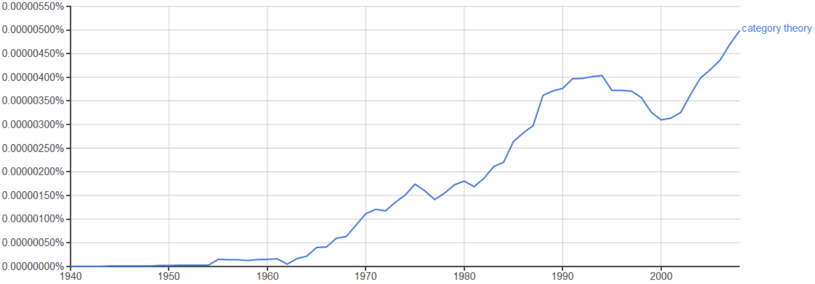
Category Theory for Modular Design: An IoT Example

Spencer Breiner & Eswaran Subrahmanian

National Institute of Standards and Technology

Ontolog 2015 - March 12

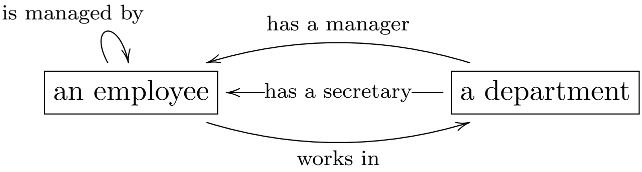
Category Theory



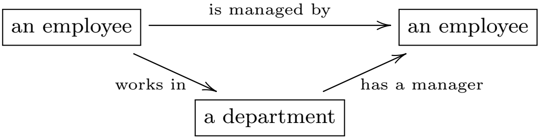
- Developed in the 1940's to connect different branches of mathematics.
- Reveals deep connections between formal logic, computer science and theoretical physics.
- Mathematical study of (implementation-independent) structure.

What is a category?

- Directed graph + path equivalence

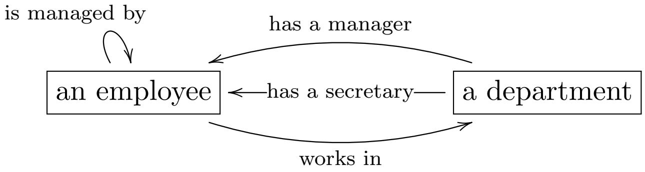


- "Every employee is managed by the manager of his department."

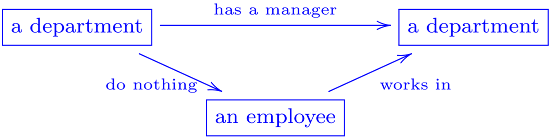


What is a category?

- Directed graph + path equivalence

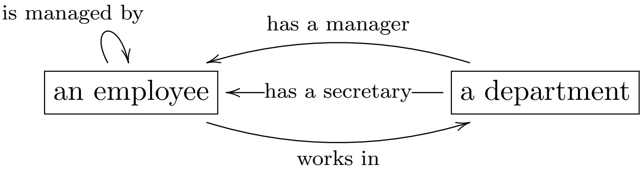


- "Every manager works in the same department she manages."



What is a category?

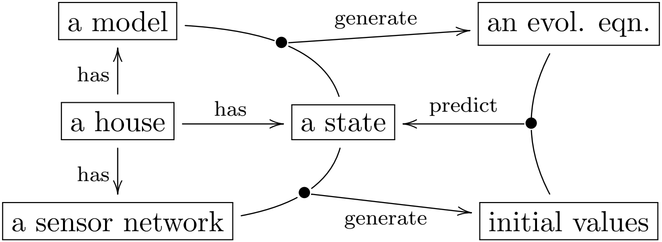
- Directed graph + path equivalence



- Intuitively, each box is a set,
  each arrow is a function.
  each path is a composition of functions,
  each equivalence is an equation between composites.

