CS 511
Formal Methods for High-Assurance Software Engineering
Homework Assignment 13

Out: 6 December 2024
Due: Thursday, 12 December 2024, by 11:59 pm

Repeated below are administrative issues already mentioned in the handout of Assignment 01:

- You need to open a Gradescope account, after which you need to add yourself to the CS511 roster for this semester. The entry code for CS511, Fall 2023, is `WWX2NW`.

  If you want to read more on adding yourself to the CS511 roster, go to Adding a Course .

- You also need to create a *GitHub repository* where you store your solutions for *coding exercises with* LEAN_4.

  To create a GitHub repository, you need to open a GitHub account. Instructions for how to do this are at the following webpages: Set Up a GitHub Account and Create a GitHub Repository.

- Typically, each weekly assignment consists of two parts:

  1. One part includes ***hand exercises***, *i.e.* ***pencil-and-paper exercises***, and
  2. One part includes ***coding exercises*** in LEAN_4.

  And each of the two parts will consist of:

  – **2 easy exercises**, and
  – **1 demanding exercise**, which we will call a **problem**,

  for a total of **4 easy exercises** and **2 problems** in each weekly assignment.

- Typeset your solutions with Latex to produce a single '`.pdf`' file containing:

  1. All your solutions for the ***hand exercises***, and
  2. Links to your ***coding exercises***, which are stored in your GitHub repository. (You should insert the links as active, *i.e.* clickable, *hyperlinks* in your '`.pdf`' file.)

  It is the '`.pdf`' file produced with Latex that you will submit in Gradescope.

  You do not need to use any particular format in naming your '`.pdf`' file, because Gradescope will keep track of who is submitting it. Nonetheless, it is nice to use suggestive names in case of a mishap and we need to recover your file. So, here is a possible naming:

  ```
  <your last name>_<your first name>.hw01.pdf
  ```

  For example, for myself, I would call my file '`kfoury_assaf.hw01.pdf`'.

For the first two exercises and the first problem in this homework assignment, you will need to consult the slides of Zach Moring's presentation on Tuesday, December 3. Zach's slides are available under Resources on Piazza.

1 By Hand

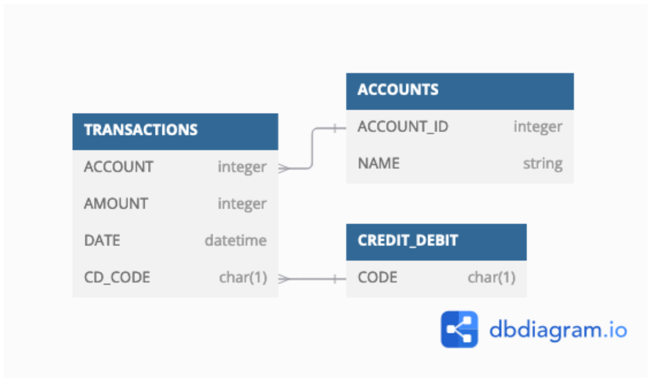
Exercise 1 Some questions related to database methods.

Part (a) The SQL standard provides an operation `EXISTS`, which can be used as an existential quantifier. For example,

```
SELECT ...  FROM ...  WHERE EXISTS (<subquery>)
```

So to express a database, we certainly need at least first-order logic. Argue as to whether or not *second*-order logic or higher is needed for any SQL operations you are familiar with. Is first-order logic sufficient for all SQL operations?

Part (b) Consider the following schema:



Draw a diagram representing this schema, using the diagram language from the lecture on December 3. Use filled circles for table vertices and empty circles for type vertices. $\square$

Exercise 2 In 1763, Leonhard Euler created a very famous graph representing the islands of the Pregel River and the seven bridges across it. (To understand the very simple question he wanted to solve which motivated one of the first problems answered by graph theory, you may read the Wikipedia article on "The Seven Bridges of Königsberg.") Here is the graph K which he drew, with some arrows added:

$$\begin{tikzcd}
 & B \arrow[dl, bend left=15, "g"] \arrow[dr, "k"] & \\
A \arrow[ur, bend left=15, "f"] \arrow[rr, "h"] \arrow[dr, bend right=15, "i"] & & C \\
 & D \arrow[ul, bend right=15, "j"] \arrow[ur, "\ell"'] &
\end{tikzcd}$$

Let $\mathcal{K}$ be the free category on K.

Part (a) List the 15 morphisms of $\mathcal{K}$ along with their domains and codomains.

