3. Databases

Database - organised system of interlocking tables

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

Department	DName	Secr
101	Sales	1
102	IT	3

(3.1)

ID

column → row elements of this column should be unique

* WorksIn, Mngr, Secr → internal references → known as "foreign keys" as they are used in ID columns of a foreign table

⇓

CAN BE CHANGED

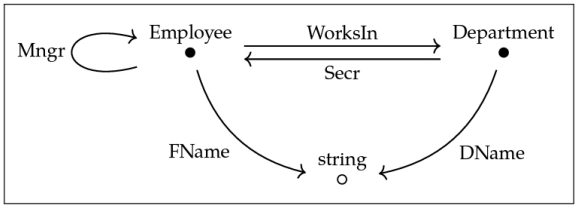
* DName, FName → external references → strings or int

⇓

CANNOT BE CHANGED

Hasse Diagram:

easySchema :=



DATABASE SCHEMA (A category)

* We can enforce constraints like every dep's secretary must work in that dep. So, in terms of eq:

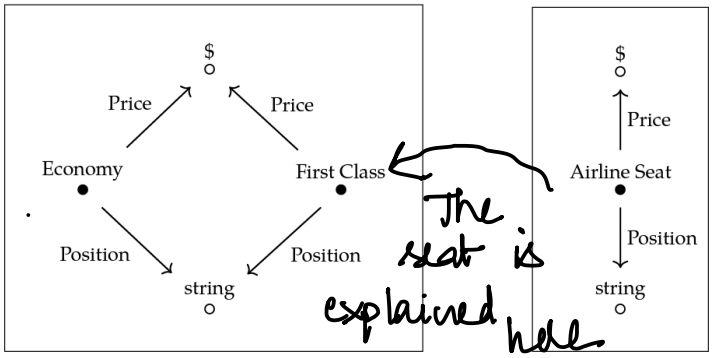
Department . Secr . WorksIn = Department

↳ used in FQL

Schema A is more detailed than B.

[There is a functor that go from A to B or viceversa)

$A :=$



$=: B$

CATEGORY :

Definition 3.6. To specify a *category* $\mathcal{C}$:

(i) one specifies a collection[2] $\mathrm{Ob}(\mathcal{C})$, elements of which are called *objects*.

(ii) for every two objects c, d, one specifies a set $\mathcal{C}(c,d)$,[3] elements of which are called *morphisms* from c to d.

(iii) for every object $c \in \mathrm{Ob}(\mathcal{C})$, one specifies a morphism $\mathrm{id}_c \in \mathcal{C}(c,c)$, called the *identity morphism* on c.

(iv) for every three objects $c, d, e \in \mathrm{Ob}(\mathcal{C})$ and morphisms $f \in \mathcal{C}(c,d)$ and $g \in \mathcal{C}(d,e)$, one specifies a morphism $f \fatsemi g \in \mathcal{C}(c,e)$, called *the composite of f and g*.

We will sometimes write an object $c \in \mathcal{C}$, instead of $c \in \mathrm{Ob}(\mathcal{C})$. It will also be convenient to denote elements $f \in \mathcal{C}(c,d)$ as $f : c \to d$. Here, c is called the *domain* of f, and d is called the *codomain* of f.

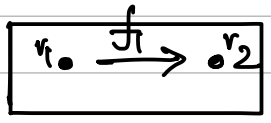
These constituents are required to satisfy two conditions:

(a) *unitality*: for any morphism $f : c \to d$, composing with the identities at c or d does nothing: $\mathrm{id}_c \fatsemi f = f$ and $f \fatsemi \mathrm{id}_d = f$.

(b) *associativity*: for any three morphisms $f : c_0 \to c_1$, $g : c_1 \to c_2$, and $h : c_2 \to c_3$, the following are equal: $(f \fatsemi g) \fatsemi h = f \fatsemi (g \fatsemi h)$. We write this composite simply as $f \fatsemi g \fatsemi h$.