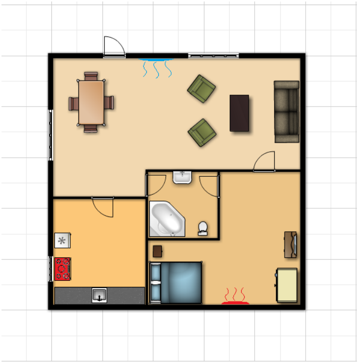
Plan

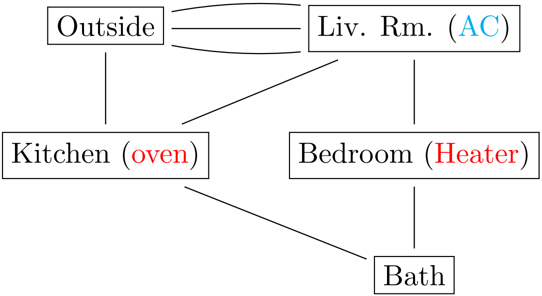
A high-level interface



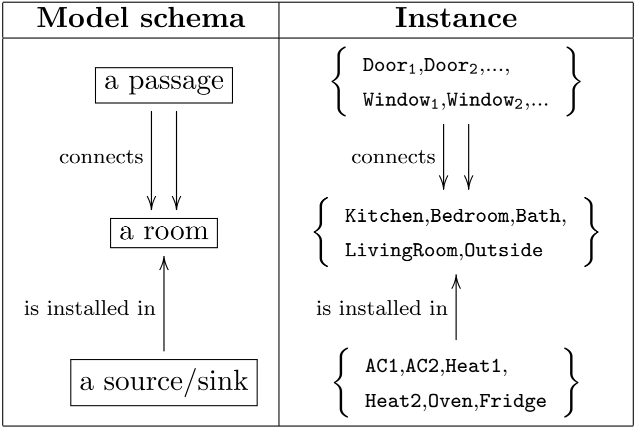
- This is a *formal* specification of an interface.
- Fill in the details piece by piece, then integrate.

A simple house model





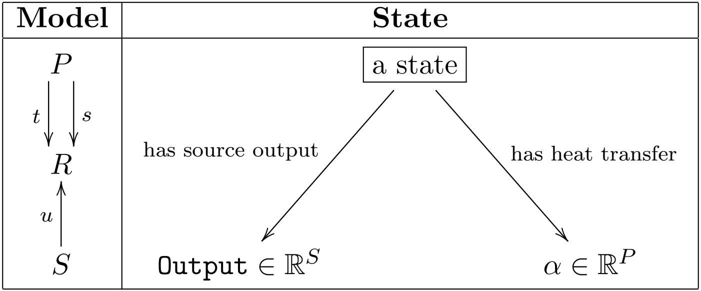
A simple house model



- Categorical models automatically provide database schemas.

Plan

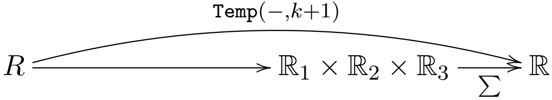
Model + State $\mapsto$ Evolution Equation



$$\texttt{Temp}(r_i, k+1) = \texttt{Temp}(r_i, k) + \sum_{u(s)=r_i} \texttt{Output}(s) + \sum_{t(p)=r_i} \alpha_p \Big(\texttt{Temp}(s(p), k) - \texttt{Temp}(r_i, k)\Big)$$

Modelling the evolution equation

$$\begin{aligned}\texttt{Temp}(r_i, k+1) &= \texttt{Temp}(r_i, k) + \sum_{u(s)=r_i} \texttt{Output}(s) \\ &+ \sum_{t(p)=r_i} \alpha_p \Big(\texttt{Temp}(s(p), k) - \texttt{Temp}(r_i, k) \Big)\end{aligned}$$



Modelling the evolution equation

$R \xrightarrow{\texttt{Temp}(-,k+1)} \mathbb{R}$; $R \longrightarrow \mathbb{R}_1 \times \mathbb{R}_2 \times \mathbb{R}_3 \xrightarrow[\Sigma]{} \mathbb{R}$	Break down sum.
$R \xrightarrow{\texttt{Temp}(-,\texttt{k})} \mathbb{R}_1$	Boundary value.

Modelling the evolution equation

$R \xrightarrow{\texttt{Temp}(-,k+1)} \mathbb{R}$ $R \longrightarrow \mathbb{R}_1 \times \mathbb{R}_2 \times \mathbb{R}_3 \xrightarrow[\Sigma]{} \mathbb{R}$	Break down sum.
$R \xrightarrow{\texttt{Temp}(-,\texttt{k})} \mathbb{R}_1$	Boundary value.
$R \xrightarrow{\texttt{curry}_u(\texttt{Output})} \prod_{u(s)=r} \mathbb{R} \xrightarrow{\Sigma} \mathbb{R}_2$	Curry, then sum.

