Applied Category Theory:
Towards a science of interdisciplinarity

David I. Spivak Brendan Fong

Department of Mathematics
Massachusetts Institute of Technology

Outline

A road to true interdisciplinarity

- Scientific disciplines are conceptual analogies of the world.
  - Science: a schematic, conceptual account of phenomena.
  - Engineering is using these accounts to channel world events.
  - But how do different disciplines and accounts cohere?
  - To solve big problems, we need to connect different approaches.

A road to true interdisciplinarity

- Scientific disciplines are conceptual analogies of the world.
  - Science: a schematic, conceptual account of phenomena.
  - Engineering is using these accounts to channel world events.
  - But how do different disciplines and accounts cohere?
  - To solve big problems, we need to connect different approaches.
- We need a shared fabric, a substrate for interdisciplinarity.
  - Interdisciplinarity consists of effective analogy-making.
  - To go further, we need to formalize the analogies themselves.

A road to true interdisciplinarity

- Scientific disciplines are conceptual analogies of the world.
  - Science: a schematic, conceptual account of phenomena.
  - Engineering is using these accounts to channel world events.
  - But how do different disciplines and accounts cohere?
  - To solve big problems, we need to connect different approaches.
- We need a shared fabric, a substrate for interdisciplinarity.
  - Interdisciplinarity consists of effective analogy-making.
  - To go further, we need to formalize the analogies themselves.
- Better yet: we need a conceptual stem-cell.
  - Something that can differentiate into huge variety of forms.
  - Find the analogies between forms as aspects within the stem cell.

Hard science requires math

Consider as you watch:

- Is a hard science of interdisciplinarity & interoperability possible?

Hard science requires math

Consider as you watch:

- Is a hard science of interdisciplinarity & interoperability possible?
- What sort of mathematical theory would it require?

Hard science requires math

Consider as you watch:

- Is a hard science of interdisciplinarity & interoperability possible?
- What sort of mathematical theory would it require?
- Does Category Theory fit the bill?

Category theory as conceptual stem-cell

Category theory (CT) can differentiate into many forms:

- All forms of pure math... (we'll briefly discuss this)

Category theory as conceptual stem-cell

Category theory (CT) can differentiate into many forms:

- All forms of pure math... (we'll briefly discuss this)
- Databases and knowledge representation (categories and functors)
- Functional programming languages (cartesian closed categories)
- Hierarchical planning (lenses and monads)
- Dynamical systems and fractals (operad-algebras, co-algebras)
- Shannon Entropy (operad of simplices)
- Partially-ordered sets and metric spaces (enriched categories)
- Higher order logic (toposes = categories of sheaves)
- Measurements of diversity in populations (magnitude of categories)
- Collaborative design (enriched categories and profunctors)
- Petri nets and chemical reaction networks (monoidal categories)
- Quantum processes and NLP (compact closed categories)

Popper's objection

"A theory that explains everything explains nothing." – Karl Popper

Popper's objection

"A theory that explains everything explains nothing." – Karl Popper

We counter this objection in two ways:

- Couldn't the same objection be made about mathematics?
  - Mathematics is the basis of hard science, used everywhere.
  - CT—like math—explains, models, formalizes many many things.
  - Conclude that math/CT explains everything and hence nothing?

Popper's objection

"A theory that explains everything explains nothing." – Karl Popper

We counter this objection in two ways:

- Couldn't the same objection be made about mathematics?
  - Mathematics is the basis of hard science, used everywhere.
  - CT—like math—explains, models, formalizes many many things.
  - Conclude that math/CT explains everything and hence nothing?
- Stem cells don't do work until they differentiate.
  - "Adult-level" work requires differentiation and optimization.
  - But the unified origins lead to impressive interoperability.

CT is the gateway to pure mathematics

CT is humanity's most powerful thought-compression language.

CT is the gateway to pure mathematics

CT is humanity's most powerful thought-compression language.

- Designed to transport theorems from one area of math to another.
  - Example: from topology (shapes) to algebra (equations).
  - This isn't mere analogy, it's analogy made rigorous.

CT is the gateway to pure mathematics

CT is humanity's most powerful thought-compression language.

- Designed to transport theorems from one area of math to another.
  - Example: from topology (shapes) to algebra (equations).
  - This isn't mere analogy, it's analogy made rigorous.
- It's revolutionized pure math since its inception in 1940s.
  - It's touched or greatly influenced all corners of mathematics.
  - It's become a gateway to learning mathematics.

CT is the gateway to pure mathematics

CT is humanity's most powerful thought-compression language.

- Designed to transport theorems from one area of math to another.
  - Example: from topology (shapes) to algebra (equations).
  - This isn't mere analogy, it's analogy made rigorous.
- It's revolutionized pure math since its inception in 1940s.
  - It's touched or greatly influenced all corners of mathematics.
  - It's become a gateway to learning mathematics.
- And it's branched out from math in a big way.
  - Databases and knowledge representation (categories and functors)
  - Functional programming languages (cartesian closed categories)
  - Hierarchical planning (lenses and monads)
  - Dynamical systems and fractals (operad-algebras, co-algebras)
  - Shannon Entropy (operad of simplices)
  - Measurements of diversity in populations (magnitude of categories)
  - Collaborative design (enriched categories and profunctors)
  - Petri nets and chemical reaction networks (monoidal categories)
  - Quantum processes and NLP (compact closed categories)

Getting to specifics

The category-theoretic stem cell is about compositional design patterns.

Getting to specifics

The category-theoretic stem cell is about compositional design patterns.

Today we'll focus on two: operads (David) and categories (Brendan)

Getting to specifics

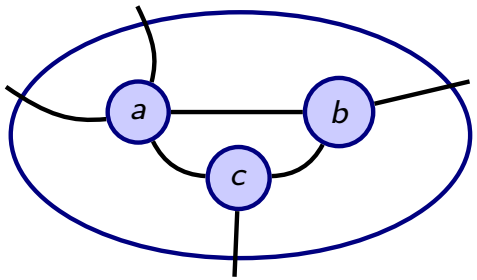
The category-theoretic stem cell is about compositional design patterns.

Today we'll focus on two: operads (David) and categories (Brendan)

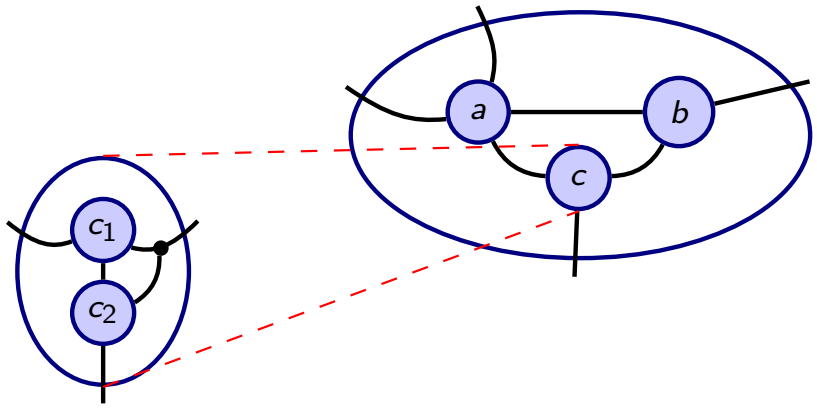
- Operads: a formalization of "operations".
- Categories: a formalization of "relationships".

