Category Theory

A pointless, yet highly functional theory

Hype for Types

Jacob Neumann

05 November 2019

Table of Contents

Note about learning category theory

Note about learning category theory

- Category theory can be a bit unusual to grasp at first – it's not quite like other styles of mathematics. I'll try to mention a few of these differences along the way.

Note about learning category theory

- Category theory can be a bit unusual to grasp at first – it's not quite like other styles of mathematics. I'll try to mention a few of these differences along the way.
- There are a lot of details to be checked here (and a semester-long course in category theory would teach you how to do this). For today, try to focus on the bigger picture of what we're doing. I'll do my best to indicate which details matter and which ones don't.

Note about learning category theory

- Category theory can be a bit unusual to grasp at first – it's not quite like other styles of mathematics. I'll try to mention a few of these differences along the way.
- There are a lot of details to be checked here (and a semester-long course in category theory would teach you how to do this). For today, try to focus on the bigger picture of what we're doing. I'll do my best to indicate which details matter and which ones don't.
- Category theory often involves some very advanced mathematical theory – don't worry too much about understanding everything. But I hope you'll walk out of next lecture understanding why category theory is an awesome and beautiful topic!

Section 1

Motivation: A Theory of Functions

The Best HOF

The Best HOF

`map`

The Best HOF

`map`

`foldl`

The Best HOF

`map`

`foldr`

`foldl`

The Best HOF

`map`

`foldr`

`foldl`

`filter`

The Best HOF

`map` `mappartiali`

`foldr`

`foldl` `filter`

The Best HOF

`map` `mappartiali`

`tabulate` `foldr`

`foldl` `filter`

The Best HOF

`map` `mappartiali`

`tabulate` `foldr`

`foldl` `filter` `exists`

The Best HOF

~~`map`~~ ~~`mappartiali`~~

~~`tabulate`~~ ~~`foldr`~~

~~`foldl`~~ ~~`filter`~~ ~~`exists`~~

The Best HOF

~~map~~ ~~mappartiali~~

~~tabulate~~ op o ~~foldr~~

~~foldl~~ ~~filter~~ ~~exists~~

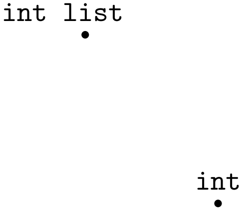
A Bird's-Eye View of SML

A Bird's-Eye View of SML

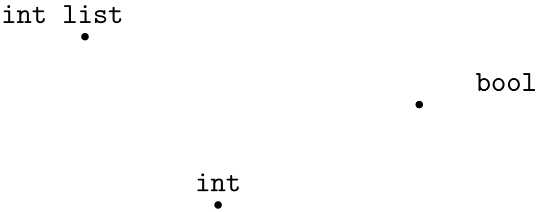
`int list`

•

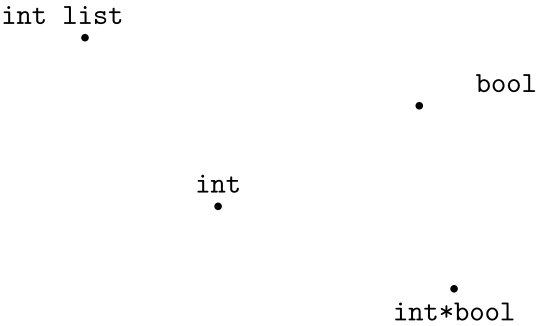
A Bird's-Eye View of SML



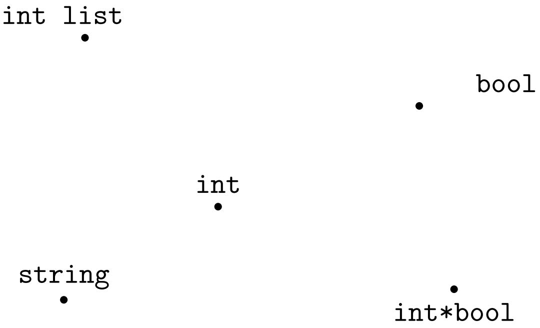
A Bird's-Eye View of SML



