#2 Assignment - Concurrency - CMPT 383

Luiz Fernando Peres de Oliveira - 301288301 - lperesde@sfu.ca

I. Programming Languages

Actor/Message Passing/OOP - Erlang

Erlang is designed for concurrency and is used in many big projects whose main concerns are related to performing asynchronous tasks, such as Facebook Messenger. Erlang is a pure functional language and features single assignment and eager evaluation. In its actor model, each object is an actor. Messages can be exchanged among actors, which will be buffered in the object's mailbox. Upon receiving a message, an action will be taken by the actor. Messages are not guaranteed to be delivered in the same order as they were sent, however they are guaranteed to be always delivered.

Concurrent programming in **Erlang** is built-in is done by sending asynchronous messages with and identifier so called *Pid*(process identifier). The caller is able to do that by executing a command responsible by spawning a new *Pid*: $pid = spawn(f)$, where f is a function. See the example below:

```
start() ->
    spawn(fun() -> sendMessage("Hi") end).

sendMessage(Message) ->
    io:fwrite("~p",[Message]).
```

The code above will call $spaw$ sending a new concurrent process that evaluates fun. The new process runs in parallel with the caller and prints the message *"Hi!"*.

Similarly, to receive a message passed by an *spawned* function f, one will use the keyword $receive$ and pattern match the input. See below:

```
spawnedFn() ->
    receive
      {X, Y} ->
        io:fwrite("X + Y is: ~p~n",[X + Y]),
        spawnedFn();
    Other ->
        io:fwrite("Unknown"),
        spawnedFn()
    end.

start() ->
    pid = spawn(fun() -> spawnedFn() end),
    pid ! {6, 10}.
```

In the code snippet above, the output will be 16, because $6, 10$ matches with the pattern x, y.

Functional Programming - Haskell
Object-oriented Programming - Java
Scripting Language - Python
Procedural - C

References

[1] "Erlang – Concurrent Programming", Erlang.org, 2017. [Online]. Available: http://erlang.org/doc/getting_started/conc_prog.html. [Accessed: 09- Oct- 2017].

[2] "The Hitchhiker's Guide to Concurrency — Learn You Some Erlang for Great Good!", Learnyousomeerlang.com, 2017. [Online]. Available: http://learnyousomeerlang.com/the-hitchhikers-guide-to-concurrency. [Accessed: 09- Oct- 2017].

[3] A. Miller, "Understanding actor concurrency, Part 1: Actors in Erlang", JavaWorld, 2017. [Online]. Available: https://www.javaworld.com/article/2077999/java-concurrency/understanding-actor-concurrency–part-1–actors-in-erlang.html. [Accessed: 09- Oct- 2017].