INTRODUCTION TO CATEGORIES

FUNCTORS AND FUNCTOR CATEGORIES

NATURAL TRANSFORMATIONS

**INTRODUCTION TO CATEGORY THEORY FOR FUNCTIONAL PROGRAMMING**

Andrew R. Plummer

Department of Linguistics
The Ohio State University

14 Oct, 2009
OUTLINE

1. INTRODUCTION TO CATEGORIES

2. FUNCTORS AND FUNCTOR CATEGORIES

3. NATURAL TRANSFORMATIONS
**Categories**

**Category Axioms (1)**

A category $\mathbf{C}$ comprises the following:

- A collection of *objects*, denoted by $\text{Ob}(\mathbf{C})$;
- A collection of *arrows*, denoted by $\text{Ar}(\mathbf{C})$;

Neither of these collections need be sets.
OPERATORS: dom and cod

- Operations dom and cod from the collection of arrows to the collection of objects:
  \[ \text{dom} : \text{Ar}(\mathcal{C}) \rightarrow \text{Ob}(\mathcal{C}) \quad \text{and} \quad \text{cod} : \text{Ar}(\mathcal{C}) \rightarrow \text{Ob}(\mathcal{C}). \]

- For any arrow \( f \in \text{Ar}(\mathcal{C}) \), dom \( f \) is called the \textit{domain} of \( f \), and cod \( f \) is called the \textit{codomain} of \( f \). If dom \( f = a \) and cod \( f = b \), we represent this as
  \[ f : a \rightarrow b \quad \text{or} \quad a \xrightarrow{f} b; \]
An operation \( \circ \) that assigns to each pair \( \langle g, f \rangle \) of arrows \( f, g \in Ar(C) \) that satisfies \( \text{cod } f = \text{dom } g \), an arrow \( g \circ f \), called the *composite* of \( f \) and \( g \), such that

\[
\text{dom } g \circ f = \text{dom } f, \quad \text{and} \quad \text{cod } g \circ f = \text{cod } g;
\]

That is,

\[
(g \circ f) : a \to c
\]
CATEGORIES

CATEGORY AXIOMS (4)

For all objects $b \in \text{Ob}(\mathbf{C})$, there is an arrow $\text{id}_b : b \to b$ in $\text{Ar}(\mathbf{C})$, called the identity arrow on $b$, such that the following identity holds for all arrows $f : a \to b$ and $g : b \to c$ in $\text{Ar}(\mathbf{C})$:

$$\text{id}_b \circ f = f \quad \text{and} \quad g \circ \text{id}_b = g;$$

That is,

$$(f \circ \text{id}_a) : a \to b \quad (\text{id}_b \circ g) : b \to c$$

$$(f \circ \text{id}_a) = f \quad (\text{id}_b \circ g) = g$$
Categories

Category Axioms (5)

For all arrows $f : a \to b$, $g : b \to c$, and $h : c \to d$ in $Ar(C)$ the following identity holds:

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

A monoid is a triple $\mathcal{M} = \langle M, \ast, e \rangle$ such that

- $M$ is a set,
- $\ast$ is an associative binary operation on $M$,
- $e$ is the identity element for $\ast$ on $M$.

Monoids are commonly used throughout computer science:

- Transition monoids and syntactic monoids in describing a finite state machine,
- Trace monoids and history monoids for process calculi and concurrent computing.
The Haskell is a purely functional programming language based on the typed $\lambda$-calculus. All of our programming examples are based on Haskell syntax.

**Lists**

Let $a$ be a type (e.g., Int). Then $L_a = \langle [a], (++) , [] \rangle$ is a monoid.

- $([], [1,2,3]) = ([1,2,3], [], []) = [1,2,3]$
- $([1,2], [3,4]) = ([3,4], [5,6]) = [1,2,3,4,5,6]$
Monoids as Categories

Monoid Category

Given a monoid $\mathcal{M} = \langle M, \ast, e \rangle$, let $\mathbf{C}_\mathcal{M}$ be the following category:

- $Ob(\mathbf{C}_\mathcal{M}) = \{0\}$ (any set with one element),
- $Ar(\mathbf{C}_\mathcal{M}) = M$,
- $g \circ f = g \ast f$.
- $id_0 = e$, for $0 \in Ob(\mathbf{C}_\mathcal{M})$.

Since $\ast$ is associative in $\mathcal{M}$, $\circ$ is associative on $Ar(\mathbf{C}_\mathcal{M})$. Since $e$ is the identity in $\mathcal{M}$, $e$ is the identity arrow for $0 \in Ob(\mathbf{C}_\mathcal{M})$. 
PREORDERS AS CATEGORIES

PREORDERS

A preorder \( \mathcal{P} \) is a set \( P \) equipped with a relation \( \sqsubseteq \) which is reflexive and transitive.

