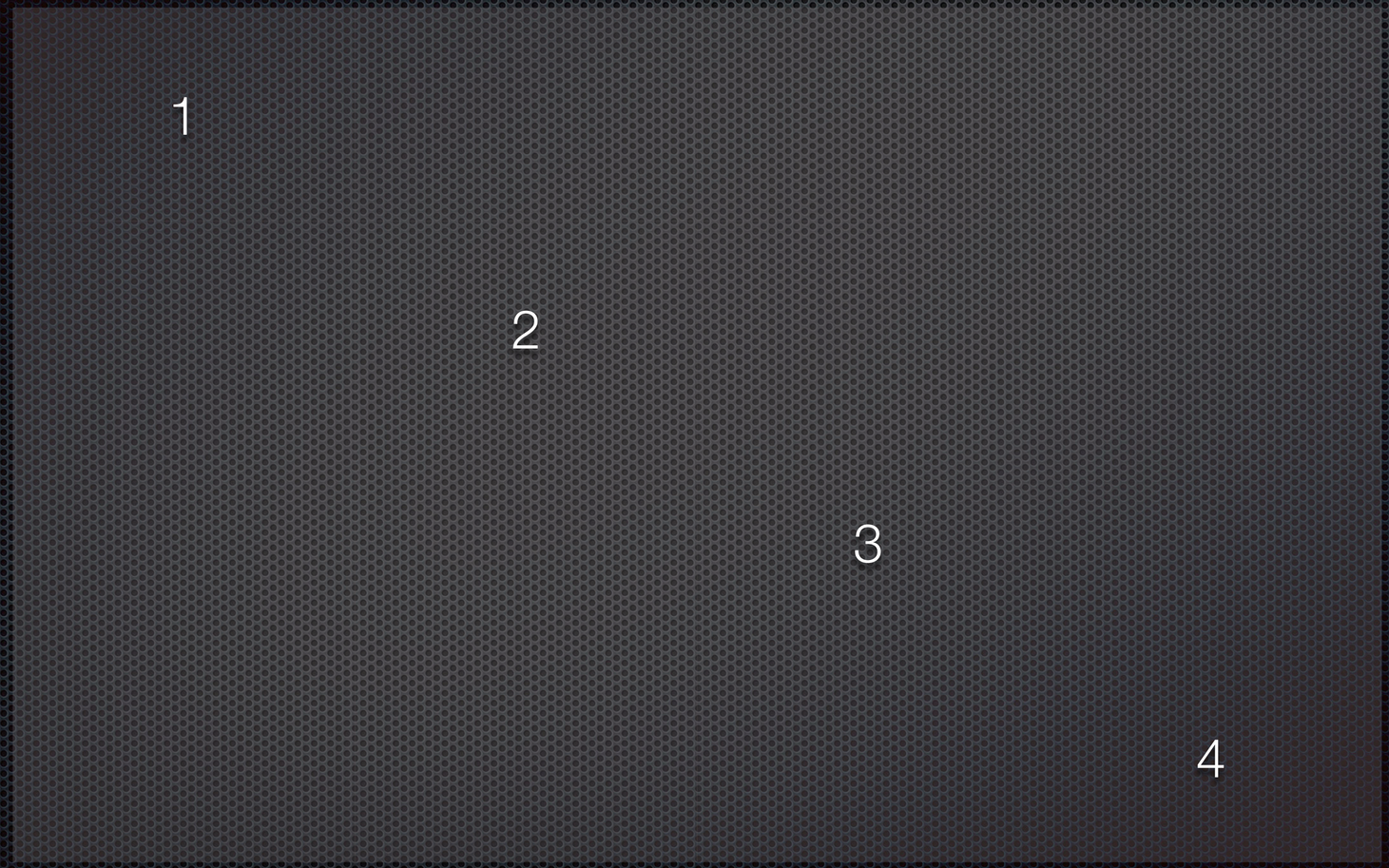
Limits and Colimits: Reasoning with Diagrams for AGI

Sridhar Mahadevan, Adobe Research and U.Mass, Amherst

Solving Sudoku



Solving Sudoku

1	4	2	3
3	2	4	1
4	1	3	2
2	3	1	4

Diagrams and AGI

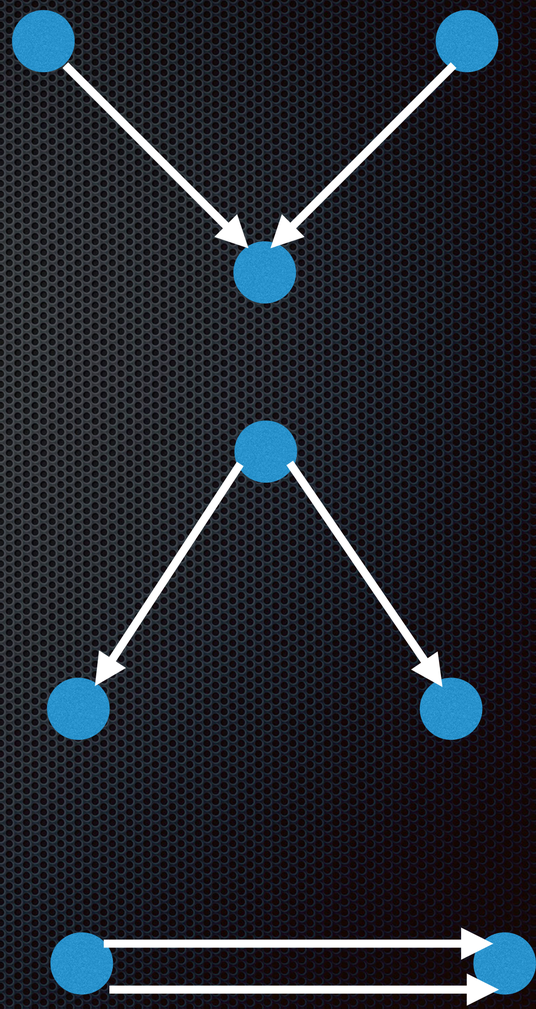
- Reasoning with diagrams is a core part of categorical AGI
- It is central to Diagrammatic Backpropagation and Geometric Transformers
- It is also essential to Topos Causal Models
- Diagrams capture constraints that are universal (Yoneda Lemma)

