket M \rrbracket = (\mathbb{N} \rightarrow \mathbb{N}, \text{a partial function})$$

$$= \text{“piping”}$$

$$0 1 2 3 \ldots$$ (countably many)
The GoI Animation

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

= “piping”

\[
\begin{array}{cccc}
0 & 1 & 2 & 3 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\end{array}
\]

\[
\cdots \quad \text{(countably many)}
\]
The GoI Animation

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

= “piping”

\[ \begin{array}{cccc}
0 & 1 & 2 & 3 \\
\downarrow & \downarrow & \downarrow & \downarrow \\
\end{array} \]

... (countably many)
The GoI Animation

- Function application $[MN]$
- by “parallel composition + hiding”
\[
[MN] = [M] \begin{array}{c}
\ldots
\end{array} [N]
\]
$[M N] = [M] \quad [N]$
\[ [M N] \]

\[ = \]

\[ [M] \]

\[ [N] \]
\[ [MN] \]

\[ = \]

\[ [M] \]

\[ [N] \]
\[ MN \]

=}

\[ M \]

\[ N \]
\[ [M] \oplus [N] \]

"parallel composition + hiding" (cf. AJM games)
\[ M N \] = \begin{array}{ll}
M &= \lambda x. x + 1 \\
N &= 2 \\
M &= \lambda x. 1 \\
N &= 2 \\
M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{array}
\[ [M N] = \]

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

\[ [M] \]

\[ [N] \]

\[ ... \rightarrow M = \lambda x. x + 1 \]

\[ \quad N = 2 \]

\[ M = \lambda x. 1 \quad N = 2 \]

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

Friday, June 24, 16
\[ MN \]
\[
\begin{align*}
M &= \lambda x. x + 1 \\
N &= 2 \\
M &= \lambda x. 1 \\
N &= 2 \\
M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{align*}
\]
\[ \begin{align*}
[MN] &= \cdots \\
M &= \lambda x. x + 1 \quad N = 2 \\
M &= \lambda x. 1 \quad N = 2 \\
M &= \lambda f. f1 \quad N = \lambda x. (x + 1)
\end{align*} \]
\[ [M N] = \]

\[
\begin{align*}
M &= \lambda x. x + 1 & N &= 2 \\
M &= \lambda x. 1 & N &= 2 \\
M &= \lambda f. f1 & N &= \lambda x. (x + 1)
\end{align*}
\]
\[ (M N) \]

\[ M = \lambda x. x + 1 \quad N = 2 \]

\[ \rightarrow M = \lambda x. 1 \quad N = 2 \]

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

\[ MN \]

\[ \begin{align*}
M &= \lambda x. x + 1 \\
N &= 2
\end{align*} \]

\[ \begin{align*}
M &= \lambda x. 1 \\
N &= 2
\end{align*} \]

\[ \begin{align*}
M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{align*} \]
\[ [MN] = \begin{align*}
&M = \lambda x. x + 1 & N = 2 \\
&M = \lambda x. 1 & N = 2 \\
&M = \lambda f. f1 & N = \lambda x. (x + 1)
\end{align*} \]
\[ [M N] = \]

\[ \begin{align*}
M &= \lambda x. x + 1 \\
N &= 2 \\
M &= \lambda x. 1 \\
N &= 2 \\
\rightarrow M &= \lambda f. f1 \\
N &= \lambda x. (x + 1)
\end{align*} \]
\[ M \]
\[ N \]

\[
[MN] = \]

\[
[M] \quad [N]
\]

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

\[
M = \lambda x. x + 1 \quad N = 2
\]

\[
M = \lambda x. 1 \quad N = 2
\]
\[
[MN] = \begin{array}{c}
\ldots
\end{array}
\]

\[
[M] = \begin{array}{c}
\ldots
\end{array}
\]

\[
[N] = \begin{array}{c}
\ldots
\end{array}
\]

\[
M = \lambda x. x + 1 \quad N = 2
\]

\[
M = \lambda x. 1 \quad N = 2
\]

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

Friday, June 24, 16
\[ [M \times N] = [M] [N] \]

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

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

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

“GoI Animation”

\[ M = \lambda x. x + 1 \quad N = 2 \]

\[ M = \lambda x. 1 \quad N = 2 \]

\[ M = \lambda f. f1 \quad N = \lambda x. (x + 1) \]
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 $\mathbb{C}$
+ other constructs $\rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category
The Categorical GoI Workflow

Traced monoidal category $C$ + other constructs $\Rightarrow$ “GoI situation” [AHS02]

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category
The Categorical GoI Workflow

- Traced monoidal category $C$ + other constructs $\Rightarrow$ “GoI situation” [AHS02]
- Categorical GoI [AHS02]
- Linear combinatory algebra
- Realizability
- Linear category

- Applicative str. + combinators
- Model of untyped calculus

Friday, June 24, 16
The Categorical GoI Workflow

Traced monoidal category $\mathbb{C}$ + 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

Friday, June 24, 16
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 $\lambda$”
The Categorical GoI Workflow

- Traced monoidal category $\mathcal{C}$
  + other constructs $\Rightarrow$ “GoI situation” [AHS02]

- Categorical GoI [AHS02]
  - Applicative str. + combinators
  - Model of untyped calculus

- Linear combinatory algebra
  - PER, $\omega$-set, assembly, ...
  - “Programming in untyped $\lambda$”

- Linear category
  - Model of typed calculus

Friday, June 24, 16
Linear Combinatory Algebra (LCA)

Defn. (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  $$\cdot : A^2 \rightarrow A$$

- a unary operator
  $$! : A \rightarrow A$$

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$ satisfying

\[
\begin{align*}
Bxyz &= x(yz) & \text{Composition, Cut} \\
Cxyz &= (xz)y & \text{Exchange} \\
Ix &= x & \text{Identity} \\
Kxy &= x & \text{Weakening} \\
Wxy &= x ! y ! y & \text{Contraction} \\
Dx &= x & \text{Dereliction} \\
\delta x &= ! ! x & \text{Comultiplication} \\
Fx &= ! (xy) & \text{Monoidal functoriality}
\end{align*}
\]

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
Linear Combinatory Algebra (LCA)

Defn. (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  $$\cdot : A^2 \rightarrow A$$

- a unary operator
  $$! : A \rightarrow A$$

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$
  satisfying

  $$\begin{align*}
  Bxyz &= x(yz) & \text{Composition, Cut} \\
  Cxyz &= (xz)y & \text{Exchange} \\
  Ix &= x & \text{Identity} \\
  Kx!y &= x & \text{Weakening} \\
  Wx!y &= x!y!y & \text{Contraction} \\
  Dx &= x & \text{Dereliction} \\
  \delta!x &= !!x & \text{Comultiplication} \\
  F!x!y &= !(xy) & \text{Monoidal functoriality}
  \end{align*}$$

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
Defn. (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  $$\cdot : A^2 \rightarrow A$$
- a unary operator
  $$! : A \rightarrow A$$
- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$ satisfying

\[
\begin{align*}
Bxyz &= x(yz) & \text{Composition, Cut} \\
Cxyz &= (xz)y & \text{Exchange} \\
Ix &= x & \text{Identity} \\
Kx!y &= x & \text{Weakening} \\
Wx!y &= x!!y & \text{Contraction} \\
D!x &= x & \text{Dereliction} \\
\delta!x &= !!x & \text{Comultiplication} \\
F!x!y &= !(xy) & \text{Monoidal functoriality}
\end{align*}
\]