Outline

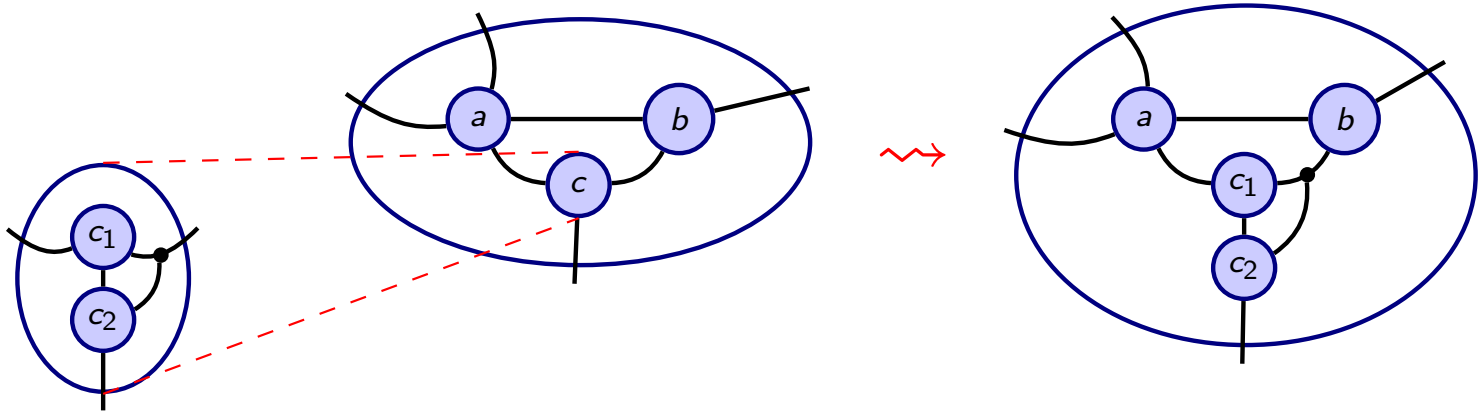
Compositionality: nestable arrangements



Compositionality: nestable arrangements



Compositionality: nestable arrangements



What are compositional operations?

An operad consists of:

- A collection of sorts $X, Y, \ldots,$
- And ways to arrange them, $\varphi\colon X_1, \ldots, X_k \to Y$,
- Such that arrangements can be nested inside each other.

What are compositional operations?

An operad consists of:

- A collection of sorts $X, Y, \ldots,$
- And ways to arrange them, $\varphi\colon X_1, \ldots, X_k \to Y$,
- Such that arrangements can be nested inside each other. (That last part is the compositionality.)

What are compositional operations?

An operad consists of:

- A collection of sorts $X, Y, \ldots,$
- And ways to arrange them, $\varphi\colon X_1, \ldots, X_k \to Y$,
- Such that arrangements can be nested inside each other.
  (That last part is the compositionality.)

If you have any questions so far, or want a more formal definition, feel free to ask me now (or later).

Operads are everywhere

Operads are used unconsciously in many fields.

- Electrical engineering: "wiring diagrams"
- Design: "set-based design"
- Computer programming: "data flow"
- Natural language processing: "grammars"
- Materials science: "hierarchical materials"
- Information theory: "Shannon entropy"

Operads are everywhere

Operads are used unconsciously in many fields.

- Electrical engineering: "wiring diagrams"
- Design: "set-based design"
- Computer programming: "data flow"
- Natural language processing: "grammars"
- Materials science: "hierarchical materials"
- Information theory: "Shannon entropy"

We want to bring operads to the fore.

- There's a common theme in the way we think.
- Operads structure this sort of thinking.
- With mathematical structure, we can go much further.

Operads are everywhere

Operads are used unconsciously in many fields.

- Electrical engineering: "wiring diagrams"
- Design: "set-based design"
- Computer programming: "data flow"
- Natural language processing: "grammars"
- Materials science: "hierarchical materials"
- Information theory: "Shannon entropy"

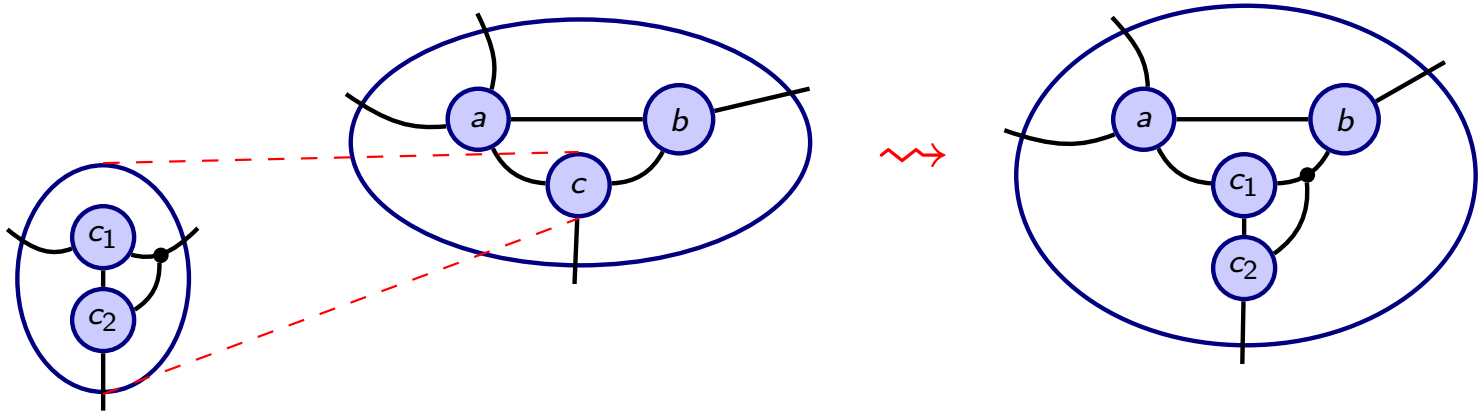
We want to bring operads to the fore.

- There's a common theme in the way we think.
- Operads structure this sort of thinking.
- With mathematical structure, we can go much further.

Let's look for sorts, arrangements, and nesting in some examples.

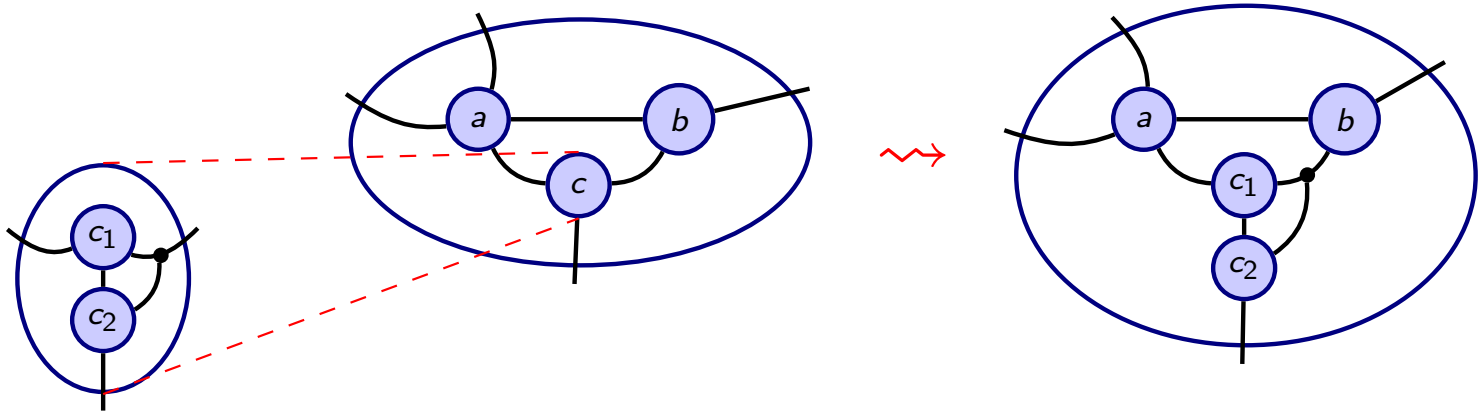
Operad 1: wiring diagrams

Sorts: circles with ports. Arrangements: wiring diagrams.