Diagrams

Diagrams are functors F: J->C

Here, J is a finite index category

We want to define how to “solve” a diagram



Database Category

We saw that a database is a functor D: I -> S

I is a category of instances

S is a database schema

[Spivak, Simplicial Databases, Arxiv]

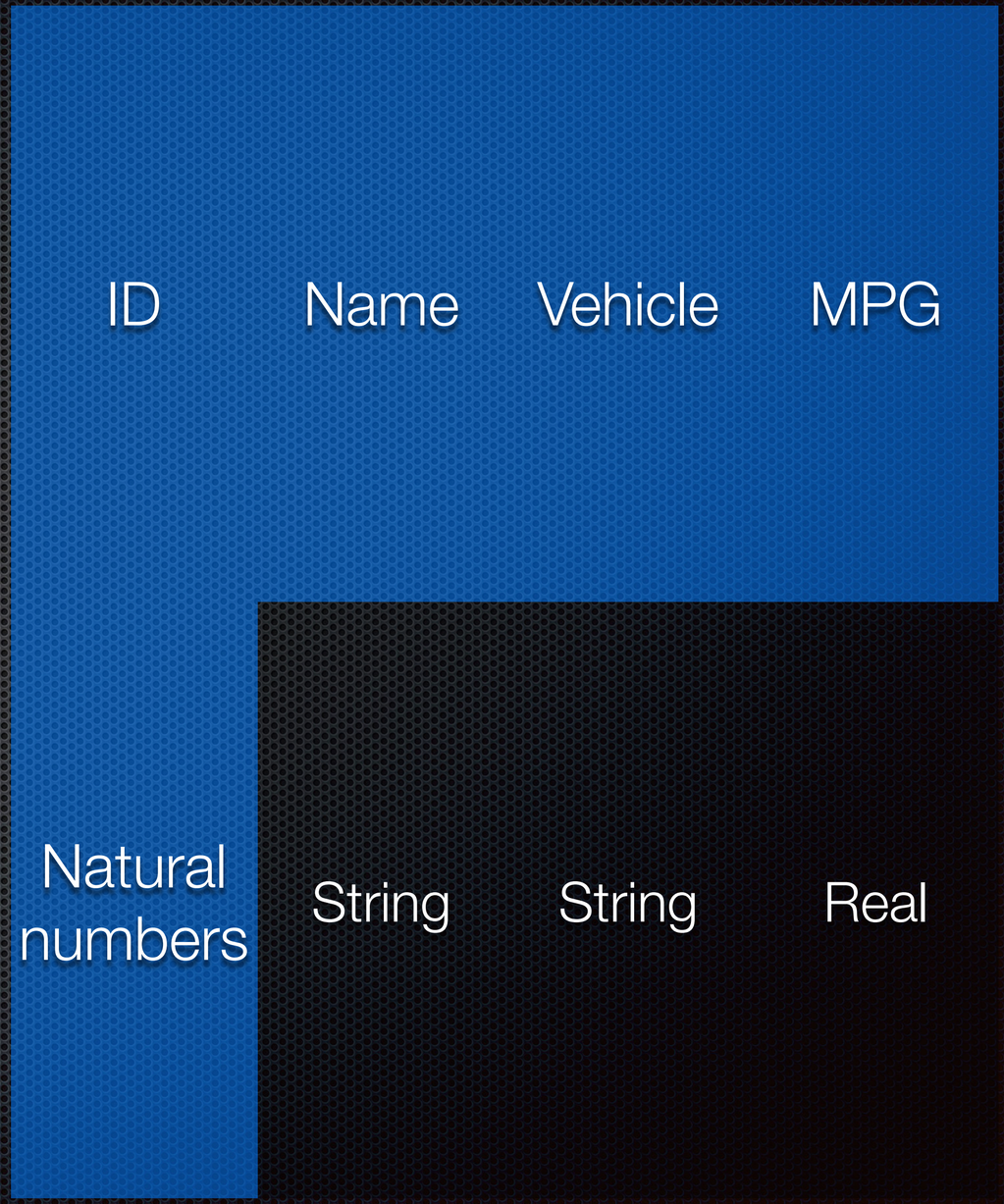
ID	Name	Vehicle	Type
1	Sridhar	Tesla	EV Sedan
2	Steve	Toyota	Gas Sedan
3	Marilyn	Honda	Gas SUV
4	Joe	Ford	Gas Pickup