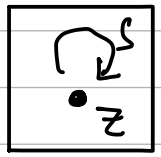
⇒ Free categories :

Defined from a graph $G = (V, A, s, t)$

Eg: $\mathcal{Q} :=$ Free ()

OBJ. - v_1, v_2

MORPHISMS - id_{v_1}, id_{v_2}, f_1

⇒ $\mathbb{N}$ as free categories : 

* It has one vertex & one morphisms but ∞ ly many paths

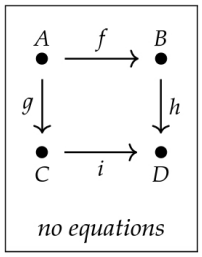
* Each path's length CORRESPONDS the natural numbers like, z is path of length 0 , S is for one, S;S

is for path of length 2, $s;s;s$ is for 3 and so on

* We can add "structures" to these free categories one is

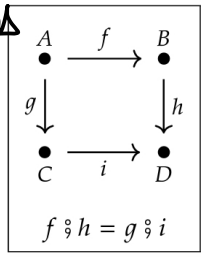
10 Morphisms

Free_square :=

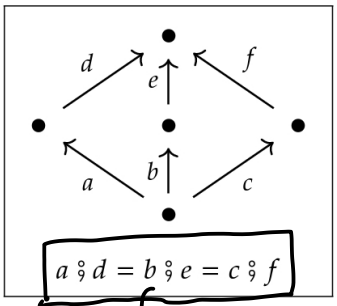


9 morphisms only

Comm_square :=



⇒ Preorders AS categories : $(P, \leq)$ be the preorder



$Ob(\mathcal{C}) = P$

$p \to q$ if $p \leq q$

id exist for all elements : $p \leq p$

$p \leq q,\ q \leq r \Rightarrow p \leq r$

↳ This is imp. to ensure transitivity & it is not free.

* Irrespective of no. of morphisms from $\bullet_q \to \bullet_q$ preorder remains same so, all are same

NOTE : Preorders & Categories are not one-one

Set - Category of sets : OBJ - Sets MORP. - Functions

Finset - OBJ - Finite Sets

There are many other categories that mathematicians care about:

- **Top**: the category of topological spaces (neighborhood)
- **Grph**: the category of graphs (connection)
- **Meas**: the category of measure spaces (amount)
- **Mon**: the category of monoids (action)
- **Grp**: the category of groups (reversible action, symmetry)
- **Cat**: the category of categories (action in context, structure)

* A monoid with every morphism an isomorphism is known as a GROUP

* Free categories have no isomorphisms except id's

⇒ Sets & functions as databases :

OBJ : Sets → ID column of a table
for a function $f : A \to B$ requires 2 tables :

one for A (ID column) & one for B (ID column)
with its f column

Eg :

Beatle	Played
George	Lead guitar
John	Rhythm guitar
Paul	Bass guitar
Ringo	Drums

(A)

f → played

Rock-and-roll instrument
Bass guitar
Drums
Keyboard
Lead guitar
Rhythm guitar

(B)

Database schema : • —PLAYED→ •

Beatle — Rock & Roll Instrument

⇒ Functors :

Def :

Definition 3.35. Let $\mathcal{C}$ and $\mathcal{D}$ be categories. To specify a *functor from* $\mathcal{C}$ *to* $\mathcal{D}$, denoted $F\colon \mathcal{C} \to \mathcal{D}$,

(i) for every object $c \in \mathrm{Ob}(\mathcal{C})$, one specifies an object $F(c) \in \mathrm{Ob}(\mathcal{D})$;
(ii) for every morphism $f\colon c_1 \to c_2$ in $\mathcal{C}$, one specifies a morphism $F(f)\colon F(c_1) \to F(c_2)$ in $\mathcal{D}$.

The above constituents must satisfy two properties:

(a) for every object $c \in \mathrm{Ob}(\mathcal{C})$, we have $F(\mathrm{id}_c) = \mathrm{id}_{F(c)}$.
(b) for every three objects $c_1, c_2, c_3 \in \mathrm{Ob}(\mathcal{C})$ and two morphisms $f \in \mathcal{C}(c_1, c_2)$, $g \in \mathcal{C}(c_2, c_3)$, the equation $F(f \fatsemi g) = F(f) \fatsemi F(g)$ holds in $\mathcal{D}$.

