Towards the Principled Design of Diagrams in Computing and Software Engineering

Kostas Tourlas

with thanks to:
Corin Gurr and John Power

October 27, 2000

Diagrams Status Quo

- In **Mathematics** and **Logic**:
  - Euler circles, Venn charts, Piercian graphs
  - Diagrammatic reasoning regarded by formal linguistic tradition as unreliable and lacking in rigour
  - Category theory
    - relies extensively on commutative diagrams
    - formalises and internalises its diagrams
    - Joyal and Street
- In **Computing**:
  - Informal/occasional uses abound (FSMs, Petri nets, . . .)
  - Virtually no diagrammatic reasoning
  - Visual Programming Languages
  - Specification: Statecharts

Lack of widely practiced diagrammatic notations sets computing apart from traditional engineering disciplines.

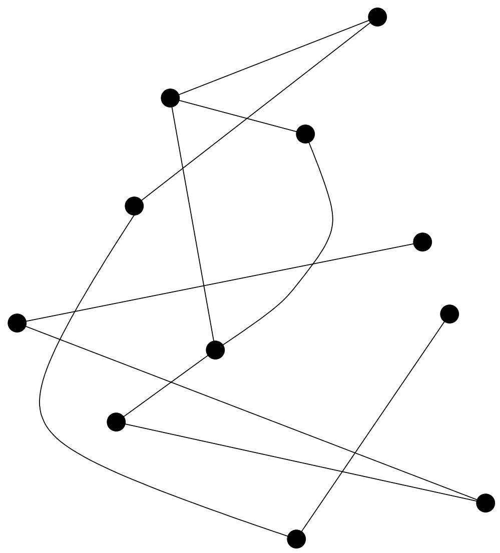
What is going on?

Recent **industrial** interest in computing with diagrams:

- IEC languages for industrial process control
  - diagrammatic **and** textual
  - derived from existing notations in domain
  - mixture of grammatical and informal definition styles
- Unified Modelling Language (UML)
  - origins in OO design
  - major standardisation effort
  - diagram-based tools (e.g. Rational Rose)
  - semantic ‘definition’ by tool specification
  - far from perfect, but a great step

Thorough, **open-minded**, application-informed **theory** is urgently needed!

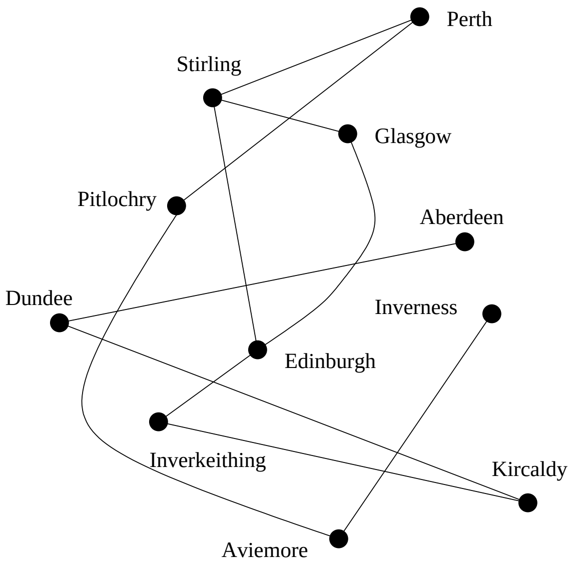
A Graph



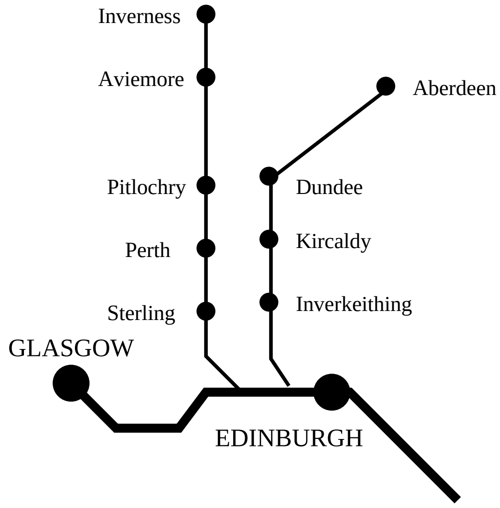
Pure diagram

11 nodes (objects), symmetric relation

A Scottish Graph



GNER Routes in Scotland



Inaccuracies do not impede task performance; rather they attempt to assist it!

