Applied category theory
The emerging science of compositionality
Category theory
Applied category theory

prepare lemon meringue pie

prepared crust

lemon
butter
sugar
egg

separate egg

yolk
white

sugar

make lemon filling

lemon filling

fill crust

unbaked lemon pie

unbaked pie

make meringue

meringue

add meringue

unbaked pie

From “Seven Sketches of Compositionality”
Fong & Spivak (2018)
Applied category theory

- Emerging field!
- Explosion of papers, research, development in last 5 years
- Builds a common vocabulary of composition across disciplines
- Enables intuitive, precise visual representations
Brand new ACT Journal (2019?)

About

Compositionality describes and quantifies how complex things can be assembled out of simpler parts. Compositionality, the journal, is an open-access journal for research using compositional ideas, most notably of a category-theoretic origin, in any discipline. Topics may concern foundational structures, an organizing principle, or a powerful tool. Example areas include but are not limited to: computation, logic, physics, chemistry, engineering, linguistics, and cognition.

Compositionality is free of cost for both readers and authors. You can find our editorial policies here, and community discussion here. Our ISSN is 2631-4444.
Seven Sketches in Compositionality:
An Invitation to Applied Category Theory

Brendan Fong     David I. Spivak
The Mathematics of Networks
9:30 am – 10:30 am
Speaker: John Baez, UC

To describe systems conceptually, Petri nets, electrical circuits, and other like. In principle all these concepts originate on category theory. This work is less rewarding, than this work.

Compositional Thermodynamics
9:30 am – 10:30 am
Speaker: Giulio Chiribella, The University of Cambridge

Thermodynamics is one of the grand engines to the grand scale of thermodynamics. This work is less rewarding, than this work.

Composition and Quantum Theory: A Conjecture, and How it Could Fail
11:00 am – 11:20 am
Speaker: Markus Mueller, Western University

Suppose that a convex-operational theory has two properties: tomographic locality and operational sensitivity. Naturally, a

A Topological Approach to Compositionality in Complex Systems
11:50 am – 12:10 pm
Speaker: Emanuela Merelli, University of Camerino

According to Frege’s principle of compositionality, the meaning of a complex expression is composed of the meanings of its parts.

Compositionality in Cybersecurity
11:25 am – 11:45 am
Speaker: Paola Mamp거리, Queen Mary University of London

Diagrammatic Semantics for Digital Circuits

We introduce a general diagrammatic theory that fills in the foundational gap in reasoning syntactically.

Funcorlral Data Migration

David I. Spivak
(Submitted on 29 Mar 2017)

(Submitted on 6 Sep 2010 (v1), last revised 3 Feb 2013 (this version, v4))
Q: Is functional programming applied category theory?
Q: Is functional programming applied category theory?

A: Yes! (...arguably)
Monads for functional programming

Philip Wadler, University of Glasgow*

Department of Computing Science, University of Glasgow, G12 8QQ, Scotland
(wadler@cs.gla.ac.uk)

Abstract. The use of monads to structure functional programs is described. Monads provide a convenient framework for simulating effects found in other languages, such as global state, exception handling, output, or non-determinism. Three case studies are looked at in detail: how monads ease the modification of a simple evaluator; how monads act as the basis of a datatype of arrays subject to in-place update; and how monads can be used to build parsers.
Plan

• I want to put applied category theory on your roadmap
• The big idea: compositionality
• Case study: Complex adaptive system design: military search & rescue
• Four concepts of composition:
  • Monoids
  • Categories
  • Operads
  • Monoidal categories
• Two graphical languages:
  • String diagrams
  • Wiring diagrams
Compositionality
Frege’s Principle of Compositionality

The meaning of a complex expression is determined by:

1. The meaning of the constituent parts
2. The rules for how those parts are combined
Compositionality lets us

Making big things
Compositionality lets us

Making big things out of little things
Compositionality lets us

Making big things

out of little things

preserving guarantees
Compositionality applied

- Little programs that work → BIG programs that work?
- Little electric circuits that work → BIG electric circuits that work?
- Little Feynman diagrams that describe real physics → BIG Feynman diagrams that describe real physics?
- Little resource plans that are feasible → BIG resource plans that are feasible?
- Little networks of agents that can solve problems → BIG networks of agents that can solve problems?
Compositionality distilled - monoids
Compose any two

\[ a \circ b \]
ab
Associativity – no need for brackets

a → b → c
Associativity – no need for brackets
Associativity – no need for brackets

\[ a \circ b \circ c \]
Associativity – no need for brackets
Associativity – no need for brackets
With associativity

Just the sum of its parts with
Without associativity

\[ abc = a \circ b \circ c + \text{or} \]

Uh-oh! We need external data.
Identity – doesn’t do anything when you compose

[Diagram of a circle with the letter 'a' next to it]
Identity – doesn’t do anything when you compose
Identity – doesn’t do anything when you compose
Compositionality distilled - categories
The building blocks are called morphisms.