Modelling the evolution equation

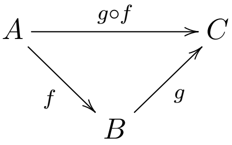
$R \longrightarrow \mathbb{R}_1 \times \mathbb{R}_2 \times \mathbb{R}_3 \xrightarrow[\sum]{} \mathbb{R}$, with $R \xrightarrow{\texttt{Temp}(-,k+1)} \mathbb{R}$	Break down sum.
$R \xrightarrow{\texttt{Temp}(-,\texttt{k})} \mathbb{R}_1$	Boundary value.
$R \xrightarrow{\texttt{curry}_u(\texttt{Output})} \prod_{u(s)=r} \mathbb{R} \xrightarrow{\sum} \mathbb{R}_2$	Curry, then sum.
$R \xrightarrow{\texttt{curry}_t\left(\alpha_p(\texttt{Temp}(t,k)-\texttt{Temp}(s,k))\right)} \prod_{t(p)=r} \mathbb{R} \xrightarrow{\sum} \mathbb{R}_3$	BVs, algebra, curry, sum.

Plan

Semantic categories

Categories via composition:

- A category consists of *objects* $(A, B, \ldots)$ and *arrows* $(f : A \to B)$.
- Arrows which match "tip-to-tail" can be composed:



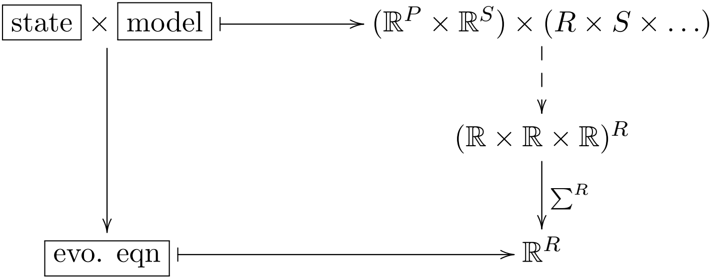
Example: **Set** has sets for objects and functions for arrows.

Maps between categories

A *functor* is a map between categories. This are just like maps between graphs, with one important difference:

Nodes map to nodes, edges map to *paths of edges*.

In the previous example,



Semantic Interpretation

The semantics for categorical models generalize the set-theoretic semantics of first-order logic.

- A "theory" is a (small) syntactic category **Syn**.
- Semantics "occur" in the category **Set**.
- A "model" or interpretation of the theory is a functor

$$\mathbf{Syn} \xrightarrow{\quad I \quad} \mathbf{Set}\ .$$

Semantic Interpretation

The semantics for categorical models generalize the set-theoretic semantics of first-order logic.

- A "theory" is a (small) syntactic category **Syn**.
- Semantics "occur" in some (large) category **Smtc**.
- A "model" or interpretation of the theory is a functor

$$\mathbf{Syn} \xrightarrow{\quad I \quad} \mathbf{Smtc}\ .$$

- The same set-up applies to more exotic semantics.

Sensors + State $\mapsto$ Initial Values

$$\boxed{\text{a state}} \xrightarrow{\text{has sensor locations}} \texttt{loc} \in R^Q$$

$$\boxed{\text{a sensor network}} \xrightarrow{\text{has sensor values}} \texttt{read} \in \mathbb{R}^Q$$

From this data we must produce an initial value map $\texttt{Temp} : R \to \mathbb{R}$.

But what is a map?

Situation: Thermostat in every room.

Semantics: $\mathbf{Syn} \longrightarrow \mathbf{Set}$.

Map: $\texttt{Temp} : R \underset{\texttt{loc}}{\overset{\texttt{loc}^{-1}}{\rightleftarrows}} Q \xrightarrow{\texttt{read}} \mathbb{R}$.

Sensors + State $\mapsto$ Initial Values

$$\boxed{\text{a state}} \xrightarrow{\text{has sensor locations}} \texttt{loc} \in R^Q$$

$$\boxed{\text{a sensor network}} \xrightarrow{\text{has sensor values}} \texttt{read} \in \mathbb{R}^Q$$

From this data we must produce an initial value map $\texttt{Temp} : R \to \mathbb{R}$.

But what is a map?

Situation: Readings over time.

Semantics: $\mathbf{Syn} \longrightarrow \mathbf{Set}^T$.

Map: $\texttt{Temp} : R \times T \longrightarrow \mathbb{R}$.

Sensors + State $\mapsto$ Initial Values

$$\boxed{\text{a state}} \xrightarrow{\text{has sensor locations}} \texttt{loc} \in R^Q$$

$$\boxed{\text{a sensor network}} \xrightarrow{\text{has sensor values}} \texttt{read} \in \mathbb{R}^Q$$

From this data we must produce an initial value map $\texttt{Temp} : R \to \mathbb{R}$.

But what is a map?

Situation: Many sensors, or few.

Semantics: $\textbf{Syn} \longrightarrow \textbf{PrLang}$.