Operad 1: wiring diagrams

Sorts: circles with ports. Arrangements: wiring diagrams.



Applications: tensor networks, databases, contracts.

Operad 2: another kind of wiring diagram

There are many kinds of wiring diagram.

- We just saw an operad of "circles arranged in a circle".

Operad 2: another kind of wiring diagram

There are many kinds of wiring diagram.

- We just saw an operad of "circles arranged in a circle".
- Next up: boxes (inputs and outputs) rather than circles (with ports).

Operad 2: another kind of wiring diagram

There are many kinds of wiring diagram.

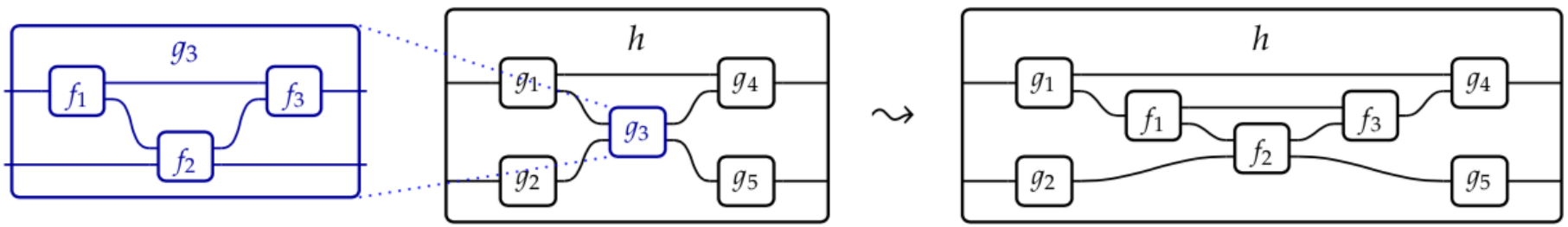
- We just saw an operad of "circles arranged in a circle".
- Next up: boxes (inputs and outputs) rather than circles (with ports).
- More structure means more restricted combination rules.

Operad 2: another kind of wiring diagram

There are many kinds of wiring diagram.

- We just saw an operad of "circles arranged in a circle".
- Next up: boxes (inputs and outputs) rather than circles (with ports).
- More structure means more restricted combination rules.

Sorts: boxes with ports. Arrangements: a different kind of wiring diagrams.

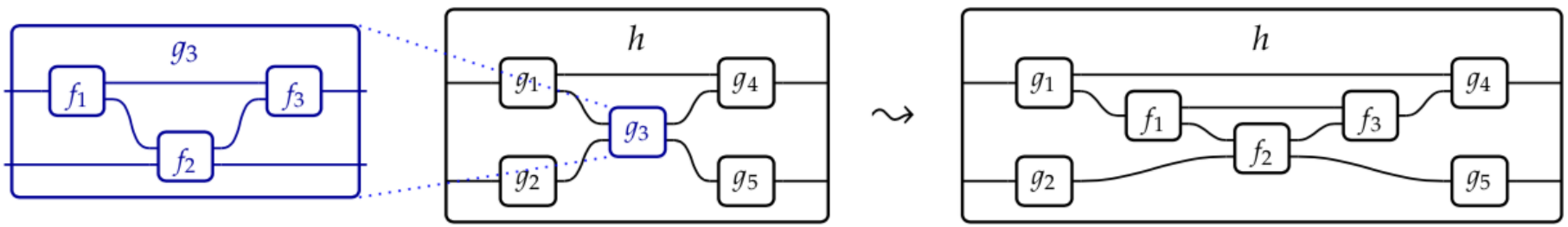


Operad 2: another kind of wiring diagram

There are many kinds of wiring diagram.

- We just saw an operad of "circles arranged in a circle".
- Next up: boxes (inputs and outputs) rather than circles (with ports).
- More structure means more restricted combination rules.

Sorts: boxes with ports. Arrangements: a different kind of wiring diagrams.



Applications: dynamical systems, control theory.

Operad 3: hierarchical protein materials

There is an operad $\mathcal{M}$ for composing hierarchical protein materials.

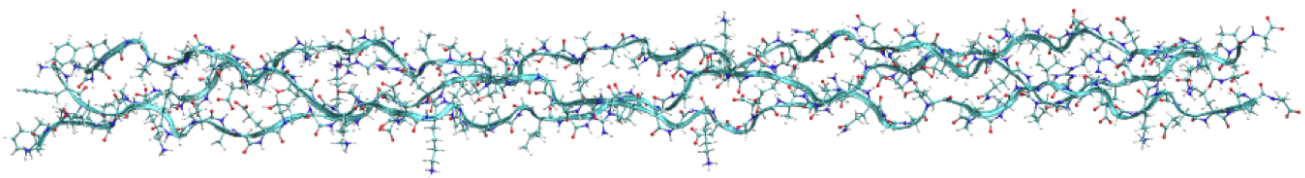
- Why protein materials?
  - Protein materials include your skin: stretchable, breathable, waterproof.
  - Materials scientists would *love* to make materials like this.

[1] Giesa, T.; Jagadeesan, R.; Spivak, D.I.; Buehler, M.J. (2015) "Matriarch: a Python library for materials architecture." *ACS Biomaterials Science & Engineering*.

Operad 3: hierarchical protein materials

There is an operad $\mathcal{M}$ for composing hierarchical protein materials.

- Why protein materials?
  - Protein materials include your skin: stretchable, breathable, waterproof.
  - Materials scientists would *love* to make materials like this.
- A protein is an arrangement of simpler proteins.
  - There are "atomic" proteins: amino acids.
  - arrange in series or parallel (H-bonds), or
  - arrange in helices, double helices, any conceivable curve, etc.



- Collagen has a nested structure: it is an array, each fiber of which is a triple helix, each strand of which is a helix, each unit of which is an amino acid.[1]

[1] Giesa, T.; Jagadeesan, R.; Spivak, D.I.; Buehler, M.J. (2015) "Matriarch: a Python library for materials architecture." *ACS Biomaterials Science & Engineering*.

Operad 4: probabilities

Even an operad with one sort can be interesting.

- Consider the operad $\mathcal{P}$ for "probabilities".

Operad 4: probabilities

Even an operad with one sort can be interesting.

- Consider the operad $\mathcal{P}$ for "probabilities".
- Say sorts=$\{$event$\}$, i.e. "event" is the only sort in $\mathcal{P}$.
- Arrangements = probability distributions. Nesting = weighted sums.

Operad 4: probabilities

Even an operad with one sort can be interesting.

- Consider the operad $\mathcal{P}$ for "probabilities".
- Say sorts=$\{$event$\}$, i.e. "event" is the only sort in $\mathcal{P}$.
- Arrangements = probability distributions. Nesting = weighted sums.
- Formally: $\mathcal{P}_k := \left\{(x_1, \ldots, x_k) \in \mathbb{R}^k_{\geq 0} \;\middle|\; x_1 + \cdots + x_k = 1\right\}$.