PREORDER CATEGORY

Given a preorder \( \mathcal{P} = \langle P, \sqsubseteq \rangle \), let \( \mathbf{C}_\mathcal{P} \) be the following category:

- \( \text{Ob}(\mathbf{C}_\mathcal{P}) = P \),
- \( \text{Ar}(\mathbf{C}_\mathcal{P}) = \{ \langle a, b \rangle : a \rightarrow b \mid a \sqsubseteq b \} \),
- \( \langle b, c \rangle \circ \langle a, b \rangle = \langle a, c \rangle \).
- \( \text{id}_a = \langle a, a \rangle \), for \( a \in \text{Ob}(\mathbf{C}_\mathcal{P}) \).
OUTLINE

1. INTRODUCTION TO CATEGORIES

2. FUNCTORS AND FUNCTOR CATEGORIES

3. NATURAL TRANSFORMATIONS
Purely functional programming is founded entirely on category theory. Functors are used to handle side effects, and real world interaction.

**Functor (1)**

A functor $F$ from a category $\mathbf{C}$ to a category $\mathbf{D}$ is a function such that:

Each object $a \in Ob(\mathbf{C})$ is mapped to an object $F(a) \in Ob(\mathbf{D})$. 


FUnctor (2)

Each arrow $f : a \rightarrow b$ in $Ar(C)$ is mapped to an arrow $F(f) : F(a) \rightarrow F(b)$ in $Ar(D)$ such that:

- For all objects $a \in Ob(C)$, the identity arrow on $a$ is mapped to the identity arrow on $F(a)$. That is, $F(id_a) = id_{F(a)}$.

- For all pairs $\langle g, f \rangle$ where $f, g \in Ar(C)$ and $\text{cod } f = \text{dom } g$, $F(g \circ f) = F(g) \circ F(f)$.
The main action of a functor is visualized with the following commutative diagrams:

If $F$ is a functor from $C$ to $D$, we write $F : C \to D$ or $C \xrightarrow{F} D$. 
Recall that $\mathcal{L}_{\text{Int}} = \langle \text{[Int]}, (++), [] \rangle$ is a monoid, and thus a category.

Similarly, $\mathcal{L}_{\text{Char}} = \langle \text{[Char]}, (++), [] \rangle$ is a category.

**The ASCII Number Functor**

Let $f :: \text{[Char]} \rightarrow \text{[Int]}$ be defined as follows:

- $f ([]) = []$
- $f (\text{char}:\text{chars}) = \text{[ascii(char)]} ++ f(\text{[chars]})$

$f$ takes a list of chars, and returns a list of ints. The $i$th element of $f(\text{[chars]})$ is the ascii number of the $i$th char in $\text{[chars]}$. 
Haskell Example

We need to verify that

\[ f(\text{[chars2]}++\text{[chars1]}) = f(\text{[chars2]})++f(\text{[chars1]}) \]
Haskell Example

Instance “blackboard”:

- \( f(\text{"black"}) = [98,108,97,99,107] \)
- \( f(\text{"board"}) = [98,111,97,114,100] \)
- \( f(\text{"black"} +\text{"board"}) = [98,108,97,99,107,98,111,97,114,100] = f(\text{"blackboard"}) \)
We can generalize the previous example.

**Monoid Homomorphisms**

Let $\mathcal{M} = \langle M, *, e_M \rangle$ and $\mathcal{N} = \langle N, *, e_N \rangle$ be monoids. A *monoid homomorphism* is a function $\phi : \mathcal{M} \to \mathcal{N}$ such that

- $\phi(e_M) = e_N$;
- $\phi(a *_M b) = \phi(a) *_N \phi(b)$.

The monoid homomorphisms are exactly the functors between monoids (when monoids are viewed as categories).
We can treat preorders in similar fashion.

**Monotonic Functions**

Let $\mathcal{P} = \langle P, \sqsubseteq_P \rangle$ and $\mathcal{Q} = \langle Q, \sqsubseteq_Q \rangle$ be (pre)orders. A monotonic function is a function $f : \mathcal{P} \to \mathcal{Q}$ such that

$$f(a \sqsubseteq_P b) = f(a) \sqsubseteq_Q f(b).$$

The monotonic functions are exactly the functors between preorders (when preorders are viewed as categories).
**Identity Functor**

The *identity functor* on a category $\mathbf{C}$ is the functor $1_{\mathbf{C}} : \mathbf{C} \to \mathbf{C}$ such that

- for all $a \in \text{Ob}(\mathbf{C})$, $1_{\mathbf{C}}(a) = a$.
- for all $f \in \text{Ar}(\mathbf{C})$, $1_{\mathbf{C}}(f) = f$. 
Since functors are functions we immediately have the following:

**Functor Composition**

Given functors $F : A \rightarrow B$ and $G : B \rightarrow C$, the composite of $F$ and $G$, denoted $G \circ F$, is the functor defined as

- $(G \circ F)(a) = G(F(a))$, for all $a \in \text{Ob}(A)$;
- $(G \circ F)(f) = G(F(f))$, for all $f \in \text{Ar}(A)$.