Data Types

A datatype T: U -> DT

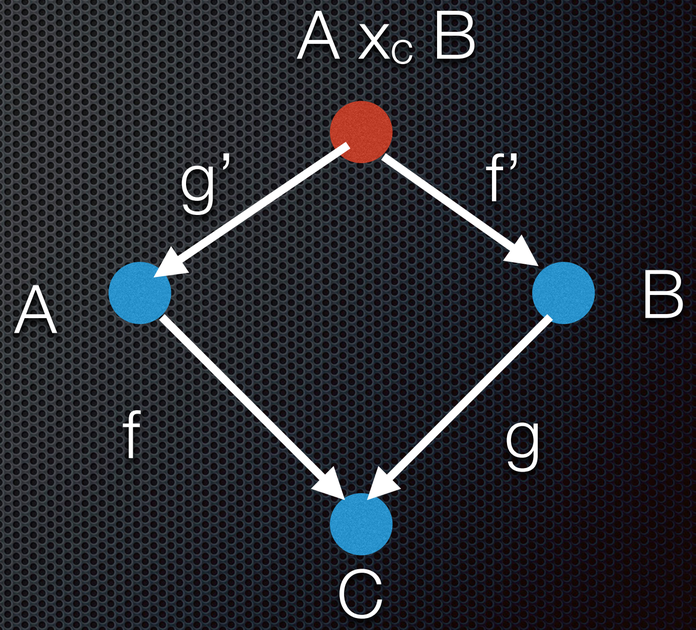
DT: Set of datatypes

U: Domains



[Spivak, Simplicial Databases, Arxiv]

Pullback



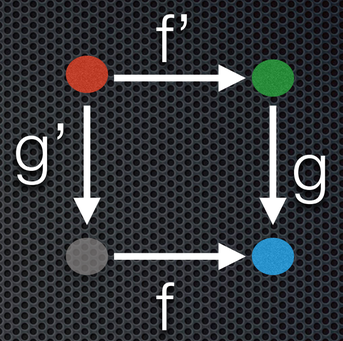
A pullback is a type of limit

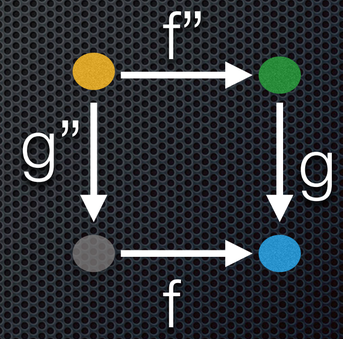
It abstracts many concrete operations: maximum, intersection, product, etc.

Pullback

In the category of sets,
pullbacks always exist

The pullback of f and g is simply
the set of ordered pairs (x,y)
such that f(x) = g(y)





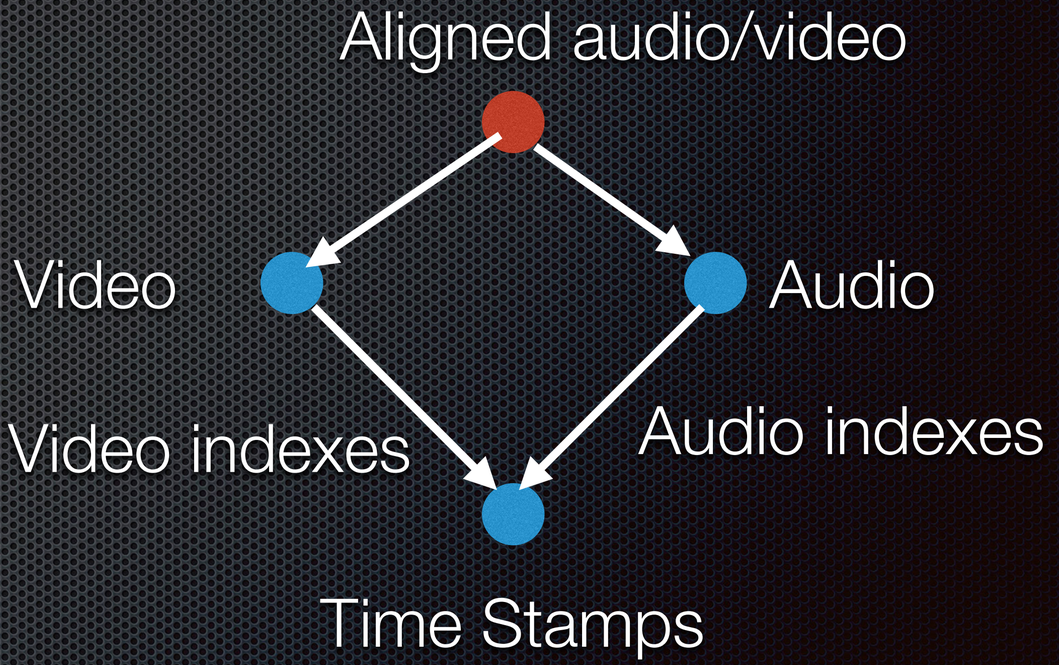
f” = f’ h

g” = g’ h

Pullback

A pullback is a type of limit

It abstracts many concrete operations: maximum, intersection, product, etc.



