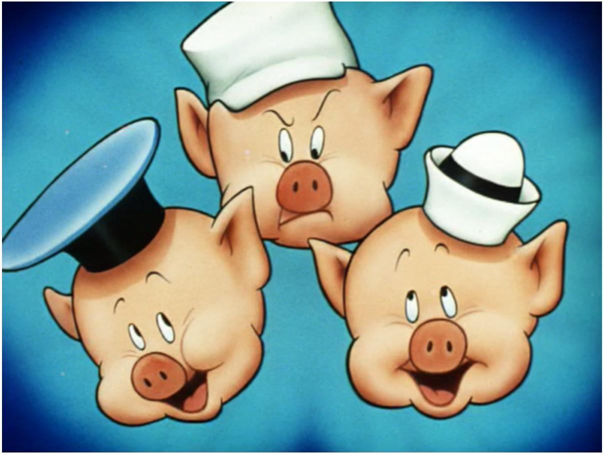
Emergent effects and contextual behaviours in categorical systems theory

Jules Hedges[1]

[1]MSP Group, University of Strathclyde, Écosse Libre
20squares.xyz cybercat.institute

4th November 2022
NIST

Open games experience report



2020 conclusions slide

- Number 1 conclusion:

  Realising the promised benefits of ACT is still hard

- Need detailed and equal dialogue between theory & domain experts
- Interdisciplinary work is very costly
- Designing good abstractions will always be an art form
- Software is necessary, string diagrams software not necessary
- String diagrams may not even be the best representation!

Categorical systems theory

We often consider categories where:

- Morphisms are some kind of open systems
- Objects are their boundaries

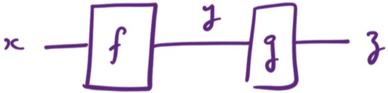
left boundary $\quad x \xrightarrow{f} y \quad$ right boundary

N.b. Why a category?

- i.e. why a strict separation into left and right boundaries?

We don't need a category! Other operad algebras work too! Categories are just convenient!

Complex systems



The composition fg is coupling along a common boundary, and yields a complex system, i.e. a system that is a complex of smaller parts

Usually $\mathcal{C}$ also admits a monoidal structure[1] for disjoint (non-coupling) composition

[1] Usually much more, eg. †-compact closed

Systems vs. processes

Side remark:
As well as systems theories we also have process theories

Most of this talk still applies, replace "left boundary" and "right boundary" with input and output

Behaviour

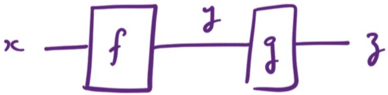
To each boundary x we associate a set $B(x)$ of possible behaviours that can be observed on that boundary

To each open system $f : x \to y$ we associate a set $B(f) \subseteq B(x) \times B(y)$

- $(a, b) \in B(f)$ means "it is possible to simultaneously observe a on the left boundary and b on the right boundary of f"

No observations can be made except on the boundary

Behaviours compose



Suppose f and g share a common behaviour on their common boundary:

$$(a, b) \in B(f) \text{ and } (b, c) \in B(g) \text{ for some } b \in B(y)$$

In most situations, this implies that $(a, c) \in B(fg)$ [2]
So:

$$B(f)B(g) \subseteq B(fg)$$

(LHS composition in Rel)

[2] If your behaviour doesn't satisfy this, you should probably try something else

In fancy terminology:

B is a lax functor[3] $\mathcal{C} \to \mathsf{Rel}$

- $\mathcal{C}$ is locally discrete 2-category (exactly one 2-cell)
- Rel is a locally thin 2-category (at most one 2-cell)

[3] Or lax pseudofunctor if being pedantic

Emergent behaviours

In many practical situations, the converse fails:

fg can exhibit "emergent" behaviours that do not arise from individual behaviours of f and g

Slick definition 1/3: An emergent behaviour of fg (w.r.t. the decomposition (f, g)) is an element of $B(fg) \setminus B(f)B(g)$

So we do not have a functor $B : \mathcal{C} \to \mathsf{Rel}$

In practice: This is much less interesting
Functoriality sometimes fails very badly in real examples

