What are Bit Strings? The View from Process

Nick Rossiter
Northumbria University
Outline

• Process as a Monad/Comonad
• Underpinning by Cartesian Closed Category
  – Adjointness
  – Composition
  – Product/Exponentiation
  – Finite products
• Generation of Strings
  – Kleisli Category
  – Free Monoids
Current State of Play

• Process
  – Viewed as monad/comonad
  – Three cycles in each direction:
    • One reflective – monad
    • Other anticipatory – comonad
  – Handles transaction concept
    • In databases (ACID)
    • In universe
Example of Adjointness

If conditions hold, then we can write $F \dashv G$

The adjunction is represented by a 4-tuple: $<F, G, \eta, \varepsilon>$

$\eta$ and $\varepsilon$ are unit and counit respectively

Endofunctor $T = GF$
Fig. 2. After three cycles $GFGFGF$ from left-hand category and three cycles $FGFGFG$ from right-hand category: $\eta$ and $\delta$ map onto other than $\bot$, $\top$ maps onto other than $\epsilon$ and $\mu$. 

Monad $= \langle T, \eta, \mu \rangle$

Comonad $= \langle S, \epsilon, \delta \rangle$
Cartesian Closed Category (CCC)

- Underpins applied category theory
- Basis of many fundamental structures in applications
  - Partial order
  - Boolean/Heyting algebras
  - Pullbacks/ Pushouts
  - Scott domains
- Also emerges in lambda calculus
- In computing functions become first-class data
  - Functional programming languages
  - Database design (normalisation)
Definition of CCC paraphrased

- CCC-1 There is a terminal object 1
- CCC-2 Each pair of objects has a product with projections
- CCC-3 There is only one path between the product and the related objects.
In more detail: CCC-1

- For every object $A$ in the category, there is exactly one arrow $A \rightarrow T$
  - $T$ is the terminal object
- Category is closed on top $T$
CCC-2

• Each pair of objects $A$ and $B$ of the category has a product $A \times B$ with projections
  \[ \pi_l : A \times B \rightarrow A \]
  \[ \pi_r : A \times B \rightarrow B \]

• Category has products and projections
  – Giving route to relationships
CCC-3a

• Notion of currying: change function on two variables to a function on one variable
• For function $f: C \times A \rightarrow B$
• Let $[A \rightarrow B]$ be set of functions from $A$ to $B$
• Then there is a function:
  $\lambda f: C \rightarrow [A \rightarrow B]$
  where $\lambda f(c)$ is the function whose value at an element $a \in A$ is $f(c,a)$

• Equivalent to typed lambda calculus
• Examples:
  $f : \text{multiply}(_{,2}) \rightarrow \mathbb{R}$ curries to $\lambda f : \text{double}(_{}) \rightarrow \mathbb{R}$
  $f : \text{exponentiate}(_{,2}) \rightarrow \mathbb{R}$ curries to $\lambda f : \text{square}(_{}) \rightarrow \mathbb{R}$
CCC-3b

• For every pair of objects $A$ and $B$, there is an object $[A \to B]$ and an arrow $\text{eval}: [A \to B] \times A \to B$ with the property that for any arrow $f: C \times A \to B$ (with $C$ being the product object) there is a unique arrow $\lambda f: C \to [A \to B]$ such that the following diagram commutes:
CCC-3c

\[\lambda f \times A \rightarrow [A \rightarrow B] \times A\]

One path from product

\([A \rightarrow B] \text{ is termed } B^A: \text{ all arrows from } A \text{ to } B, \ A \text{ is the exponent of } B\]
Uniqueness

• The category is CCC if (other conditions satisfied) and:
  – $\lambda f$ is unique (one path)

• Notes
  – eval is also a function
  – eval: $[A \rightarrow B] \rightarrow B$
    refers to one $A$ object and its associated $B$ object
  -- eval: $[A \rightarrow B] \times A \rightarrow B$
    refers to all $A$ objects and their associated $B$ objects
Finite Products

- CCC is not restricted to binary products
- Can have finite products
- For any objects $A_1, \ldots, A_n$ and $A$ of a CCC and any $i=1,\ldots,n$, there is an object $[A_i \to A]$ and an arrow:
  $$\text{eval} : [A_i \to A] \times A_i \to A$$
- such that for any $f: \prod A_k \to A$, there is a unique arrow:
  $$\lambda_i f : \prod A_k \to [A_i \to A] \quad (k > 1)$$

Finite products give construction of n-tuples which can represent strings.