See Deep Learning demo on pullback of AV on GitHub repo

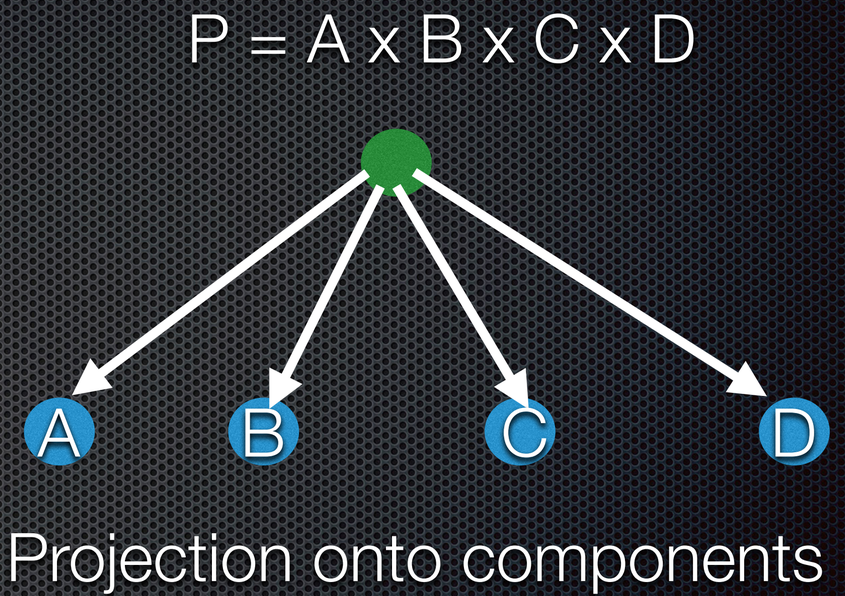
https://colab.research.google.com/github/sridharmahadevan/Category-Theory-for-AGI-UMass-CMPSCI-692CT/blob/main/notebooks/week3_synthetic_av_pullback-2.ipynb

Cartesian Product as a Limit

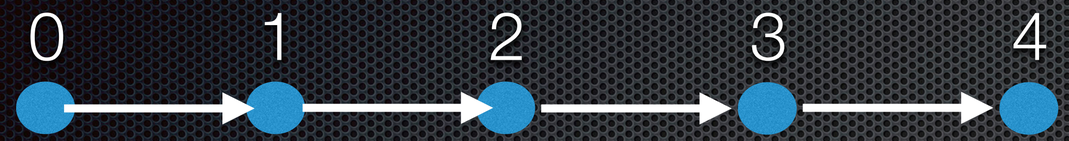
A Cartesian product of sets is defined universally in terms of a limit

It is the unique object that has projections onto component sets

Every function onto component sets must factor uniquely through the Cartesian product