Example: Open graph reachability[4]

OGph = structured cospan category of open graphs

- Objects: finite sets
- Morphisms: cospans of graph homomorphisms

$$L(X) \xrightarrow{\iota_1} G \xleftarrow{\iota_2} L(Y)$$

$L(-)$ = discrete graph on a set

[4] Not really systems theory, but easy to understand and easy to visualise

Reachability

Define $B : \mathsf{OGph} \to \mathsf{Rel}$ by:

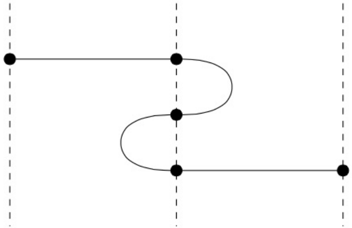
- On objects: $B(X) = X$
- On morphisms: $B(X \xrightarrow{\iota_1} G \xleftarrow{\iota_2} Y) =$

$$\{(x, y) \in X \times Y \mid \iota_1(x) \text{ and } \iota_2(y) \text{ are connected in } G\}$$

Proposition. B is a lax functor

Reachability is not a functor

Minimal counterexample: the zig-zag



$$L(1) \longrightarrow f \longleftarrow L(3) \longrightarrow g \longleftarrow L(1)$$

$$B(f)B(g) = \varnothing \subsetneq \{(*,*)\} = B(fg)$$

Better but handwaved examples

In general, naive compositionality is not enough
Relevant effects can cut across the obvious compositional structure

- High frequency electronics: behaviour “jumps” across the logical circuit structure between physically nearby components
- Safety/failure analysis: catastrophic failures can cascade through physically nearby components
- A system (e.g. an agent inside it) reasons about its situation instead of passively reacting

Not this talk: A grand challenge

Given a lax functor to Rel [5] associate some mathematical object (cohomology?) that 'describes' how it fails to be a functor (i.e. how the laxator fails to be an iso), in a useful way

"Useful" = encodes something worth knowing about emergent behaviour

[5] Or linear/additive relations etc. as convenient

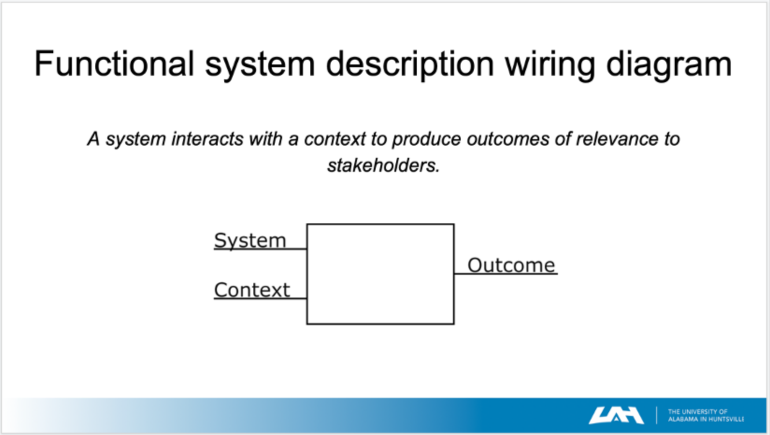
Pre-feature cartoon

Categorical systems theory

A theory of contexts

Contextual functors

Outcomes depend on contexts too



[6]

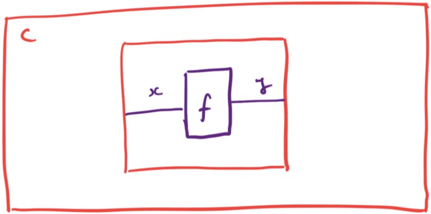
[6] This slide courtesy of David Perner, University of Alabama in Huntsville, dp0101@uah.edu