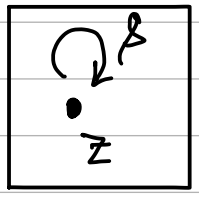
It is basically a map from one cat to another which connects all the obj of first to some or all of the other cat & preserves the str. & the morphisms of first gets mapped to that of 2nd (auto-)

* for preorders, if the map preserve str., then the functor is basically the monotone map b/w 2 preorders

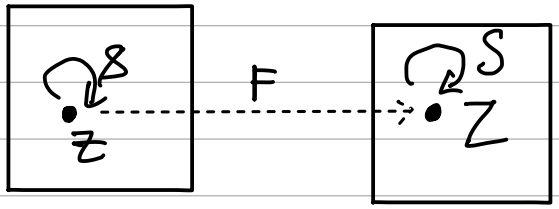
⇒ For a schema i.e. finitely presented category $\mathcal{C}$ the functor $I : \mathcal{C} \to Set$ is called "$\mathcal{C}$-instance" or "set valued functor on $\mathcal{C}$"

Eg: $F: 1 \longrightarrow Set$
↳ "set category"

- F(1) is simple a set

* $\mathcal{C}$ = 

$S;S = S$

$\mathcal{C} \to Set$ will look like


Eg $S;S = S$

Z is a set
S is a function : $Z \to Z$

⇒ Natural Transformation :

Def :

Definition 3.49. Let $\mathcal{C}$ and $\mathcal{D}$ be categories, and let $F, G : \mathcal{C} \to \mathcal{D}$ be functors. To specify a *natural transformation* $\alpha : F \Rightarrow G$,

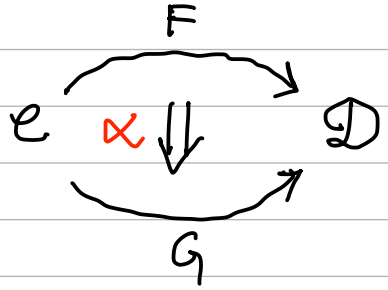
(i) for each object $c \in \mathcal{C}$, one specifies a morphism $\alpha_c : F(c) \to G(c)$ in $\mathcal{D}$, called the *c-component of* α.

These components must satisfy the following, called the *naturality condition*:

(a) for every morphism $f : c \to d$ in $\mathcal{C}$, the following equation must hold:

$$F(f) \fatsemi \alpha_d = \alpha_c \fatsemi G(f).$$

A natural transformation $\alpha : F \to G$ is called a *natural isomorphism* if each component α_c is an isomorphism in $\mathcal{D}$.



The commutative diagram for $c, d \in \text{Obj}(\mathcal{C})$ looks like →

$$\begin{array}{ccc} F(c) & \xrightarrow{\alpha_c} & G(c) \\ {\scriptstyle F(f)}\downarrow & & \downarrow{\scriptstyle G(f)} \\ F(d) & \xrightarrow[\alpha_d]{} & G(d) \end{array}$$

Here,

$f \rightarrow$ morphism

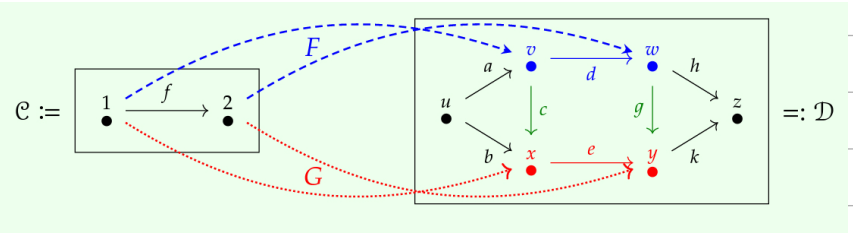
$F, G \rightarrow$ Functors

$\alpha_c, \alpha_d \equiv \alpha \rightarrow$ natural transformation

⇒ Def :

Definition 3.51. A *diagram* D in $\mathcal{C}$ is a functor $D : \mathcal{J} \to \mathcal{C}$ from any category $\mathcal{J}$, called the *indexing category* of the diagram D. We say that D *commutes* if $D(f) = D(f')$ holds for every parallel pair of morphisms $f, f' : a \to b$ in $\mathcal{J}$.[7]

Eg:



⇒ Functor Category :

Definition 3.54. Let $\mathcal{C}$ and $\mathcal{D}$ be categories. We denote by $\mathcal{D}^{\mathcal{C}}$ the category whose objects are functors $F : \mathcal{C} \to \mathcal{D}$ and whose morphisms $\mathcal{D}^{\mathcal{C}}(F, G)$ are the natural transformations $\alpha : F \to G$. This category $\mathcal{D}^{\mathcal{C}}$ is called the *functor category*, or the *category of functors from* $\mathcal{C}$ *to* $\mathcal{D}$.