The colours are called objects.
Can only compose when the colours match.
The middle bit disappears
Associativity – no need for brackets
Associativity – no need for brackets
Associativity – no need for brackets
Associativity – no need for brackets
Associativity – no need for brackets
Identities – don’t do anything when you compose
Identities – don’t do anything when you compose
Identities – don’t do anything when you compose
Military research: search & rescue

• Systems of systems
• Many different kinds of networks
• Complex realtime behaviour
• How to connect them all?
Helicopter knows the target and home base

In comms range

Intent to rescue

My port

Port 1

Helo 1

Target 1
UAV knows the target, and the helicopter as a home base

My port

Intent to rescue

In comms range

Helo 1

UAV 1

Target 1
Overlay the networks

We now have a bigger network working together
Overlaying is monoidal; you can always overlay

Helo network

UAV network
Overlaying is monoidal; you can always overlay
Overlaying is monoidal; you can always overlay
How to sew together unrelated networks?

Rescue-Scene
- Helo 1
- UAV 1
- Target 1

UAV-Search-Net
- UAV 2
- UAV 3
- UAV 4
- UAV 5
- In comms range
- In comms range

Homebase
- Helo 2
- Havana
- My port
How to sew together unrelated networks?

“Take 3 networks of 3, 4 and 2 vertices; return a network of 9 vertices”
Helo 1

UAV 1

Target 1

UAV 2

UAV 3

UAV 4

UAV 5

Helo 2

My port

Havana

In comms range

Intent to rescue

Intent to rescue
Is that a category?
Nope!
You can’t compose the networks themselves.
Morphisms with lots of inputs?

The "type" is the number of vertices expected in subnetworks.

"Take 3 networks of 3, 4 and 2 vertices; return a network of 9 vertices"
Compositionality distilled – operads
Obvious example: types & functions

- Username
- Password
- User
- Invoice
- Receipt
- Signature
- Shoebox
- login
- receiveInvoice
- storeReceipt
def fullInvoiceFlow(username, password, invoice, signature, shoebox):
    user = login(username, password)
    receipt = receiveInvoice(user, invoice, signature)
    return storeReceipt(receipt, shoebox)
Mapping it to reality

To actually use an operad, translate it to the operad of sets & functions, preserving composition & identity

```
Rescue-Scene := ...
UAV-Search-Net := ...
Homebase := ...

joinNetworks(Rescue-Scene, UAV-Search-Net, Homebase)
```
String diagrams

prepare lemon meringue pie

prepared crust

lemon
butter
sugar
egg
separate egg
sugar

make lemon filling

fill crust

lemon filling
unbaked lemon pie

add meringue

meringue
unbaked pie

From “Seven Sketches of Compositionality”
Fong & Spivak (2018)
prepare lemon meringue pie

prepared crust

lemon
butter
sugar
egg
sugar

separate egg

make lemon filling

fill crust

lemon filling
unbaked lemon pie
unbaked pie

make meringue

meringue

add meringue

unbaked pie

morphisms
prepare lemon meringue pie

1. make lemon filling
   - separate egg
   - yolk
   - white
   - sugar
   - butter
   - lemon
2. make meringue
   - egg white
   - sugar
3. fill crust
   - unbaked lemon pie
4. add meringue
   - unbaked pie
prepare lemon meringue pie

- prepared crust
  - lemon
  - butter
  - sugar
- egg
  - separate egg
  - sugar
- make lemon filling
  - fill crust
  - add meringue
  - unbaked pie
prepare lemon meringue pie

1. make lemon filling
   - separate egg
   - lemon
   - butter
   - sugar
   - egg yolk

2. make meringue
   - egg white
   - sugar

3. fill crust
   - lemon filling

4. add meringue
   - meringue

5. unbaked lemon pie
Tensoring forms a monoid (up to isomorphism)

... on the objects of the category

lemon \otimes butter \otimes sugar

make lemon filling

lemon filling
Monoidal categories

Category $\mathcal{C}$

Tensor operation
$\otimes : (\text{object, object}) \rightarrow \text{object}$
such that $(a \otimes b) \otimes c \cong a \otimes (b \otimes c)$

Unit object
$I : \text{object}$
such that $a \otimes I \cong I \otimes a \cong a$
Monoidal categories

**Horizontal composition**

\[ x \xrightarrow{f} y \xrightarrow{g} z \]

**Vertical composition**

\[ x \xrightarrow{f \otimes g} y \xrightarrow{g} z \]

\[ x \xrightarrow{f \otimes g} y \xrightarrow{f \otimes g} z \]
Diagrams inside diagrams