Operad 4: probabilities

Even an operad with one sort can be interesting.

- Consider the operad $\mathcal{P}$ for "probabilities".
- Say sorts=$\{$event$\}$, i.e. "event" is the only sort in $\mathcal{P}$.
- Arrangements = probability distributions. Nesting = weighted sums.
- Formally: $\mathcal{P}_k := \left\{(x_1, \ldots, x_k) \in \mathbb{R}^k_{\geq 0} \;\middle|\; x_1 + \cdots + x_k = 1\right\}$.

Arrangement: "In this event, there's a distribution on next events."

Operad 4: probabilities

Even an operad with one sort can be interesting.

- Consider the operad $\mathcal{P}$ for "probabilities".
- Say sorts=$\{$event$\}$, i.e. "event" is the only sort in $\mathcal{P}$.
- Arrangements = probability distributions. Nesting = weighted sums.
- Formally: $\mathcal{P}_k := \left\{(x_1, \ldots, x_k) \in \mathbb{R}^k_{\geq 0} \;\middle|\; x_1 + \cdots + x_k = 1\right\}$.

Arrangement: "In this event, there's a distribution on next events."

- coin flip: $f = (\frac{1}{2}, \frac{1}{2}) \in \mathcal{P}_2$.
  - In the event coin flip, there's a 50-50 distribution on next events.
- die roll: $r = (\frac{1}{6}, \ldots, \frac{1}{6}) \in \mathcal{P}_6$.
- card selection: $p = (\frac{1}{52}, \ldots, \frac{1}{52}) \in \mathcal{P}_{52}$.

The nesting rule composes distributions by weighted sum:

- Flip a coin: result decides whether to roll a die or pick a card.

$$f \circ (r, p) = \left(\underbrace{\frac{1}{12}, \ldots, \frac{1}{12}}_{\text{6 times}}, \underbrace{\frac{1}{104}, \ldots, \frac{1}{104}}_{\text{52 times}} \right) \in \mathcal{P}_{58}$$

A zoo of operads: Grammars

Any context-free grammar is an operad.

A zoo of operads: Grammars

Any context-free grammar is an operad.

$$
\begin{array}{rcl}
\langle \text{sentence} \rangle & ::= & \langle \text{noun-phrase} \rangle \ \langle \text{verb-phrase} \rangle \\
\langle \text{noun-phrase} \rangle & ::= & \langle \text{pronoun} \rangle \mid \langle \text{proper-noun} \rangle \mid \langle \text{determiner} \rangle \ \langle \text{nominal} \rangle \\
\langle \text{nominal} \rangle & ::= & \langle \text{noun} \rangle \mid \langle \text{nominal} \rangle \ \langle \text{noun} \rangle \\
\langle \text{verb-phrase} \rangle & ::= & \langle \text{verb} \rangle \mid \langle \text{verb} \rangle \ \langle \text{noun-phrase} \rangle \mid \langle \text{verb} \rangle \ \langle \text{prep-phrase} \rangle \\
\langle \text{prep-phrase} \rangle & ::= & \langle \text{preposition} \rangle \ \langle \text{noun-phrase} \rangle
\end{array}
$$

Of course, it doesn't have to be English

A zoo of operads: Grammars

Any context-free grammar is an operad.

$$
\begin{array}{rcl}
\langle \text{sentence} \rangle & ::= & \langle \text{noun-phrase} \rangle \ \langle \text{verb-phrase} \rangle \\
\langle \text{noun-phrase} \rangle & ::= & \langle \text{pronoun} \rangle \mid \langle \text{proper-noun} \rangle \mid \langle \text{determiner} \rangle \ \langle \text{nominal} \rangle \\
\langle \text{nominal} \rangle & ::= & \langle \text{noun} \rangle \mid \langle \text{nominal} \rangle \ \langle \text{noun} \rangle \\
\langle \text{verb-phrase} \rangle & ::= & \langle \text{verb} \rangle \mid \langle \text{verb} \rangle \ \langle \text{noun-phrase} \rangle \mid \langle \text{verb} \rangle \ \langle \text{prep-phrase} \rangle \\
\langle \text{prep-phrase} \rangle & ::= & \langle \text{preposition} \rangle \ \langle \text{noun-phrase} \rangle
\end{array}
$$

Of course, it doesn't have to be English

- Grammars are used to design custom languages.
- Legal arguments, arithmetic expressions, programming languages.

A zoo of operads: Grammars

Any context-free grammar is an operad.

⟨sentence⟩	::=	⟨noun-phrase⟩ ⟨verb-phrase⟩
⟨noun-phrase⟩	::=	⟨pronoun⟩ | ⟨proper-noun⟩ | ⟨determiner⟩ ⟨nominal⟩
⟨nominal⟩	::=	⟨noun⟩ | ⟨nominal⟩ ⟨noun⟩
⟨verb-phrase⟩	::=	⟨verb⟩ | ⟨verb⟩ ⟨noun-phrase⟩ | ⟨verb⟩ ⟨prep-phrase⟩
⟨prep-phrase⟩	::=	⟨preposition⟩ ⟨noun-phrase⟩

Of course, it doesn't have to be English

- Grammars are used to design custom languages.
- Legal arguments, arithmetic expressions, programming languages.

And how is this an operad?

- The sorts are the parts of speech.
- The arrangements are the production rules.
- Nesting is given by "syntax trees".

Outline

Another theme of category theory

The category-theoretic stem cell also matures to a method for organizing relationships.