Eg :

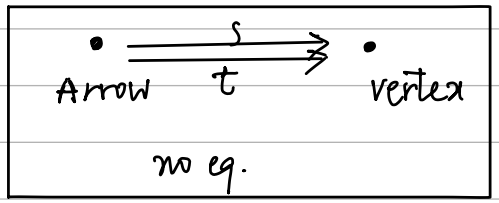
$\mathcal{N} = (\mathbb{N}, \leq)$
↳ preorder

* $\mathcal{N}^{\mathcal{N}}$ is also a preorder of monotone maps from $\mathbb{N} \to \mathbb{N}$

⇒ The category of instances in a schema :

Definition 3.60. Suppose that $\mathcal{C}$ is a database schema and $I, J : \mathcal{C} \to \mathbf{Set}$ are database instances. An *instance homomorphism* between them is a natural transformation $\alpha : I \to J$. Write $\mathcal{C}$-**Inst** $:= \mathbf{Set}^{\mathcal{C}}$ to denote the functor category as defined in Definition 3.54.

Eg: (1) Category of Graphs (Gr) as a functor category

Gr: 

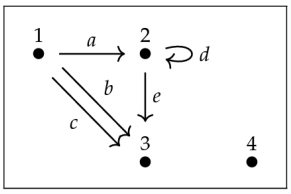
Arrow	source	target
a	1	2
b	1	3
c	1	3
d	2	2
e	2	3

Vertex
1
2
3
4

⇗ AS A DATABASE

Gr-instance $I : Gr \to Set$, ⇘ AS A GRAPH

Here $I(\text{Arrow}) = \{a, b, c, d, e\}$, and $I(\text{Vertex}) = \{1, 2, 3, 4\}$. One can draw the instance I as a graph:

$I =$ 

So, the obj. in the category of Gr-Inst are graphs as above & the morphism b/w these graphs, i.e, natural transformations are called "Graph Homomorphisms"

If $G, H : Gr \to Set$ i.e, $G, H \in Obj(Gr\text{-}Inst)$

$\alpha : G \to H$ contains $\alpha_{arrow} : G(arrow) \to H(arrow)$

& $\alpha_{vertex} : G(vertex) \to H(vertex)$

for α to be a graph homomorphism it must respect source & target functions.

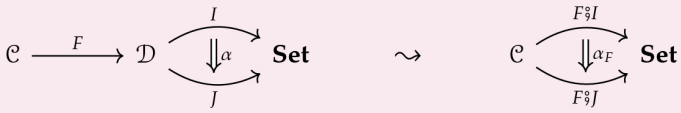
$$\begin{array}{ccc} G(\text{Arrow}) & \xrightarrow{\alpha_{\text{Arrow}}} & H(\text{Arrow}) \\ {\scriptstyle G(\text{source})}\downarrow & & \downarrow{\scriptstyle H(\text{source})} \\ G(\text{Vertex}) & \xrightarrow[\alpha_{\text{Vertex}}]{} & H(\text{Vertex}) \end{array} \qquad \begin{array}{ccc} G(\text{Arrow}) & \xrightarrow{\alpha_{\text{Arrow}}} & H(\text{Arrow}) \\ {\scriptstyle G(\text{target})}\downarrow & & \downarrow{\scriptstyle H(\text{target})} \\ G(\text{Vertex}) & \xrightarrow[\alpha_{\text{Vertex}}]{} & H(\text{Vertex}) \end{array}$$

DATA TRANSFER:

* Pulling back along a functor:

Definition 3.68. Let $\mathcal{C}$ and $\mathcal{D}$ be categories and let $F\colon \mathcal{C} \to \mathcal{D}$ be a functor. For any set-valued functor $I\colon \mathcal{D} \to \mathbf{Set}$, we refer to the composite functor $F \fatsemi I\colon \mathcal{C} \to \mathbf{Set}$ as the *pullback of I along F*.

