Problems of Interoperability in Information Systems

Nick Rossiter and Michael Heather
CEIS, Northumbria University, UK
nick.rossiter@unn.ac.uk
Four Challenges

• Enterprise Interoperability
• Knowledge-oriented Collaboration
• Web Technologies
• Interoperability Service Utility

Need dynamic connections
What is Underlying Logic?

• Not set theory
  – OK for closed local systems
  – But falls foul of Gödel as higher-order operations needed
  – Neither complete nor decidable outside FOPC
  – CWA is not realistic
  – But experimental verification is valuable

• Not pure category theory
  – Axiomatic
  – So also falls foul of Gödel
Process Logic

• Strong candidate
• Long pedigree
  – Heraclites
  – Whitehead
  – Category theory
  – Cartesian closed categories
Uses of Category Theory

– Cartesian closed categories (CCC, naturality)
– Systems theory with Heyting logic (open systems)
– Topos (SoS)
– Monad (transaction logic, process)
– Adjointness (relationships)
– 2-categories (vertical + horizontal composition)
– Higher-order logic in CCC
  • Without axioms and reliance on number
  • Gödel free in connecting systems in our view
– For good practice, avoid categorification
Twin-track Approach

• Two subsystems

1. Data Structures and Rules
   – 3-level architecture
   – In terms of mappings A → B → C → D
     • With dual D → C → B → A

2. Behaviour
   – 3-level architecture
   – In terms of cycles F: A → B; G: B → A
     • GF 3 times
     • FG 3 times
Example of Adjointness

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

The adjunction is represented by a 4-tuple: $\langle F, G, \eta, \varepsilon \rangle$.

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

$L, R$ are categories; $F, G$ are functors.
Data Structures and Rules

A is category for Concepts
B is category for Constructs
C is category for Schema
D is category for Data

Adjunctions compose naturally
F-|G is one of 6 adjunctions (if they hold)
Principles

- Have pairs of abstractions
- Each level is defined by level above
- Adjunctions permit relationships less than equivalence between the levels
- Having more than three levels of abstraction does not achieve greater precision
- Can be viewed as multi-level type subsystem
Six Possible Adjunctions

\[ \begin{align*}
F &\mid G \\
\overline{F} &\mid \overline{G} \\
\overline{F} &\overline{F} - \mid \overline{G} \overline{G} \\
\overline{F} \overline{F} \overline{F} &\mid \overline{G} \overline{G} \overline{G} \\
\overline{F} \overline{F} \overline{F} \overline{F} &\mid \overline{G} \overline{G} \overline{G} \overline{G}
\end{align*} \]
Adjunctions in More Detail
Simple Pairs

We can define these in more detail with their units and counits of adjunction as follows:

\[ < F, G, \eta_a, \epsilon_b > : A \rightarrow B \]  (1)
\[ \eta_a \text{ is the unit of adjunction } 1_a \rightarrow GFa \text{ and } \epsilon_b \text{ is the counit of adjunction } FGb \rightarrow 1_b \]

\[ < \tilde{F}, \tilde{G}, \tilde{\eta}_b, \tilde{\epsilon}_c > : B \rightarrow C \]  (2)
\[ \tilde{\eta}_b \text{ is the unit of adjunction } 1_b \rightarrow \tilde{G}\tilde{F}b \text{ and } \tilde{\epsilon}_c \text{ is the counit of adjunction } \tilde{F}\tilde{G}c \rightarrow 1_c \]

\[ < \tilde{F}, \tilde{G}, \tilde{\eta}_c, \tilde{\epsilon}_d > : C \rightarrow D \]  (3)
\[ \tilde{\eta}_c \text{ is the unit of adjunction } 1_c \rightarrow \tilde{G}\tilde{F}c \text{ and } \tilde{\epsilon}_d \text{ is the counit of adjunction } \tilde{F}\tilde{G}d \rightarrow 1_d \]
Adjunctions in More Detail

Doubles

\[ \langle \tilde{F} \tilde{F}, G \tilde{G}, G \tilde{\eta}_a F \cdot \eta_a, \tilde{\epsilon}_c \cdot \tilde{F} \epsilon_c \tilde{G} \rangle : A \longrightarrow C \]

\( G \tilde{\eta}_a F \cdot \eta_a \) is the unit of adjunction \( 1_a \longrightarrow G \tilde{G} \tilde{F} F a \) and \( \tilde{\epsilon}_c \cdot \tilde{F} \epsilon_c \tilde{G} \) is the counit of adjunction \( \tilde{F} \tilde{F} G \tilde{G} c \longrightarrow 1_c \).

The unit of adjunction is a composition of \( \eta_a : 1_a \longrightarrow G F a \) with \( G \tilde{\eta}_a F : G F a \longrightarrow G \tilde{G} \tilde{F} F a \).

The counit of adjunction is a composition of \( \tilde{F} \epsilon_c \tilde{G} : \tilde{F} \tilde{F} G \tilde{G} c \longrightarrow \tilde{F} \tilde{G} c \) with \( \tilde{\epsilon}_c : \tilde{F} \tilde{G} c \longrightarrow 1_c \).

