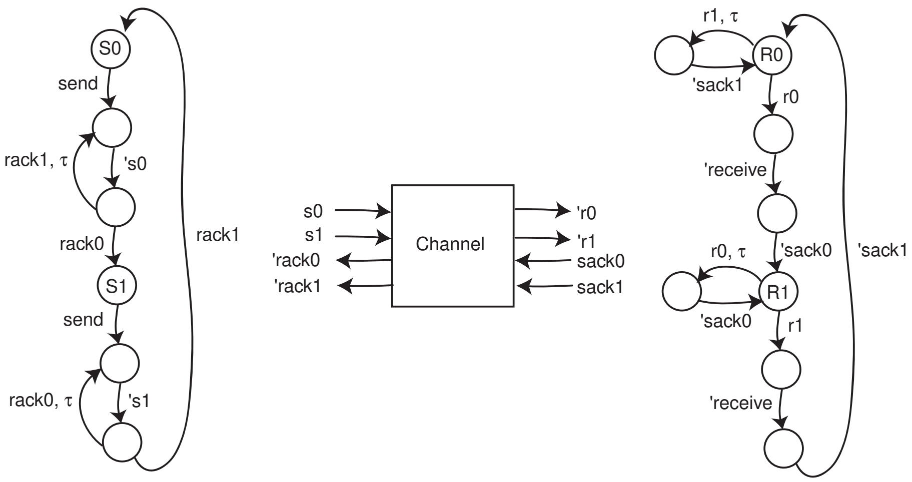
Calculi of Concurrency
Homework Problems 2

The **deadline** is Friday, June 2. Please deliver your **individual** solutions to Ulf. To collect credits you have (1) to solve all exercises, (2) to attend the homework session on June 6, and to be ready to present your solutions at this session.

1. The *alternating bit protocol* is a well-known communication protocol, based on the sliding window technique, which is intended to provide reliable transfer on a lossy channel. The sending and receiving of messages alternate between two modes (0 and 1) as illustrated below:



When receiving an external request to send a message (*send*) the sender S0 first sends the message, together with its current mode (*s0*). The sender then awaits an acknowledgment from the receiver. If the receiver acknowledges reception in mode 0 (*rack0*) the sender switches to mode 1 and behaves in a similar way (but in the alternative mode). If the receiver acknowledges reception in the opposite mode (*rack1*) the sender re-transmits *s0*; the sender may also time-out (represented by a reaction), in which case *s0* is also re-transmitted.

The receiver R0 either receives the message *r0*, the message *r1* or times out. In the last two cases the receiver sends the acknowledgment *sack1* (corresponding to an error message). In the first case the receiver informs the environment that it has received a message (*receive*) and sends an acknowledgment (*sack0*) followed by a switch to mode 1 etc.

We consider two systems $(S0|C1|R0)$ and $(S0|C2|R0)$ consisting of one sender, one receiver and one communication channel. The former uses a channel C1 capable of transmitting one message at a time, never losing any messages. The latter uses a lossy channel C2 which behaves like C1 but may in addition fail to forward a message (including acknowledgments). In both cases all interaction with the channel is unobservable.

The CWB code for both systems can be found at the home page. Try to understand how it works and then use the modal μ-calculus (and/or CTL) to check whether the systems satisfy the following properties:

- Once the sender obtains a request to dispatch a message (action *send*) the receiver is always capable of outputing a message (action *receive*).
- Messages sent (by the sender) are always eventually received by the receiver.
- After the sender dispatches a message it is not possible to send a new (different) message (unless the first message has been received by the receiver).
- It is impossible to receive two messages without an intermediate *send*.
- The systems can not deadlock.

Try to explain the outcomes. Is it what you expected? If not, can you explain the results?

What if we want to verify that whatever happens it is always possible to return to a configuration where the sender is in state S0 and the receiver is in state R0? Is it possible, or is it possible to modify the system to facilitate this?

2. Design a process P with observable actions a, b, c plus reactions satisfying the following properties expressed in the modal μ-calculus and CTL:

   - $P \models \mathrm{AG}\ \langle - \rangle \mathrm{tt}$
   - $P \models [a]\langle b \rangle \mathrm{tt}$
   - $P \models \langle a \rangle \langle \tau \rangle \langle a \rangle \mathrm{tt}$
   - $P \models \mu X.(\langle c \rangle \mathrm{tt} \vee (\langle a \rangle \mathrm{tt} \wedge \langle\langle - \rangle\rangle X))$
   - $P \models \langle - \rangle \mathrm{tt} \wedge [-a]\mathrm{ff}$
   - $P \models \nu X.(\mu Y.(\langle a \rangle \mathrm{tt} \vee \langle - \rangle Y) \wedge [-]X)$

   Explain in natural language what the properties mean. Draw the automaton corresponding to P. (Try to make the automaton as small as possible.)

3. A system consists of senders S_i, $i = 1, ..., m$, receivers R_j, $j = 1, ..., n$ and a switch that cyclically establishes one-to-one connections between arbitrary senders and receivers. At most k sender-receiver pairs can be connected simultaneously. When connection between S_i and R_j is established S_i sends a specific signal s_i received by R_j. Then R_j sends a confirmation r_j to the switch and the connection terminates.

- Describe this system in the language of π-calculus, using recursive definitions. Explain your definition and draw flow graphs for sample states of the system.
- Eliminate recursive definitions from your description by using replicators.

4. Summarize (at most one A4 printed page) the main points of the "strategic directions in concurrency research" article:

   `ftp://ftp.cs.sunysb.edu/pub/TechReports/smolka/sdcr.ps.Z`

   Taking into account this article and the experience of this course what in your opinion should be included in a 4 points course on concurrency, provided that only one such course is given in the Ph.D. curriculum. The article makes a suggestion for such a course, but we would like to know **your** opinion, related to the contents of **this** course.