Formal Models for Parallel and Distributed Systems Exercise 1 (April 27)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.uni-linz.ac.at

March 15, 2009

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

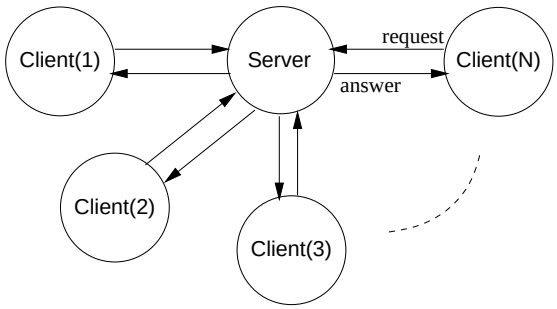
1. a PDF file with a decent cover page (mentioning the title of the course, your full name and Matrikelnummer) with
   - listings of the model/configuration files and
   - the output of the TLC model checker runs,
2. all .tla and .cfg files used in the specification and model checking.

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 := 0; waiting := {}
  loop
    sender := receiveRequest()
    if sender = given then
      if waiting = {} then
        given := 0
      else
        choose given from waiting;
        waiting := waiting \ { given };
        sendAnswer(given)
      endif
    elsif given = 0 then
      given := sender
      sendAnswer(given)
    else
      waiting :=
        waiting U {sender}
    endif
  endloop
end Server
```

```
Client(p):
  param ident
begin
  loop
    ...
    1: sendRequest()
    2: receiveAnswer()
    ... // critical region
    3: 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 ($1 \to 2$, $2 \to 3$, $3 \to 1$). The "send/receive" calls in above pseudo-code mark the writing to/reading from the respective channels.

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

$$
\begin{array}{l}
\mathit{Next} \;\equiv \\
\quad \mathit{ServerAction} \lor \\
\quad \exists p \in \{1 \ldots N\} : \mathit{ClientAction}(p) \lor \mathit{ClientServerCommunication}(p)
\end{array}
$$

where each action is a disjunction of some basic actions. Since the model checker of TLA+ version 1 does not support instantiation, 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 channels of buffer size $B = 1$ 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.

On the virtual machine, the TLC model checker can be invoked by the command `tlc <ModelName>` which expands to

```
/software/java6/bin/java -cp /software/tla tlc.TLC <ModelName>
```

See the installation directory of TLA+ (e.g. `/software/tla/examples`) for numerous TLA+ examples 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.