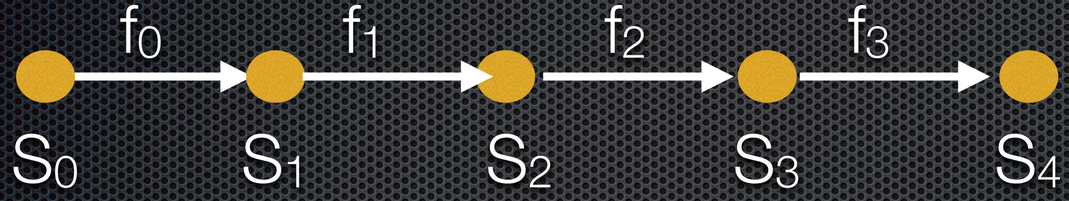


Limits over a Time Series



Indexing category

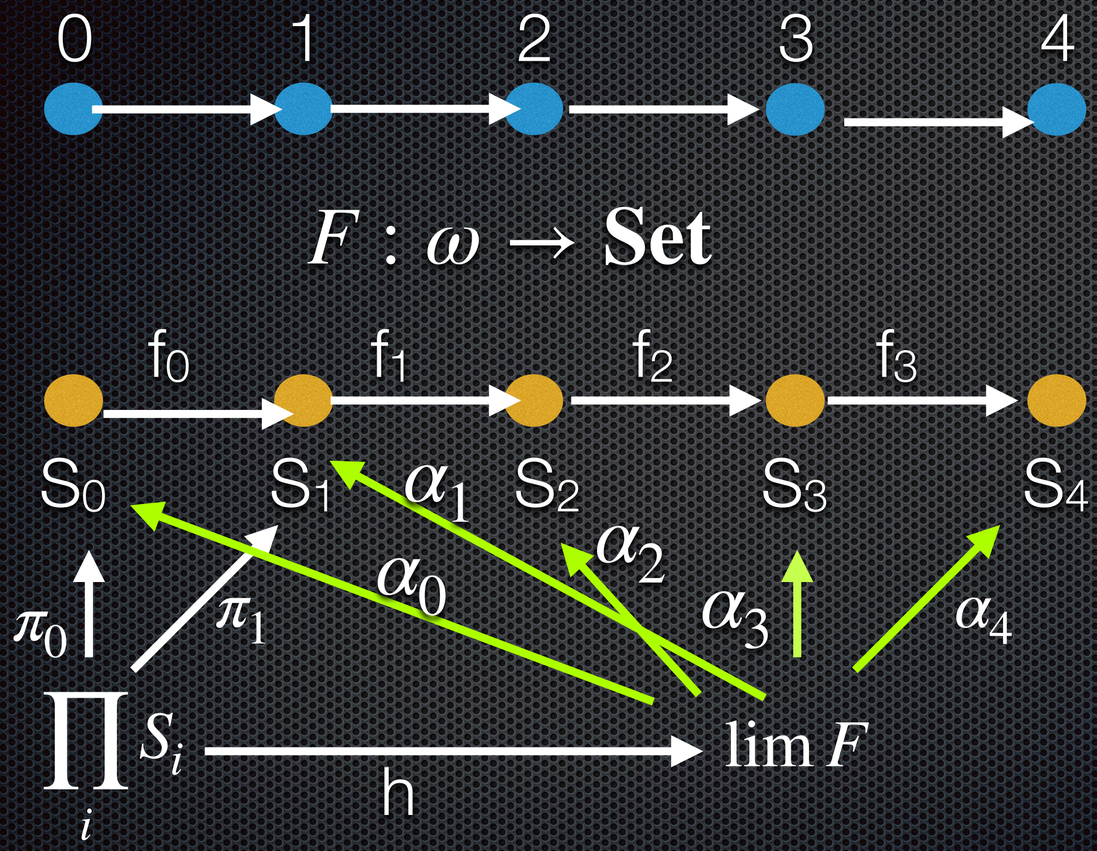
$$F : \omega \to \mathbf{Set}$$



Sets

Note that the limit here cannot be Cartesian product!

Limits over a Time Series



Indexing category

Sets

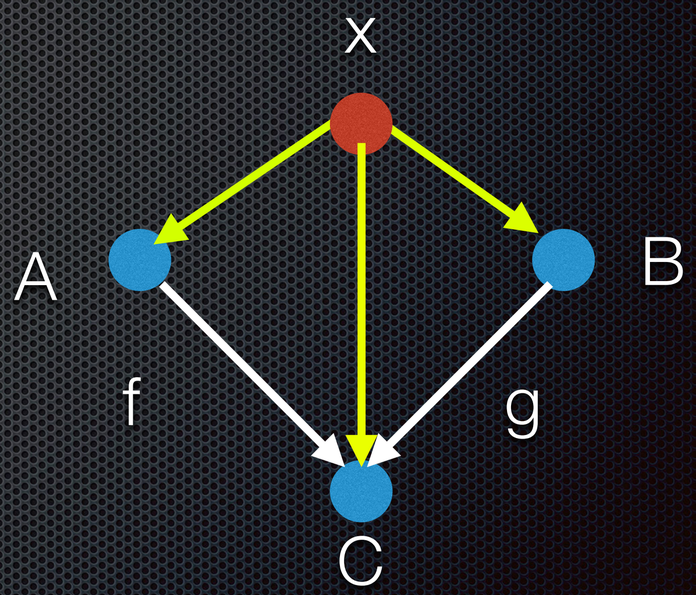
$\alpha_{n+1} = f_{n+1}\alpha_n$

Cones

A cone over diagram F: J->C with apex x is a natural transformation from the constant functor at x to F

The constant functor at x maps every object in J to x, and every arrow in J to the identity arrow at x

$$\Delta : C \to C^J$$



Cones and Cocones: Formal Definition

DEFINITION 3.1.2. A **cone over** a diagram $F\colon \mathsf{J} \to \mathsf{C}$ with **summit** or **apex** $c \in \mathsf{C}$ is a natural transformation $\lambda\colon c \Rightarrow F$ whose domain is the constant functor at c. The components $(\lambda_j\colon c \to Fj)_{j\in\mathsf{J}}$ of the natural transformation are called the **legs** of the cone. Explicitly:

- The data of a cone over $F\colon \mathsf{J} \to \mathsf{C}$ with summit c is a collection of morphisms $\lambda_j\colon c \to Fj$, indexed by the objects $j \in \mathsf{J}$.
- A family of morphisms $(\lambda_j\colon c \to F_j)_{j\in\mathsf{J}}$ defines a cone over F if and only if, for each morphism $f\colon j \to k$ in J, the following triangle commutes in C:

(3.1.3)

$$\begin{array}{ccc} & c & \\ {\scriptstyle\lambda_j}\swarrow & & \searrow{\scriptstyle\lambda_k} \\ Fj & \xrightarrow[Ff]{} & Fk \end{array}$$

Dually, a **cone under** F with **nadir** c is a natural transformation $\lambda\colon F \Rightarrow c$, whose **legs** are the components $(\lambda_j\colon F_j \to c)_{j\in\mathsf{J}}$. The naturality condition asserts that, for each morphism $f\colon j \to k$ of J, the triangle

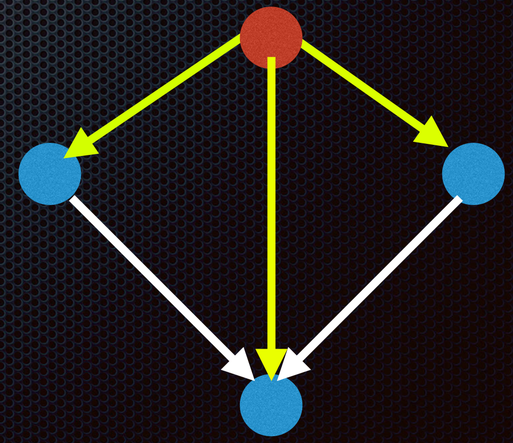
$$\begin{array}{ccc} Fj & \xrightarrow{Ff} & Fk \\ & {\scriptstyle\lambda_j}\searrow \quad \swarrow{\scriptstyle\lambda_k} & \\ & c & \end{array}$$

commutes in C.