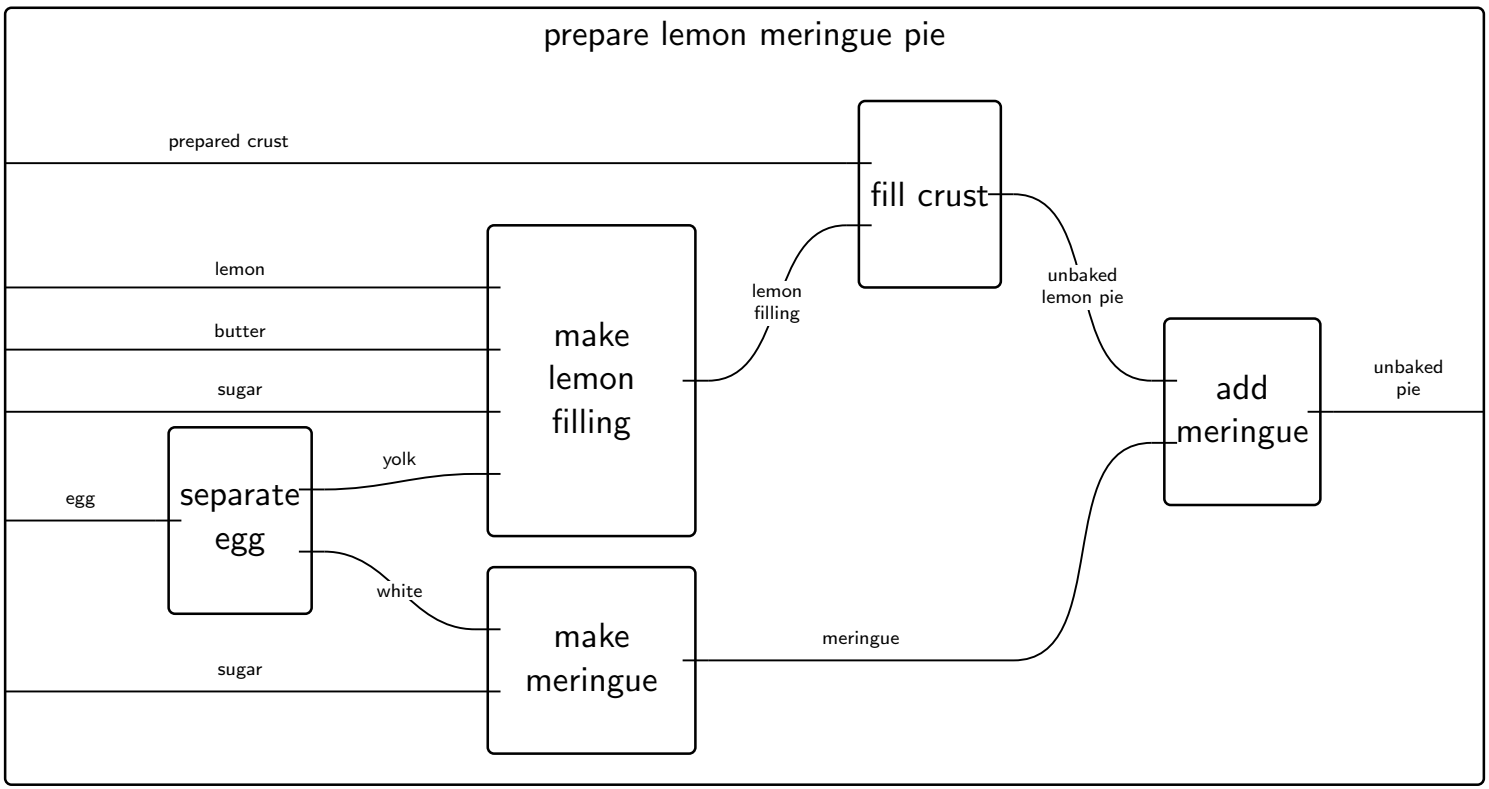
Another theme of category theory

The category-theoretic stem cell also matures to a method for organizing relationships.

Let's talk about categories and databases.

- Categories: provide a map of relationships.
- A finite category is a database schema.
- CQL is a category theoretic database query language.

Another look at wiring diagrams



Instead of focusing on the way the boxes are connected, let's focus on what's in the boxes.

What are relationships?

A category consists of:

- A collection of objects $X, Y, \ldots,$
- And relationships between them, $\varphi\colon X \to Y$,
- Such that relationships can be composed $X \xrightarrow{\varphi_1} Y \xrightarrow{\varphi_2} Z$.

What are relationships?

A category consists of:

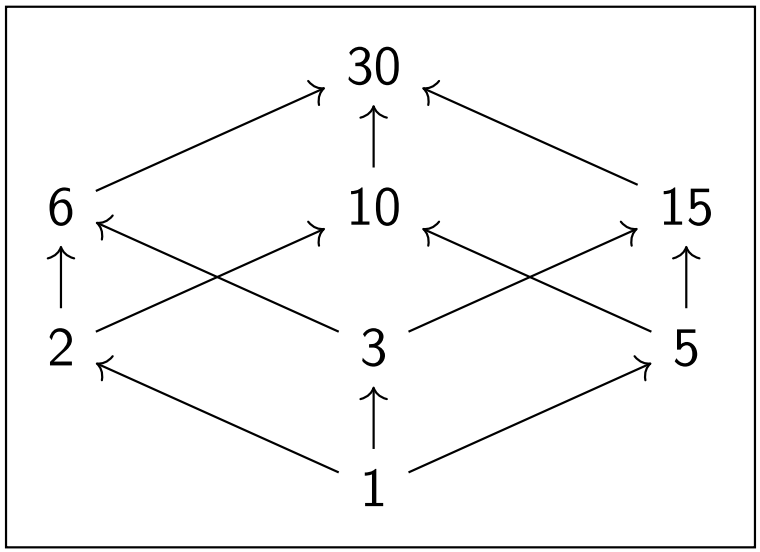
- A collection of objects $X, Y, \ldots,$
- And relationships between them, $\varphi\colon X \to Y,$
- Such that relationships can be composed $X \xrightarrow{\varphi_1} Y \xrightarrow{\varphi_2} Z.$

(Notice the similarity with the definition of operad:

- A collection of sorts $X, Y, \ldots,$
- And ways to arrange them, $\varphi\colon X_1, \ldots, X_k \to Y,$
- Such that arrangements can be nested inside each other.)

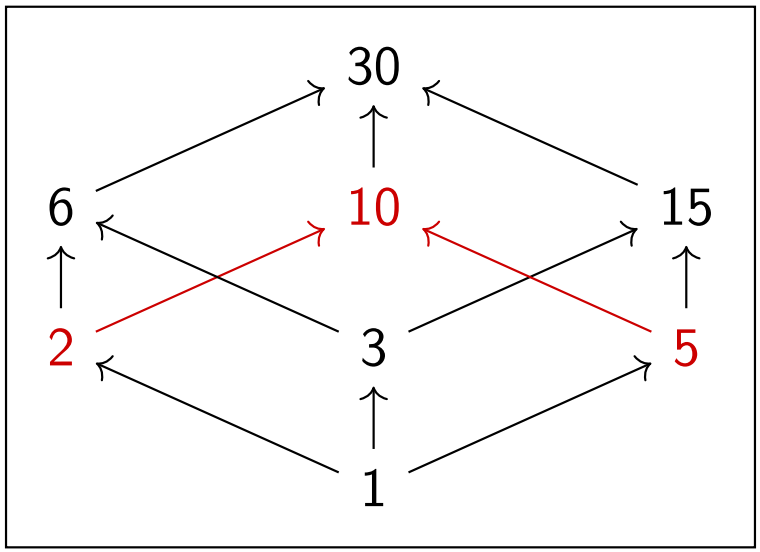
Categories 1: ordering

There is a category where the objects are numbers, and relationships are divisibility.



Categories 1: ordering

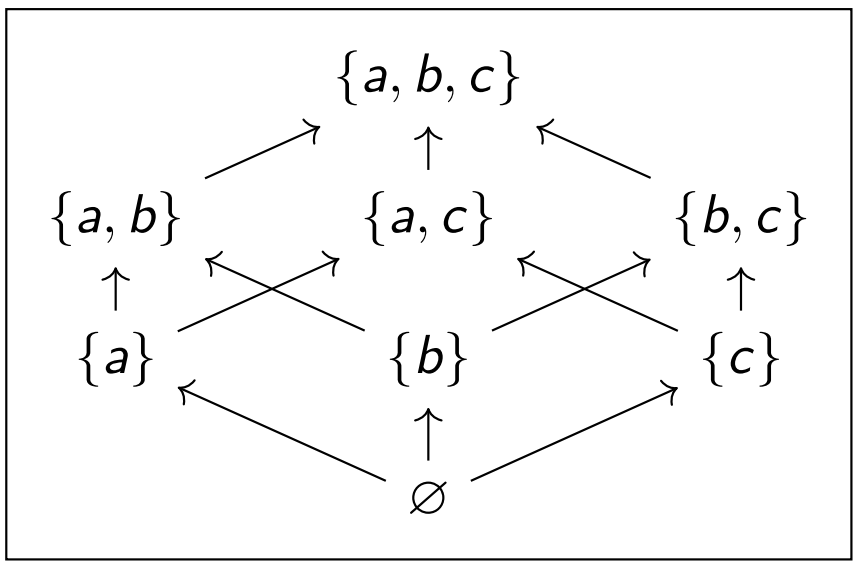
There is a category where the objects are numbers, and relationships are divisibility.