Part (b) A diagram is called a "commutative" diagram if all paths with the same start and end point are equal; that is, if all "parallel" paths through the diagram produce the same result. Let $\mathcal{K}'$ be the *commutative* free category on K; list the morphisms of $\mathcal{K}'$. Using the answers of part (a) should make this a trivial exercise.

Part (c) Consider the following category $\mathcal{V}$:

$$\begin{tikzcd} U \arrow[r, "p"] & V & W \arrow[l, "q"'] \end{tikzcd}$$

Define a functor $F : \mathcal{V} \to \mathcal{K}$.

Part (d) Imagine a category that looks like this:

$$\begin{tikzcd} \bullet \arrow[r] & \bullet \arrow[d] \\ \bullet \arrow[u] & \bullet \arrow[l] \end{tikzcd}$$

Argue why there cannot be a functor from this category to $\mathcal{K}$. $\square$

Before you proceed with Problem 1 below, read carefully the following definition. The *category of sets*, denoted **Set**, is specified as follows:

(i) Ob(**Set**) is the collection of all sets.

(ii) If S and T are sets, then $\mathbf{Set}(S, T) := \{\ f : S \to T \mid f \text{ is a function }\}$.

(iii) For every set S, the identity morphism is the function $\mathrm{id}_S : S \to S$ given by $\mathrm{id}_S(s) := s$ for every $s \in S$.

(iv) Given $f : S \to T$ and $g : T \to U$, their composite is the function $(f\ ;\ g) : S \to U$ given by $(f\ ;\ g)(s) := g(f(s))$.

Note that, for all $f : S \to T$, $g : T \to U$, and $h : U \to V$, and all $s \in S$, we have:

$(f\ ;\ \mathrm{id}_T)(s) = f(s) = (\mathrm{id}_S\ ;\ f)(s)$, which is called the *unitality condition*,

$\big((f\ ;\ g)\ ;\ h\big) = \big(f\ ;\ (g\ ;\ h)\big)$, which is called the *associativity condition*.

Satisfaction of these two conditions by **Set**, which is indeed the case, is required in order that **Set** be called a *category* – but you don't need to worry about this in this assignment.

Before proceeding with Problem 1, also read the following example. Let $B \in \mathrm{Ob}(\mathbf{Set})$ be any set. There is an adjunction called "currying B," after logician Haskell Curry, expressed as follows:

$$\begin{tikzcd}\mathbf{Set} \arrow[r, bend left, "- \times B"] \arrow[r, phantom, "\Longrightarrow"] & \mathbf{Set} \arrow[l, bend left, "(-)^B"]\end{tikzcd} \qquad \mathbf{Set}(A \times B, C) \cong \mathbf{Set}(A, C^B)$$

Abstractly, we write it as on the left, but what this means is that for any sets A and C, there is a natural isomorphism (denoted here by "$\cong$") 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, which we can denote by writing C^B. It is written this way because if C has 10 elements and B has 3 elements, then C^B has 10^3 elements – which is just the number of ways, *i.e.*, $10 \times 10 \times 10$, of placing the 3 elements of the domain B in the 10 slots of the codomain C – so that, more generally for any two finite sets, we have $\left|C^B\right| = \left|C\right|^{|B|}$.

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 $A \to C^B$, with the latter being equivalent to $A \to (B \to C)$. Intuitively, if we have a function f of a pair of variables (a, b), then we can "put off" entering the second variable, *i.e.*, if we are given only $a \in A$, we will return a function $B \to C$ which is wating for the second input $b \in B$. This is what is called the *curried* version of f.

PROBLEM 1 The preceding example about *currying* illustrates of what is called an *adjunction* between functors, here the functor $- \times B$ and the functor $(-)^B$. We only said how each of these two functors works on objects: For an arbitrary set X, the first functor returns the set $X \times B$ while the second returns the set X^B. There are three parts in this problem – these may look scary to you, but the answer to each part takes at most 2 (or perhaps 3) lines:

1. Given a morphism $f : X \to Y$, what morphism should $- \times B : X \times B \to Y \times B$ return?
2. Given a morphism $f : X \to Y$, what morphism should $(-)^B : X^B \to Y^B$ return?
3. Consider the function $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$, which maps (a, b) to $a + b$. Currying $+$ we get a function $p : \mathbb{N} \to \mathbb{N}^{\mathbb{N}}$. What is $p(3)$? □

2 With Lean_4

Exercise 3 From Macbeth's book: Exercise 10.1.5.4 □

Exercise 4 From Macbeth's book: Exercise 10.1.5.5 □

PROBLEM 2 From Macbeth's book: Exercise 10.1.5.6 □