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
**Linear Combinatory Algebra (LCA)**

Defn. (LCA)
A linear combinatory algebra (LCA) is a set $A$ equipped with

- a binary operator (called an applicative structure)
  
  $\cdot : A^2 \rightarrow A$

- a unary operator
  
  $! : A \rightarrow A$

- (combinators) distinguished elements $B, C, I, K, W, D, \delta, F$ satisfying
  
  $\begin{align*}
  Bxyz &= x(yz) & \text{Composition, Cut} \\
  Cxyz &= (xz)y & \text{Exchange} \\
  Ix &= x & \text{Identity} \\
  Kx!y &= x & \text{Weakening} \\
  Wx!y &= x!y!y & \text{Contraction} \\
  Dx &= x & \text{Dereliction} \\
  \delta!x &= !!x & \text{Comultiplication} \\
  F!x!y &= !(xy) & \text{Monoidal functoriality}
  \end{align*}$

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.
A linear combinatory algebra (LCA) is a set $A$ equipped with:

- A binary operator (called an applicative structure)
  $$\cdot : A^2 \to A$$
- A unary operator
  $$! : A \to A$$

(Combinators) Distinguished elements $B, C, I, K, W, D, \delta, F$ satisfying:

- $Bxyz = x(yz)$ (Composition, Cut)
- $Cxyz = (xz)y$ (Exchange)
- $Ix = x$ (Identity)
- $Kx!y = x$ (Weakening)
- $Wx!y = x!!y!y$ (Contraction)
- $D!x = x$ (Dereliction)
- $\delta!x = !!x$ (Comultiplication)
- $F!x!y = !(xy)$ (Monoidal functoriality)

Here: $\cdot$ associates to the left; $\cdot$ is suppressed; and $!$ binds stronger than $\cdot$ does.

Model of untyped linear $\lambda$-term:

- $a \in A \Rightarrow$ closed linear $\lambda$-term

No $S$ or $K$ (linear!)

Combinatory completeness:

- $\lambda xyz. zxy$

Designates an elem. of $A$
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);
- \(F : C \rightarrow C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  \[\begin{array}{ll}
  e : FF \triangleright F : e' & \text{Comultiplication} \\
  d : \text{id} \triangleleft F : d' & \text{Dereliction} \\
  c : F \otimes F \triangleright F : c' & \text{Contraction} \\
  w : K_I \triangleleft F : w' & \text{Weakening}
  \end{array}\]

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.
  \[\begin{array}{ll}
  j : U \otimes U \triangleleft U : k \\
  I \triangleleft U \\
  u : FU \triangleleft U : v
  \end{array}\]
**GoI situation**

* Monoidal category \((\mathcal{C}, \otimes, I)\)

* String diagrams

---

**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).
  - \(e : FF < F : e'\) Comultiplication
  - \(d : \text{id} < F : d'\) Dereliction
  - \(c : F \otimes F < F : c'\) Contraction
  - \(w : K_I < F : w'\) 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 retraction.
  - \(j : U \otimes U < U : k\)
  - \(I < U\)
  - \(u : FU < U : v\)
**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);
- \(F : C \rightarrow C\) is a traced symmetric monoidal functor, equipped with the following retractions (which are monoidal natural transformations).
  - \(e : FF \triangleright F : e'\) - Comultiplication
  - \(d : \text{id} \triangleleft F : d'\) - Dereliction
  - \(c : F \otimes F \triangleright F : c'\) - 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.
  - \(j : U \otimes U \triangleright U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleright U : v\)

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

**String diagrams**
**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).
  
  \[
  
  \begin{align*}
  e : & \quad FF \triangleleft F : e' \quad \text{Comultiplication} \\
  d : & \quad \text{id} \triangleleft F : d' \quad \text{Dereliction} \\
  c : & \quad F \otimes F \triangleleft F : c' \quad \text{Contraction} \\
  w : & \quad K_I \triangleleft F : w' \quad \text{Weakening}
  \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 : & \quad U \otimes U \triangleleft U : k \\
  I \triangleleft U \\
  u : & \quad FU \triangleleft U : v
  \end{align*}
  
  \]

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

* String diagrams

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

\[
A \xrightarrow{g \circ f} C
\]

\[
A \xrightarrow{f} B \\
\downarrow{g} & \downarrow{g} \\
C & \rightarrow & D
\]

\[
A \otimes C \xrightarrow{f \otimes g} B \otimes D
\]

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

* **Traced monoidal category**

* "feedback"

\[
\begin{array}{ccc}
A \otimes C & \xrightarrow{f} & B \otimes C \\
\downarrow \text{tr}(f) & & \downarrow \text{tr}(f) \\
A & \xrightarrow{\text{tr}(f)} & B
\end{array}
\]

that is

\[
\begin{array}{ccc}
A & \xrightarrow{\text{tr}(f)} & B \\
\downarrow & & \downarrow \\
A & \xrightarrow{\text{tr}(f)} & B
\end{array}
\]

**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).
  - \(e : FF < F : e'\) Comultiplication
  - \(d : \text{id} < F : d'\) Dereliction
  - \(c : F \otimes F < F : c'\) Contraction
  - \(w : K_I < F : w'\) 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.
  - \(j : U \otimes U < U : k\)
  - \(I < U\)
  - \(u : FU < U : v\)
String Diagram vs. "Pipe Diagram"

* I use two ways of depicting partial functions $\mathbb{N} \rightarrow \mathbb{N}$

Pipe diagram

String diagram
String Diagram vs. “Pipe Diagram”

I use two ways of depicting partial functions $\mathbb{N} \rightarrow \mathbb{N}$

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 $X \rightarrow Y$ in Pfn

$X \rightarrow Y, \text{ partial function}$

Hasuo (Tokyo)
Traced Sym. Monoidal Category 
\((\text{Pfn}, +, 0)\)

* Category Pfn of **partial functions**
  
* **Obj.** A set \(X\)
  
* **Arr.** A partial function
  
\[
\frac{X \to Y \text{ in Pfn}}{X \to Y, \text{ partial function}}
\]

* is traced symmetric monoidal
Traced Sym. Monoidal Category $(\text{Pfn}, +, 0)$

* \( X + Z \xrightarrow{f} Y + Z \) in $\text{Pfn}$

* \( X \xrightarrow{\text{tr}(f)} Y \) in $\text{Pfn}$

How?
Traced Sym. Monoidal Category
\((\text{Pfn}, +, 0)\)

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

\[f\]

X \quad \quad \quad \quad Z

Y \quad \quad \quad \quad Z

How?
Traced Sym. Monoidal Category
(Pfn, +, 0)

\[
\begin{align*}
X + Z & \xrightarrow{f} Y + Z \quad \text{in Pfn} \\
X & \xrightarrow{\text{tr}(f)} Y \quad \text{in Pfn}
\end{align*}
\]
Traced Sym. Monoidal Category
\((\text{Pfn}, +, 0)\)

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

How?