Given a natural transformation $\alpha\colon I \Rightarrow J$, there is a natural transformation $\alpha_F\colon F\fatsemi I \Rightarrow F \fatsemi J$, whose component $(F \fatsemi I)(c) \to (F \fatsemi J)(c)$ for any $c \in \mathrm{Ob}(\mathcal{C})$ is given by $(\alpha_F)_c := \alpha_{Fc}$.

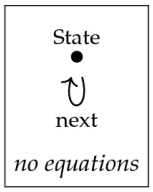


This uses the data of F to define a functor $\Delta_F\colon \mathcal{D}\text{-}\mathbf{Inst} \to \mathcal{C}\text{-}\mathbf{Inst}$.

Eg: I: DDS → Set

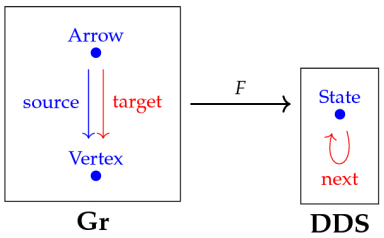
↓

Discrete Dynamical Sys.



State	next
1	4
2	4
3	5
4	5
5	5
6	7
7	6

F: Gr → DDS



→ { Arrow, Vertex to State

source to id_{state}

target to next

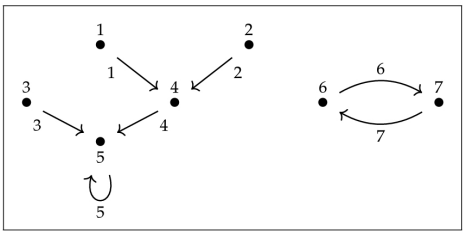
* Gr $\xrightarrow{F;I}$ Set

↓

Pullback of data I along F

Arrow	source	target
1	1	4
2	2	4
3	3	5
4	4	5
5	5	5
6	6	7
7	7	6

Vertex
1
2
3
4
5
6
7



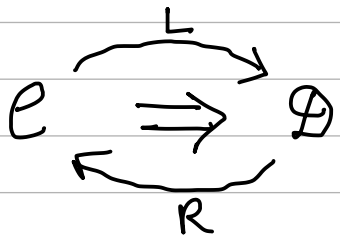
⇒ Adjunctions :

Definition 3.70. Let $\mathcal{C}$ and $\mathcal{D}$ be categories, and $L\colon \mathcal{C} \to \mathcal{D}$ and $R\colon \mathcal{D} \to \mathcal{C}$ be functors. We say that L *is left adjoint to* R (and that R *is right adjoint to* L) if, for any $c \in \mathcal{C}$ and $d \in \mathcal{D}$, there is an isomorphism of hom-sets

$$\alpha_{c,d}\colon \mathcal{C}(c, R(d)) \xrightarrow{\cong} \mathcal{D}(L(c), d)$$

that is natural in c and d.[8]

Given a morphism $f\colon c \to R(d)$ in $\mathcal{C}$, its image $g := \alpha_{c,d}(f)$ is called its *mate*. Similarly, the mate of $g\colon L(c) \to d$ is f.



⇒ → indicates dir^n of left adjoint

* Galois connection is adjunctions b/w preorders simply

Example 3.72. Let $B \in \mathrm{Ob}(\mathbf{Set})$ be any set. There is an adjunction called 'currying B,' after the logician Haskell Curry:

$$\mathbf{Set} \underset{(-)^B}{\overset{-\times B}{\rightleftarrows}} \mathbf{Set} \qquad\qquad \mathbf{Set}(A \times B, C) \cong \mathbf{Set}(A, C^B)$$

Abstractly we write it as on the left, but what this means is that for any sets A, C, there is a natural isomorphism as on the right.