The idea of contexts

Idea: a context is a hole into which a morphism can be put

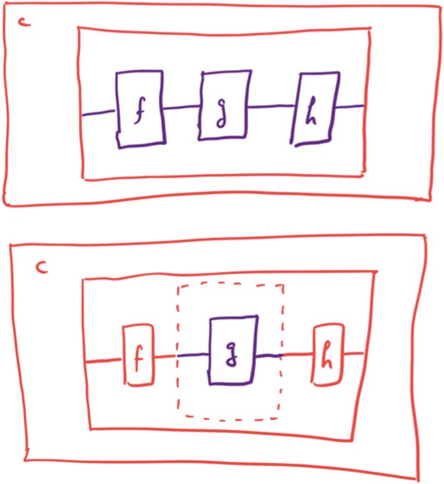
Later we'll make behaviours depend on both a system and a context

Standard ACT methodology: axiomatise the structure they must have, leaving freedom to do domain-specific things later



Contexts form a functor

Write $\overline{\mathcal{C}}(X, Y)$ for the set of possible contexts for morphisms $X \to Y$



$\overline{\mathcal{C}} : \mathcal{C} \times \mathcal{C}^{\mathrm{op}} \to \mathrm{Set}$

Generalised states

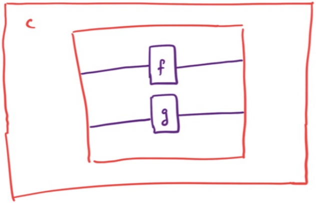
A functor $F : \mathcal{C} \to \mathrm{Set}$ describes a theory of generalised states (things that can be stuck to a left boundary) [7]
An element $e \in F(X)$ behaves like a morphism $e : I \to X$

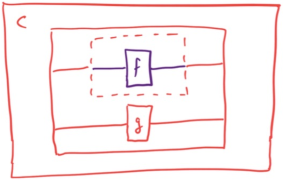
Similarly, a functor $G : \mathcal{C}^{\mathrm{op}} \to \mathrm{Set}$ describes a theory of generalised costates (things that can be stuck to a right boundary)
An element $e \in G(X)$ behaves like a morphism $e : X \to I$

A context could be a pair of these: $\overline{\mathcal{C}}(X, Y) = F(X) \times G(Y)$
This describes a thing stuck to the left boundary and another (disjoint) thing stuck to the right boundary

[7] It also ought to be lax monoidal to $(\mathrm{Set}, \times)$

Contexts for a monoidal category

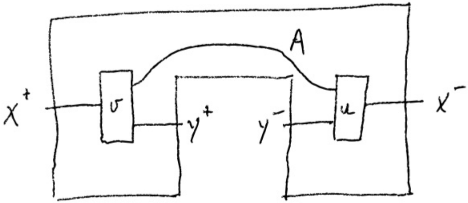




This is less obviously described by functoriality
In particular this is not just a monoidal functor

The obligatory optics slide

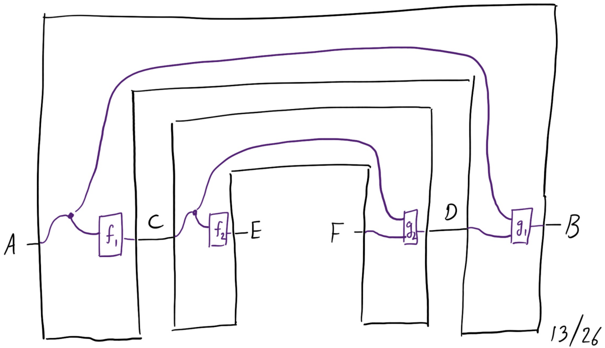
An optic $(X^+, X^-) \to (Y^+, Y^-)$ in a monoidal category is a pair of morphisms like this:



modulo sliding things along the A wire:

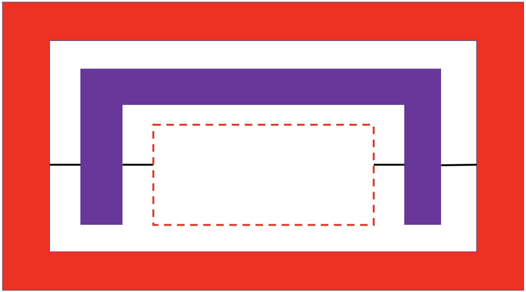
$$\mathrm{Optic}(\mathcal{C})(X, Y) = \int^{A:\mathcal{C}} \mathcal{C}(X^+, A \otimes Y^+) \times \mathcal{C}(A \otimes Y^-, X^-)$$