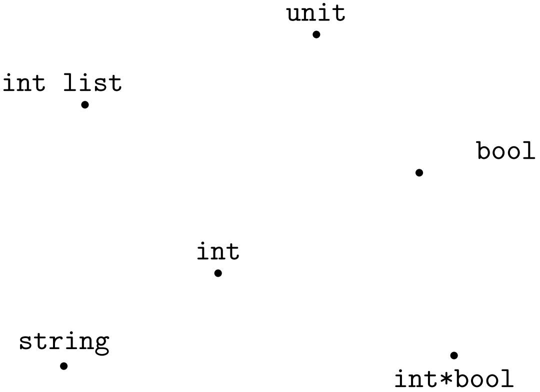
A Bird's-Eye View of SML



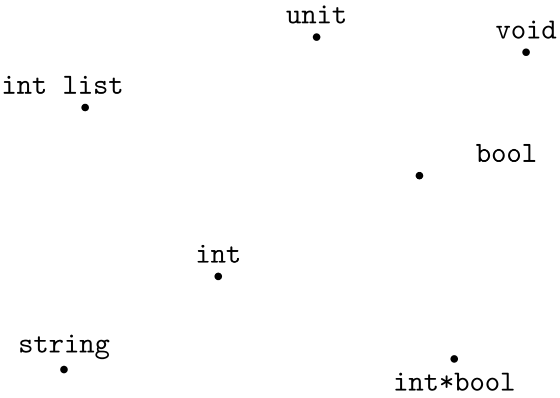
A Bird's-Eye View of SML



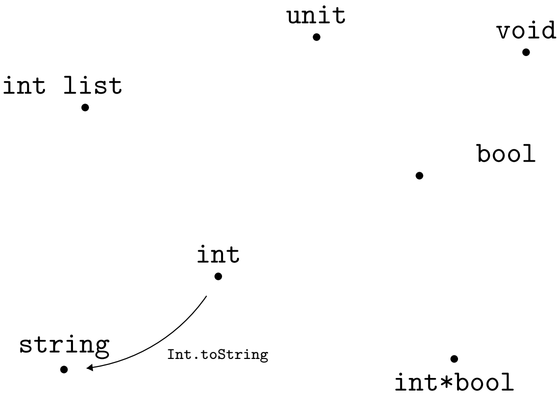
A Bird's-Eye View of SML



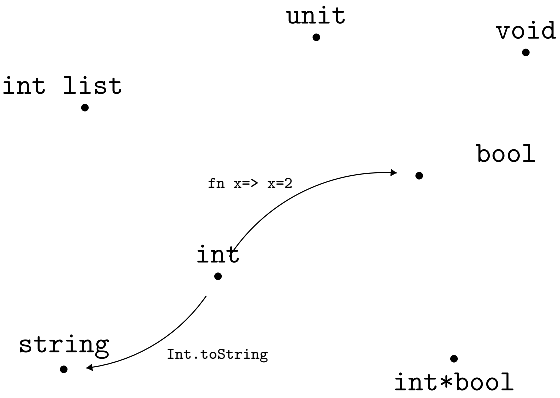
A Bird's-Eye View of SML



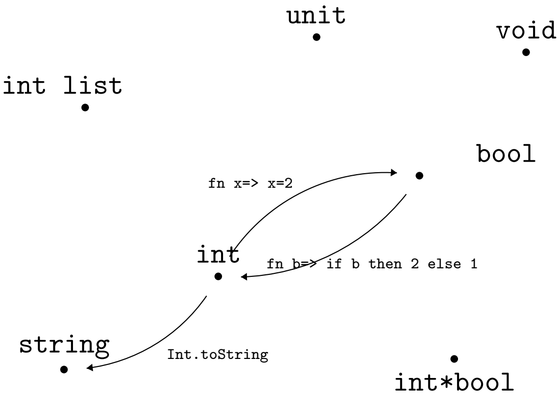
A Bird's-Eye View of SML



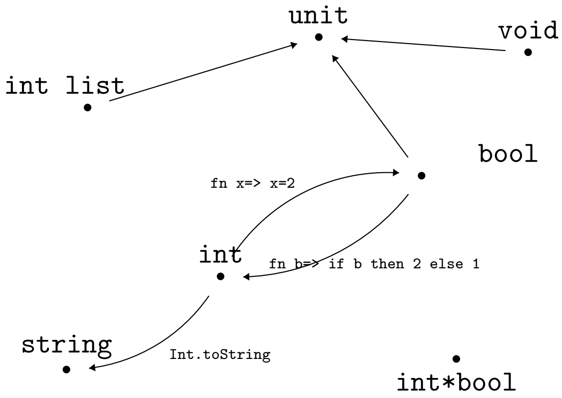
A Bird's-Eye View of SML



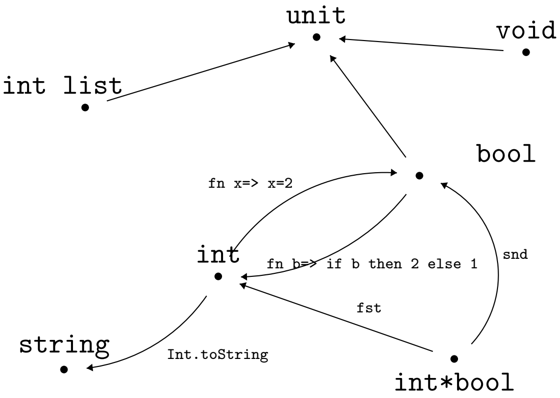
A Bird's-Eye View of SML



A Bird's-Eye View of SML



A Bird's-Eye View of SML



Theorem

There exists a total function of type $\mathtt{int} \to \mathtt{bool}$.

Theorem

There exists a total function of type $\texttt{int} \to \texttt{bool}$.

Theorem

For all types τ, τ', *there exist total functions* $\texttt{fst} : \tau * \tau' \to \tau$ *and* $\texttt{snd} : \tau * \tau' \to \tau'$

Theorem

There exists a total function of type $\mathtt{int} \to \mathtt{bool}$.

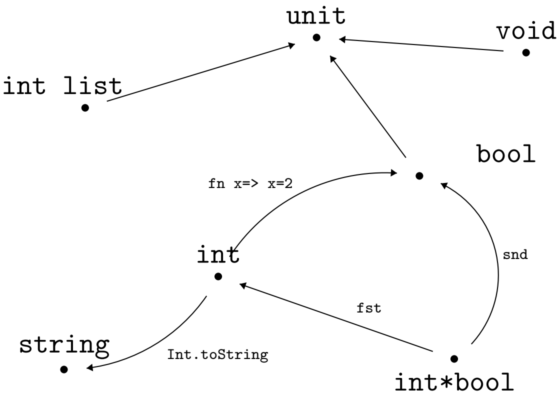
Theorem

For all types τ, τ', *there exist total functions* $\mathtt{fst} : \tau * \tau' \to \tau$ *and* $\mathtt{snd} : \tau * \tau' \to \tau'$

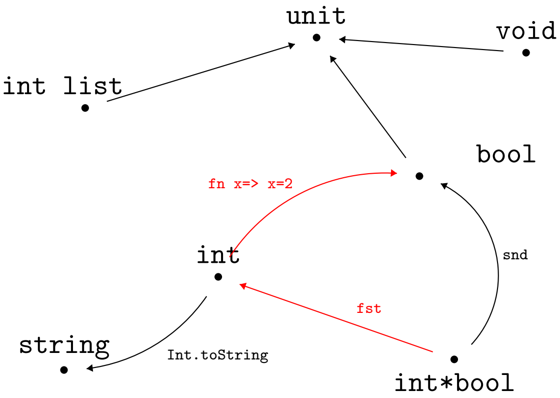
Theorem

For all types τ, *there exists a unique function* $\mathtt{u}_\tau : \tau \to \mathtt{unit}$