For all functors $F : A \rightarrow B$, $G : B \rightarrow C$, and $H : C \rightarrow D$, we have $H \circ (G \circ F) = (H \circ G) \circ F$. 
HASKELL EXAMPLE

\[ \mathcal{L}_{\text{Bool}} = \langle [\text{Bool}], (++) , [\ ] \rangle \text{ is a category.} \]

THE is98 FUNCTOR

Let \( g :: [\text{Int}] \to [\text{Bool}] \) be defined as follows:

- \( g ([\ ]) = [\ ] \)
- \( g (\text{int}:\text{ints}) = [\text{is98(int)}] \text{ ++ } g([\text{ints}]) \)

\( g \) takes a list of ints, and returns a list of bools. The \( i \)th element of \( g([\text{ints}]) \) is True if and only if the \( i \)th int in \([\text{ints}]\) is 98.
Haskell Example

In Haskell syntax, the composition operator \( \circ \) is denoted by \( \cdot \).

**Composition**

Since \( g \) and \( f \) are functors, then so is

\[
h = g \cdot f :: [\text{Char}] \rightarrow [\text{Bool}]
\]

\( h \) takes a list of chars, and returns a list of bools. The \( i \)th element of \( h([\text{chars}]) \) is True if and only if the \( i \)th char in \([\text{chars}]\) is 'b'.

\[
h(\text{"tabby"}) = g \cdot f (\text{"tabby"}) = [\text{False}, \text{False}, \text{True}, \text{True}, \text{True}, \text{False}]
\]
We can again generalize the previous example.

Since functor composition is associative, and identity functors satisfy the identity law for arrows, we can treat categories as objects and functors as arrows, yielding categories of categories.
The category of monoids $\text{Mon}$ is defined as follows:

- $\text{Ob}(\text{Mon}) = \{ \mathcal{M} | \mathcal{M} \text{ is a monoid} \}$,
- $\text{Ar}(\text{Mon}) = \{ \phi | \phi \text{ is a monoid homomorphism} \}$,
- $\phi \circ \psi$ is function composition,
- $\text{id}_\mathcal{M}$ is the identity function on $\mathcal{M}$, for $\mathcal{M} \in \text{Ob}(\text{Mon})$. 
Category of Preorders

The category of preorders $\textbf{PreOrd}$ is defined as follows:

- $\text{Ob}(\textbf{PreOrd}) = \{P \mid P \text{ is a preorder}\}$,
- $\text{Ar}(\textbf{PreOrd}) = \{f \mid f \text{ is a monotonic function on preorders}\}$,
- $f \circ g$ is function composition,
- $\text{id}_P$ is the identity function on $P$, for $P \in \text{Ob}(\textbf{PreOrd})$. 
**Functor Categories**

**Looking Ahead**
- We can define functors between categories of categories, yielding an even higher level of abstraction.
- At this level, the functors themselves are objects. We need to define the arrows.
INTRODUCTION TO CATEGORIES

FUNCTORS AND FUNCTOR CATEGORIES

NATURAL TRANSFORMATIONS
Natural Transformations

The arrows of functor categories are called *natural transformations*.

Natural transformations are used in defining monads (next week's topic). We simply provide definitions and examples. The programming will be discussed next week.
Natural Transformations

Lead in
Let $\textbf{C}$ and $\textbf{D}$ be categories, and let $F : \textbf{C} \to \textbf{D}$ and $G : \textbf{C} \to \textbf{D}$ be functors from $\textbf{C}$ to $\textbf{D}$.

- The images of $F$ and $G$ are representations of the category $\textbf{C}$ within the category $\textbf{D}$.
- We want to translate the representation $F$ yields onto the representation that $G$ yields.
- We want to do so in a way that preserves the structure of the representation that $F$ yields.
Consider \( a, b \in \text{Ob}(C) \) and \( f \in \text{Ar}(C) \) where

\[
\begin{array}{ccc}
\text{a} & \xrightarrow{F(a)} & \text{F(a)} \\
\downarrow{f} & & \downarrow{F(f)} \\
\text{b} & \xrightarrow{F(b)} & \text{G(b)} \\
\end{array}
\]

\[
\begin{array}{ccc}
\text{a} & \xrightarrow{G(a)} & \text{G(a)} \\
\downarrow{f} & & \downarrow{G(f)} \\
\text{b} & \xrightarrow{F(b)} & \text{G(b)} \\
\end{array}
\]
We translate $F(a)$ onto $G(a)$ and $F(b)$ onto $G(b)$ using arrows in $\mathbf{D}$. Call these arrows $\tau_a : F(a) \rightarrow G(a)$ and $\tau_b : F(b) \rightarrow G(b)$.

\[ a \quad \xymatrix{ F(a) \ar[r]^{\tau_a} & G(a) & F(b) \ar[r]_{\tau_b} & G(b) } \quad b \]
Let \( \mathbf{C} \) and \( \mathbf{D} \) be categories, and let \( F : \mathbf{C} \to \mathbf{D} \) and \( G : \mathbf{C} \to \mathbf{D} \) be functors from \( \mathbf{C} \) to \( \mathbf{D} \).