Optic composition goes outside-in



Generalised states in optics

A functor $\mathrm{Optic}(\mathcal{C}) \to \mathrm{Set}$ describes things that can be stuck to the outside boundary, and can be extended like this:



Slick definition 2/3: A context functor for a monoidal category $\mathcal{C}$ is a lax monoidal functor $\overline{\mathcal{C}} : \mathrm{Optic}(\mathcal{C}) \to (\mathrm{Set}, \times)$

Examples of context functors

- The representable one [8]:

$$\overline{\mathcal{C}}(X, Y) = \mathrm{Optic}(\mathcal{C})((I, I), (X, Y))$$

$$= \int^{A:\mathcal{C}} \mathcal{C}(I, A \otimes X) \times \mathcal{C}(A \otimes Y, I)$$

- If $\mathcal{C}$ is traced, $\overline{\mathcal{C}}(X, Y) = \mathcal{C}(Y, X)$ is a context functor
- If $\mathcal{C}$ is compact closed, $\overline{\mathcal{C}}(X, Y) = \mathcal{C}(I, X \otimes Y)$ is a context functor
- Domain-specific examples can be tailored to individual problems

[8] Open games uses this one (where $\mathcal{C}$ is itself a category of optics!)

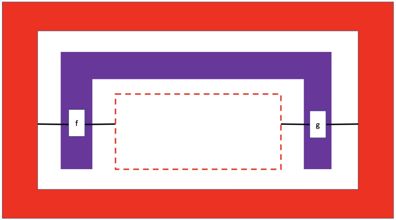
Optics can usually be avoided

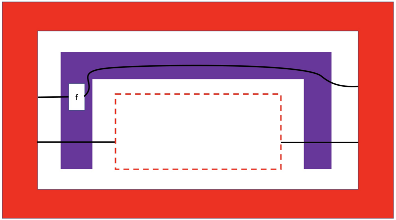
Useful lemma: If $\mathcal{C}$ is compact closed, then $\mathrm{Optic}(\mathcal{C}) \cong \mathrm{Int}(\mathcal{C})$

Int-construction[9]: objects are pairs, morphisms $(X^+, X^-) \to (Y^+, Y^-)$ are morphisms $X^+ \otimes Y^- \to Y^+ \otimes X^-$

[9] Universal property: forget the compact closed structure, freely add a new one generating the same trace

How a context transforms





Morphisms in context

Given:

- A monoidal category $\mathcal{C}$ ("systems")
- A context functor $\overline{\mathcal{C}} : \mathrm{Optic}(\mathcal{C}) \to \mathrm{Set}$
- A monoidal category $\mathcal{D}$ ("semantics")

we can form a new category $\mathcal{D}[\overline{\mathcal{C}}]$

- Objects are pairs $(X \in \mathcal{C}, A \in \mathcal{D})$
- A morphism $(X, A) \to (Y, B)$ is a pair $f : \mathcal{C}(X, Y)$ and

$$\langle - | f \rangle : \overline{\mathcal{C}}(X, Y) \to \mathcal{D}(A, B)$$

Thats a system together with a behaviour for every possible context [10]

[10] It looks kinda like a fibred category / Grothendieck construction, but it's not

The yoga of contexts

$$(X, A) \xrightarrow{f, \langle - | f \rangle} (Y, B) \xrightarrow{g, \langle - | g \rangle} (Z, C)$$

We need to get a function

$$\langle - | fg \rangle : \overline{\mathcal{C}}(X, Z) \to \mathcal{D}(A, C)$$

From $c \in \overline{\mathcal{C}}(X, Z)$ and $g : \mathcal{C}(Y, Z)$ we can get $g^* c \in \overline{\mathcal{C}}(X, Y)$, and then $\langle g^* c | f \rangle : \mathcal{D}(A, B)$
From $c \in \overline{\mathcal{C}}(X, Z)$ and $f : \mathcal{C}(X, Y)$ we can get $f_* c \in \overline{\mathcal{C}}(Y, Z)$, and then $\langle f_* c | g \rangle : \mathcal{D}(B, C)$
Contextual composition law: [11]

$$\boxed{\langle c | fg \rangle = \langle g^* c | f \rangle \, \langle f_* c | g \rangle}$$