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

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

Hasuo (Tokyo)
Traced Sym. Monoidal Category
\((\mathsf{Pfn}, +, 0)\)

\[
\begin{array}{c}
X + Z \xrightarrow{f} Y + Z \quad \text{in } \mathsf{Pfn} \\
X \xrightarrow{\text{tr}(f)} Y \quad \text{in } \mathsf{Pfn}
\end{array}
\]

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

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

Trace operator:

Hasuo (Tokyo)

Friday, June 24, 16
Traced Sym. Monoidal Category
(Pfn, +, 0)

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

How?

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

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

Trace operator:

\[
\text{tr}(f) = \bigcup_{n \in \mathbb{N}} f_{ZY} \circ (f_{ZZ})^n \circ f_{XZ}
\]
Traced Sym. Monoidal Category
(Pfn, +, 0)

How?

Trace operator:

\[ X + Z \overset{f}{\to} Y + Z \quad \text{in Pfn} \]
\[ X \overset{\text{tr}(f)}{\to} Y \quad \text{in Pfn} \]

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

Execution formula (Girard)

Partiality is essential (infinite loop)

\[ \text{tr}(f) = f_{XY} \sqcup \left( \bigsqcup_{n \in \mathbb{N}} f_{ZY} \circ (f_{ZZ})^n \circ f_{XZ} \right) \]
GoI situation

* Traced sym. monoidal cat.

* Where one can “feedback”

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).
  - $e : FF \otimes F \rightarrow F : e'$, Comultiplication
  - $d : \text{id} \otimes F \rightarrow F : d'$, Dereliction
  - $c : F \otimes F \rightarrow F : c'$, Contraction
  - $w : K_I \otimes F \rightarrow F : w'$, 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.
  - $j : U \otimes U \rightarrow U : k$
  - $I \otimes U$
  - $u : FU \rightarrow U : v$

* Why for GoI?
\[
\begin{bmatrix}
M & N
\end{bmatrix}
= \begin{bmatrix}
M \\
\end{bmatrix} 
\begin{bmatrix}
N
\end{bmatrix}
\]
\[ [M N] \]

\[ = \]

in string diagram
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).
  
  \[
  \begin{align*}
  e &: \text{Comultiplication} \\
  d &: \text{Dereliction} \\
  c &: \text{Contraction} \\
  w &: \text{Weakening}
  \end{align*}
  \]

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.

\[
\begin{align*}
  j &: \text{Reflexivity} \\
  I &: \text{Identity} \\
  u &: \text{Weakening}
  \end{align*}
\]

**Traced sym. monoidal cat.**

**Where one can “feedback”**

**Why for GoI?**

**Leading example: Pfn**
**GoI situation**

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

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

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

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

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

\[
f : X \triangleleft Y : g
\]

is a pair of arrows

\[
id \quad X \quad Y
\]

\[
f \quad g
\]

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

- **Functor** \(F\)
- **For obtaining** \(! : A \to A\)
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).
  - \(e : FF \triangleleft F : e'\) \text{ Comultiplication}
  - \(d : \text{id} \triangleleft F : d'\) \text{ Dereliction}
  - \(c : F \otimes F \triangleleft F : c'\) \text{ Contraction}
  - \(w : K_I \triangleleft F : w'\) \text{ 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 rejections.
  - \(j : U \otimes U \triangleleft U : k\)
  - \(I \triangleleft U\)
  - \(u : FU \triangleleft U : v\)
**GoI situation**

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

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

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

Here \(K_I\) is the constant functor into the monoidal unit \(I\);
3. \(U \in C\) is an object (called reflexive object), equipped with the following retractions.
   - \(j : U \otimes U \triangleleft U \triangleleft k\)
   - \(I \triangleleft U\)
   - \(u : FU \triangleleft U \triangleleft v\)

**The reflexive object** \(U\)

**Retr.** \(U \otimes U \overset{j}{\rightarrow} U \overset{k}{\leftarrow} U\)

with \(j = k = \text{id}\)
GoI situation

- The reflexive object $U$
- Why for GoI?
- Example in 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).
  
  - $e : FF \triangleleft F : e'$, Comultiplication
  - $d : \text{id} \triangleleft F : d'$
  - $c : F \otimes F \triangleleft F : c'$
  - $w : K_I \triangleleft F : w'$
  
  Here $K_I$ is the constant functor.

- $U \in C$ is an object (called reflexive object), equipped with the following retractions.
  
  - $j : U \otimes U \triangleleft U : k$
  - $I \triangleleft U$
  - $u : FU \triangleleft U : v$
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).
  
  \[
  \begin{align*}
  e &: FF \otimes F : e' \\
  d &: \text{id} \otimes F : d' \\
  c &: F \otimes F \otimes F : c' \\
  w &: K_I \otimes F : w'
  \end{align*}
  \]

  Here \(K_I\) is the constant functor.

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

The reflexive object \(U\)

Why for GoI?

Example in Pfn:
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).
  
  \[
  \begin{align*}
  e & : FF < F : e' & \text{Comultiplication} \\
  d & : \text{id} < F : d' \\
  c & : F \otimes F < F : c' \\
  w & : K_I < F : w'
  \end{align*}
  \]

  Here \(K_I\) is the constant functor.

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

- Why for GoI?

- Example in Pfn:

  \[N \in \text{Pfn}, \text{ with} \]
  \[
  \begin{align*}
  N + N & \cong N, \\
  N \cdot N & \cong N
  \end{align*}
  \]
GoI Situation: Summary

**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).
  - \(e : FF \cdot F \cdot e'\) Comultiplication
  - \(d : \text{id} \cdot F \cdot d'\) Dereliction
  - \(c : F \otimes F \cdot F \cdot c'\) Contraction
  - \(w : K_I \cdot F \cdot w'\) 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.
  - \(j : U \otimes U \cdot U \cdot k\)
  - \(I \cdot U\)
  - \(u : FU \cdot U \cdot v\)

\[\text{(Pfn, } N \cdot _{-} , N)\]

**Categorical axiomatics of the “GoI animation”**

**Example:**

\[\text{(Pfn, } N \cdot _{-} , N)\]
**Hasuo (Tokyo)**

---

**Categorical axiomatics of the “GoI animation”**

---

**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).
  
  - $e : FF < F : e'$ Comultiplication
  - $d : \text{id} < F : d'$ Dereliction
  - $c : F \otimes F < F : c'$ Contraction
  - $w : K_I < F : w'$ 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.
  