Concepts such as ‘least common multiple’ can be expressed using relationships.

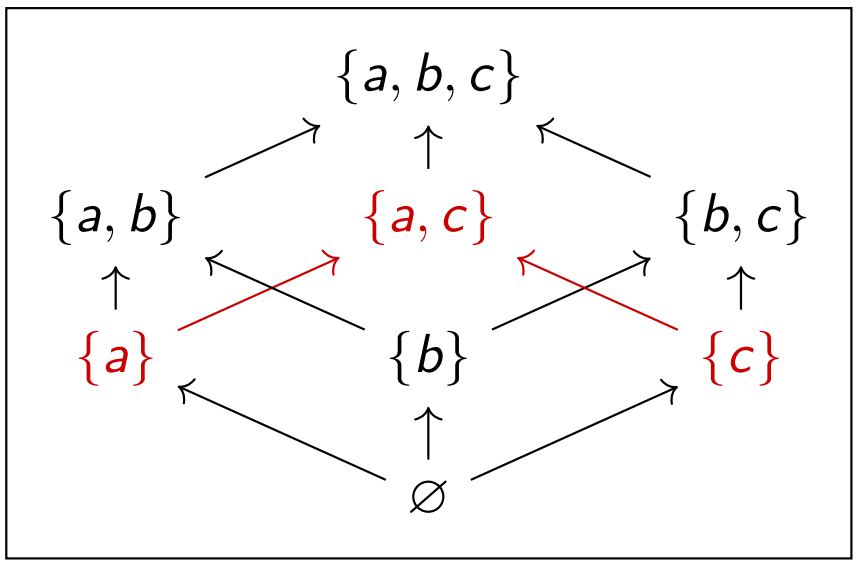
Categories 1: ordering

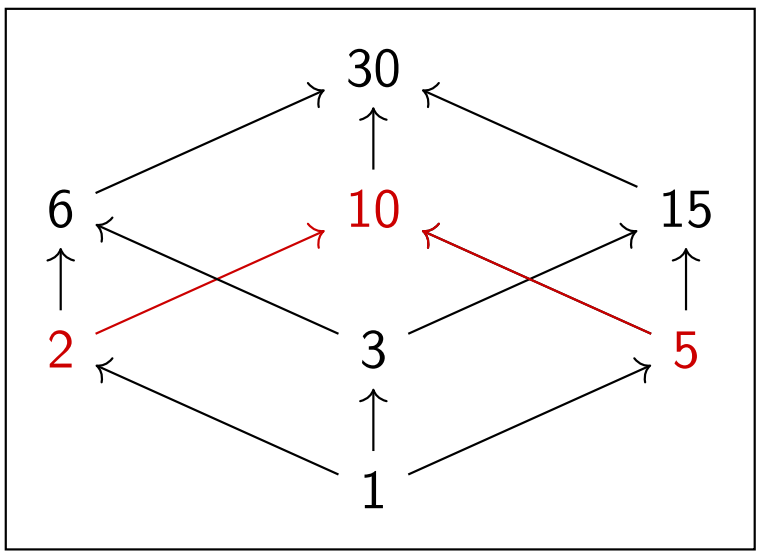
Similarly, there is a category where the objects are sets, and relationships are containment.



Categories 1: ordering

Similarly, there is a category where the objects are sets, and relationships are containment.



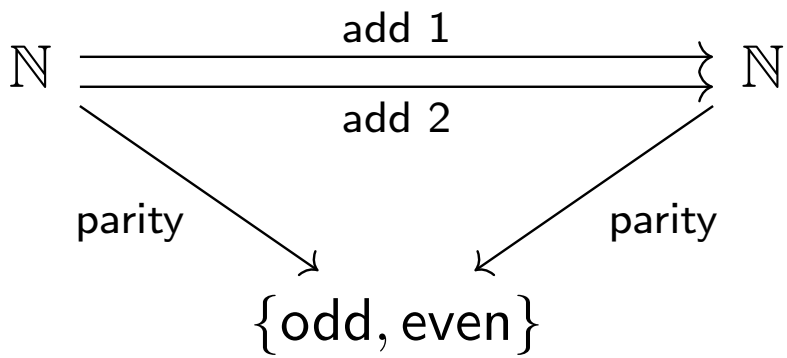


'Union' is analogous to 'least common multiple': they have the same universal property.

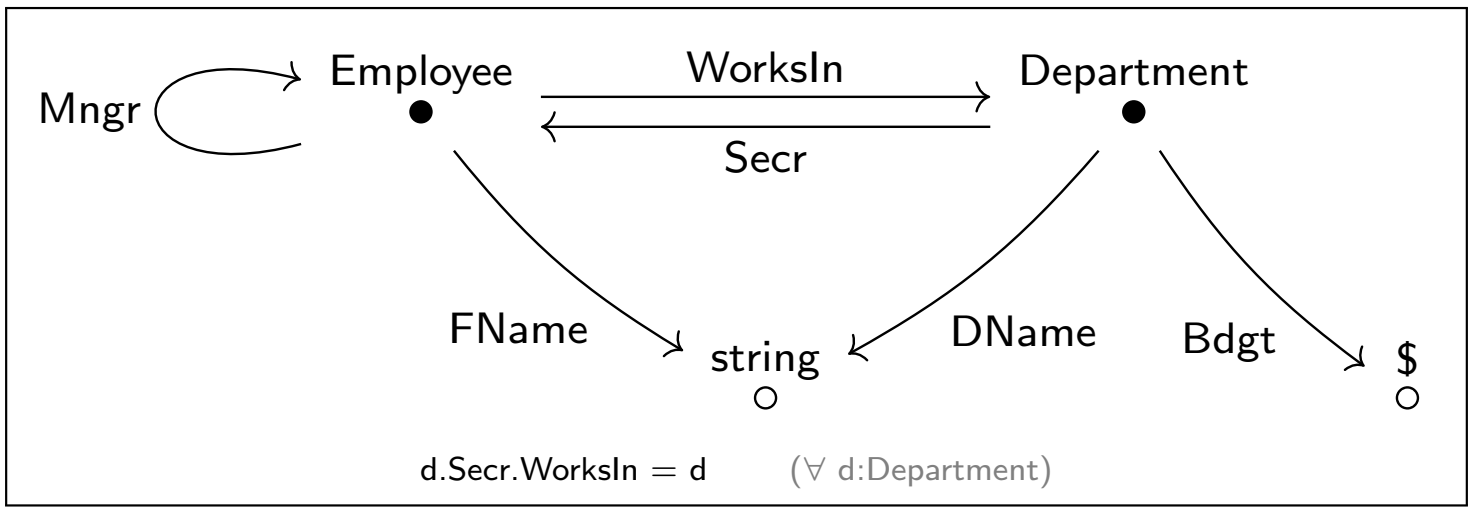
Category 2: sets and functions

The category **Set** of sets and functions is very important

- The objects of the category are sets X.
- The relationships are functions $f : X \to Y$.
- The composition rules are function composition.



Categories 3: database schemas



- Database schemas are categories!
  - The objects of the category are tables.
  - The relationships are columns, connecting one table to another.
  - The composition rules are integrity constraints.

