Formal Models for Parallel and Distributed Systems

Exercise 1 (April 27)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

March 24, 2011

The exercise is to be submitted by the deadline stated above via the Moodle interface as a single .zip or .tgz file containing

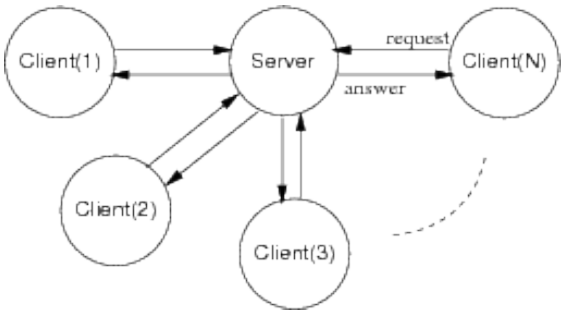
1. a single PDF file with a decent cover page (mentioning the title of the course, your full name and Matrikelnummer) with
   - nicely formatted listings of the commented model/configuration files and
   - the output of the TLA+ model checker runs (if run from the command line) and/or screenshots of the corresponding window (if run from the toolbox),
   - an explicit interpretation of the results (does the modelcheck highlight an error or not, how can the result be explained):
2. all .tla and (if used, also .cfg) files used in the exercise.

1 A Client/Server System

Write a TLA+ model of a distributed system of a server and N clients where the server maintains a shared resource which it grants to at most one of the clients at a time, as depicted by the following pseudo-code:

```
Server:
  local given, waiting, sender
begin
  given := <none>; waiting := {}
  loop
    sender := receiveRequest()
    if sender = given then
      if waiting = {} then
        given := <none>
      else
        choose given from waiting;
        waiting := waiting \ { given };
        sendAnswer(given)
      endif
    elsif given = <none> then
      given := sender
      sendAnswer(given)
    else
      waiting :=
        waiting U {sender}
    endif
  endloop
end Server
```

```
Client(p):
  param ident
begin
  loop
    ...
    c1: sendRequest()
    c2: receiveAnswer()
    ... // critical region
    c3: sendRequest()
  endloop
end Client
```



The system operates on the variables *given*, *waiting*, *sender* (describing the internal state of the server component), the variables *pc(p)* (the "program counter" of client p) and four buffered channels *request1*, *request2*, *answer1*, and *answer2* holding the requests sent from each client to the server respectively the answers returned from the server.

The system can be modeled by four server actions (corresponding to the combined effect of each conditional branch in above pseudo-code) and, for each client, three client actions (corresponding to the transition from one labeled command to the next ($c_1 \to c_2$, $c_2 \to c_3$, $c_3 \to c_1$). The "send/receive" calls in above pseudo-code mark the use of the the respective channels.

The overall structure of the system's next state relation is therefore

$$\begin{array}{l} \mathit{Next} \equiv \\ \quad \mathit{ServerAction} \lor \\ \quad \exists p \in \{p_1, \ldots, p_N\} : \mathit{ClientAction}(p) \lor \mathit{ClientServerCommunication}(p) \end{array}$$

where each action is a disjunction of some basic actions. Write your specification as a single module with parameterized state variables ($pc(p)$) and actions ($\mathit{ClientAction}(p)$).

Model-check your TLA+ specification for $N = 3$ clients and with respect to the following correctness properties:

- *Type correctness:* all state variables maintain reasonable types.
- *Mutual exclusion:* it is never the case that two processes are at the same time in the criticial region (at position c_3).

On the virtual machine, the TLA+ command line tools can be invoked by the commands `tla2tex`, `sany`, and `tlc`, e.g. the command "`tlc <ModelName>`" expands to

```
java -cp /software/tla tlc2.TLC <ModelName>
```

Likewise, the command `toolbox` starts the graphical TLA+ toolbox.

See the installation directory of TLA+ (e.g. `/software/tla/examples`) for numerous TLA+ examples, the "Documentation" section of the TLA+ toolbox web site and the online version of Lamport's book (especially chapters 14 and 15) for more information on the syntax of TLA+ and on the use of the model checker.