To explain this, we need to talk about exponential objects in **Set**. Suppose that B and C are sets. Then the set of functions $B \to C$ is also a set; let's denote it C^B. It's written this way because if C has 10 elements and B has 3 elements then C^B has 10^3 elements, and more generally for any two finite sets $|C^B| = |C|^{|B|}$.

The idea of currying is that given sets A, B, and C, there is a one-to-one correspondence between functions $(A \times B) \to C$ and functions $A \to C^B$. Intuitively, if I have a function f of two variables a, b, I can "put off" entering the second variable: if you give me just a, I'll return a function $B \to C$ that's waiting for the B input. This is the curried version of f. As one might guess, there is a formal connection between exponential objects and what we called hom-elements $b \multimap c$ in Definition 2.79.

* Terminal objects & products refer prev. notes

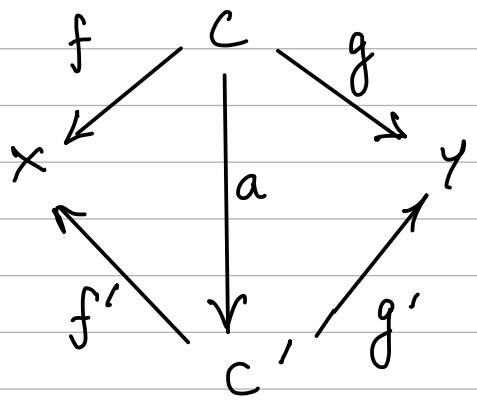
⇒ Limits :

first we define "Cone". Cone(x, y) is a category of cones over x & y in $\mathcal{C}$.

An obj. in Cone(x, y) is simply a pair of maps:

$x \xleftarrow{f} c \xrightarrow{g} y$.

A morphism from $X \xleftarrow{f} C \xrightarrow{g} Y$ to $X \xleftarrow{f'} C' \xrightarrow{g'} Y$ in cone(X,Y) is a morphism $a : C \to C'$ in $\mathcal{C}$ such that the following diagram commutes.



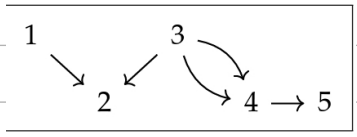
* for products $C' = X \times Y$ & there is unique morphism from any C to $C'(X \times Y)$ such that the triangles commute f', g' are projection maps.

TERMINOLOGY : A diagram in $\mathcal{C}$ is a functor $D : \mathcal{J} \to \mathcal{C}$. Here $\mathcal{J}$ is called indexing category of the diagram D.

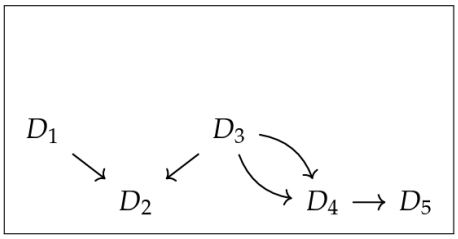
Def :

Definition 3.92. Let $D : \mathcal{J} \to \mathcal{C}$ be a diagram. A *cone* (C, c_*) over D consists of
(i) an object $C \in \mathcal{C}$;
(ii) for each object $j \in \mathcal{J}$, a morphism $c_j : C \to D(j)$.
To be a cone, these must satisfy the following property:
(a) for each $f : j \to k$ in $\mathcal{J}$, we have $c_k = c_j \fatsemi D(f)$.
A *morphism of cones* $(C, c_*) \to (C', c'_*)$ is a morphism $a : C \to C'$ in $\mathcal{C}$ such that for all $j \in \mathcal{J}$ we have $c_j = a \fatsemi c'_j$. Cones over D, and their morphisms, form a category **Cone**(D).
The *limit* of D, denoted $\lim(D)$, is the terminal object in the category **Cone**(D). Say it is the cone $\lim(D) = (C, c_*)$; we refer to C as the *limit object* and the map c_j for any $j \in \mathcal{J}$ as the j^{th} *projection map*.