op o is a (partial) binary operation on functions

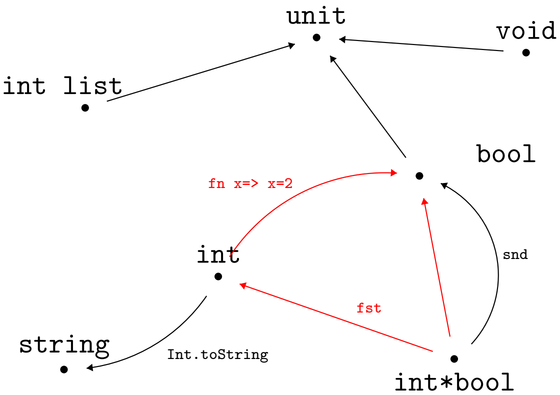


`(op o):('b -> 'c) * ('a -> 'b) -> ('a -> 'c)`


unit
void
int list
bool
fn x=> x=2
snd
int
fst
string
Int.toString
int*bool

(op o):('b -> 'c) * ('a -> 'b) -> ('a -> 'c)

op o is a (partial) binary operation on functions



`(op o):('b -> 'c) * ('a -> 'b) -> ('a -> 'c)`

op o Theory

op o Theory

Notice:

$$\texttt{(fn x=> x=2) o (fn b=> if b then 2 else 1)} = \texttt{id}_{\texttt{bool}}$$

This is an equation of functions, and it tells us information about the types `bool` and `int`.

op o Theory

Notice:

`(fn x=> x=2) o (fn b=> if b then 2 else 1)` $= \texttt{id}_{\texttt{bool}}$

This is an equation of functions, and it tells us information about the types `bool` and `int`.
`op o` is important enough that it has it's own theory.

op o Theory

Notice:

$$\texttt{(fn x=> x=2) o (fn b=> if b then 2 else 1)} = \texttt{id}_{\texttt{bool}}$$

This is an equation of functions, and it tells us information about the types `bool` and `int`.
`op o` is important enough that it has it's own theory.
The theory of `op o` is called **category theory**.

Section 2

Categories

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$
- A collection of morphisms f, g, h, each with a specified domain and codomain (e.g. $f : X \to Z$)

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$
- A collection of morphisms f, g, h, each with a specified domain and codomain (e.g. $f : X \to Z$)

such that:

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$
- A collection of morphisms f, g, h, each with a specified domain and codomain (e.g. $f : X \to Z$)

such that:

1. For each object X, there exists a morphism $\mathrm{id}_X : X \to X$ called the *identity morphism* (on X)

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$
- A collection of morphisms f, g, h, each with a specified domain and codomain (e.g. $f : X \to Z$)

such that:

1. For each object X, there exists a morphism $\mathrm{id}_X : X \to X$ called the *identity morphism* (on X)
2. For each pair of morphisms $f : X \to Y$ and $g : Y \to Z$, there exists a morphism $(g \circ f) : X \to Z$, called the *composition* of g *after* f

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$
- A collection of morphisms f, g, h, each with a specified domain and codomain (e.g. $f : X \to Z$)

such that:

1. For each object X, there exists a morphism $\mathrm{id}_X : X \to X$ called the *identity morphism* (on X)
2. For each pair of morphisms $f : X \to Y$ and $g : Y \to Z$, there exists a morphism $(g \circ f) : X \to Z$, called the *composition* of g *after* f
3. Identity morphisms are units under composition: for all $f : X \to Y$,

$$\mathrm{id}_Y \circ f = f = f \circ \mathrm{id}_X$$

Definition

A **category** $\mathbb{C}$ consists of:

- A collection of objects $X, Y, Z, \ldots$
- A collection of morphisms f, g, h, each with a specified domain and codomain (e.g. $f : X \to Z$)

such that:

1. For each object X, there exists a morphism $\mathrm{id}_X : X \to X$ called the *identity morphism* (on X)
2. For each pair of morphisms $f : X \to Y$ and $g : Y \to Z$, there exists a morphism $(g \circ f) : X \to Z$, called the *composition* of g *after* f
3. Identity morphisms are units under composition: for all $f : X \to Y$,

$$\mathrm{id}_Y \circ f = f = f \circ \mathrm{id}_X$$

4. Composition is associative: for all $f : A \to B$, $g : B \to C$, $h : C \to D$,

$$(h \circ g) \circ f = h \circ (g \circ f)$$

Example 0: Types!