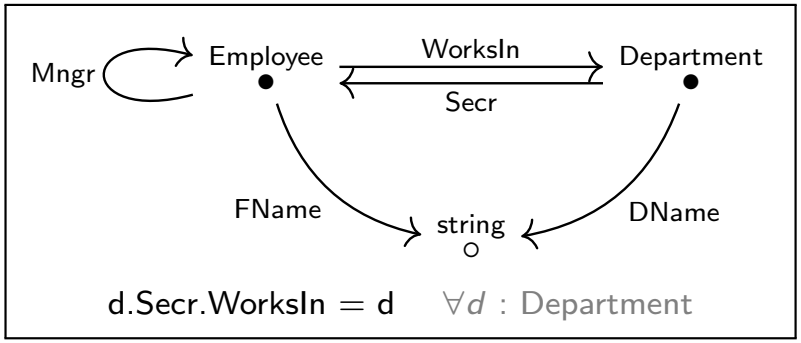
Database instances as functors

A functor is a relationship between *categories*.

Database instances as functors

A functor is a relationship between *categories*.

$\mathcal{C} :=$



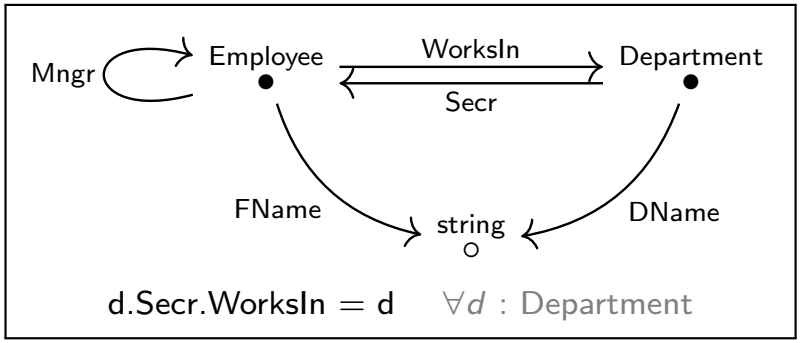
A functor $I : \mathcal{C} \to \mathbf{Set}$ consists of

- A set for each object of $\mathcal{C}$ and
- a function for each relationship of $\mathcal{C}$, such that
- the declared equations hold.

Database instances as functors

A functor is a relationship between *categories*.

$\mathcal{C} :=$



A functor $I : \mathcal{C} \to \mathbf{Set}$ consists of

- A set for each object of $\mathcal{C}$ and
- a function for each relationship of $\mathcal{C}$, such that
- the declared equations hold.

In other words, I fills the schema with compatible data.

$I :=$

Employee	FName	WorksIn	Mngr
1	Alan	101	2
2	Ruth	101	2
3	Kris	102	3

Department	DName	Secr	Bdgt
101	Sales	1	$10
102	IT	3	$5

The category of instances

- Given a schema $\mathcal{C}$, the *category of instances* on $\mathcal{C}$ is denoted $\mathcal{C}$-**Inst**.
  - The objects of $\mathcal{C}$-**Inst** are functors (instances) $I : \mathcal{C} \to$ **Set**.
  - The arrows are insertions and deduplications of data.

The category of instances

- Given a schema $\mathcal{C}$, the *category of instances* on $\mathcal{C}$ is denoted $\mathcal{C}$-**Inst**.
  - The objects of $\mathcal{C}$-**Inst** are functors (instances) $I : \mathcal{C} \to$ **Set**.
  - The arrows are insertions and deduplications of data.
- Functors between schemas allow us to move data between them.
  - Given a functor $F : \mathcal{C} \to \mathcal{D}$, there are automatically three data transformation functors

$$\mathcal{C}\text{-}\mathbf{Inst} \;\; \begin{array}{c} \xrightarrow{\Sigma_F} \\ \xleftarrow{\Delta_F} \\ \xrightarrow{\Pi_F} \end{array} \;\; \mathcal{D}\text{-}\mathbf{Inst}$$

  - Δ=project, delete; Σ=sum, union; Π=product, join.
  - Δ, Σ, Π were known by mathematicians in 1960 as "Kan extensions".
  - But they had no idea Δ, Σ, Π correspond to database relational algebra (1970).

The category of instances

- Given a schema $\mathcal{C}$, the *category of instances* on $\mathcal{C}$ is denoted $\mathcal{C}$-**Inst**.
  - The objects of $\mathcal{C}$-**Inst** are functors (instances) $I : \mathcal{C} \to$ **Set**.
  - The arrows are insertions and deduplications of data.
- Functors between schemas allow us to move data between them.
  - Given a functor $F : \mathcal{C} \to \mathcal{D}$, there are automatically three data transformation functors

$$\mathcal{C}\text{-}\mathbf{Inst} \;\; \begin{array}{c} \xrightarrow{\Sigma_F} \\ \xleftarrow{\Delta_F} \\ \xrightarrow{\Pi_F} \end{array} \;\; \mathcal{D}\text{-}\mathbf{Inst}$$

  - Δ=project, delete; Σ=sum, union; Π=product, join.
  - Δ, Σ, Π were known by mathematicians in 1960 as "Kan extensions".
  - But they had no idea Δ, Σ, Π correspond to database relational algebra (1970).

There's a lot of mathematics ready made for hygienically moving data.

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.
- ETL functionality: transform data (Σ, Δ, Π) at scale.

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.
- ETL functionality: transform data (Σ, Δ, Π) at scale.
- Incorporates any function from across JVM languages (JavaScript, Java, Python, etc).
  - Arbitrary user-defined functions, e.g. edit-distance.
  - Can reason about them, e.g. for database transformations.

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.
- ETL functionality: transform data (Σ, Δ, Π) at scale.
- Incorporates any function from across JVM languages (JavaScript, Java, Python, etc).
  - Arbitrary user-defined functions, e.g. edit-distance.
  - Can reason about them, e.g. for database transformations.
- Import / export CSV, SQL, JSON, RDF, XML, etc.

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.
- ETL functionality: transform data (Σ, Δ, Π) at scale.
- Incorporates any function from across JVM languages (JavaScript, Java, Python, etc).
  - Arbitrary user-defined functions, e.g. edit-distance.
  - Can reason about them, e.g. for database transformations.
- Import / export CSV, SQL, JSON, RDF, XML, etc.
- Data integration and warehousing: compute limits and colimits.