  - $j : U \otimes U < U : k$
  - $I < U$
  - $u : FU < U : v$

---

**Example:**

$(\text{Pfn}, \mathcal{N} \cdot \_ \cdot, \mathcal{N})$
**Categorical axiomatics of the “GoI animation”**

**Example:**

\[(\text{Pfn}, \mathbb{N} \cdot \_ \cdot \mathbb{N})\]
**Categorical axiomatics of the “GoI animation”**

**Example:**

\[(\text{Pfn, } \mathbb{N} \cdot _, \mathbb{N})\]
**Categorical axiomatics of the “GoI animation”**

**Example:**

\[(\text{Pfn}, \ N \cdot \_ \ , \ N)\]
Categorical GoI: Constr. of an LCA

**Thm. ([AHS02])**
Given a GoI situation \((C, F, U)\), the homset

\[ C(U, U) \]

carries a canonical LCA structure.
**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, ...
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. •
- ! operator
- Combinators B, C, I, ...
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. ⋅
- ! operator
- Combinators B, C, I, ...

\[ g \cdot f := \text{tr} \left( (U \otimes f) \circ k \circ g \circ j \right) \]

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

\[
\begin{array}{c}
\mathcal{C}(U, U) \\
\end{array}
\]

\[
\begin{array}{c}
f \\
Ff \\
v \\
U
\end{array}
\]

\[
\begin{array}{c}
\in \mathcal{C}(U, U)
\end{array}
\]

\[
\begin{array}{c}
\star \quad \text{Applicative str.} \\
\star \quad \text{! operator} \\
\star \quad \text{Combinators } B, C, I, \ldots
\end{array}
\]

\[
\begin{array}{c}
!f := u \circ Ff \circ v
\end{array}
\]

\[
\begin{array}{c}
\quad = \\
\quad =
\end{array}
\]

Pipe diagram
Categorical GoI: Constr. of an LCA

* Combinator \( B_{xyz} = x(yz) \)

Figure 7: Composition Combinator B

from [AHS02]
Categorical GoI: Constr. of an LCA

* Combinator $Bxyz = x(yz)$
Categorical GoI:

\[ Bxyz = x(yz) \]
Categorical GoI: Constr. of an LCA

\[ B_{xyz} = x(yz) \]
Bxyz = x(yz)
Categorical GoI: Constr. of an LCA

* Combinator \( B_{xyz} = x(yz) \)

Figure 7: Composition Combinator B

from [AHS02]
Categorical GoI: Constr. of an LCA

* Combinator $Bxyz = x(yz)$

Nice dynamic interpretation of (linear) computation!!

Figure 7: Composition Combinator B

from [AHS02]
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).
  - $e : FF \triangleleft F : e'$ 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 $I$;
- $U \in \mathbb{C}$ is an object (called reflexive object), equipped with the following retractions.
  - $j : U \otimes U \triangleleft U : k$
  - $I \triangleleft U$
  - $u : FU \triangleleft U : v$

**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]
Outline

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

“GoI Animation”

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

Categorical GoI [Abramsky, Haghverdi & Scott, MSCS’02]
**Why Categorical Generalization?: Examples Other Than Pfn** [AHS02]

- Strategy: find a TSMC!
- "Wave-style" examples
  - $\otimes$ is Cartesian product(-like)
  - in which case,
    - `trace` $\approx$ `fixed point operator` [Hasegawa/Hyland]
- An example: $\left( (\omega\text{-Cpo}, \times, 1), (\_)^N, A^N \right)$
- (... less of a dynamic flavor)
Why Categorical Generalization?: Examples Other Than \( \text{Pfn} \) [AHS02]

- "Particle-style" examples
  - Obj. \( X \in C \) is set-like; \( \otimes \) is coproduct-like
  - The GoI animation is valid

- Examples:
  - Partial functions: \( (\text{Pfn}, +, 0), \mathbb{N} \cdot \_, \mathbb{N}) \)
  - Binary relations: \( (\text{Rel}, +, 0), \mathbb{N} \cdot \_, \mathbb{N}) \)
  - "Discrete stochastic relations": \( (\text{DSRel}, +, 0), \mathbb{N} \cdot \_, \mathbb{N}) \)
Why Categorical Generalization?: Examples Other Than \textit{Pfn} \cite{AHS02}

\begin{itemize}
  \item \textbf{Pfn (partial functions)}
    \[
    \begin{align*}
    X \rightarrow Y \text{ in Pfn} \\
    X \rightarrow Y, \text{ partial function} \\
    X \rightarrow \mathcal{L}Y \text{ in Sets}
    \end{align*}
    \]
    where \( \mathcal{L}Y = \{\bot\} + Y \)
  
  \item \textbf{Rel (relations)}
    \[
    \begin{align*}
    X \rightarrow Y \text{ in Rel} \\
    R \subseteq X \times Y, \text{ relation} \\
    X \rightarrow \mathcal{P}Y \text{ in Sets}
    \end{align*}
    \]
    where \( \mathcal{P} \) is the powerset monad
  
  \item \textbf{DSRel}
    \[
    \begin{align*}
    X \rightarrow Y \text{ in DSRel} \\
    X \rightarrow \mathcal{D}Y \text{ in Sets}
    \end{align*}
    \]
    where \( \mathcal{D}Y = \{d : Y \rightarrow [0, 1] | \sum_y d(y) \leq 1\} \)
\end{itemize}
Why Categorical Generalization? 

Examples Other Than Pfn [AHS02]

* Pfn (partial functions)

\[
\begin{align*}
X & \to Y \text{ in } \text{Pfn} \\
\frac{X \to Y, \text{ partial function}}{X \to \mathcal{L}Y \text{ in } \text{Sets}} \\
\end{align*}
\]
where \( \mathcal{L}Y = \{ \bot \} + Y \)

* Rel (relations)

\[
\begin{align*}
X & \to Y \text{ in } \text{Rel} \\
\frac{R \subseteq X \times Y, \text{ relation}}{X \to \mathcal{P}Y \text{ in } \text{Sets}} \\
\end{align*}
\]
where \( \mathcal{P} \) is the powerset monad

* DSRel

\[
\begin{align*}
X & \to Y \text{ in } \text{DSRel} \\
X & \to \mathcal{D}Y \text{ in } \text{Sets} \\
\end{align*}
\]
where \( \mathcal{D}Y = \{ d : Y \to [0, 1] \mid \sum_y d(y) \leq 1 \} \)
Why Categorical Generalization? 
Examples Other Than Pfn

- **Pfn** (partial functions)
  
  \[
  \frac{X \to Y \text{ in } \text{Pfn}}{X \to Y, \text{ partial function}} \quad \text{where } \mathcal{LY} = \{\bot\} + Y
  \]
  
- **Rel** (relations)
  
  \[
  \frac{X \to Y \text{ in } \text{Rel}}{R \subseteq X \times Y, \text{ relation}} \quad \text{where } \mathcal{P} \text{ is the powerset monad}
  \]
  
  \[
  \frac{X \to \mathcal{PY} \text{ in } \text{Sets}}{}
  \]

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

---

Categories of sets and (functions with different branching/partiality)

(Potential) non-termination

Non-determinism

Probabilistic branching

Hasuo (Tokyo)

Friday, June 24, 16
Different Branching in The GoI Animation

- **Pfn** (partial functions)
- Pipes can be stuck
- **Rel** (relations)
- Pipes can branch
- **DSRel**
- Pipes can branch probabilistically

Hasuo (Tokyo)
Different Branching in The GoI Animation

- **Pfn** (partial functions)
- Pipes can be stuck
- **Rel** (relations)
- Pipes can branch
- **DSRel**
- Pipes can branch probabilistically

Hasuo (Tokyo)
Different Branching in The GoI Animation

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

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

- **DSRel**
  - Pipes can branch probabilistically
Different Branching in The GoI Animation

- Pfn (partial functions)
- Pipes can be stuck
- Rel (relations)
- Pipes can branch
- DSRel
- Pipes can branch probabilistically
Different Branching in The GoI Animation

- \textbf{Pfn} (partial functions)
- Pipes can be stuck
- \textbf{Rel} (relations)
- Pipes can branch
- \textbf{DSRel}
  - Pipes can branch probabilistically
Different Branching in The GoI Animation

- **Pfn** (partial functions)
- Pipes can be stuck
- **Rel** (relations)
- Pipes can branch
- **DSRel**
  - Pipes can branch probabilistically
Different Branching in The GoI Animation

- \textbf{Pfn} (partial functions)
- Pipes can be stuck
- \textbf{Rel} (relations)
- Pipes can branch
- \textbf{DSRel}
- Pipes can branch probabilistically
A Coalgebraic View

- Theory of coalgebra =
  Categorical theory of state-based dynamic systems
  (LTS, automaton, Markov chain, ...)