Figure source: Riehl, Category Theory in Context

Cones as set-valued functors

- Given a diagram F: J->C, we can define a set-valued functor
  - Cone(-, F): the set of cones over F with summit c
  - Cone(F, -): the set of cones under F with nadir c



Limits and Colimits: Representable functors using the Yoneda Lemma

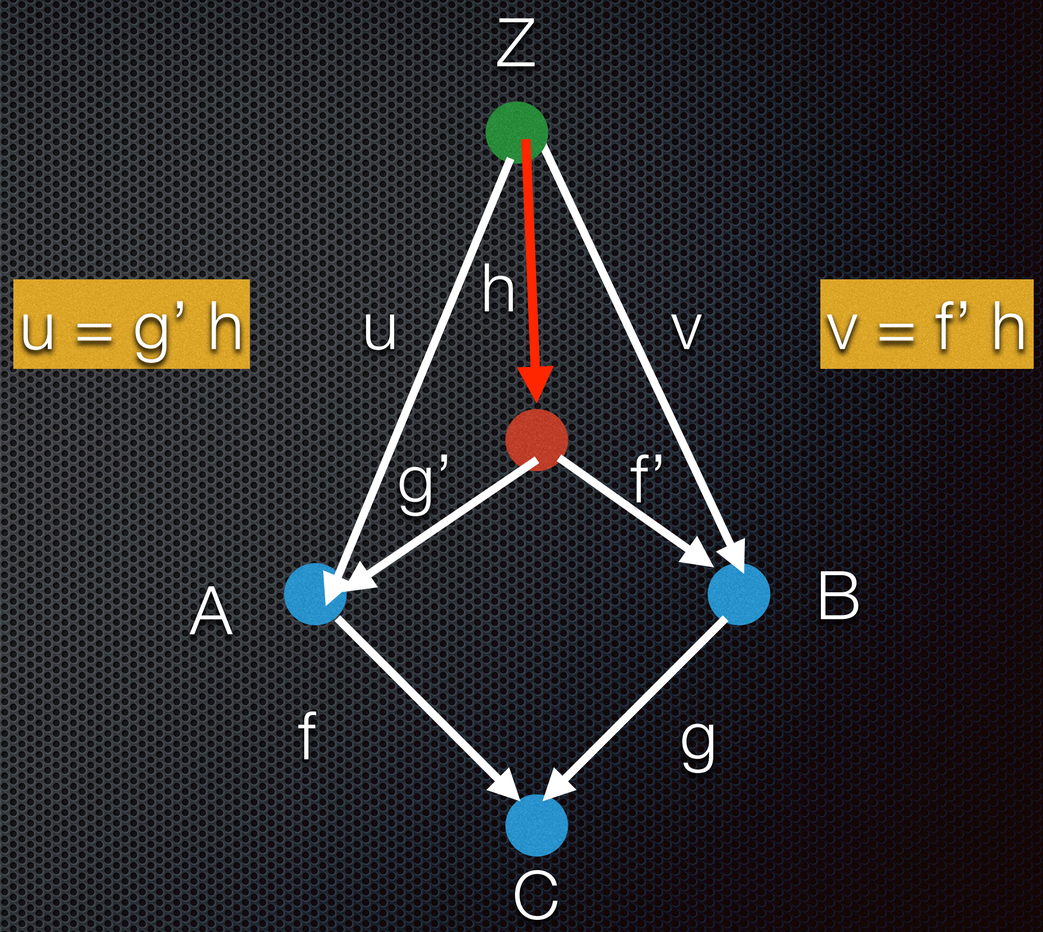
- A limit of diagram F is a representation for Cone(-, F)
  - C(-, lim F) ~ Cone(-, F)
- A colimit of diagram F is a representation of Cone(F, -)
  - C(colim F, -) ~ Cone(F, -)

Limits using Category of Elements

- We can model limits in terms of a category of elements
  - $\int Cone(-, F)$
- Each object in this category is an actual cone Cone(x, F)
- Arrows are mappings from one cone to another
- A limit is the terminal object in the category of elements $\int Cone(-, F)$