Map:

$$\texttt{Temp}(r) = \texttt{if has\_sensors}(r):$$
$$\texttt{return avg}(\texttt{read}(\texttt{sensors}(r)))$$
$$\texttt{else}: \texttt{return avg}(\texttt{read}(\texttt{all\_sensors}))$$

Sensors + State $\mapsto$ Initial Values

$$\boxed{\text{a state}} \xrightarrow{\text{has sensor locations}} \texttt{loc} \in R^Q$$

$$\boxed{\text{a sensor network}} \xrightarrow{\text{has sensor values}} \texttt{read} \in \mathbb{R}^Q$$

From this data we must produce an initial value map $\texttt{Temp} : R \to \mathbb{R}$.

But what is a map?

Situation: Unreliable sensors.

Semantics: $\mathbf{Syn} \longrightarrow \mathbf{Prob}$.

Map: $\texttt{Temp} : R \longrightarrow \mathbf{ProbDist}(\mathbb{R})$.

Plan

Integration via Pushouts

Formal tools called *colimits* allow us to automatically integrate high-level and low-level models.

1) Begin with two (or more) models.	M_1 M_2

Integration via Pushouts

Formal tools called *colimits* allow us to automatically integrate high-level and low-level models.

1) Begin with two (or more) models. 2) Identify the overlap between these.	M_1 M_2 O

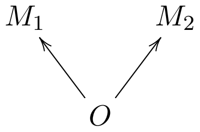
Integration via Pushouts

Formal tools called *colimits* allow us to automatically integrate high-level and low-level models.

1) Begin with two (or more) models.

2) Identify the overlap between these.

3) Map the overlap into each piece.



Integration via Pushouts

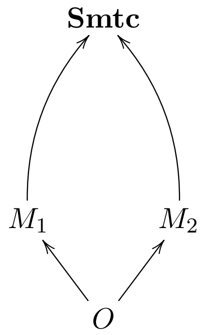
Formal tools called *colimits* allow us to automatically integrate high-level and low-level models.

1) Begin with two (or more) models.

2) Identify the overlap between these.

3) Map the overlap into each piece.

4) Aggregates are defined semantically.



Integration via Pushouts

Formal tools called *colimits* allow us to automatically integrate high-level and low-level models.

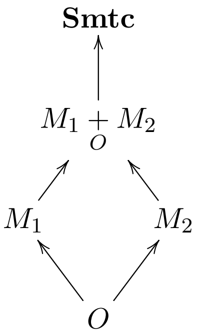
1) Begin with two (or more) models.

2) Identify the overlap between these.

3) Map the overlap into each piece.

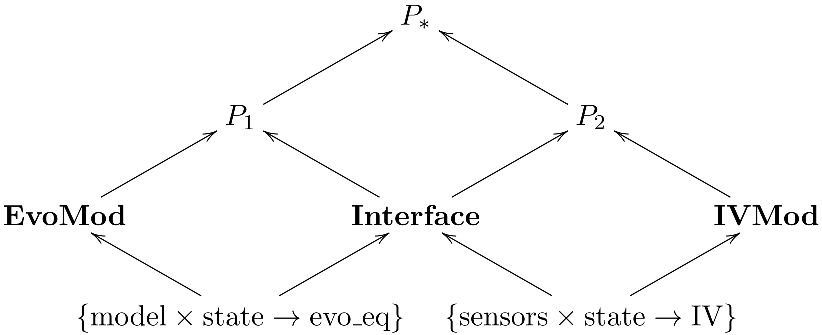
4) Aggregates are defined semantically.

5) Push out.



Iterating Pushouts

Theorems on colimits guarantee correctness for iterated constructions:



Plan

(Some) Advantages of CT

- Extensibility.
- Modularity.
- Close connections with formal logic and programming languages.
- Automatic database integration.
- Rich semantics.

(Some) Advantages of CT

- Extensibility.
- Modularity.
- Close connections with formal logic and programming languages.
- Automatic database integration.
- Rich semantics.
- Tools for studying semantic relationships (natural transformations).
- Tools for studying "the same" data in different contexts (adjoints).
- Tools for modelling effects in a functional context (monads).
- Automatic methods for data migration (Kan extension).

Ready for Primetime?

The mathematics is solid; the software is not (yet).

Haskell - A functional programming language using monads.

FQL - A query language and IDE for building categorical databases.

OPL - A graphical programming language based on operads.

DOL - Distributed Ontology Language based on institutions.

No general purpose environments for implementing and analysing categorical models.

Reaching Critical Mass

New developments:

Ologs	-	A human-readable graphical specification for categories.
Spivak	-	*Category Theory for the Sciences*

Want to learn more, or help?

Email: sub@cmu.edu

spencer.breiner@nist.gov

Web: Categorical data project