We have retained the symbol \( \cdot \) indicating vertical composition as distinct from normal horizontal composition indicated by the symbol \( \circ \) [13].

\[ \langle \tilde{F} \tilde{F}, \tilde{G} \tilde{G}, \tilde{G} \tilde{\eta}_b F \cdot \tilde{\eta}_b, \tilde{\epsilon}_d \cdot \tilde{F} \epsilon_d \tilde{G} \rangle : B \longrightarrow D \]

\( \tilde{G} \tilde{\eta}_b \tilde{F} \cdot \tilde{\eta}_b \) is the unit of adjunction \( 1_b \longrightarrow \tilde{G} \tilde{G} \tilde{F} F B \) and \( \tilde{\epsilon}_d \cdot \tilde{F} \epsilon_d \tilde{G} \) is the counit of adjunction \( \tilde{F} \tilde{F} G \tilde{G} d \longrightarrow 1_d \).

The unit of adjunction is a composition of \( \tilde{\eta}_b : 1_b \longrightarrow \tilde{G} \tilde{F} b \) with \( \tilde{G} \tilde{\eta}_b \tilde{F} : \tilde{G} \tilde{F} b \longrightarrow \tilde{G} \tilde{G} \tilde{F} F b \).

The counit of adjunction is a composition of \( \tilde{F} \epsilon_d \tilde{G} : \tilde{F} \tilde{F} G \tilde{G} d \longrightarrow \tilde{F} \tilde{G} d \) with \( \tilde{\epsilon}_d : \tilde{F} \tilde{G} d \longrightarrow 1_d \).
Adjunctions in More Detail
Triples

\[ < \tilde{F} F F, G G G, G G \tilde{\eta}_a F F \bullet G \tilde{\eta}_a F \bullet \eta_a, \tilde{\epsilon}_d \bullet \tilde{F} \tilde{\epsilon}_d G \bullet \tilde{F} F \tilde{\epsilon}_d G \tilde{G} > : A \to D \]  

The unit of adjunction is a composition of:
\[ \eta_a : 1_a \to G F_a \text{ with } G \tilde{\eta}_a F : G F_a \to G G F F a \text{ with } G G \tilde{\eta}_a F F : G G F F a \to G G G F F F a \]

The counit of adjunction is a composition of:
\[ \tilde{F} F \tilde{\epsilon}_d G G : \tilde{F} F F G G G d \to \tilde{F} F G G d \text{ with } \tilde{F} \tilde{\epsilon}_d G : \tilde{F} F G G d \to \tilde{F} G d \text{ with } \tilde{\epsilon}_d : \tilde{F} \tilde{G} d \to 1_d \]
Desired Properties

• If all adjunctions hold
  – Have clearly-defined multi-level type subsystem

• Can relate one subsystem to another by
  – Natural transformation
    • Maps between functors

• Provides interoperability between subsystems for
  – Data structures and rules
Natural Transformation

$\alpha$ is natural transformation comparing $F$ and $F'$
Behaviour/Anticipation
Monad/Comonad

• Define subsystem
  – Handle transactions
    • ACID properties
      • Atomicity, Consistency, Isolation, Durability
  – Have 3 cycles
    • 1. make changes
    • 2. review changes
    • 3. holistic check that all is well
  – Example with Bank ATM:
    • 1. debit account
    • 2. check funds available
    • 3. holistic check that all changes recorded safely
Monad

- Construction for transactions is the Monad
- Monad is a triple \(<T, \eta, \mu>\)
  - \(T\) is an endofunctor (functor with same source and target)
    - e.g. \(GF : A \rightarrow B \rightarrow A\)
  - \(\eta\) is unit of adjunction e.g. \(1_L \rightarrow GF(L)\)
    - Compares initial value for object \(L\) with value for \(L\) after one cycle
  - \(\mu\) is multiplication \(T^2 \rightarrow T\)
    - comparing result from 2\(^{nd}\) cycle with 1\(^{st}\)
    - e.g. \(GFGF \rightarrow GF\)
- Full details of definition involve \(T^3\) (GFGFGGF)
Comonad

- Monad gives left-hand-perspective (L)
- Comonad gives right-hand perspective (R)
- Comonad is a triple \( <S, \varepsilon, \delta> \)
  - \( S \) is FG
    - e.g. \( B \rightarrow A \rightarrow B \)
  - \( \varepsilon \) is counit of adjunction e.g. \( FG(R) \rightarrow 1_R \)
  - \( \delta \) is comultiplication \( T \rightarrow T^2 \)
    - Anticipation – looking forward
Figure 2: After three cycles $GFGFGF$ from left-hand category and three cycles $FGFGFG$ from right-hand category: $\eta$ and $\delta$ map onto other than $\bot$, $T$ maps onto other than $\epsilon$ and $\mu$. 
System Viewpoint for Interoperability

• Have a system formed from 2 subsystems
  – For data structures/rules
    • 3 levels of mapping as functors between categories
    • Each mapping represents a level-pair of abstractions
  – For behaviour
    • 3 cycles as a monad/comonad structure

• Interoperability
  – Comparing one system with another by natural transformations or higher-order categories

• Recent work on Security by PhD student Dimitris Sisiaridis with category theory produces the system unification
Possible Way Forward

• Not for everybody to learn category theory!
• Development of tool
  – Assist with interoperability
  – Based on process category theory
  – Graphical
  – Haskell is a candidate
    • Facilities include monads