- In my thesis (2008):
  - Coalgebras in a Kleisli category $Kl(T)$
    
    $X \rightarrow Y$ in $Kl(T)$
    
    $X \rightarrow TY$ in $\text{Sets}$

- Generic theory of trace and simulations
Why Categorial Generalization?

Examples Other Than Pfn

* Pfn (partial functions)

\[
\begin{align*}
X \to Y & \text{ in } \text{Pfn} \\
X \to Y, \text{ partial function} & \implies X \to \mathcal{L}Y \text{ in } \text{Sets} \\
\text{where } \mathcal{L}Y = \{\bot\} + Y
\end{align*}
\]

* Rel (relations)

\[
\begin{align*}
X \to Y & \text{ in } \text{Rel} \\
R \subseteq X \times Y, \text{ relation} & \implies X \to P Y \text{ in } \text{Sets} \\
\text{where } P & \text{ is the powerset monad}
\end{align*}
\]

* DSRel

\[
\begin{align*}
X \to Y & \text{ in } \text{DSRel} \\
X \to \mathcal{D}Y & \text{ in } \text{Sets} \\
\text{where } \mathcal{D}Y = \{d : Y \to [0, 1] \mid \sum_y d(y) \leq 1\}
\end{align*}
\]

Categories of sets and (functions with different branching/partiality)

(Potential) non-termination

Non-determinism

Probabilistic branching
Why Categorical Generalization?

Examples Other Than Pfn

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

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

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

\[ \text{KL}(T) \] for different branching monads \( T \)

(Potential) non-termination

Non-determinism

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

$$(\mathcal{K}\ell(T), +, 0)$$

is

- a unique decomposition category
  [Haghverdi, PhD00], hence is

- a traced symmetric monoidal category.

Cor.
$$( (\mathcal{K}\ell(T), +, 0), \mathbb{N} \cdot \_ , \mathbb{N} )$$ is a GoI situation.
**Thm.** ([Jacobs, CMCS10])

Given a “branching monad” $T$ on $\text{Sets}$, the monoidal category

$$(K\ell(T), +, 0)$$

is

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

**Cor.**

$$((K\ell(T), +, 0), N \cdot _, N)$$ is a GoI situation.

---

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

- $Kl(T)$ is $\text{Cpo}_\perp$-enriched
- like $L$, $P$, $D$
**Thm.** ([Jacobs,CMCS10])

Given a “branching monad” $T$ on $\text{Sets}$, the monoidal category

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

is

- a *unique decomposition category* [Haghverdi,PhD00], hence is

- a traced symmetric monoidal category.

**Cor.**

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

---

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

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

- Traced monoidal category $\mathbb{C}$
- + other constructs $\rightarrow$ "GoI situation" [AHS02]
- Categorical GoI [AHS02]
- Linear combinatory algebra
- Realizability
- Linear category
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
The Categorical GoI Workflow

Branching monad $B$

$\Rightarrow$ Coalgebraic trace semantics

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

Categorical GoI [AHS02]

Linear combinatory algebra

Realizability

Linear category

Model of fancy language

Hasuo (Tokyo)

Friday, June 24, 16
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 LCA

Model of fancy language

Hasuo (Tokyo)
The Categorical GoI Workflow

- Branching monad B
  - Coalgebraic trace semantics
  - Traced monoidal category $\mathcal{C}$
    + other constructs $\Rightarrow$ "GoI situation" [AHS02]
  - Categorical GoI [AHS02]
  - Linear combinatory algebra
  - Realizability
  - Linear category

Hasuo (Tokyo)

Fancy
TSMC

Fancy
LCA

Model of fancy language

Friday, June 24, 16
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

Hasuo (Tokyo)
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

Workflow

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 :(

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

where

\[ R = \left\{ p_0, q_0, p_1, q_1 \mid p_\alpha, q_\alpha \in [0, 1] \right\} \]
The Categorical GoI Workflow

- Model for (a variant of) the Selinger-Valiron quantum λ-calculus
  (linear λ + prep./Unitary/meas.)
  [Hasuo & Hoshino, LICS’11 & APAL’16]
- via the quantum branching monad
- ... with considerable complication :

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

where

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

\[
\begin{align*}
\vdots & \quad \vdots & \quad \vdots & \quad \vdots \\
p_\varepsilon & \quad q_\varepsilon
\end{align*}
\]
Challenge: Memorizing Effects

Already w/ nondeterminism!
Challenge: Memorizing Effects

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

\[ \begin{align*}
\lambda x. x + x \\
3 \sqcup 5
\end{align*} \]

Already w/ nondeterminism!
Challenge: Memorizing Effects

Already w/ nondeterminism!

Query \((\lambda x. x + x)(3 \downarrow 5)\)

Answer \(3 + 3, 3 + 5, 5 + 3, 5 + 5\)

Answer 3 or 5

Answer 3 or 5

Answer 3 or 5

Answer 3 or 5
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
Challenge: Memorizing Effects

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

\[
\begin{align*}
\lambda x & x + x \\
3 \sqcup 5
\end{align*}
\]

- 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
Challenge: Memorizing Effects

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

Already w/ nondeterminism!

[\[\lambda x. x + x\]]

[\[3 \sqcup 5\]]

- 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 \(5 + 5\)
Challenge: Memorizing Effects

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

Already w/ nondeterminism!

\[\lambda x. x + x\]

\[3 \sqcup 5\]

- 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
Challenge: Memorizing Effects

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

Already w/ nondeterminism!

- Query \((\lambda x. x + x)(3 \uplus 5)\)
- Query \(x\)
- Answer \(3\) or \(5\)
- Query \(x\)
- Answer \(3\) or \(5\)
- Answer \(3 + 3, 3 + 5, 5 + 3\) or \(5 + 5\)
Challenge: Memorizing Effects

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

```
[\lambda x. \text{red} + x] [3 \sqcup 5]
```

- 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

Already w/ nondeterminism!
Challenge: Memorizing Effects

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

- 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
Challenge: Memorizing Effects

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

- 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

Already w/ nondeterminism!
Challenge: Memorizing Effects

* Nondeterministic choice is resolved
→ we must **stick to it**!

* Is CBV to blame?
  (GoI is inherently CBN...)

\[(\lambda x. x + x)(3 \sqcup 5) \rightarrow_{\text{CBV}} 6 \text{ or } 10\]
Challenge: Memorizing Effects

* Nondeterministic choice is resolved ➔ we must **stick to it**!