The type system of SML defines a category:

Example 0: Types!

The type system of SML defines a category:

- The objects are types

Example 0: Types!

The type system of SML defines a category:

- The objects are types
- The morphisms are total functions

Example 0: Types!

The type system of SML defines a category:

- The objects are types
- The morphisms are total functions

1. For any type τ, the identity morphism on τ is given by:

$$\texttt{val id}_\tau : \tau \to \tau \texttt{ = fn x:}\tau \texttt{ => x}$$

Example 0: Types!

The type system of SML defines a category:

- The objects are types
- The morphisms are total functions

1. For any type τ, the identity morphism on τ is given by:

$$\texttt{val id}_\tau : \tau \to \tau \texttt{ = fn x:}\tau \texttt{ => x}$$

2. For `f:X->Y`, `g:Y->Z`, `(g o f):X-> Z`

Example 0: Types!

The type system of SML defines a category:

- The objects are types
- The morphisms are total functions

1. For any type τ, the identity morphism on τ is given by:

$$\texttt{val id}_\tau : \tau \to \tau \ \texttt{= fn x:}\tau\ \texttt{=> x}$$

2. For `f:X->Y`, `g:Y->Z`, `(g o f):X-> Z`
3. $$\texttt{id}_\texttt{Y}\ \texttt{o f} = \texttt{f} = \texttt{f o id}_\texttt{X}$$

Example 0: Types!

The type system of SML defines a category:

- The objects are types
- The morphisms are total functions

1. For any type τ, the identity morphism on τ is given by:

$$\texttt{val id}_\tau : \tau \to \tau \texttt{ = fn x:}\tau \texttt{ => x}$$

2. For `f:X->Y`, `g:Y->Z`, `(g o f):X-> Z`
3. $$\texttt{id}_\texttt{Y} \texttt{ o f} = \texttt{f} = \texttt{f o id}_\texttt{X}$$
4. `op o` is associative

Example 1: Set

The collection of *all sets* is a category:

Example 1: Set

The collection of *all sets* is a category:

- The objects are sets

Example 1: Set

The collection of *all sets* is a category:

- The objects are sets
- The morphisms are total functions

Example 1: Set

The collection of *all sets* is a category:

- The objects are sets
- The morphisms are total functions

1. For any set X, the function $\mathrm{id}_X : X \to X$ given by $\mathrm{id}_X(x) = x$ is the identity function

Example 1: Set

The collection of *all sets* is a category:

- The objects are sets
- The morphisms are total functions

1. For any set X, the function $\mathrm{id}_X : X \to X$ given by $\mathrm{id}_X(x) = x$ is the identity function
2. Composition is just the usual composition of functions.

Example 1: Set

The collection of *all sets* is a category:

- The objects are sets
- The morphisms are total functions

1. For any set X, the function $\mathrm{id}_X : X \to X$ given by $\mathrm{id}_X(x) = x$ is the identity function
2. Composition is just the usual composition of functions.
3. Identity is a unit for composition

Example 1: Set

The collection of *all sets* is a category:

- The objects are sets
- The morphisms are total functions

1. For any set X, the function $\mathrm{id}_X : X \to X$ given by $\mathrm{id}_X(x) = x$ is the identity function
2. Composition is just the usual composition of functions.
3. Identity is a unit for composition
4. Function composition is associative

But why?

But why?

- As we'll see, abstractly defining the notion of a "category" allows us to specify a particular kind of structure: compositional structure.