**Natural Transformations**

A *natural transformation* from \( F \) to \( G \) is a collection of arrows \( \tau \), contained in \( \text{Ar}(\mathbf{D}) \), such that:

- for each object \( a \in \text{Ob}(\mathbf{C}) \), there is an arrow \( \tau_a : F(a) \to G(a) \) in \( \tau \), called a *component* of \( \tau \), such that
- for any arrow \( f : a \to b \) in \( \text{Ar}(\mathbf{C}) \), given \( \tau_b : F(b) \to G(b) \in \tau \), we have
  \[
  G(f) \circ \tau_a = \tau_b \circ F(f).
  \]
That is, the following diagram commutes:

\[
\begin{array}{ccc}
F(a) & \xrightarrow{\tau_a} & G(a) \\
\downarrow & & \downarrow \\
F(f) & & G(f) \\
\downarrow & & \downarrow \\
F(b) & \xrightarrow{\tau_b} & G(b)
\end{array}
\]

If \( \tau \) is a natural transformation from \( F \) to \( G \), we write \( \tau : F \to G \), or \( F \xrightarrow{\tau} G \),
Let $\mathbf{C}$ and $\mathbf{D}$ be the categories represented by the following diagrams (identities and compositions are suppressed):

\begin{align*}
\text{C} & \quad \text{b} \\
\quad \downarrow^f & \quad \downarrow^v
\end{align*}

\begin{align*}
\text{D} & \quad \text{z} \\
\quad \downarrow^y & \quad \downarrow^u
\end{align*}
Let $F : \mathbf{C} \to \mathbf{D}$ and $G : \mathbf{C} \to \mathbf{D}$ be functors from $\mathbf{C}$ to $\mathbf{D}$, defined as follows:

\[
\begin{align*}
F(a) &= x & G(a) &= y \\
F(b) &= y & G(b) &= z \\
F(f) &= u & G(f) &= v
\end{align*}
\]

The reader may infer the remaining arrow assignments.
That is, we have the following representations of $\mathbf{C}$ in $\mathbf{D}$:
Now, we define a natural transformation $\tau$ from $F$ to $G$ as the collection of arrows:

$$\tau_a = U$$
$$\tau_b = V.$$
That is, $\tau$ is the following “shift”:

\[
\begin{array}{c}
F(C) \\
\downarrow v \\
F(b) \\
\downarrow F(f) \\
F(a)
\end{array}
\quad z \quad
\begin{array}{c}
G(b) \\
\downarrow \tau_b \\
G(a) \\
\downarrow \tau_a \\
G(C)
\end{array}
\quad
\begin{array}{c}
G(C) \\
\downarrow x \\
F(a) \\
\downarrow u \\
F(f)
\end{array}
\]
**Natural Transformations**

**Identity Transformation**

The *identity transformation* on a functor $F : \mathcal{C} \to \mathcal{D}$ is the natural transformation $1_F : F \to F$ such that for all $a \in \text{Ob}(\mathcal{C})$, 

$$\tau_a = \text{id}_{F(a)} : F(a) \to F(a).$$
Natural Transformation Composition

Let $F$, $G$, and $H$ be functors from category $\mathbf{C}$ to category $\mathbf{D}$, and let $\tau : F \to G$ and $\sigma : G \to H$ be natural transformations. The composite of $\tau$ and $\sigma$, denoted $\sigma \circ \tau$, is the collection of arrows

$$(\sigma \circ \tau)_a = \sigma_a \circ \tau_a$$

for all $a \in \text{Ob}(\mathbf{C})$.

- Natural transformation composition is, of course, associative. The proof is left to the enthusiastic reader.
- Composition together with identity gives us the functor category $\mathbf{D}^\mathbf{C}$ – the category of all functors from $\mathbf{C}$ to $\mathbf{D}$. 
THE HASKELL

More information on Haskell Syntax and Theory:

- The Haskell: http://www.haskell.org
- Haskell Beginners:
  http://www.haskell.org/mailman/listinfo/beginners
- Haskell Cafe:
  http://www.haskell.org/mailman/listinfo/haskell-cafe