Purpose and Anatomy of Diagrams

Good diagrammatic notations are **effective**.

- Effectiveness is:
  - **cognitive** in nature
  - relative to **task(s)**
- Common tasks are:
  - communication
  - reasoning/inference
- Main aspects of diagrams:
  - **Syntax**: what the diagram consists of
    - visual objects (nodes, icons, shapes)
    - visual relations (connectivity, containment, adjacency)
  - **Semantics**: how the diagram is interpreted
  - **Pragmatics** (secondary features): layout, colour, line thickness

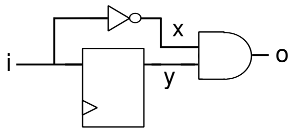
Objectives of a Theory

- Typically, diagrams are
  - **analogical** representations
  - ... of **static** artifacts or situations
- In computing, diagrams are
  - typically **specifications** or **programs**
  - interpreted as computational processes or structures (dynamics)
  - less analogical in nature
- Mathematical understanding of computational interpretation of diagrams
- Cognitive understanding of their use
- The theory should:
  - yield principles for designing effective diagrams
  - justify informal practice and reasoning
  - underpin tools to support best practice
- Notation and domain specificity is the key.

Formalisation: the Practitioner's Joy

Fisler's example . . .

- a single pulser:



- its "typical" formal spec:

$$
\begin{aligned}
SP(I,O) &= \exists x\colon \exists y\colon\ delay(I,x) \wedge \\
&\quad\ not(i,y) \wedge and(x,y,O) \\
delay(I,O) &= \forall t\colon\ O(t+1) = I(t) \\
not(I,O) &= \forall t\colon\ O(t) = 1 - I(t) \\
and(I_1,I_2,O) &= \forall t\colon\ O(t) = I_1(t) \times I_2(t)
\end{aligned}
$$

Maths as a tool of the notation designer, not as imposition on the user.

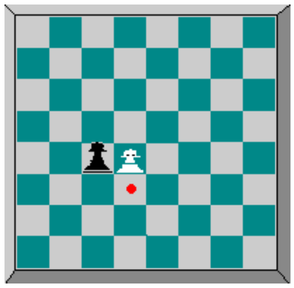
Limitations

Few diagrammatic notations are purely such.

Pure diagrams typically enjoy **specificity**, or **limited abstraction** (K. Stenning)

Chess configurations:

$$\exists x.\exists y.\mathrm{Pawn}(x) \wedge \mathrm{Pawn}(y)$$



Limited expressiveness is often key to effectiveness!

Might be why UML is succeeding where VDM failed . . . (OCL least popular part of UML!)

Diagrams as Programs

Nardi's view

- DSL

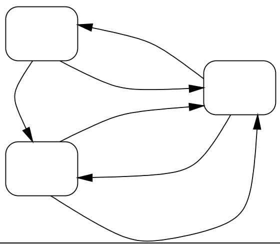
  = "core" visual formalism (table, graph, . . .)
  \+ domain extension/specialisation
  \+ linguistic labelling

- Issues
  - Cognitive
    - limits of extension
    - expressiveness vs. effectiveness
  - Modularity
  - Semantic (property preservation)

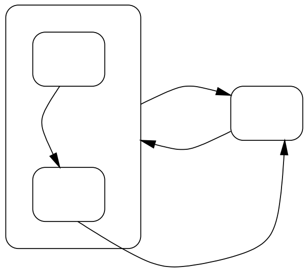
higraphs = graphs + depth

Domain: Reactive system specification.