Note: this notation may offend Gödel’s theorems!
Abstract View of CCC

• An adjoint relationship
  – $F \dashv G$
  – Free functor $F$ creates binary products
  – Underlying functor $G$ checks for exponentials
    (one path)
Adjointness

• Left adjoint -- free functor on category $\mathbf{C}$:
  $\_ \times A : \mathbf{C} \to \mathbf{C}$
  
  For fixed object $A$ and an object $B$, this gives binary product $B \times A$ and an arrow:
  $f \times id_A : B \times A \to C \times A$

• Right adjoint – underlying functor $G$ on value of object $C$ on right-hand side:
  – Unique arrow $\lambda f : B \to G(C)$ such that $\text{eval } o (\lambda f \times A) = g$

• Adjointness requires both left and right adjoints to exist
Composition for there to be a Right Adjoint

\[ B \times A \xrightarrow{\lambda f \times A} G(C) \times A \]

\[ g \]

One path from product
Compositions for Adjointness with unit/counit

Unit of adjunction $\eta$

Counit of adjunction $\varepsilon$

$\varepsilon$ is eval

$\_XA(G(C))$ is $G\_X(A)$

$\_XA(B)$ is $B\_X(A)$

$\_XA(f)$ is $F\lambda f$
Locally CCC

• Satisfied when:
  – The category $\mathbf{C}$ has pullbacks and either:
    • The pullback functor has a right adjoint OR
    • For every object $A$ in $\mathbf{C}$, the slice category $\mathbf{C}/A$ is cartesian closed

• Pullbacks express relationships over objects in a particular context

• Locally CCC provide more expressiveness in capturing the real world
Product vs Pullback

Product and projections

Pullback of A and B in the context of C
Kleisli Category

• Free algebra
• Based on monad earlier \( T = <T, \eta, \mu> \)
  – where \( T \) is endofunctor \( GF \) for adjoint functors \( F \dashv G \)
  – \( \eta \) is unit of adjunction \( \eta : 1_A \Rightarrow GFA \)
  – \( \mu \) is multiplication \( \mu : GFGF \Rightarrow GF \)
    compares results of 2nd and 1st cycles
  – \( T \) is a category
  – \( A \) is an object in left-hand category
Kleisli Category 2

• In Kleisli category
  – $T = \langle T, \eta, \mu \rangle$
  – The arrows are substitutions
  – $\mu$ can be thought of as carrying out a computation

• For arrow $f : A \to B$
  – then $A \to TB$
  – where $T$ defines the substitutions as functions
Kleisli example

- For \( f : A \to TB \)
  \[ A = \{g, h\} \text{ and } B = \{i, j, k\} \]
  \[ f(g) = \text{cddc}, \ f(h) = \text{ec} \]

- \( Tf: TA \to TTB \)
  \( TA \) is for example string ‘ghhg’
  \( TTB \) is (cddc), (ec), (ec), (cddc) (concatenations)
  \( \mu : TT \to T \) is ‘cddcececcddc’ \( \Rightarrow \) ‘ghhg’

- In the comonad:
  \( \delta : T \to TT \) is ‘ghhg’ \( \Rightarrow \) ‘cddcececcddc’

- So we have string generation through substitution
Kleene Closure

• Given a set $A$:
  – The Kleene closure $A^*$ of a set $A$ is defined as
    • the set of strings of finite length of elements of $A$

• In adjointness terms:
  $F : A \rightarrow A^*$
  $G : A^* \rightarrow A$

• The closure is then $GFA$
• $F$ is the free functor, adding structure
• $G$ is the underlying functor, removing structure
Example

• $A = \{a, b, c, d, \ldots, z\}$ (alphabet)
• $F(A) = A^* = \text{all finite strings constructed from } A \text{ by } F$
• $G(A^*)$ returns the alphabet
• The closure relies on adjointness
  – $F$ can be free and open (all possibilities)
  – $G$ can check for language rules
Example 2

- The adjoint (if it exists) is $<F, G, \eta, \varepsilon>$
  - $F$ constructs all possibilities
  - $G$ applies the language rules
  - $\eta$ defines the change from $A \rightarrow GFA$ in the alphabet
  - $\varepsilon$ defines the change from $FGA^* \rightarrow A^*$ in the language
Summary

• Category Theory provides a number of routes for generating strings:
  – n-tuples through cartesian closed categories
  – String expansion through substitution as in Kleisli categories
  – String generation through free functors as in the Kleene closure