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

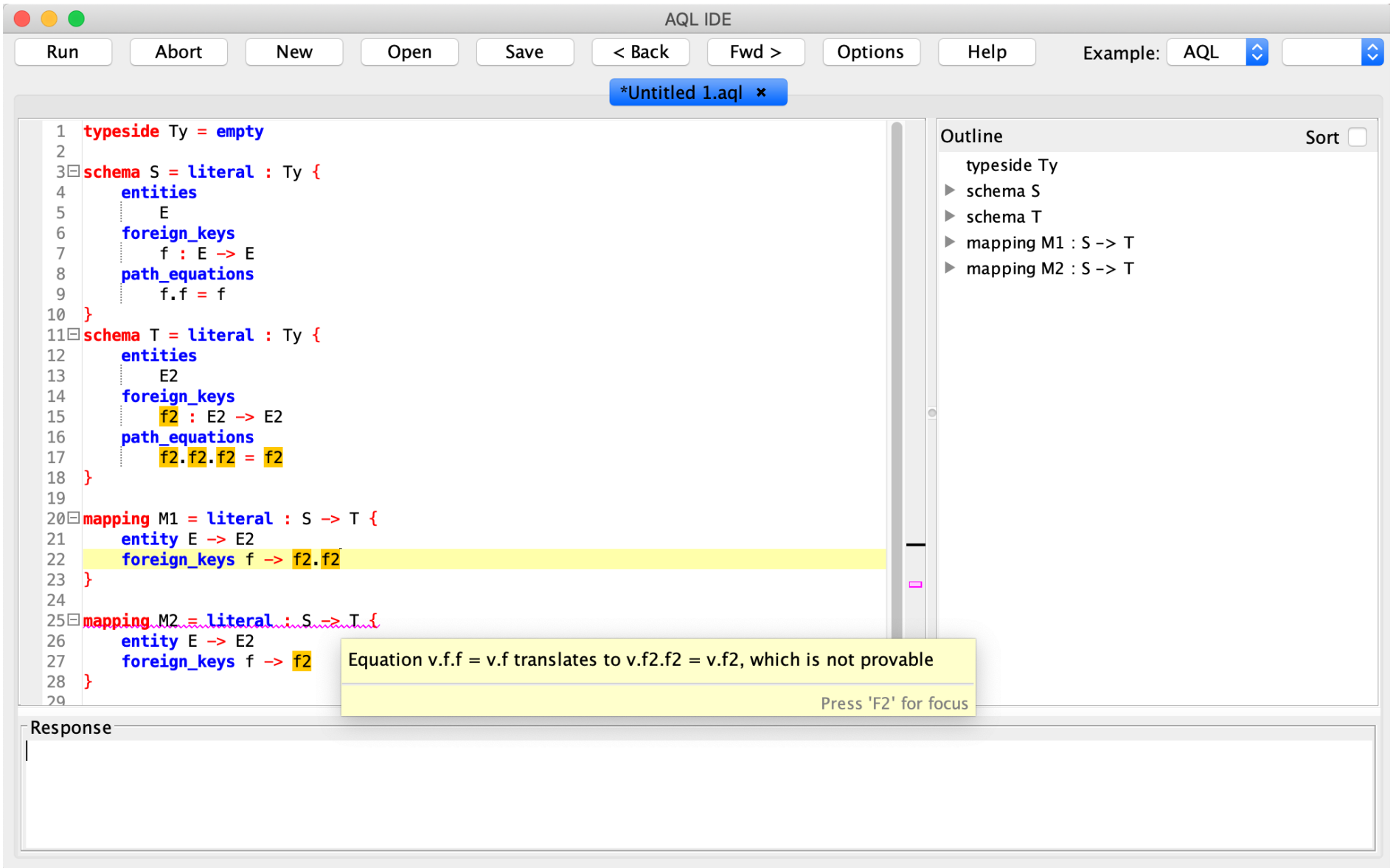
- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.
- ETL functionality: transform data (Σ, Δ, Π) at scale.
- Incorporates any function from across JVM languages (JavaScript, Java, Python, etc).
  - Arbitrary user-defined functions, e.g. edit-distance.
  - Can reason about them, e.g. for database transformations.
- Import / export CSV, SQL, JSON, RDF, XML, etc.
- Data integration and warehousing: compute limits and colimits.
- Check, clean, or repair against rich data integrity constraints
  - Repair using Chase algorithm on existential Horn clauses (EDs)

Categorical Query Language (CQL)

CQL is category theory-based database software, with features including:

- Check schema mappings and queries for functoriality, at compile time.
  - Achieved using various automated theorem provers.
  - Know in advance that landed data will satisfy all target constraints.
- ETL functionality: transform data (Σ, Δ, Π) at scale.
- Incorporates any function from across JVM languages (JavaScript, Java, Python, etc).
  - Arbitrary user-defined functions, e.g. edit-distance.
  - Can reason about them, e.g. for database transformations.
- Import / export CSV, SQL, JSON, RDF, XML, etc.
- Data integration and warehousing: compute limits and colimits.
- Check, clean, or repair against rich data integrity constraints
  - Repair using Chase algorithm on existential Horn clauses (EDs)
- And more (natural transformations, algebraic theories, profunctors, Grothendieck construction, (co-) monads..., simply-typed lambda calculus)

Screenshot


AQL IDE
Run
Abort
New
Open
Save
< Back
Fwd >
Options
Help
Example:
AQL
*Untitled 1.aql
typeside Ty = empty
schema S = literal : Ty {
entities
E
foreign_keys
f : E -> E
path_equations
f.f = f
}
schema T = literal : Ty {
entities
E2
foreign_keys
f2 : E2 -> E2
path_equations
f2.f2.f2 = f2
}
mapping M1 = literal : S -> T {
entity E -> E2
foreign_keys f -> f2.f2
}
mapping M2 = literal : S -> T {
entity E -> E2
foreign_keys f -> f2
}
Outline
Sort
typeside Ty
schema S
schema T
mapping M1 : S -> T
mapping M2 : S -> T
Equation v.f.f = v.f translates to v.f2.f2 = v.f2, which is not provable
Press 'F2' for focus
Response

Outline

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

Functors = formal mappings between languages

- A richly interconnected network of languages.

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

Functors = formal mappings between languages

- A richly interconnected network of languages.
- Some languages are more expressive, others are easier to reason about.
- Think rounding: real numbers = more expressive, integers = easier.

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

Functors = formal mappings between languages

- A richly interconnected network of languages.
- Some languages are more expressive, others are easier to reason about.
- Think rounding: real numbers = more expressive, integers = easier.

What is the right mathematics to underlie a science of interdisciplinarity?

- Category theory has an impressive record inside of math and out.
- Recently highlighted by US agencies such as NIST and DARPA.
- Whether or not it can really do the job at scale remains to be seen.

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

Functors = formal mappings between languages

- A richly interconnected network of languages.
- Some languages are more expressive, others are easier to reason about.
- Think rounding: real numbers = more expressive, integers = easier.

What is the right mathematics to underlie a science of interdisciplinarity?

- Category theory has an impressive record inside of math and out.
- Recently highlighted by US agencies such as NIST and DARPA.
- Whether or not it can really do the job at scale remains to be seen.

If you want to work towards a science of interdisciplinarity,

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

Functors = formal mappings between languages

- A richly interconnected network of languages.
- Some languages are more expressive, others are easier to reason about.
- Think rounding: real numbers = more expressive, integers = easier.

What is the right mathematics to underlie a science of interdisciplinarity?

- Category theory has an impressive record inside of math and out.
- Recently highlighted by US agencies such as NIST and DARPA.
- Whether or not it can really do the job at scale remains to be seen.

If you want to work towards a science of interdisciplinarity,

Category Theory should be on your radar.

CT for a science of interdisciplinarity

Operads and categories can be used for many things.

- People are already using them in many different fields.
- We can describe them all in a common language.
- We can also invent our own new compositional languages.

Functors = formal mappings between languages

- A richly interconnected network of languages.
- Some languages are more expressive, others are easier to reason about.
- Think rounding: real numbers = more expressive, integers = easier.

What is the right mathematics to underlie a science of interdisciplinarity?

- Category theory has an impressive record inside of math and out.
- Recently highlighted by US agencies such as NIST and DARPA.
- Whether or not it can really do the job at scale remains to be seen.

If you want to work towards a science of interdisciplinarity,
Category Theory should be on your radar.

Thanks! Questions and comments welcome.