* Is CBV to blame? (GoI is inherently CBN...)

* Not really: it's also hard to get

\[(\lambda x. x + x)(3 \sqcup 5) \rightarrow_{CBV} 6 \text{ or } 10\]

\[(M \sqcup N)L = ML \sqcup NL\]
Challenge: Memorizing Effects

* Nondeterministic choice is resolved
  → we must **stick to it**!

* Is CBV to blame?
  (GoI is inherently CBN...)

* Not really: it’s also hard to get
  \[(M \sqcup N)L = ML \sqcup NL\]

* Mathematically:

  \[\text{Given } \begin{array}{c|c|c} A & C & A \\ B & C & B \end{array}, \begin{array}{c|c|c} A & C & A \\ B & C & B \end{array} : A + C \rightarrow \mathcal{P}(B + C),\]

  \[\text{tr}(f \cup g) \neq \text{tr}(f) \cup \text{tr}(g)\]
Outline

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

“GoI Animation”

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

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

Friday, June 24, 16
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]

Memoryful GoI

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

* Equip piping with internal states, or memory
Memoryful GoI

* Equip piping with internal states, or memory

* not $[3 \sqcup 5]: \mathbb{N} \rightarrow \mathcal{P}\mathbb{N}, \ q \mapsto \{3, 5\}$

but a transducer (Mealy machine)

$[3 \sqcup 5]: X \times \mathbb{N} \rightarrow \mathcal{P}(X \times \mathbb{N}), \ q/3 \xrightarrow{s_l} q/3 \xrightarrow{s_0} q/5 \xrightarrow{s_r} q/5$
Memoryful GoI

* Equip piping with internal states, or memory

* not

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

but a transducer (Mealy machine)

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

Hasuo (Tokyo)
Memoryful GoI

* Equip piping with internal states, or memory

* not \([3 \sqcup 5] : \mathbb{N} \rightarrow \mathcal{P}\mathbb{N} , \ q \mapsto \{3, 5\}\)

but a transducer (Mealy machine)

\([3 \sqcup 5] : X \times \mathbb{N} \rightarrow \mathcal{P}(X \times \mathbb{N}) , \ q/3 \xrightarrow{s_i} q/3 \xrightarrow{s_0} q/5 \xrightarrow{s_r} q/5\)

* Not a new idea:

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

* Resumption GoI [Abramsky, CONCUR’96]

Hasuo (Tokyo)
\[(\lambda x . x + x)(3 \sqcup 5)\]
Challenge: Memorizing Effects

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

\[
\begin{align*}
\lambda x. x + x & \\
3 \sqcup 5 & \\
3 + 3, & 3 + 5, 5 + 3, \text{ or } 5 + 5
\end{align*}
\]

• Query \((\lambda x. x + x)(3 \sqcup 5)\)
• Query \(x\)
• Answer 3 \text{ or } 5
• Query \(x\)
• Answer 3 \text{ or } 5
• Answer 3 + 3, 3 + 5, 5 + 3 \text{ or } 5 + 5
Challenge: Memorizing Effects

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

- 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 \(5 + 5\)
Challenge: Memorizing Effects

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

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

- 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
Challenge: Memorizing Effects

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

- 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 \(5 + 5\)
Challenge: Memorizing Effects

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

[\lambda x. x + x]

[3 \sqcup 5]

• 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
Challenge: Memorizing Effects

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

\[ [\lambda x. x + x] \quad [3 \sqcup 5] \]

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer \(3\) or \(5\) \textbf{and remember the choice}

- Query \(x\)
- Answer \(3\) or \(5\)
- Answer \(3 + 3, 3 + 5, 5 + 3\) or \(5 + 5\)

Friday, June 24, 16
Challenge: Memorizing Effects

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

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer 3 or 5 and remember the choice
- Query \(x\)
- Answer 3 or 5
- Answer \(3 + 3, 3 + 5, 5 + 3\) or \(5 + 5\)
Challenge: Memorizing Effects

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

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer 3 or 5 and remember the choice
- Query \(x\)
- Answer 3 or 5
- Answer 3 + 3, 3 + 5, 5 + 3 or 5 + 5
Challenge: Memorizing Effects

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

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer 3 or 5 and remember the choice
- Query \(x\)
- Answer 3 or 5
- Answer 3 or 5
- Answer 3 + 3, 3 + 5, 5 + 3 or 5 + 5
Challenge: Memorizing Effects

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

\[ \lambda x. (\bullet + x) \]

\[ 3 \sqcup 5 \]

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer 3 or 5 and remember the choice
- Query \(x\)
- Answer 3 or 5 following the prev. choice
- Answer 3 + 3, 3 + 5, 5 + 3 or 5 + 5
\[(\lambda x. x + x)(3 \sqcup 5)\]

**Challenge: Memorizing Effects**

- Query \((\lambda x. x + x)(3 \sqcup 5)\)
- Query \(x\)
- Answer 3 or 5 **and remember the choice**
- Query \(x\)
- Answer 3 or 5 **following the prev. choice**
- Answer \(3 + 3, 3 + 5, 5 + 3\) or 5 + 5
Memoryful GoI

* That is...
a traversing token rearranges piping!
Memoryful GoI

* We introduce memory in a structured manner...

→ the “traced monoidal category” of transducers

\[
\begin{array}{|c|c|}
\hline
\text{Trans}(T) & \text{Objects: sets } A, B, \ldots \\
\hline
\text{Arrows:} & A \rightarrow B \text{ in Trans}(T) \\
( X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X ), \ T\text{-transducer} \\
\hline
\end{array}
\]
We introduce memory in a structured manner... 
→ the “traced monoidal category” of transducers

\[
\text{Objects: sets } A, B, \ldots \]
\[
A \to B \text{ in Trans}(T)
\]
\[
(X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X), \ T\text{-transducer}
\]

∗ with operations like

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

composition \( \circ \)

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

tensor \( \otimes \)

\[
\begin{array}{c}
A \\
\downarrow \ \\
(X,c,x) \\
B \\
\downarrow \ \\
C \\
\text{trace}
\end{array}
\]
Trans($T$) by Coalgebraic Component Calculus

[Barbosa ’03][IH & Jacobs ’11]

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

\[\begin{array}{cc}
A & C \\
\downarrow & \downarrow \\
(X,c,x) & (X,c,x)
\end{array}\]

\[\begin{array}{cc}
A & C \\
\downarrow & \downarrow \\
(Y,d,y) & (Y,d,y)
\end{array}\]

\[\begin{array}{cc}
A & C \\
\downarrow & \downarrow \\
B & C
\end{array}\]

\textbf{composition} \quad \circ \quad \textbf{tensor} \quad \otimes \\

\textbf{trace}
Trans($T$) by Coalgebraic Component Calculus

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

<table>
<thead>
<tr>
<th>Trans($T$)</th>
<th>Objects: sets $A, B, \ldots$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arrows:</td>
<td>$A \rightarrow B$ in 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>

$$
\begin{align*}
(X, c, x) \\
A \\
\downarrow \\
B \\
(Y, d, y) \\
C \\
\downarrow \\
D \\
\end{align*}
$$