But why?

- As we'll see, abstractly defining the notion of a "category" allows us to specify a particular kind of structure: compositional structure.
- Also, by making an abstract definition, we open up the possibility of massively reducing redundancy in theorem proving: if I can prove something holds for an arbitrary category, it automatically holds for both sets and for SML types – I don't have to prove it twice!

But why?

- As we'll see, abstractly defining the notion of a "category" allows us to specify a particular kind of structure: compositional structure.
- Also, by making an abstract definition, we open up the possibility of massively reducing redundancy in theorem proving: if I can prove something holds for an arbitrary category, it automatically holds for both sets and for SML types – I don't have to prove it twice!
- Also, notice that the definition of category never mentioned the objects having "elements". This is intentional. This forces us to adopt a "pointfree" mindset, and define everything in terms of functions.

Section 3

Isomorphisms

Example: Number of elements

When do we say that two sets have the "same number of elements"? Well, to account for the possibility of infinite sets, we make some definition like the following:

Example: Number of elements

When do we say that two sets have the "same number of elements"? Well, to account for the possibility of infinite sets, we make some definition like the following:

Definition

Two sets X, Y are said to be **equinumerous** if there exists a funtion $f : X \to Y$ such that

- Injectivity: For all $x, x' \in X$, if $f(x) = f(x')$, then $x = x'$
- Surjectivity: For all $y \in Y$, there exists some $x \in X$ such that $f(x) = y$.

Example: Number of elements

When do we say that two sets have the "same number of elements"? Well, to account for the possibility of infinite sets, we make some definition like the following:

Definition

Two sets X, Y are said to be **equinumerous** if there exists a funtion $f : X \to Y$ such that

- Injectivity: For all $x, x' \in X$, if $f(x) = f(x')$, then $x = x'$
- Surjectivity: For all $y \in Y$, there exists some $x \in X$ such that $f(x) = y$.

This definition sucks!!!! It constantly refers to the elements of X and Y!

Example: Number of elements

When do we say that two sets have the "same number of elements"? Well, to account for the possibility of infinite sets, we make some definition like the following:

Definition

Two sets X, Y are said to be **equinumerous** if there exists a funtion $f : X \to Y$ such that

- Injectivity: For all $x, x' \in X$, if $f(x) = f(x')$, then $x = x'$
- Surjectivity: For all $y \in Y$, there exists some $x \in X$ such that $f(x) = y$.

This definition sucks!!!! It constantly refers to the elements of X and Y! Can we define equinumerosity *pointfree*?

Pointfree Equinumerosity

Yes we can!

Pointfree Equinumerosity

Yes we can!

Definition

Two sets X, Y are said to be **equinumerous** if there exist functions $f : X \to Y$ and $g : Y \to X$ such that

Pointfree Equinumerosity

Yes we can!

Definition

Two sets X, Y are said to be **equinumerous** if there exist functions $f : X \to Y$ and $g : Y \to X$ such that

$$g \circ f = \mathrm{id}_X \qquad \text{and} \qquad f \circ g = \mathrm{id}_Y$$

In this case, f and g are called *bijections*.

Generalization: Isos

Generalization: Isos

Definition

Two objects X, Y of a category $\mathbb{C}$ are said to be **isomorphic** if there exist morphisms $f : X \to Y$ and $g : Y \to X$ such that

$$g \circ f = \mathrm{id}_X \qquad \text{and} \qquad f \circ g = \mathrm{id}_Y$$

In this case, f and g are called *isos*.

Generalization: Isos

Definition

Two objects X, Y of a category $\mathbb{C}$ are said to be **isomorphic** if there exist morphisms $f : X \to Y$ and $g : Y \to X$ such that

$$g \circ f = \mathrm{id}_X \qquad \text{and} \qquad f \circ g = \mathrm{id}_Y$$

In this case, f and g are called *isos*.

Notice this is exactly the same definition as before!