Graph:



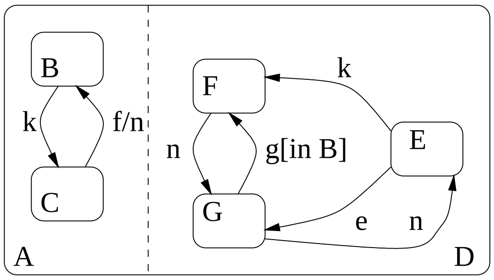
Higraph:



Depth accommodates the domain-specific concepts of **subsystem** and **interrupt**.

. . . + labelling

Statecharts:

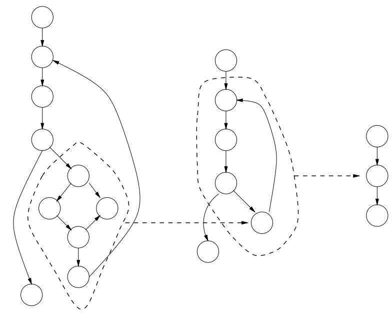


- Labels encode:
  - domain-specific events
  - synchronisation
- Linguistic elements "understand" diagrammatic primitives!
- Rich interaction between modalities.

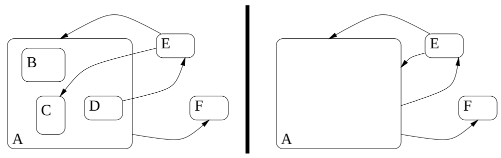
Tangled semantics?

Common Manipulations

- Chunking:



- Zooming out:



Not all diagrams,or semantics, support either!

Summary: Main Issues

- Semantic admissibility of
  - extensions
  - manipulations
- Effects on reasoning
- Mathematical structure of
  - manipulations
  - inter-modality interaction

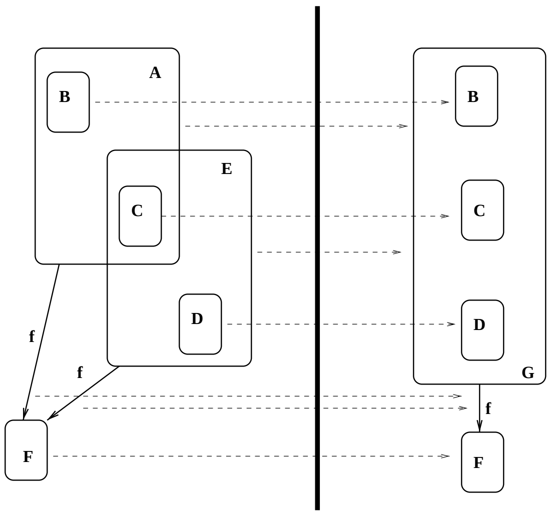
Why Categories?

Categories on a "need-to-know" basis

Diagrams as tuples $(S_1, \ldots, S_n; r_1, \ldots, r_m)$ of sets and relations.

A **category** has both **objects** and **mappings** between objects which preserve object features.

Example morphism between higraphs:



Higraphs, categorically

Tuples (B, σ, E, s, t) with $s, t : E \to B$ giving source and target, $\sigma : B \to 2^B$ giving "sub-blobs".

σ is a partial order on B ($b \leq_\sigma b'$) and induces one on E too:

$$e \leq e' \iff s(e) \leq_\sigma s(e') \text{ and } t(e) \leq_\sigma t(e')$$

Moreover, s, t are monotone maps.

Two posets B, E and two monotone maps $s, t : E \to B$.

Objects: functors χ from $\bullet \rightrightarrows \bullet$ to $\mathbf{Poset}$.

