

Homework 1

Questions

1. What is computational model?
Computational models provides values and operation for the programing language, it means that power or the functionality depends on the computational model.
2. What is the most ancient computational model?
I think that the most ancient computational model is simple arithmetic. The set of values is the set of natural numbers and the set of operation is just addition, subtraction, multiplication and division.
3. What types of computational models there are? Give short description of main difference in your own words.
There are three main models of computation: functional, logical and imperative. Functional and Imperative are quite similar but they differ in a way they treat variables. Each function in a functional model is stateless and can access only variables from the local scope. On the contrary, Imperative has a state and the result of the function may depend on it. Logical model is different from both the imperative and the functional model. Some logical statements are given to it and apart from other models it can guess using deduction method.
4. What is programming language?
Programming Language is a way of writing programs, which are essentially just some computations to achieve certain results.
5. What programming language consists of?
Programming language consists of Computational Model, Syntax and Semantics.
6. What is syntax and semantics?
Syntax defines structure of the program, i.e. the rules of how the program must be written.
Semantics defines the meaning of words.
7. It terms of decidability, are those models different?
No, they are the same. For me it makes sense because modern programming languages operate on similar sets of instruction architectures and on the binary CPUs

Miranda – Functional Language

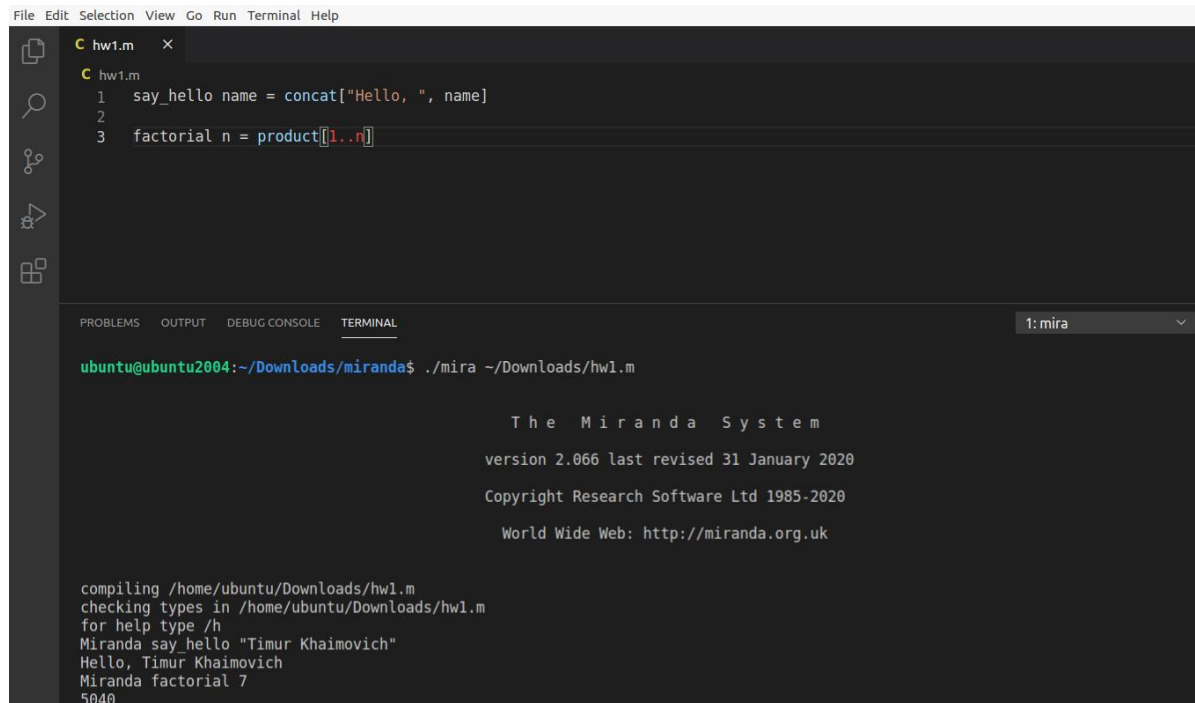
I wanted to choose [Hope](#) functional programming language, because this is the only way I can have an A (just kidding), but I had problems finding a compiler and a neat tutorial. Then I decided to choose [Miranda](#) because this language was in some way inspired by Hope and also Haskell was influenced by Miranda.

