Week Twelve

Siva Sundar, EE23B151

December 22, 2024

16^{nd} December

- A **database** is an organised system of interlocking tables. **Database Scheme** are categories $\mathcal{C}$, data itself is given by a *'set-valued'* functor $\mathcal{C} \longrightarrow$**Set**, and databases can be mapped to each other via functors $\mathcal{C} \longrightarrow \mathcal{D}$.
- Category theory formalizes **data migration** between databases using **adjoint functors**.
- A category $\mathcal{C}$ consists of four constituents:
  - ⋆ a collection $\mathrm{Ob}(\mathcal{C})$, whose elements are **objects**.
  - ⋆ $\forall c, d \in \mathrm{Ob}(\mathcal{C})$, we specify the **hom-set** $\mathcal{C}(c, d)$, whose elements are called **morphisms**.
  - ⋆ $\forall c \in \mathrm{Ob}(\mathcal{C})$, we can specify the **identity morphism** on c: $id_c \in \mathcal{C}(c, c)$.
  - ⋆ $\forall c, d, e \in \mathrm{Ob}(\mathcal{C})$ and morphisms $f \in \mathcal{C}(c, d)$ and $g \in \mathcal{C}(d, e)$, we can define the **composite morphism** $f \circ g \in \mathcal{C}(c, c)$.

  It should also satisfy two properties:

  - ⋆ *Unitality*: $id_c \circ f = f \circ id_c = f$.
  - ⋆ *associativity*: $(f \circ g) \circ h = f \circ (g \circ h)$.
- For any graph $G = (V, A, s, t)$, we can define a **free category Free**(G) whose objects are *vertices* V, and morphisms from c to d are the *paths* from c to d.
- A category with one object is called **monoid**. (see page 83)
- A finite graph with path equations is called a *finite presentation* and this category is called *finitely-presented category.* (see difference between free category and commutative square in page 84)
- A preorder is a category where every two parallel arrows are the same (ie, between two points, there is *at-most* one morphism, Page 85-86). Similarly, any category can be converted to a preorder by destroying the distinction between any two parrelel morphisms.

19^{th} December

- The cateogory of finite sets, is called **FinSet**. There are many other categories as well. (See page 87)
- $f : A \longrightarrow B$ is an **isomorphism** if there exists another morphism $g : B \longrightarrow A$ satisfying $f \circ g = id_A$ and $g \circ f = id_B$. g is said to be inverse of f and A and B are said to be **isomorphic** objects.

- **Functor** is a mapping between categories. It maps objects as well as morphisms from one category to another. It obeys the *structure-preserving* rule: $F(f \circ g) = F(f) \circ F(g)$. (See page 91)

- Let $\mathcal{C}$ be a finitely-presented category. A **$\mathcal{C}$-instance** is a functor $I : \mathcal{C} \longrightarrow \textbf{Set}$. (It is a state of the database, "at an instant in time", see ex 3.45)

  The takeway is that: 'a database schema is a category, and an instance on that schema (the data itself), is a set-valued functor. The constraints (biz rules, etc) are ensured by the structure preserving rule.'

- **Natural transformation** (say α) is a mapping between two functors (say $F : \mathcal{C} \longrightarrow \mathcal{D}$ and $G : \mathcal{C} \longrightarrow \mathcal{D}$), denoted as $\alpha : F \Longrightarrow G$:

  - ⋆ $\forall c \in \mathcal{C}$, *c-component of* α is the morphism $\alpha_c : F(c) \longrightarrow G(c)$ in $\mathcal{D}$.
  - ⋆ **Naturality condition:** $\forall f : c \longrightarrow d$ in $\mathcal{C}$, $F(f) \circ \alpha_d = \alpha_c \circ G(f)$.

$$
\begin{tikzcd}
F(c) \arrow[r, "\alpha_c"] \arrow[d, "F(f)"'] & G(c) \arrow[d, "G(f)"] \\
F(d) \arrow[r, "\alpha_d"'] & G(d)
\end{tikzcd}
$$

Figure 1: Diagram commutes $\Longleftrightarrow$ naturality condition

$\alpha : F \longrightarrow G$ is said to be **natural isomorphism** if each of its component α_c is an **isomorphism in** $\mathcal{D}$.