Universal Property

Universal property of a pullback defines it as a type of limit



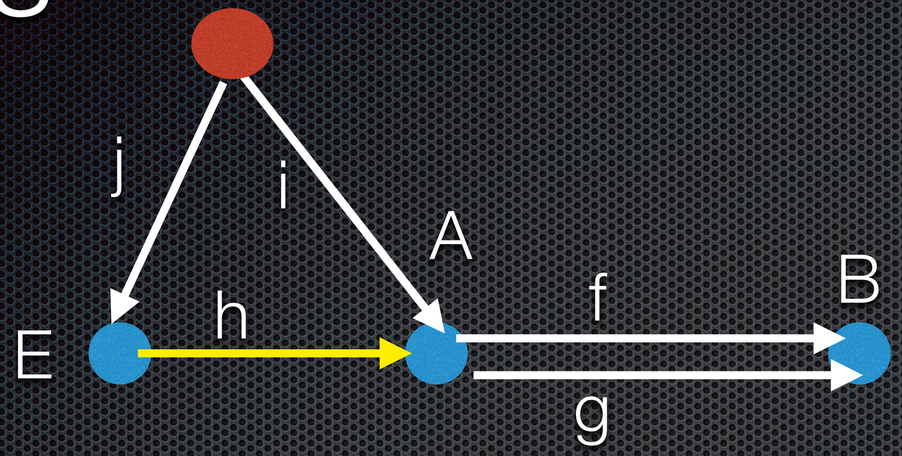
Colimits using Category of Elements

- We can model colimits in terms of a category of elements
  - $\int Cone(F, -)$
- Each object in this category is an actual cone Cone(x, F)
- Arrows are mappings from one cone to another
- A colimit is the initial object in the category of elements $\int Cone(F, -)$

Cartesian Products as Limits

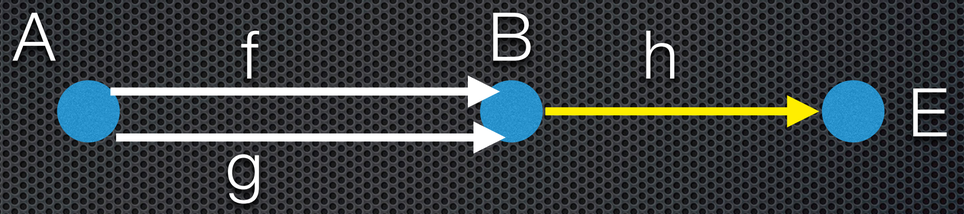
- A Cartesian product is a limit of a diagram indexed by a discrete category with only identity arrows
  - Limit is $\prod_j F_j$
  - The legs of the limit cone are projections $\pi_k : \prod_j F_j \to F_k$

Equalizers



- An equalizer is a limit of a diagram with two parallel arrows
- Note that equalizers are “right cancellable”:
  - f h = g h implies that f = g

Coequalizers



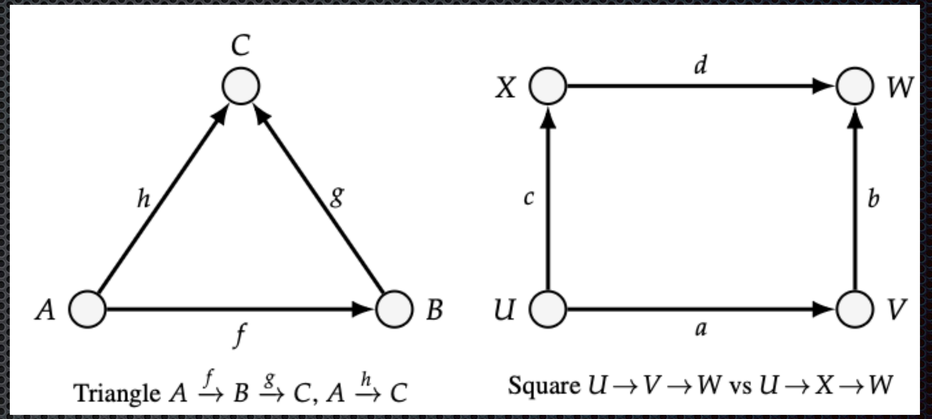
- A coequalizer is a colimit of a diagram with two parallel arrows
- Note that equalizers are “left cancellable”:
  - h f = h g implies that f = g

(Co)complete Categories

- A digram is small if its indexing category is small
- A category is complete if it admits limits of all small diagrams
- A category is cocomplete if it admits colimits of all small diagrams
- The category of Sets is complete and cocomplete

Diagrammatic Backpropagation

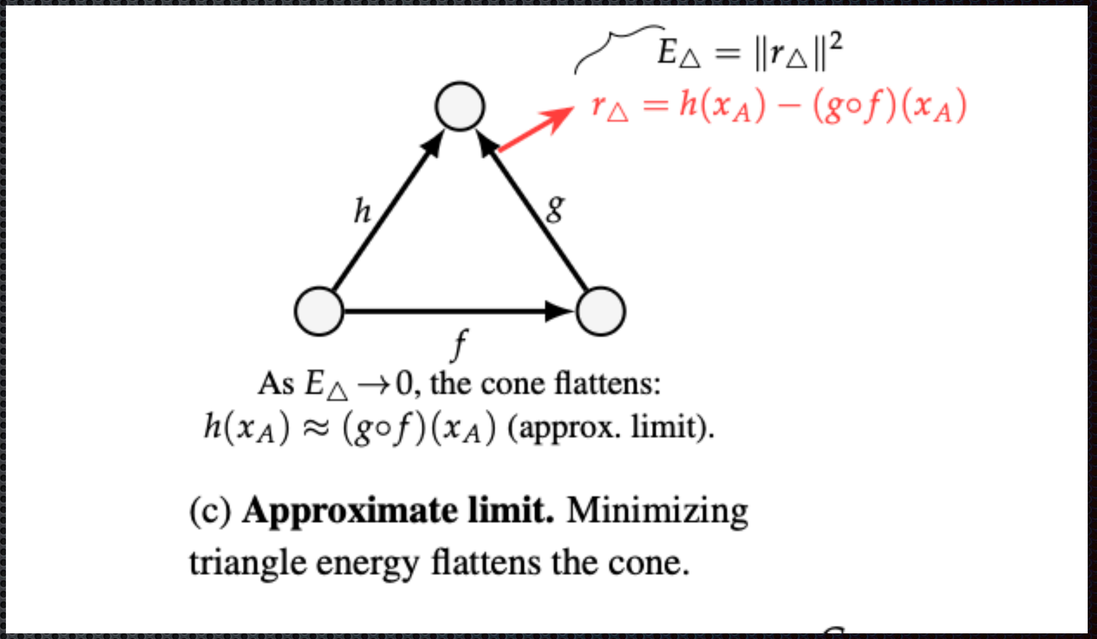
DB extends vanilla backpropagation to include diagrammatic reasoning