**Composition**: $\circ$

**Tensor**: $\otimes$

**Trace**: $\bigodot$

\[
(X \times Y) \times A \xrightarrow{\simeq} (X \times A) \times Y \\
\xrightarrow{c \times Y} T(X \times B) \times Y \\
\xrightarrow{\text{str}^t} T((X \times B) \times Y) \\
\xrightarrow{\simeq} T(X \times (Y \times B)) \\
\xrightarrow{\text{str}^t} T(X \times T(Y \times C)) \\
\xrightarrow{T \text{str}} TT(X \times (Y \times C)) \\
\xrightarrow{\mu^T} T(X \times (Y \times C)) \\
\xrightarrow{\text{trace}} T((X \times Y) \times C)
\]

Hasuo (Tokyo)

Friday, June 24, 16
Trans(T) by Coalgebraic Component Calculus

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

\[
\begin{align*}
\text{Trans}(T) \quad \text{Objects:} & \quad \text{sets } A, B, \ldots \\
\text{Arrows:} & \quad A \rightarrow B \text{ in Trans}(T) \\
& \quad (X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X), \ T\text{-transducer}
\end{align*}
\]

\[
\begin{array}{c}
\text{composition} \\
\text{tensor} \\
\text{trace}
\end{array}
\]

\* Trans(T) is a “category”...
Trans($T$) by Coalgebraic Component Calculus

[Barbosa ’03][IH & Jacobs ’11]

Objects: sets $A, B, \ldots$

Arrows: $A \rightarrow B$ in Trans($T$)

$(X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X)$, $T$-transducer

Composition

Tensor

Trace

∗ Trans($T$) is a "category"...

∗ Fix: quotient modulo behavioral equivalence (homomorphisms of $T$-transducers) $\Rightarrow$ resumptions [Abramsky]
The Memoryful GoI Framework

- Given:
  - a monad $T$ on $\text{Sets}$, s.t. $\text{Kl}(T)$ is Cppo-enriched
  - an alg. signature $\Sigma$ with algebraic operations on $T$
    ([Plotkin & Power])

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

- For the calculus: $\lambda c + (\text{alg. opr. from } \Sigma) + (\text{co})\text{products} + \text{arith.}$

- We give
The Memoryful GoI Framework

* Given:

* a monad $T$ on $\text{Sets}$, s.t. $\text{Kl}(T)$ is Cppo-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: $\lambda_c + \text{(alg. opr. from } \Sigma) + \text{(co)products + arith.}$

* We give

- Exception $1 + E + (_{\text{-ary opr.}})$
  - with 0-ary opr. $\text{raise}_e (e \in E)$

- Nondeterminism $\mathcal{P}$
  - with binary opr. $\sqcup$

- Probability $\mathcal{D}$, where
  $$\mathcal{D}X = \{d : X \rightarrow [0, 1] \mid \sum_x d(x) \leq 1\}$$
  - with binary opr. $\sqcup_p (p \in [0, 1])$

- Global state $(1 + S \times (_)_S)$
  - with $|V|$-ary lookup $l$ and unary update $l,v$

Hasuo (Tokyo)

Friday, June 24, 16
The Memoryful GoI Framework

* **Given:**

* a monad $T$ on $\text{Sets}$, s.t. $\text{Kl}(T)$ is Cppo-enriched
* an alg. signature $\Sigma$ with algebraic operations on $T$

[Plotkin & Power]

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

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

- **Exception** $1 + E + (\_)$
  - with 0-ary opr. $\text{raise}_e (e \in E)$
- **Nondeterminism $\mathcal{P}$**
  - with binary opr. $\sqcup$
- **Probability $\mathcal{D}$**, where
  \[ \mathcal{D} X = \{ d : X \rightarrow [0, 1] \mid \sum_x d(x) \leq 1 \} \]
  - with binary opr. $\cup_p (p \in [0, 1])$
- **Global state** $(1 + S \times \_)^S$
  - with $|V|$-ary $\text{lookup}_l$ and unary $\text{update}_{l,v}$
The Memoryful GoI Framework

Given:

- A monad $T$ on $\text{Sets}$, s.t. $\text{Kl}(T)$ is Cppo-enriched
- An alg. signature $\Sigma$ with algebraic operations on $T$

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

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

* Given:

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

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

in $\text{Trans}(T)$

- Exception $1 + E + (\_)$
  - with $0$-ary opr. $\text{raise}_e (e \in E)$
- Nondeterminism $\mathcal{P}$
  - with binary opr. $\sqcup$
- Probability $\mathcal{D}$, where

  \[ \mathcal{D}X = \{ d : X \rightarrow [0, 1] \mid \sum_x d(x) \leq 1 \} \]

  - with binary opr. $\sqcup_p (p \in [0, 1])$
- Global state $(1 + S \times \_)^S$
  - with $|V|$-ary $\text{lookup}_l$ and unary $\text{update}_{l,v}$
Trans($T$) by Coalgebraic Component Calculus

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

Objects: sets $A, B, \ldots$

Arrows: $A \rightarrow B$ in Trans($T$)

$(X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X)$, $T$-transducer

Composition

Tensor

Trace

\[
\begin{pmatrix}
(X \times Y) \times A & \xrightarrow{\cong} (X \times A) \times Y \\
\xrightarrow{c \times Y} T(X \times B) \times Y \\
\xrightarrow{\text{str}'} T((X \times B) \times Y) \\
\xrightarrow{\cong} T(X \times (Y \times B)) \\
\xrightarrow{T(X \times d)} T(X \times T(Y \times C)) \\
\xrightarrow{T\text{str}} TT(X \times (Y \times C)) \\
\xrightarrow{\mu_T} T(X \times (Y \times C)) \\
\xrightarrow{\cong} T((X \times Y) \times C)
\end{pmatrix},
\begin{pmatrix}
X \times Y, \\
(x, y)
\end{pmatrix}
\]
**Missing Ingredient I: Alg. Opr.**

* $\alpha \in \Sigma_n$ an alg. operation

\[
\alpha(\begin{array}{c}
\alpha(A | (X_1, c_1, x_1), \ldots, (X_n, c_n, x_n) ) \\
\end{array})
\]

\[
= \left\{ * \right\} + X_1 + \cdots + X_n,
\]

\[
\left(\begin{array}{c}
\left(\begin{array}{c}
\left(\begin{array}{c}
\ldots
\end{array}\right)
\end{array}\right)
\end{array}\right)
\]

\[
\left(\begin{array}{c}
x_1 \\
\cdots
\end{array}\right)
\]

\[
\left(\begin{array}{c}
x_i \\
\cdots
\end{array}\right)
\]

\[
\left(\begin{array}{c}
x_n \\
\end{array}\right), *
\]

\[
\left(\begin{array}{c}
c_1 \\
\cdots
\end{array}\right)
\]

\[
\left(\begin{array}{c}
c_i \\
\cdots
\end{array}\right)
\]

\[
\left(\begin{array}{c}
c_n
\end{array}\right)
\]
* \( \alpha \in \Sigma_n \) an alg. operation

\[
\alpha\left(\begin{array}{c}
A \mid (X_1, c_1, x_1) \\
B \\
\end{array}\right), \ldots, \begin{array}{c}
A \mid (X_n, c_n, x_n) \\
B \\
\end{array}\right)
\]