- **Functor category:** Let $\mathcal{C}$ and $\mathcal{D}$ be categories. We define the **category of functors** $\mathcal{D}^{\mathcal{C}}$ with *objects* as functors $F : \mathcal{C} \longrightarrow \mathcal{D}$ and with *morphisms* $\mathcal{D}^{\mathcal{C}}(F, G)$ as the natural transformations $\alpha : F \longrightarrow G$. (See ex 3.72)

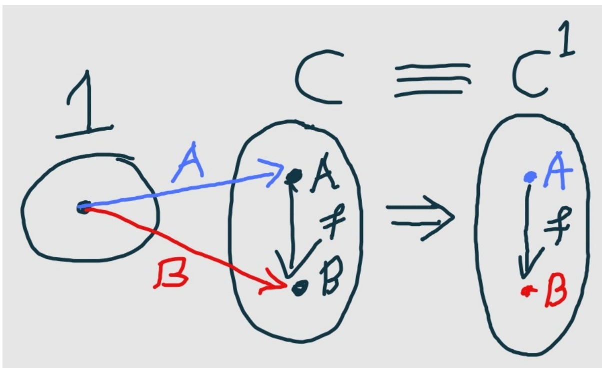

Figure 2: The category $\mathcal{C}^1$ is equivalent to $\mathcal{C}$.

The category of preorders is equivalent to category of **Bool**-categories. (Page 97)

22^{nd} *December*

- Let $\mathcal{C}$ be a *database scheme* and $I, J : \mathcal{C} \longrightarrow \textbf{Set}$ be *database instances*. An **instance homomorphism** between I and J is a *natural transformation* $\alpha : I \longrightarrow J$ and these are included in the *functor category* $\mathcal{C} - \textbf{Inst} := \textbf{Set}^{\mathcal{C}}$.

- The objects in the functor category **Gr-Inst** are graphs and the morphisms init are called *graph homomorphisms* which must *respect source and target*. (See page 98)
- Let $\mathcal{C}$ and $\mathcal{D}$ be categories and let $F : \mathcal{C} \longrightarrow \mathcal{D}$ be a functor. For any set-valued functor $I : \mathcal{D} \longrightarrow \textbf{Set}$, the composite functor $F \circ I$ is called **pullback of I along F**.
- Let $L : \mathcal{C} \longrightarrow \mathcal{D}$ and $R : \mathcal{D} \longrightarrow \mathcal{C}$ be functors. L is the **left adjoint** to R and R is the **right adjoint** to L if there exist an isomorphism (see page 102):

$$\forall c \in \mathcal{C} \;\;\&\;\; d \in \mathcal{D}, \qquad \alpha_{c,d} : \mathcal{C}(c, R(d)) \xrightarrow{\cong} \mathcal{C}(L(c), d)$$

  In set theory, given a set B, we have an adjunction called **currying** B: (see page 103)

$$\textbf{Set}(A \times B, C) \cong \textbf{Set}(A, C^B)$$

- Given $F : \mathcal{C} \longrightarrow \mathcal{D}$, the **data migration functor** Δ_F turns $\mathcal{D}$-instances into $\mathcal{C}$-instances.

$$\begin{tikzcd}
\mathcal{C}\textbf{-Inst} \arrow[r, bend left, "\Sigma_F"] \arrow[r, bend right, "\Pi_F"'] & \mathcal{D}\textbf{-Inst} \arrow[l, "\Delta_F" description]
\end{tikzcd}$$

Figure 3: Σ_F and Π_F are the Left and Right (Pushforward) adjoint functors resp. (See page 105)