Mahadevan, Categories for AGI, 2026

Diagrammatic Backpropagation

DB extends vanilla backpropagation to include diagrammatic reasoning

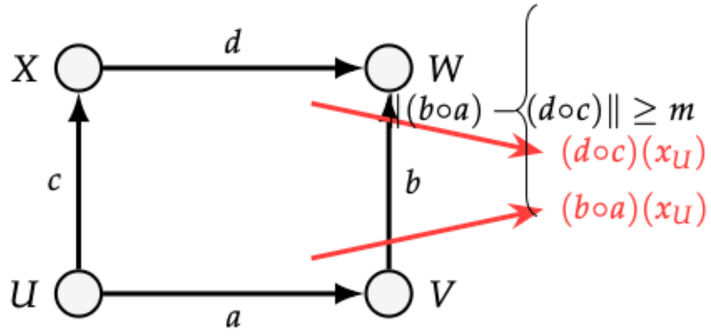


(c) **Approximate limit.** Minimizing triangle energy flattens the cone.

Mahadevan, Categories for AGI, 2026

Diagrammatic Backpropagation

DB extends vanilla backpropagation to include diagrammatic reasoning



Contrastive margin m prevents collapse of distinct branches (approx. colimit separation).

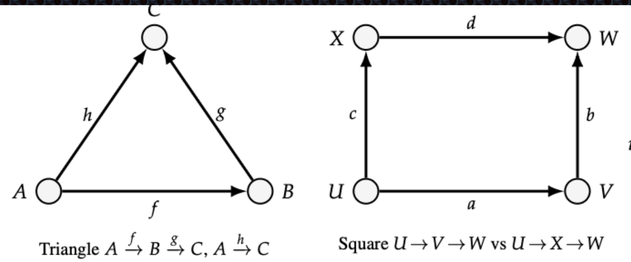
(d) **Approximate colimit.** Margin keeps branches distinct.

Mahadevan, Categories for AGI, 2026

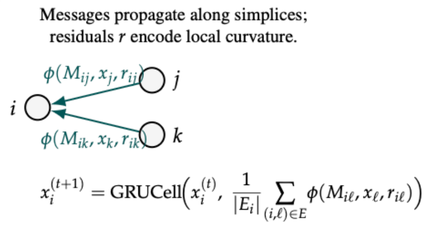
Diagrammatic Backpropagation

DB extends vanilla backpropagation to include diagrammatic reasoning

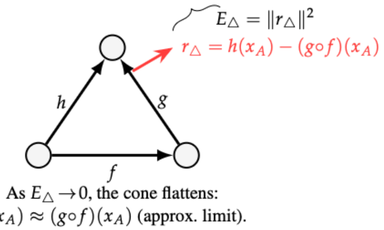
Mahadevan, Categories for AGI, 2026



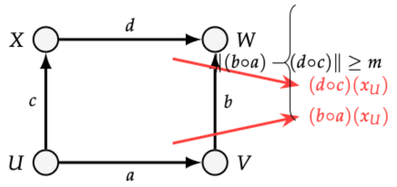
(a) **Simplicial diagrams.** Edges carry morphisms $M_f, M_g, \ldots$.



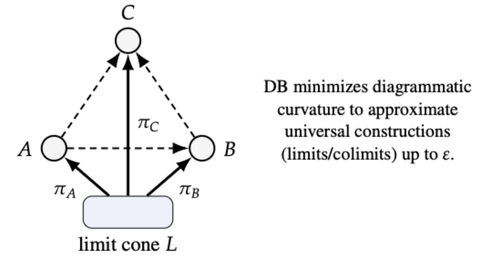
(b) **Simplicial message passing.** Curvature diffuses via messages.



(c) **Approximate limit.** Minimizing triangle energy flattens the cone.



(d) **Approximate colimit.** Margin keeps branches distinct.



(e) **Universal view.** Curvature $\to$ 0 $\Rightarrow$ numeric limit; margin $\Rightarrow$ numeric colimit.

Figure 12: **Simplicial message passing and limit/colimit approximation in DB.** (A) Triangle and square simplices encode compositional constraints. (B) DB aggregates messages along simplices; residuals carry local curvature. (C) Minimizing triangle energy $\|h(x_A) - (g \circ f)(x_A)\|^2$ flattens the cone (approx. limit). (D) A margin on square paths prevents collapse (approx. colimit).

Background Reading

- Chapter 3, Emily Riehl, Category Theory in Context
- Chapter 3, Sridhar Mahadevan, Categories for AGI
- GitHub code:
  - Synthetic AV demo of pullback using deep learning
  - Python Suduko GT solver uses diagrammatic constraints