Arrows: $f : \chi \to \chi'$ where f is an "oplax" natural transformation:

$$\begin{tikzcd} E \arrow[r, "f"] \arrow[d, "s"'] \arrow[dr, phantom, "\leq"] & E' \arrow[d, "s'"] \\ B \arrow[r, "f"'] & B' \end{tikzcd} \qquad \begin{tikzcd} E \arrow[r, "f"] \arrow[d, "t"'] \arrow[dr, phantom, "\leq"] & E' \arrow[d, "t'"] \\ B \arrow[r, "f"'] & B' \end{tikzcd}$$

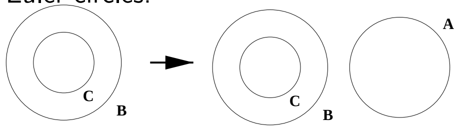
Category: $\mathit{Oplax}\left[\bullet \rightrightarrows \bullet, \mathbf{Poset}\right]$

"Theory" of a Diagram

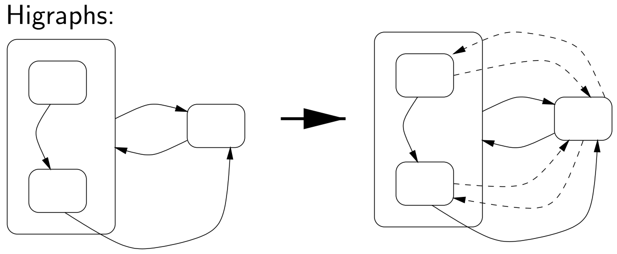
theory = axioms (facts) + consequences

"Emergent features" and "free rides" in diagrams

- Euler circles:



- Higraphs:



Theory of a Higraph

$$\chi: \quad \begin{tikzcd} E \arrow[d, shift right, "s"'] \arrow[d, shift left, "t"] \\ B \end{tikzcd} \qquad T(\chi): \quad \begin{tikzcd} E' \arrow[d, shift right, "s'"'] \arrow[d, shift left, "t'"] \\ B \end{tikzcd}$$

Common poset B of blobs.

E' is characterised as a "comma object" in $\mathit{Oplax}\left[\bullet \rightrightarrows \bullet, \mathbf{Poset}\right]$: the universal poset E' such that

$$\begin{tikzcd} E' \arrow[r] \arrow[d, "\langle s', t' \rangle"'] \arrow[dr, phantom, "\leq"] & E \arrow[d, "\langle s, t \rangle"] \\ B \times B \arrow[r, "\mathrm{id}"'] & B \times B \end{tikzcd}$$

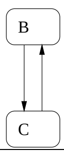
commutes.

The mapping T extends to a functor, right adjoint to the inclusion

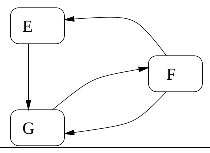
$$\left[\bullet \rightrightarrows \bullet, \mathbf{Poset}\right] \xrightarrow{J} \mathit{Oplax}\left[\bullet \rightrightarrows \bullet, \mathbf{Poset}\right]$$

An Operation on Higraphs

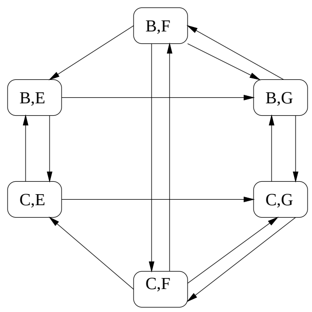
χ_1:



χ_2:



$\chi_1 \otimes \chi_2$:



Graphs in a Category

For C any category, $\mathbf{Grph}(C) = [\bullet \rightrightarrows \bullet, C]$.

When C is **cartesian closed** and has **binary coproducts**, $\mathbf{Grph}(C)$ is **symmetric monoidal**.

In $\mathbf{Grph}(\mathbf{Poset})$, the monoidal product $\otimes$ yields the operation of higraph interleaving.

Benefit: may specialise higraphs (e.g. take $\mathbf{SemiLatt}$ instead of $\mathbf{Poset}$) but structure and properties of interleaving remain the same!

Conclusion

- Presented
  - roadmap to a theory of diagrams in computing
  - initial work on "austere" higraphs
- Motivated by earlier work on PLC diagrams and need for Occam's razor
- Category theory
  - emphasises relations among diagrams as morphisms
  - may help to draw general lessons and conclusions
- Future work
  - realistic graph-based diagrams
  - supporting tools