Category Theory Seminar

Databases: Schemas and Instances

Stephen Diadamo - January 10, 2017

Contents

1 Introduction

In this report, the concept of a database schema will be introduced. To do so, since concepts in databases heavily rely on graph theory, some basic definitions and concepts of graphs will be introduced. We then see how we can categorize a schema by using special types of data table which will allow a schema to meet all the criteria of being a category. Finally we will examine the equivalence between the category of schemas and the category of categories.

2 Graphs

Definition. A **graph** G is a quadruple $G := (V, A, src, tgt)$ such that

1. V is a set called *the set of vertices of* G.
2. A is a set called *the set of arrows of* G.
3. $src : E \to V$ is a function called *the source function for* G.
4. $tgt : E \to V$ is a function called *the target function for* G.

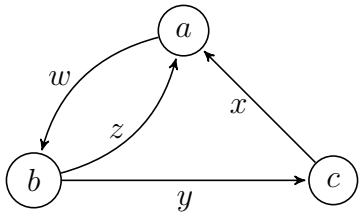
For an arrow $a \in A$, We call $src(A)$ *the source vertex* of a and $tgt(A)$ *the target vertex* of a.

Example. Let $G = (\{a, b, c\}, \{w, x, y, z\}, src, tgt)$ be a graph, where src and tgt are defined as follows,

src(A)	V
w	a
x	c
y	b
z	b

tgt(A)	V
w	b
x	a
y	c
z	a

The pictorial representation is given below.



Definition. Let $G = (V, A, src, tgt)$ be a graph. A **path of length n** in G denoted $p \in \mathrm{Path}_G^{(n)}$ is a head-to-tail sequence

$$p = (v_0 \xrightarrow{a_1} v_1 \xrightarrow{a_2} v_2 \xrightarrow{a_3} ... \xrightarrow{a_n} v_n)$$

of arrows in G. We denote by Paths_G the set of paths in G,

$$\mathrm{Paths}_G = \bigcup_{n \in \mathbb{N}} \mathrm{Path}_G^{(n)}$$

Definition. Let $G := (V, A, src, tgt)$ be a graph. A **path equivalence declaration** is an expression of the form $p \simeq q$ where $p, q \in \mathrm{Paths}_G$ have the same source and target (i.e. $src(p) = src(q)$ and $tgt(p) = tgt(q)$). A **congruence** on G is a relation $\simeq$ on Paths_G that has the following properties,

1. $\simeq$ is an equivalence relation
2. if $p \simeq q$, then $src(p) = src(q)$
3. if $p \simeq q$, then $tgt(p) = tgt(q)$
4. Suppose $p, q : a \to b$ are paths, and $m : a \to b$ is an arrow, then $mp \simeq mq$ if $p \simeq q$
5. Suppose $p, q : a \to b$ are paths, and $n : b \to c$ is an arrow, then $pn \simeq qn$ if $p \simeq q$

Definition. Let $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ be two graphs. A **graph homomorphism** $f : G \to G'$ consists of two functions $f_0 : V \to V'$ and $f_1 : A \to A'$ such that the following diagrams commute,

$$\begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ {\scriptstyle src}\downarrow & & \downarrow{\scriptstyle src'} \\ V & \xrightarrow[f_0]{} & V' \end{array} \qquad \begin{array}{ccc} A & \xrightarrow{f_1} & A' \\ {\scriptstyle tgt}\downarrow & & \downarrow{\scriptstyle tgt'} \\ V & \xrightarrow[f_0]{} & V' \end{array}$$

Remark. This definition encodes at a high level that "arrows are bound to their vertices", which means arrows should not change which vertices they are connected to, or in other words, the source and target should be preserved under the mapping.

3 What are Databases?

Briefly, a database consists of tables and relationships between them. The tables of a database contain data which are called the "records" or "tuples." For each table, there is a particular set of attributes that each record should have. One attribute that each record must contain is a unique identifier which is called the "primary key." Records can also contain as an attribute a reference to the primary key of another record from a different table called "foreign keys." Attributes that are not keys are called "pure data" and contain for example a name, date, or any other form of immediately usable information. There can also be a set of rules that set of tables together must obey which are sometimes called the "business rules." The set of tables and their respective attributes along with the business rules form what is called the "database schema" or just the "schema."

Example.

Students			
ID	**first**	**last**	**Faculty**
10	John	Smith	maths
11	Bob	Small	informatics

Category Theory Talks		
ID	**student**	**date**
20	10	10.10.2016
21	11	11.10.2016

- Students table rules: For each student s, the faculty must be maths or informatics
- Category Theory Talks table rules: For each student s, there is at least one talk date

This example demonstrates a database schema. There are two tables, Students and Category Theory talks, and two records in each table. Each record in the Students table has four attributes and each record in the Category Theory Talks has three. In particular, in the Category Theory Talks table, the "student" attribute is a foreign key, and the data must match an ID of a record from the Students table. Each table also has a set of rules which must be enforced. All of this together forms the schema.