Generalization: Isos

Definition

Two objects X, Y of a category $\mathbb{C}$ are said to be **isomorphic** if there exist morphisms $f : X \to Y$ and $g : Y \to X$ such that

$$g \circ f = \mathrm{id}_X \qquad \text{and} \qquad f \circ g = \mathrm{id}_Y$$

In this case, f and g are called *isos*.

Notice this is exactly the same definition as before! This fact is summarized by saying:

Bijections are isos in the category of sets & functions

Type Isos

The key strength of category theory is that we can take general notions (like isos), and then study them in the categories that interest us!

Type Isos

The key strength of category theory is that we can take general notions (like isos), and then study them in the categories that interest us!
So what's an iso in the category of types?

Type Isos

The key strength of category theory is that we can take general notions (like isos), and then study them in the categories that interest us!
So what's an iso in the category of types?

Theorem

The types `unit+unit` and `bool` are isomorphic

Type Isos

The key strength of category theory is that we can take general notions (like isos), and then study them in the categories that interest us!
So what's an iso in the category of types?

Theorem

The types `unit+unit` and `bool` are isomorphic

Proof.

```
fun f(inl ())= true
  | f(inr ())= false
fun g(b)     = if b then inl() else inr()
```

□

Section 4

Diagrams

Commutative Diagrams

We can depict isos more graphically, using a *commutative diagram*:

Commutative Diagrams

We can depict isos more graphically, using a *commutative diagram*:

$$\begin{tikzcd} X \arrow[loop left, "\mathrm{id}_X"] \arrow[r, bend left, "f"] & Y \arrow[l, bend left, "g"] \arrow[loop right, "\mathrm{id}_Y"] \end{tikzcd}$$

Commutative Diagrams

We can depict isos more graphically, using a *commutative diagram*:

$$\begin{tikzcd} X \arrow[loop left, "\mathrm{id}_X"] \arrow[r, bend left, "f"] & Y \arrow[l, bend left, "g"] \arrow[loop right, "\mathrm{id}_Y"] \end{tikzcd}$$

"Commutative Diagrams" get their names from the fact that they're assumed to *commute*: any two paths through the diagram which start at the same place and end at the same place (e.g. $f \circ g$ and id_Y both start and end at Y) are assumed to be equal.

Commutative Diagrams

We can depict isos more graphically, using a *commutative diagram*:

$$\begin{tikzcd} X \arrow[loop left, "\mathrm{id}_X"] \arrow[rr, bend left, "f"] & & Y \arrow[loop right, "\mathrm{id}_Y"] \arrow[ll, bend left, "g"] \end{tikzcd}$$

"Commutative Diagrams" get their names from the fact that they're assumed to *commute*: any two paths through the diagram which start at the same place and end at the same place (e.g. $f \circ g$ and id_Y both start and end at Y) are assumed to be equal.
Note that we usually don't draw the identity arrows in commutative diagrams – they're left implicit.

Example: Commutative Triangles

Consider the following diagram (in some category $\mathbb{C}$):

Example: Commutative Triangles

Consider the following diagram (in some category $\mathbb{C}$):

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "h"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

Example: Commutative Triangles

Consider the following diagram (in some category $\mathbb{C}$):

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "h"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

What does it mean to say that this diagram commutes?

Example: Commutative Triangles

Consider the following diagram (in some category $\mathbb{C}$):

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, "h"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

What does it mean to say that this diagram commutes?

$$g \circ f = h$$

Example: Commutative Squares

Consider the following diagram (in some category $\mathbb{C}$):

Example: Commutative Squares

Consider the following diagram (in some category $\mathbb{C}$):

$$\begin{tikzcd} W \arrow[r, "f"] \arrow[d, "h"'] & X \arrow[d, "g"] \\ Y \arrow[r, "k"'] & Z \end{tikzcd}$$

Example: Commutative Squares

Consider the following diagram (in some category $\mathbb{C}$):

$$\begin{tikzcd} W \arrow[r, "f"] \arrow[d, "h"'] & X \arrow[d, "g"] \\ Y \arrow[r, "k"'] & Z \end{tikzcd}$$

What does it mean to say that this diagram commutes?