[11] It looks a tiny bit like a product rule for a derivation if you squint really hard

The contextual category

This rule is associative
The corresponding identity on (X, A) is id_X together with

$$\langle c|\mathrm{id}_X\rangle = \mathrm{id}_A \text{ for all } c \in \overline{\mathcal{C}}(X, X)$$

That is: the identity system may only perform the identity behaviour, no matter what context it is in. This is not as innocent as it sounds!

$\mathcal{D}[\overline{\mathcal{C}}]$ is also a monoidal category - this is where we seriously use Optic($\mathcal{C}$)
Fun fact: This isolates 1 of 3 ingredients making up open games [12]

[12] Bonus fun fact: optics appear in compositional game theory for 2 apparently unrelated reasons !!

Contextual functors

There's a forgetful functor $U : \mathcal{D}[\overline{\mathcal{C}}] \to \mathcal{C}$

- $U(X, A) = X$
- $U(f, \langle - | f \rangle) = f$

Slick definition 3/3: A $\overline{\mathcal{C}}$-contextual functor $C \to D$ is a section $\mathcal{C} \to \mathcal{D}[\overline{\mathcal{C}}]$ of U

Unpacking the definition

To specify a contextual functor $\mathcal{C} \to \mathcal{D}$ is equivalently to give:

- For each object X of $\mathcal{C}$, an object $F(X)$ of $\mathcal{D}$
- For each morphism $f : \mathcal{C}(X, Y)$ and context $c \in \overline{\mathcal{C}}(X, Y)$, a morphism $\langle c|f\rangle : \mathcal{D}(F(X), F(Y))$

satisfying 2 conditions:

- (weird-unitality) $\langle c|\mathrm{id}_X\rangle = \mathrm{id}_{F(X)}$
- (weird-associativity) $\langle c|fg\rangle = \langle g^* c|f\rangle \langle f_* c|g\rangle$

The punchline: when I originally tried to define a "contextual functor" by brute force, this is almost the definition I came up with [13]

[13] I left out the weird-uniltality law, which is a headache

Contexts for graph reachability

It's enough to know that some boundary nodes are connected via other components

Idea: define $\overline{\mathsf{OGph}} : \mathsf{Int}(\mathsf{OGrph}) \to \mathsf{Set}$ by

$$\overline{\mathsf{OGph}}(X, Y) = \{\text{partitions of } X\} \times \{\text{partitions of } Y\}$$

Work needed to check this really is a functor!

Note: I find

$$\overline{\mathsf{OGph}}(X, Y) = \{\text{partitions of } X + Y\} = \{\text{corelations } X \to Y\}$$

more intuitive, but it goes wrong for subtle reasons

How not to do reachability

Obvious idea: define a contextual functor OGph $\to$ Rel by

$$\langle L, R | G \rangle = \{\text{reachability in } G \text{ after identifying } L\text{-equivalent and } R\text{-equivalent nodes}\}$$

Weird-unitality fails: $\langle L, R | \mathrm{id} \rangle$ might not be the identity relation!

This seems to be a common phenomenon

How maybe to do reachability

Brute-forcing the problem from the previous slide:

$$\langle L, R | G \rangle = \{(x, y) \mid (x, y) \text{ reachable in } G + \text{edges from } L, R,$$

$$\text{by a path either of length 0, or taking at least one step in } G\}$$

Bunch of fiddly combinatorics needed to prove this really is a contextual functor

Outlook

- Most importantly: actually compelling examples needed
- Normally, a functor leads to a linear time divide-and-conquer algorithm
- Contextual functors do not yield efficient algorithms!
- This is the fundamental bamboozle of open games: how to get "compositionality of Nash equilibria" without implying that $P = NP$
- I think this could become a standard tool of ACT (similar to decorated cospans)