4 Categorizing Schemas

In this section we will see how a schema can be categorized. The objects will be the tables and arrows the relationships between them. We firstly consider a special

class of single attribute tables that allow pure data elements of general tables to be considered as foreign keys. Adding these special tables to a schema allows us to consider each table as a composition of keys where one is primary and the rest are foreign.

Example.

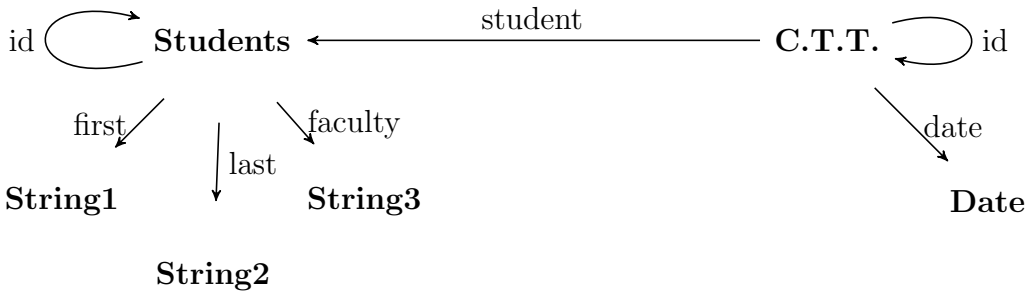
Integers
ID
0
1
⋮
10
⋮
1000
⋮

Dates
ID
01.01.0000
02.01.0000
⋮
01.01.0001
⋮
01.01.2020
⋮

Strings
ID
a
b
⋮
Aab Baa
⋮
John Smith
⋮

A schema with these special tables can then be considered as a family of tables linked by foreign keys. Thus a schema is a category where the objects are tables, the morphisms are the columns for each table, the identity arrow comes from the primary key back to itself, and composition comes from composition of paths. To enforce the business rules in the category, we use path equivalences, which we defined in the graph section.

Example. Let $\mathcal{C} :=$



Student $\simeq$ C.T.T., faculty $\simeq$ String3.maths or faculty $\simeq$ String3.informatics

This example is a depiction of the example we saw previously. There are six tables and for each foreign key, there is an arrow to another table. Business rules are enforced with path equivalence declarations.

Example. *Loop* :=

This simple schema generates the free monoid on one generator, which is isomorphic to $\mathbb{N}$. It has just one object $\mathbf{s}$, but has an infinite family of morphisms of the form $f^n : s \to s, n \in \mathbb{N}$ which can be thought of as "how many times to go around the loop."

Example. To populate data in our schemas, we can define a functor $I : \mathcal{C} \to \mathbf{Set}$ consisting of

- A set for each object in $\mathcal{C}$
- A function for each arrow in $\mathcal{C}$
- The declared path equivalences hold

In words, this is a functor that fills the database with data, searching **Set** for data that satisfies the schema rules. These categorical databases are also called functional databases.

5 Equivalence of Sch and Cat, Sch $\simeq$ Cat

5.1 Defining Sch

In this section, we start by determining what the category of schemas **Sch** is, and then showing that it is equivalent to the category of categories **Cat**. We begin with a more compact definition of a schema.

Definition. A database schema $\mathcal{C}$ consists of a pair $\mathcal{C} := (G, \simeq)$ where $G = (V, A, src, tgt)$ is a graph representing the set of tables and the relationships between them and $\simeq$ a congruence on G which declares path equivalences, to enforce business rule.

If we think schemas are analogous to categories, we should have some concept of functors in schemas. Since a schema is composed of a graph and path equivalences, one would think that graph homomorphisms would be the correct candidate since they preserve graph structure, but this is not entirely complete. The reason for this is because graph homomorphisms require arrows to be mapped to arrows while preserving their mapped source and target, but paths may not be preserved. The morphisms we desire therefore need to preserve paths, in order to preserve path equivalence declarations.

We then construct such morphisms so that path endpoints, path concatenation, and path equivalences are preserved, which will result in the desired morphisms. Because a path is a concatenation of arrows, it suffices to give a function F from the arrows of schema $\mathcal{C} = (G, \simeq_G)$ to the paths of schema $\mathcal{C}' = (G', \simeq_{G'})$. F then automatically preserves endpoints and concatenation. So the only requirement that must explicitly be met is that F preserves path equivalences. Before we construct the desired morphisms we explore two examples.