Example: Commutative Squares

Consider the following diagram (in some category $\mathbb{C}$):

$$\begin{tikzcd} W \arrow[r, "f"] \arrow[d, "h"'] & X \arrow[d, "g"] \\ Y \arrow[r, "k"'] & Z \end{tikzcd}$$

What does it mean to say that this diagram commutes?

$$g \circ f = k \circ h$$

Section 5

Universal Properties

Universal Properties

To talk about the interesting properties of various categories, we make use of *universal properties*.

Universal Properties

To talk about the interesting properties of various categories, we make use of *universal properties*.

Assuming we have some commutative diagram:

$$\begin{tikzcd}
A \arrow[drr, bend left, "q_2"] \arrow[ddr, bend right, "q_1"'] & & \\
& W \arrow[r, "f"] \arrow[d, "h"'] & X \arrow[d, "g"] \\
& Y \arrow[r, "k"'] & Z
\end{tikzcd}$$

To talk about the interesting properties of various categories, we make use of *universal properties*.

Assuming we have some commutative diagram:

$$\begin{tikzcd}
A \arrow[dr, dashed] \arrow[drr, bend left, "q_2"] \arrow[ddr, bend right, "q_1"'] & & \\
& W \arrow[r, "f"] \arrow[d, "h"'] & X \arrow[d, "g"] \\
& Y \arrow[r, "k"'] & Z
\end{tikzcd}$$

This implies other object/arrows (indicated by the dashed lines) must exist, and commute with the rest of the diagram as depicted.

Example: Composition

Composition is the simplest example of a universal property:

$$\begin{tikzcd} X \arrow[r, "f"] \arrow[dr, dashed, "g \circ f"'] & Y \arrow[d, "g"] \\ & Z \end{tikzcd}$$

Example: Composition

Composition is the simplest example of a universal property:

$$
\begin{tikzcd}
X \arrow[r, "f"] \arrow[dr, dashed, "g \circ f"'] & Y \arrow[d, "g"] \\
& Z
\end{tikzcd}
$$

Read aloud: "For all objects X, Y, Z of a category $\mathbb{C}$, and all morphisms $f : X \to Y$ and $g : Y \to Z$, there **exists a** morphism $(g \circ f) : X \to Z$

Example: Factorization

What does this diagram say?

Example: Factorization

What does this diagram say?

$$\begin{tikzcd}
 & U \arrow[dr, dashed] & \\
X \arrow[ur, dashed] \arrow[rr, "f"'] & & Y
\end{tikzcd}$$

Example: Factorization

What does this diagram say?

$$\begin{tikzcd} & U \arrow[dr, dashed] & \\ X \arrow[ur, dashed] \arrow[rr, "f"'] & & Y \end{tikzcd}$$

"f can be written as $g_2 \circ g_1$ for some $g_1 : X \to U$ and some $g_2 : U \to Y$"

Example: Factorization

What does this diagram say?

$$
\begin{tikzcd}
 & U \arrow[dr, dashed] & \\
X \arrow[ur, dashed] \arrow[rr, "f"'] & & Y
\end{tikzcd}
$$

"f can be written as $g_2 \circ g_1$ for some $g_1 : X \to U$ and some $g_2 : U \to Y$"
"f *factors through* U"

Example of Factorization

Theorem

A total SML function $f : \tau_1 \to \tau_2$ is constant (in the sense that $f(x) = f(y)$ for all $x, y : \tau_1$) iff f factors through `unit`

Example of Factorization

Theorem

A total SML function $f : \tau_1 \to \tau_2$ is constant (in the sense that $f(x) = f(y)$ for all $x, y : \tau_1$) iff f factors through `unit`

$$\begin{tikzcd} & \texttt{unit} \arrow[dr, dashed] & \\ \tau_1 \arrow[ur, dashed] \arrow[rr, "f"'] & & \tau_2 \end{tikzcd}$$

Sanity Break