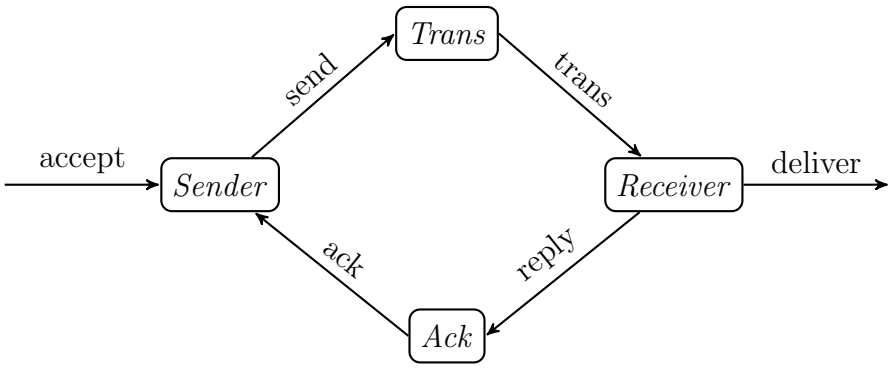
RIO 2023 Summer School of Informatics
Course 2: Modeling and Analyzing Concurrent Systems

Thomas Noll, RWTH Aachen University, Germany, rio2023@i2.informatik.rwth-aachen.de

Homework: The Alternating-Bit Protocol

The Protocol In this homework you are asked to model the *Alternating-Bit Protocol (ABP)* in the CCS language and verify it using the CAAL tool[1]. ABP is a simple yet effective protocol for reliably transmitting data from a sender to a receiver over unreliable communication channels. This is accomplished by retransmitting lost messages if required. An overview of the overall system is given in the following diagram.



ABP works as follows.

- *Sender* transfers data to *Receiver* via transport channel *Trans*, and *Receiver* confirms reception via acknowledgment channel *Ack*;
- Channels are unidirectional and can transfer only one message each time. Channels are not perfect, i.e., messages can be lost. However transmission errors are detected.
- After accepting data, *Sender* sends it item via *Trans* with a control bit b (which is initially set to 0) repeatedly until the acknowledgment b is received over *Ack*. For the next data item, control bit $1 - b$ is used, and so on. (That is where the name "*Alternating-Bit* Protocol" comes from.)
- *Receiver* gets the data item with a control bit b or the information that the data is corrupted. In the first case, if b is the expected value of the control bit (which is initially set to 0 also on the receiving side), b is returned via *Ack*. Otherwise, the transmission is re-initiated by returning the "wrong" control bit $1 - b$ to Sender.
- When a message is received for the first time, *Receiver* delivers it, while subsequent messages with the same control bit are simply acknowledged.
- When *Receiver* receives an acknowledgment containing the same bit as the message it is currently transmitting, it stops transmitting that message, flips the protocol bit, and repeats the protocol for the next message.

[1] `https://caal.cs.aau.dk`

There is no direct communication between *Sender* and *Receiver*; all messages must travel through the channels. The only actions visible to the environment are accept and deliver steps on the *Sender* and *Receiver* side; everything else is considered to be internal communication.

Your Tasks Your tasks are as follows:

(1) Implement ABP as a CCS process definition in the CAAL tool. You can abstract away from the content of the messages and focus only on the additional control bit. To model the decision when the sender retransmits the message, use either non-determinism or, even better, a special process called *Timer*. This process will communicate with the sender using an action called *timeout*, and will signal when a message should be retransmitted.

(2) Suggest a specification of the protocol in CCS and check whether it is equivalent to your implementation by checking a suitable form of bisimulation supported by the CAAL tool. If interested, you might moreover want to add the following degrees of (un-)reliability and answer this question for all of these extensions:

   (a) lossy channels (received messages can be lost without any warning) and

   (b) lossy and duplicating channels (in addition, the transported message can possibly be delivered several times).

(3) Check for possible deadlocks (stuck configurations) and livelocks (a possibility of an infinite sequence of τ-labelled transitions) by formulating the properties as recursive formulae in Hennessy-Milner Logic, and by verifying whether the implementation satisfies these formulae.

Your Deliverables Write a short report, which must be in PDF format and should contain:

- Full names and e-mail addresses of all people that worked in your team (at most two students per team),
- your commented implementation and specification of the protocol (just copy/paste it directly into the report),
- a short conclusion about verification of the protocol using bisimulation checking (including lossy and duplicating channels, if applicable) – what equivalence did you choose and why?, and
- formulae in CAAL syntax for deadlock and livelock and the information whether your implementation does or does not satisfy these formulae.

The report does not have to be long at all but it should contain (at least to some extent) all the points mentioned above. Send the report together with the CAAL project file containing the implementation and specification of the protocol to rio2023@i2.informatik.rwth-aachen.de, no later than March 31, 2023.