Example. Let $G = (V, A, src, tgt)$ be a graph. With the set Paths_G of all paths in G, and functions $\overline{src} : \text{Paths}_G \to V$ and $\overline{tgt} : \text{Paths}_G \to V$, it is possible to define a new graph called the paths-graph of G as,

$$\text{Paths}(G) := (V, \text{Paths}_G, \overline{src}, \overline{tgt}).$$

The arrows in G are considered length 1 paths, so the structure of G is contained in $\text{Paths}(G)$. Indeed, G is homomorphic to $\text{Paths}(G)$. It turns out that Paths: **Grph** $\to$ **Grph** is a functor.

Example. For any graph G, we can construct the paths-graph, and further, since the paths-graph is itself a graph, we can construct the paths-graph of the paths-graph,

$$\text{Paths}(\text{Paths}(G)) = (V, \text{Paths}_{\text{Paths}(G)}, \overline{src}', \overline{tgt}').$$

$\text{Paths}(\text{Paths}(G))$ can be thought of as paths of paths in G, or head-to-tail sequences of paths. For example, in the first example of a graph, a path of paths could be,

$$(b \to c \to a) \to (a \to b \to c).$$

From this, we can see that it is possible to compose this sequence of paths into one big path, i.e.

$$b \to c \to a \to b \to c,$$

and so one could consider the graph homomorphism,

$$\mu_G : \text{Paths}(\text{Paths}(G)) \to \text{Paths}(G)$$

which can be considered as a concatenation of paths.

The original goal was to find a morphism F to map the paths of a graph to the paths of another graph, i.e. $F : \text{Paths}(G) \to \text{Paths}(G')$. Since the structure of a graph G is contained in its paths-graph $\text{Paths}(G)$, a graph homomorphism $f : G \to G'$ can be considered as a map $f : G \to \text{Paths}(G')$. Now, with some graph homomorphism $f : G \to \text{Paths}(G')$, to accomplish the goal, we can construct a function $\text{Paths}_f : \text{Paths}(G) \to \text{Paths}(G')$ using a combination of the previous maps in the examples. Explicitly,

$$G \xrightarrow{\text{Paths}} \text{Paths}(G) \xrightarrow{\text{Paths}(f)} \text{Paths}(\text{Paths}(G')) \xrightarrow{\mu_{G'}} \text{Paths}(G')$$

This says that given a function from the arrows in G to the paths in G', a path in G becomes a path of paths in G', which can be concatenated to just a path in G'. Formally, we have the definition of a schema morphism.

Definition. (Schema morphism). Let $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$ be graphs and let $\mathcal{C} = (G, \simeq_G)$ and $\mathcal{C}' = (G', \simeq_{G'})$ be schemas. A **schema morphism** $F : \mathcal{C} \to \mathcal{C}'$ is a graph homomorphism $F : G \to \text{Paths}(G')$ that satisfies the following condition for any paths p and q in G:

$$p \simeq_G q \Rightarrow \text{Paths}_F(p) \simeq_{G'} \text{Paths}_F(q),$$

where $\text{Paths}_F(p)$ represents the path that p is mapped to under F.

With this, we can again define the category of schemas, **Sch**, where the objects are schemas, the morphisms are schema morphisms, and the identity morphism on a schema $\mathcal{C}$ is the schema morphism $\text{id}_{\mathcal{C}} : G \to \text{Paths}(G)$. Composition of schema morphisms $F : \mathcal{C} \to \mathcal{C}'$ and $F' : \mathcal{C}' \to \mathcal{C}''$ on their objects is simply the composition of the graph homomorphism mapping vertices to vertices. For the arrows, given an arrow of $\mathcal{C}$ it is mapped to a path in $\mathcal{C}'$ and then mapped to a path in the paths-graph (i.e. a path of paths) of $\mathcal{C}''$. Using $\mu_{G''}$, we can concatenate the paths of paths to one path in G'', and thus a complete map from the paths in $\mathcal{C}$ to the paths in $\mathcal{C}''$.

5.2 Explanation of Equivalence

Here we investigate how **Sch** is equivalent to **Cat**. Recall that two categories, $\mathcal{C}$ and $\mathcal{D}$, are equivalent if there are two functors $L : \mathcal{C} \to \mathcal{D}$ and $R : \mathcal{D} \to \mathcal{C}$ such that there is a pair of natural isomorphisms $\alpha : 1_{\mathcal{C}} \xrightarrow{\sim} R \circ L$ and $\beta : 1_{\mathcal{D}} \xrightarrow{\sim} L \circ R$. We will need to construct the two functors L and R for **Sch** and **Cat** and state the theorem of equivalence with a proof sketch.