Prepare lemon meringue pie

- Prepared crust
- Lemon
- Butter
- Sugar
- Egg
- Yolk
- White
- Sugar
- Meringue
- Make lemon filling
  - Lemon
  - Butter
  - Sugar
  - Yolk
  - Beat
- Combine lemon filling
- Fill crust
  - Lemon filling
  - Unbaked lemon pie
- Add meringue
  - Unbaked pie
  - Meringue
Diagrams inside diagrams

prepare lemon meringue pie

- prepared crust
- lemon
- butter
- sugar
- egg
  - yolk
  - white
- sugar
- make meringue
- Beat
- Blend
- Combine
- fill crust
- lemon filling
- unbaked lemon pie
- add meringue
- unbaked pie
- meringue
- lemon
- add meringue
prepare lemon meringue pie

unbaked pie

baked pie

oven

bake pie
Surprise! It’s another operad
Lemon Meringue Pie operad

- sugar $\otimes$ white $\rightarrow$ meringue
- egg $\rightarrow$ yolk $\otimes$ white
- lemon $\otimes$ butter $\otimes$ sugar $\otimes$
- yolk $\rightarrow$ lemon filling
- prepared crust $\otimes$ lemon filling $\rightarrow$ unbaked lemon pie
- unbaked lemon pie $\otimes$ meringue $\rightarrow$ unbaked pie
- prepared crust $\otimes$ lemon $\otimes$ butter $\otimes$ sugar $\otimes$ egg $\otimes$ sugar $\rightarrow$ unbaked pie
String diagrams in software architecture

- Fantastic for software design
- I’ve used them for years to train junior & senior developers
- Modularity, boundaries
The C4 model for visualising software architecture

Context, Containers, Components and Code

Abstractions  Core diagrams  Supplementary diagrams  Notation  Examples  FAQ  Diagramming vs modelling  On-site training  Online training  Tooling

👋 In a hurry? Read the 5 minute introduction to the C4 model at InfoQ:

The C4 model for software architecture
O modelo C4 de documentação para Arquitetura de Software
用于软件架构的C4模型
ソフトウェアーキテクチャのためのC4モデル

Simon Brown
Level 1

System Context diagram for Internet Banking System

The system context diagram for the Internet Banking System.
Operad! (maybe)
Operad of wiring diagrams

from
“The operad of wiring diagrams: formalizing a graphical language for databases, recursion and plug-and-play circuits”
by David I. Spivak (2013)
Inputs $x_1$, $x_2$, $x_3$
Input types
\{s, r, t\}
\{w, x, y, z\}
\{u, v\}

Output type
\{a, b, c, d, e\}
\[ \phi' \circ (\phi_1, \phi_2, \phi_3) = \]
Compositional SQL queries

SELECT L.student, L.address
FROM   attends a1, gender g1, attends a2, gender g2, lives L
WHERE  a1.student=g1.student AND a2.student=g2.student
AND    L.student=g1.student AND a1.course=a2.course
      AND g1.gender='male' AND g2.gender='female'
SELECT L.student, L.address
FROM   attends a1, gender g1, attends a2, gender g2, lives L
WHERE  a1.student=g1.student AND a2.student=g2.student
AND    L.student=g1.student AND a1.course=a2.course
AND g1.gender='male' AND g2.gender='female'
SELECT L.student, L.address
FROM attends a1, gender g1, attends a2, gender g2, lives L
WHERE a1.student=g1.student AND a2.student=g2.student
AND L.student=g1.student AND a1.course=a2.course
AND g1.gender='male' AND g2.gender='female'
SELECT L.student, L.address
FROM   attends a1, gender g1, attends a2, gender g2, lives L
WHERE  a1.student=g1.student AND a2.student=g2.student
AND    L.student=g1.student AND a1.course=a2.course
AND g1.gender='male' AND g2.gender='female'
SELECT L.student, L.address
FROM attends a1, gender g1, attends a2, gender g2, lives L
WHERE a1.student=g1.student AND a2.student=g2.student
AND L.student=g1.student AND a1.course=a2.course
AND g1.gender='male' AND g2.gender='female'
In conclusion

- **Applied category theory** is an exciting new field that seeks to mine the benefits of **compositionality** wherever they may be found.
- Supports marvelous graphical languages
- Early days!
- Watch this space
The future

- Anything where “making big things out of little things” is an important part
- So far: category theorists discovering applications and promoting
- Industry might eventually begin to lead the charge
- Computer science & programming will be at the forefront: software has eaten the world
- Functional programming will be at the forefront: we are early adopters
People to follow on Twitter

John C. Baez @johncarlosbaez
Dr. Eugenia Cheng @DrEugeniaCheng
Tai-Danae Bradley @math3ma

Jade Master @JadeMasterMath
Jules Hedges @_julesh_
Joe Moeller @CreeepyJoe

Statebox @statebox
Jelle Herold @wires_wires
Fabrizio Remano Genovese @fabgenovese
Further reading

• “Seven Sketches of Compositionality” by Brendan Fong, David I. Spivak (2018)
• “Applied category theory course” by John C. Baez (2018)
• “What is applied category theory?” by Tai-Danae Bradley (2018)
• “Network models” by Baez, Foley, Moeller, Pollard (2017)
• “The operad of wiring diagrams: formalizing a graphical language for databases, recursion and plug-and-play circuits” by David I. Spivak (2013)