Christian-Albrechts-Universität zu Kiel
Institut für Informatik

Priv.-Doz. Dr. Frank Huch, Marius Rasch

Institut für Informatik

7. Übung zur Vorlesung „Concurrent and Distributed Programming"

Abgabe am Monday, 27. May 2019 - 18:00

Aufgabe 1 - Turing machine 1 Punkt

Implement a Turing machine by using a tail-recursive Erlang program (after the recursiv call no other calculation must follow). Your implementation shall only use Pids and a finite number of atoms, but no lists or nested tuples. Represent the tape of the Turing machine by concurrent server processes, holding the current letter and the pids of there adjacent processes. Another control process represents the finite control of the Turing machine, which communicates with the tape process that is currently at the reading/writing position. Use the implementation from the last exercise to test this implementation.

Aufgabe 2 - Stacks by using the mailbox 1 Punkt

Implement a stack by using the mailbox of a process. Provide `pop`, `push` and `newStack` as interface functions. Only use two processes and finite data types and do not use the function stack to implement the tape of the Turing machine. Change the Turing maschine from the lecture in a way, that is uses your stack implementation.

Aufgabe 3 - Tupel server in Erlang 1 Punkt

Extend the tuple server from the lecture by the functions `in/3`, `out/3` and `rd/3`, accepting a timeout as third parameter. For reading operations the timeout represents the maximal duration of the attempt to read. For `out/3` this timeout corresponds to the maximal lifespan of the tuple in the tuple space.

Try to avoid additional computational cost for the tuple server.

An implentation for `in/3` like the one below runs into a problem, when a `{found,Res}` is send after the timeout happens. The result might be seen as result for the next request this client does. Think about a way to solve this problem.

```
in(TS,P,Timeout) -> TS!{in,P,self(),{just,Timeout}},
  receive
    {found,Res} -> {just,Res}
  after Timeout -> nothing
  end.
```