Construction. (From Schema to Category) Define $L : \mathbf{Sch} \to \mathbf{Cat}$ such that for schema $\mathcal{C} = (G, \simeq), G = (V, A, src, tgt)$, $L(\mathcal{C})$ is the category such that $\mathrm{Ob}(\mathrm{L}(\mathcal{C})) = V$, and for $v_1, v_2 \in V$, $\mathrm{Hom}_{\mathrm{L}(\mathcal{C})}(v_1, v_2) := \mathrm{Path}_G(v_1, v_2)/\simeq$, where $\mathrm{Hom}_{L(\mathcal{C})}(v_1, v_2)$ is the set of morphisms between v_1 and v_2. Composition of morphisms is simply the concatenation of paths. We should ensure that these sets of morphisms are well-defined, so we can refer to lemma 3.5.2.5 in [1]. Therefore we have defined L on the objects of **Sch**.

Now we define L on the morphisms of **Sch**, which are schema morphisms. Given a schema morphism $F : \mathcal{C} \to \mathcal{C}'$ for schemas $\mathcal{C}$ defined above and $\mathcal{C}' = (G', \simeq')$, $G' = (V', A', src', tgt')$, we define the functor $L(F) : L(\mathcal{C}) \to L(\mathcal{C}')$. Since F is a schema morphism, paths equivalences are preserved via Paths_F (the map from paths in G to paths in G'). A morphism in $L(\mathcal{C})$ is then an equivalence class of paths in $\mathcal{C}$, so by definition of schema morphisms, for any representative path $p \in \mathrm{Paths}(G)$ such that $\mathrm{Paths}_F(p) \in \mathrm{Paths}(G')$, if $p \simeq q$ then $\mathrm{Paths}_F(p) \simeq' \mathrm{Paths}_F(q)$. Therefore Paths_F does indeed map $\mathrm{Hom}_{L(\mathcal{C})} \to \mathrm{Hom}_{L(\mathcal{C}')}$. With this construction L preserves composition via path concatenation and preserves identities, and so it is a functor.

Construction. (From Category to Schema) Define R : **Cat** $\to$ **Sch**, where for category $\mathcal{D} = (\mathrm{Ob}(\mathcal{D}), \mathrm{Hom}_{\mathcal{D}}, dom, cod, \mathrm{ids}, \circ)$, $R(\mathcal{D}) = (G, \simeq)$, where $G = (\mathrm{Ob}(\mathcal{D}), \mathrm{Hom}_{\mathcal{D}}, dom, cod)$ is a graph and $\simeq$ is defined as the congruence generated by the following path equivalence declarations: for any composable sequence of morphisms $f_1, f_2, ..., f_n$ such that $dom(f_{i+1}) = cod(f_i), 1 \leq i \leq n-1$, we declare the path equivalence

$$[f_1, f_2, ..., f_n] = [f_n \circ ... \circ f_2 \circ f_1].$$

We now define R on the morphisms of **Cat**. A functor $F : \mathcal{D} \to \mathcal{E}$, for categories $\mathcal{D}$ and $\mathcal{E}$ induces a schema morphism $R(F) : R(\mathcal{D}) \to R(\mathcal{E})$ because vertices in $R(\mathcal{D})$, $\mathrm{Ob}(\mathcal{D})$, are mapped to the vertices of $R(\mathcal{E})$, $\mathrm{Ob}(\mathcal{E})$, and arrows are mapped to arrows as length one paths. Path equivalences are preserved since F

preserves what is called the composition formula. It is also clear that R preserves compositions and therefore it is indeed a functor.

Theorem. The functors $L : \mathbf{Sch} \to \mathbf{Cat}$ and $R : \mathbf{Cat} \to \mathbf{Sch}$ are mutually inverse equivalences of categories.

Proof. (Sketch) It is clear that there is a natural isomorphism $\alpha : \mathrm{id}_{\mathbf{Cat}} \xrightarrow{\simeq} L \circ R$.

We describe $R(L(\mathcal{S})) =: (G', \simeq')$ for a schema $\mathcal{S} = (G, \simeq)$. Let $G = (V, A, src, tgt)$ and $G' = (V', A', src', tgt')$. The vertices under $R \circ L$ are equal, that is $V' = V$. For the arrows, they are preserved up to congruence $\simeq$, that is $A' = \mathrm{Paths}_G / \simeq$. $\simeq'$ is imposed by the composition formula, which I do not state here, but can be seen in 5.14, [1]. Under $\simeq'$, the paths in Paths(Paths(G)) are declared equivalent to their concatenations and considered length one paths in G'.

There is then a natural transformation $\beta : \mathrm{id}_{\mathbf{Sch}} \to R \circ L$ such that for schema $\mathcal{S}$ defined above, it sends the arrows in G to length one paths in G'. To show β is isomorphic we find its inverse. For any arrow f in A', f is an equivalence class of paths in $\mathcal{S}$. One selects any representative in f and maps f to that representative. By the definition of a schema morphism, this is a well defined action, and will give the identical morphism of schemas. □

References

[1] David Spivak - Category Theory for Scientists (2013)

[2] David Spivak - Talk on "Databases as Categories" - http://math.mit.edu/∼dspivak/informatics/talks/CTDBIntroductoryTalk