= \left\{ * \right\} + X_1 + \cdots + X_n, \quad \begin{array}{c}
x_1 \\
\vdots \\
x_i \\
\vdots \\
x_n \\
\end{array}, \quad * 

Fresh initial state

* \( \alpha \in \Sigma_n \) an alg. operation

\[
\alpha\left( \begin{array}{c}
A \mid (X_1, c_1, x_1) \\
\hline
B \\
\end{array} , \ldots , \begin{array}{c}
A \mid (X_n, c_n, x_n) \\
\hline
B \\
\end{array} \right)
\]

\[
= \left\{ \ast \right\} + X_1 + \cdots + X_n , \quad \begin{array}{c}
x_1 \\
\hline
\end{array} \quad \cdots \quad \begin{array}{c}
x_i \\
\hline
\end{array} \quad \cdots \quad \begin{array}{c}
x_n \\
\hline
\end{array} , \quad \ast
\]

\( T \)-branching given by

\[
\{ \alpha_{A,B} : (A \Rightarrow TB)^{|\alpha|} \rightarrow (A \Rightarrow TB) \}_A \in \text{Sets}, B \in \mathcal{K}(T)
\]
Missing Ingredient II: Recursion

Girard style
fixed point operator

Mackie style
fixed point operator

* Obviously a fixed point
* Fixed-point induction
* Finitary string diagram
Missing Ingredient II: Recursion

Girard style
fixed point operator

Mackie style
fixed point operator

* Obviously a fixed point
* Fixed-point induction

** Theorem** The two coincide. (for any suitable \( T! \))

Friday, June 24, 16
The Memoryful GoI Framework

* Given:
  * a monad \( T \) on \( \text{Sets} \), s.t. \( \text{KL}(T) \) is \( \text{Cppo} \)-enriched
  * an alg. signature \( \Sigma \) with algebraic operations on \( T \)
    [Plotkin & Power]
    \[
    \left\{ \alpha_{A,B} : (A \Rightarrow TB)^{|\alpha|} \rightarrow (A \Rightarrow TB) \right\}_{A \in \text{Sets}, B \in \text{KL}(T)}
    \]
  * For the calculus: \( \lambda_c + (\text{alg. opr. from } \Sigma) + (\text{co})\text{products} \)
  * We give
    \[
    \frac{|\Gamma|}{\frac{\Gamma \vdash M_1 : \tau \quad \cdots \quad \Gamma \vdash M_{|\alpha|} : \tau}{\Gamma \vdash \alpha(M_1, \ldots, M_{|\alpha|}) : \tau}} \alpha \in \Sigma
    \]

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

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

- Given:
  - a monad \( T \) on \( \text{Sets} \), s.t. \( \mathcal{K}l(T) \) is \( \text{Cppo} \)-enriched
  - an alg. signature \( \Sigma \) with algebraic operations on \( T \)

[Plotkin & Power]

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

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

- We give

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

\[ (\Gamma \vdash M : \tau) \]

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

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

Hasuo (Tokyo)
Theorem (Adequacy)

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

$$
\left(\left\langle \begin{array}{c}
\vdash M : \text{nat} \\
\mathbb{N}
\end{array} \right\rangle \right)^{\dagger} = \left[ |M| \right].
$$
The Memoryful GoI Framework

**Theorem** (Adequacy)

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

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

feeding a query and observing the outcome
Theorem (Adequacy)

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

\[
\left( \begin{array}{c}
\vdash M : \text{nat} \\
\mathbb{N}
\end{array} \right)

\overset{\dagger}{\mapsto}

\left[ \left\lfloor |M| \right\rfloor \right].
\]

feeding a query and observing the outcome

Opr. sem.: Plotkin-Power effect-value. E.g.

\[
\left\lfloor 3 \uplus (5 \uplus \text{div}) \right\rfloor =

\begin{array}{c}
3 \\
\uplus
\end{array}

\begin{array}{c}
5 \\
\downarrow
\end{array}
\]
The Memoryful GoI Framework

**Theorem (Adequacy)**

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

$$
\left\lceil \left( \left\lceil \vdash M : \text{nat} \right\rceil \right) \right\rceil^\dagger = \left\lceil \left[ \right\rceil \vdash M \right\rceil .
$$

**Interpretation**

$$
\left[ \_ \right] : \text{EffVal}_\Sigma^\mathbb{N} \rightarrow T(\mathbb{N})
$$

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

**Opr. sem.:** Plotkin-Power

**Effect-value.** E.g.

$$
\left\lceil 3 \sqcup (5 \sqcup \text{div}) \right\rceil = \begin{array}{c}
\sqcup \\
3 \\
\sqcup \\
\ \ 5 \\
\sqcap
\end{array}
$$

feeding a query and observing the outcome
**Trans**(T) by Coalgebraic Component Calculus

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

**Objects:** sets \( A, B, \ldots \)

**Arrows:**
\[
A \rightarrow B \text{ in } \text{Trans}(T)
\]
\[
(X, X \times A \xrightarrow{c} T(X \times B), x_0 \in X) \), \ T\text{-transducer}
\]

Diagram:

- **Composition**
- **Tensor**
- **Trace**

\[
(X \times Y) \times A \xrightarrow{\cong} (X \times A) \times Y
\]
\[
\xrightarrow{c \times Y} T(X \times B) \times Y
\]
\[
\xrightarrow{\text{str}^t} T((X \times B) \times Y)
\]
\[
\xrightarrow{\cong} T(X \times (Y \times B))
\]
\[
\xrightarrow{T(x \times d)} T(X \times T(Y \times C))
\]
\[
\xrightarrow{T \text{str}} TT(X \times (Y \times C))
\]
\[
\xrightarrow{\mu^T} T((X \times (Y \times C))
\]
\[
\xrightarrow{\eta^T} T((X \times Y) \times C)
\]

Hasuo (Tokyo)

Friday, June 24, 16
TtT (Terms to Transducers)

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

(rec(flip(loopSimple x) (choose(0.4) x (flip(loopSimple x)))) 0)

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.

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

$x \in X$

Hasuo (Tokyo)
$\ast$ GoI + algebraic effects [Plotkin & Power]

- Coalgebra meets higher-order computation
- 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]

$$f : X \times \mathbb{N} \to T(X \times \mathbb{N})$$
GoI + algebraic effects [Plotkin & Power]

via $T$-branching transducers
• GoI + algebraic effects [Plotkin & Power]
• via $T$-branching transducers
• Compositional translation
  \[(\Gamma \vdash M : \tau)\]
  $\Rightarrow$ implementation $TtT$
\* GoI + algebraic effects [Plotkin & Power]

\* via \( T \)-branching transducers

\* Compositional translation

\[(\Gamma \vdash M : \tau)\]

\(\Rightarrow\) implementation \( \text{TT} \text{TT} \)

\* (Categorical GoI + realizability)

\(\Rightarrow\) categorical model

\* Thm. Adequacy

\* “Correct-by-construction” compilation!
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 [Mi175], which led in a fairly direct line to his enormously influential work on ccs [Mi180, Mi189]. I will take (to the extreme) the liberty of 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

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

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

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

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