Assignment 2: Modelling and analysis of cyber-physical systems now with monads

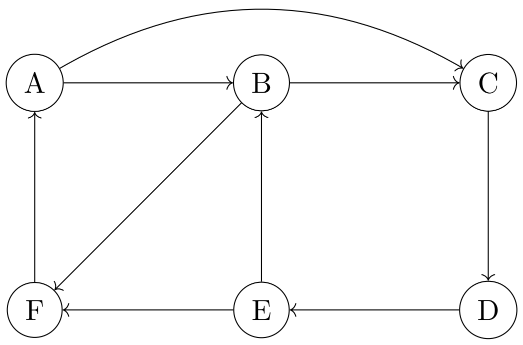
Renato Neves
Arquitectura e Cálculo – 2020/2021

This assignment revolves around the travelling salesman problem (TSP). This is a famous example of a question that can be easily formulated but whose answer cannot be *efficiently* obtained. Note that TSP has numerous applications in surprisingly different domains, such as mail distribution and prediction of protein functions[1].

Your task in this assignment is to model and analyse different variants of TSP *via monads*. The details of this task are given next.

First part (Preliminaries, 30% of the grade)

TSP is based on important concepts of graph theory [2]: namely those of *Hamiltonian path* and *Hamiltonian cycle*. Consider a directed graph. A path in this graph is called Hamiltonian iff it visits each vertex (a.k.a. node) of the graph *exactly once*. For example, consider the following directed graph:



The paths $ABCDEF$ and $ACDEBF$ are Hamiltonian, but the path $ABCDEBF$ is not. Then a *Hamiltonian cycle* is a path $X_1 \ldots X_n$ such that $X_1 = X_n$ and $X_2 \ldots X_n$ is Hamiltonian. For example, the paths $ABCDEFA$ and $ACDEBFA$ are Hamiltonian cycles, but $ABCDEF$ is not.

Hamitonian paths and Hamiltonian cycles can be computed via *backtracking* using the list monad.

Exercise 1 (Warmup). Recall the slides of the previous lectures. Then consider the λ-term,

$$x : \mathbb{N} \vdash_{\mathtt{c}} y \leftarrow \text{choice}(\mathtt{return}(x-1), \mathtt{return}(x+1)); \text{choice}(\mathtt{return}(y-1), \mathtt{return}(y+1)) : \mathbb{N}$$

and its interpretation using the list monad. What is the output of this λ-term for input 0?

[1]http://www.math.uwaterloo.ca/tsp/index.html
[2]https://en.wikipedia.org/wiki/Graph_theory

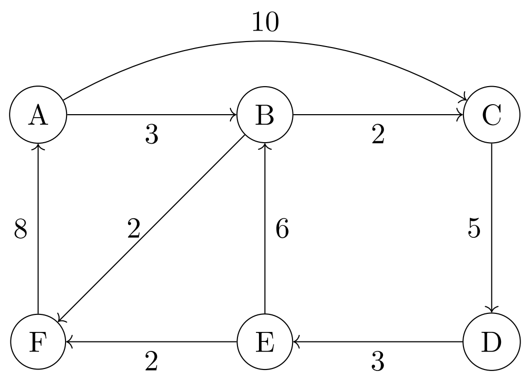
Exercise 2 (Computation of Hamiltonian cycles). Fill-in the definitions of the functions `addtoEnd` and `hCycles` in `Problem1.hs`, using in both cases the list monad.

A hint. Recall previous homeworks and the Knight's quest example.

N.B. Do not change the names of the functions in the code, because we will use these names for automatic testing.

Second part (TSP, 30% of the grade)

Let us now go over the actual formulation of TSP. Consider a directed graph whose edges are labelled by natural numbers. Consider also a node A in the graph. Intuitively, the nodes in the graph can be regarded as cities and the labels in the edges can be regarded as the time necessary to move from one city to another. Then TSP asks for the Hamiltonian cycle starting in A that has the *least cost*. The cost of a cycle is calculated by summing all the labels encountered when traversing the edges stipulated by the cycle. For example, consider the following directed graph:



The cost of $ABCDEFA$ is $3+2+5+3+2+8 = 23$. In the previous task you saw that Hamiltonian cycles can be computed via the list monad. The cost of paths, on the other hand, can be computed via the duration monad. Both monads working together thus allow to compute a solution for TSP.

Exercise 3 (Warmup). Recall the duration monad and the interpretation rules of effectful λ-calculus. Then show that the equation,

$$[\![x : \mathbb{A} \vdash_{\mathsf{c}} (\lambda y : \mathbb{A} \to \mathbb{A}.\ \mathtt{wait}_2(y\ x))(\lambda z : \mathbb{A}.\ \mathtt{wait}_1(\mathtt{return}\ z)) : \mathbb{A}]\!] = [\![x : \mathbb{A} \vdash_{\mathsf{c}} \mathtt{wait}_3(\mathtt{return}\ x) : \mathbb{A}]\!]$$

holds. Essentially this means that we can *simplify* the program on the left to the one on the right without changing its behaviour.

Exercise 4 (Solution of TSP). Fill-in the definitions of the functions `tadjacentNodes`, `taddToEnd`, and `hCyclesCost` in `Problem2.hs`, using the list and duration monads.

N.B. Do not change the names of the functions in the code, because we will use these names for automatic testing.

Third part (Essay, 15% of the grade)

Exercise 5. Write a small essay (around 400 words) on the (dis)advantages of using monads in programming.

Fourth part (Open Exercise, 25% of the grade)

Exercise 6.[3] Copy the code that you developed in `Problem1.hs` to `Problem3.hs`. After this, extend the code in `Problem3.hs` to accomodate any functionality that you may find relevant for TSP. We will value those functionalities that recur to monadic machinery. Here are two examples of this:

1. To read graphs from a file (uses the IO monad, among other things);
2. Recall that TSP can be applied to routing problems. One can then postulate the existence of an *electric vehicle* and that each visit to a city allows to recharge the vehicle's battery by a certain amount. A natural question then to ask is: which are the routes that the vehicle can perform without *emptying completely* the battery? This problem can be naturally tackled via the state monad, among other things.

What to submit: A single report in PDF that contains the essay proposed in Exercise 5 and that presents/justifies all the steps and design choices made to complete the remaining exercises. Submit also the completed source files (`Problem1.hs`, `Problem2.hs`, and `Problem3.hs`). Send by email (nevrenato@gmail.com) a unique zip file "`ac1920-N1_N2.zip`", where `N1` and `N2` are your student numbers. The subject of the email should be "`ac1920 N1 N2`"

Deadline: 13th June 2021 @ 23h59

[3]This exercise is more time consuming than the other exercises. So I recommend it to be done only after all other exercises are completed.