Important Note

Programs in Miranda are called scripts and they **only contain definitions**, this is important because it means that **we can only use functions from the interpreter**

How to setup Miranda and run my script, tested on Ubuntu 20.04:

1. [Download](#) archive with Miranda and unpack it
2. Open terminal and navigate to the folder with Miranda
3. Use *make* command to build Miranda's interpreter
4. Run *hw1.m* using *./mira* command



The screenshot shows a code editor with a file named `hw1.m` containing the following Miranda code:

```
1 say_hello name = concat["Hello, ", name]
2
3 factorial n = product[1..n]
```

Below the code editor is a terminal window titled `1: mira`. It shows the command `./mira ~/Downloads/hw1.m` being executed. The output of the command is:

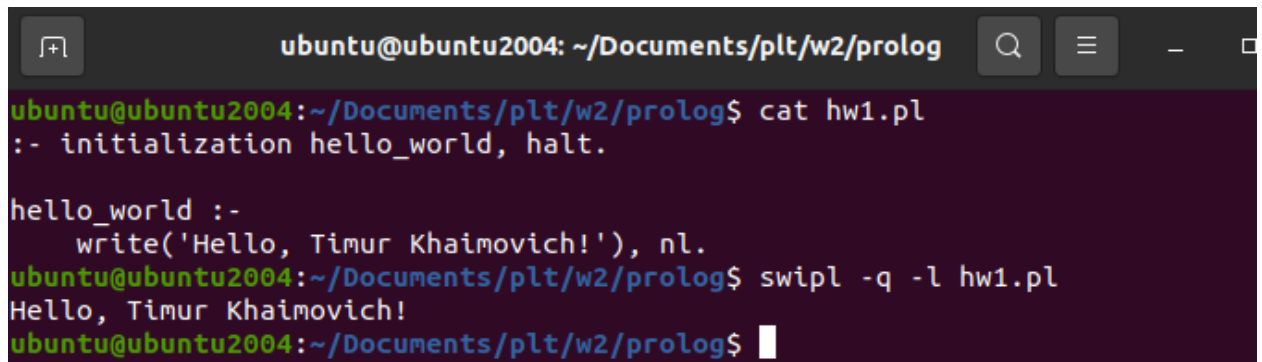
```

      T h e  M i r a n d a  S y s t e m
version 2.066 last revised 31 January 2020
Copyright Research Software Ltd 1985-2020
World Wide Web: http://miranda.org.uk

compiling /home/ubuntu/Downloads/hw1.m
checking types in /home/ubuntu/Downloads/hw1.m
for help type /h
Miranda say_hello "Timur Khaimovich"
Hello, Timur Khaimovich
Miranda factorial 7
5040
```

Prolog – Logic Language

After 4 hours of suffering with [Mozart](#), I had given up and picked up [Prolog](#).

A terminal window titled 'ubuntu@ubuntu2004: ~/Documents/plt/w2/prolog'. The prompt is 'ubuntu@ubuntu2004:~/Documents/plt/w2/prolog\$'. The user enters 'cat hw1.pl', and the terminal displays the contents of the file: ':- initialization hello_world, halt.' followed by 'hello_world :-' and an indented line 'write('Hello, Timur Khaimovich!'), nl.'. The user then enters 'swipl -q -l hw1.pl', and the terminal outputs 'Hello, Timur Khaimovich!'. The prompt returns to 'ubuntu@ubuntu2004:~/Documents/plt/w2/prolog\$' with a cursor.

```
ubuntu@ubuntu2004: ~/Documents/plt/w2/prolog
ubuntu@ubuntu2004:~/Documents/plt/w2/prolog$ cat hw1.pl
:- initialization hello_world, halt.

hello_world :-
    write('Hello, Timur Khaimovich!'), nl.
ubuntu@ubuntu2004:~/Documents/plt/w2/prolog$ swipl -q -l hw1.pl
Hello, Timur Khaimovich!
ubuntu@ubuntu2004:~/Documents/plt/w2/prolog$
```

Side Note

I tried to make use of the language called Mozart which is an implementation of Oz. This was quite painful experience and I do not suggest you to try it. Firstly, the best and maybe the only way to write and execute your programs is by using Emacs, an ancient IDE, but this is not the main problem of the Mozart. The syntax is VERY inconvenient and it has lots of limitations.