Formal Models for Parallel and Distributed Systems

Exercise 2 (May 26, 2025)

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.

TLA$^+$ Model of a Seat Booking 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 sold; furthermore there are multiple client terminals each of which has a local copy of *free* and a set *sold* of seats that have on this terminal been sold to some customer.
- When a customer arrives at a terminal, she may select some of the seats from the local *free* set; the client then sends a "book" 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 "booked" message with the reserved seats; if not all requested seats are free (because another client has reserved some of the seats), the server leaves its set unchanged and returns a "notbooked" message.
- If a client receives a "booked" message, the customer may buy the booked seats or abort the purchase. 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 a "return" message to the server which adds them to its *free* set.
- From time to time, the server sends an "update" message to every client to inform it about the current *free* set of the server.

To limit the state space, model the network by two variables *server* and *client* each of which may be "empty" or hold a single message to the server respectively to some client (together with the number of the client that sent the message or is to receive it); the sender of a message has to wait until the corresponding variable is empty before it can fill it.

Model-check your specification for a small 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 sold:* a seat never appears in the *sold* sets of two different clients.
- *Seats are permanently sold:* once a seat arrives in the *sold* set of a client, it stays there forever.
- *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.
- *Sold seats disappear:* every seat that is in the *sold* set of some client eventually disappears from the *free* set of every client.
- *Not sold seats appear:* every seat that is not in the *sold* set of any client eventually appears in the *free* set of every client.
- *All seats are sold:* 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.