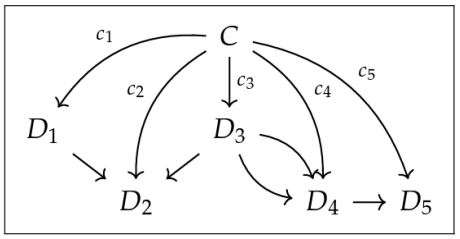
Eg : $\mathcal{J}$ is :



the diagram
Then, inside $\mathcal{C}$:



Cone over D:



Note :

Example 3.93. Terminal objects are limits where the indexing category is empty, $\mathcal{J} = \varnothing$.

Example 3.94. Products are limits where the indexing category consists of two objects v, w and no arrows, $\mathcal{J} =$ 

.

Alt : Finite Limit in set

Theorem 3.95. Let $\mathcal{J}$ be a category presented by the finite graph (V, A, s, t) together wit some equations, and let $D : \mathcal{J} \to \mathbf{Set}$ be a set-valued functor. Write $V = \{v_1, \ldots, v_n$ The set

$$\lim_{\mathcal{J}} D := \{(d_1, \ldots, d_n) \mid d_i \in D(v_i) \text{ for all } 1 \le i \le n \text{ and for all } a : v_i \to v_j \in A, \text{ we have } D(a)(d_i) = d_j\}.$$

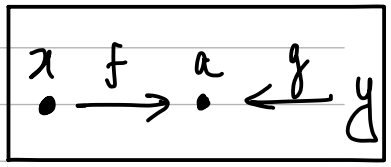
together with the projection maps $p_i : (\lim_{\mathcal{J}} D) \to D(v_i)$ given by $p_i(d_1, \ldots, d_n) := d$ is a limit of D.

for $\mathcal{J} =$  $\Rightarrow \lim_{\mathcal{J}} =$ {1} → singleton set

$\mathcal{J} =$ 

$\Rightarrow \lim_{\mathcal{J}} = \{(d_1, d_2),\ d_1 \in A \ \&\ d_2 \in B\}$

⇒ Pullbacks :

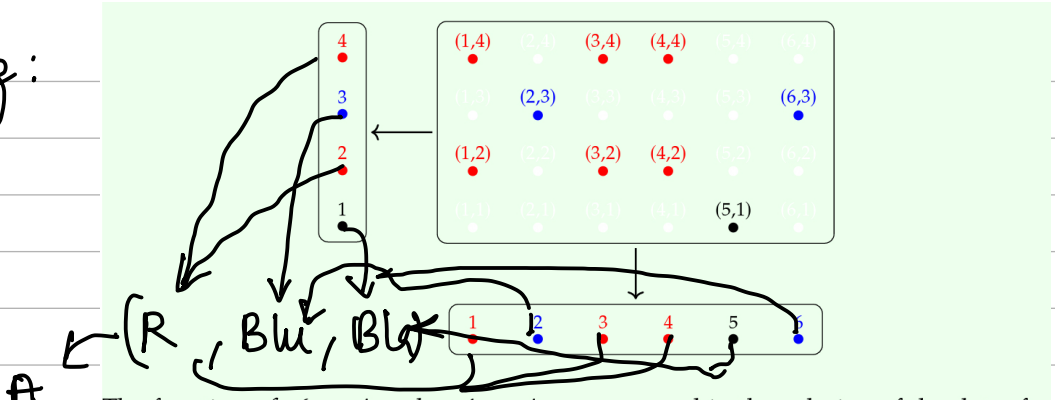
The limit of the cospan graph 

pullback ↑

$$\begin{array}{ccc} C & \xrightarrow{c_y} & Y \\ {\scriptstyle c_x}\downarrow & \searrow^{c_a} & \downarrow{\scriptstyle g} \\ X & \xrightarrow[f]{} & A \end{array} \qquad \begin{array}{ccc} X \times_A Y & \xrightarrow{c_y} & Y \\ {\scriptstyle c_x}\downarrow & \lrcorner & \downarrow{\scriptstyle g} \\ X & \xrightarrow[f]{} & A \end{array}$$

c_a is superfluous as the diagram must commute

Eg:



The functions $f : \underline{6} \to A$ and $g : \underline{4} \to A$ are expressed in the coloring of the dots: for example, $g(2) = g(4) =$ red, while $f(5) =$ black. The pullback selects pairs $(i, j) \in \underline{6} \times \underline{4}$ such that $f(i)$ and $g(j)$ have the same color.