Formal Models for Parallel and Distributed Systems

Exercise 1 (May 20)

Wolfgang Schreiner
Wolfgang.Schreiner@risc.jku.at

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 (screenshots of the relevant toolbox windows),
   - an explicit justification of the chosen fairness properties,
   - 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.

A Seat Reservation System

Write a TLA+ model of the following distributed system for booking the seats of a theater:

- There is a central server with a set *free* of seats that are are ready to be booked; furthermore there are multiple client terminals each of which has a local copy of *free* and a set *sold* of seats that have been booked by some customer on this terminal.
- When a customer arrives at a terminal, she may select some of the seats from the local *free* set; the client then sends a "reservation" message with the selected seats to the server. If all selected seats are also in the *free* set of the server, the server removes them from its *free* set and returns a "reserved" message with the reserved seats; if not all requested seats are free (because in the mean time another client has reserved the seats), the server leaves its set unchanged and returns a "notreserved" message to the client. Both kinds of messages transfer as an additional payload the server's version of the *free* set to the client which updates its local copy correspondingly.
- If a client receives a "reserved" message, the customer may decide to ultimately book the reserved seats or to abort the reservation. In the first case, the seats are moved into the local set *sold* of the client; in the second case, the seats are returned by an "abort" message to the server which returns them to its *free* set.
- From time to time the server sends "update" messages to all clients to inform them about the current *free* set of the server.

To limit the state space, you may model communication by an "instantenous" transfer of messages between the message buffers of the communication partners (i.e., without modeling the intermediate network): the sender writes a message into a local variable from which the communication subsystem can transfer it in a single step into a corresponding variable in the receiver.

Model-check your specification for a reasonable numbers of seats and clients (start with two seats and two clients and then try to increase as much as possible to yield reasonable checking times) with respect to the following correctness properties:

- *Type correctness:* all state variables maintain the expected types.
- *Seats are not doubly booked:* a seat never appears in the *sold* sets of two different clients.
- *Seats are not lost:* every seat is either in some *sold* set or in the *free* set of the server or in the payload of some message (or in some other variable of your model).
- *Reservations are not permanent:* every seat that is not in the *free* set of the server ultimately arrives in the *sold* set of some client or returns to the *free* set of the server.
- *Booked seats disappear:* every seat that is in the *sold* set of some client eventually disappears from the *free* set of every client.
- *Not booked seats appear:* every seat that is not in the *sold* set of any client eventually appears in the *free* set of every client.
- *Seats are permanently booked:* once a seat arrives in the *sold* set of a client, it stays there forever.
- *All seats are booked:* every seat eventually arrives in the *sold* set of some client.

Interpret the results of the model checks and whether they indicate an error in your model or not (because the specified property should actually not be expected to be true). Make sure, however, that you choose reasonable fairness properties to ensure appropriate system progress and justify your choice.