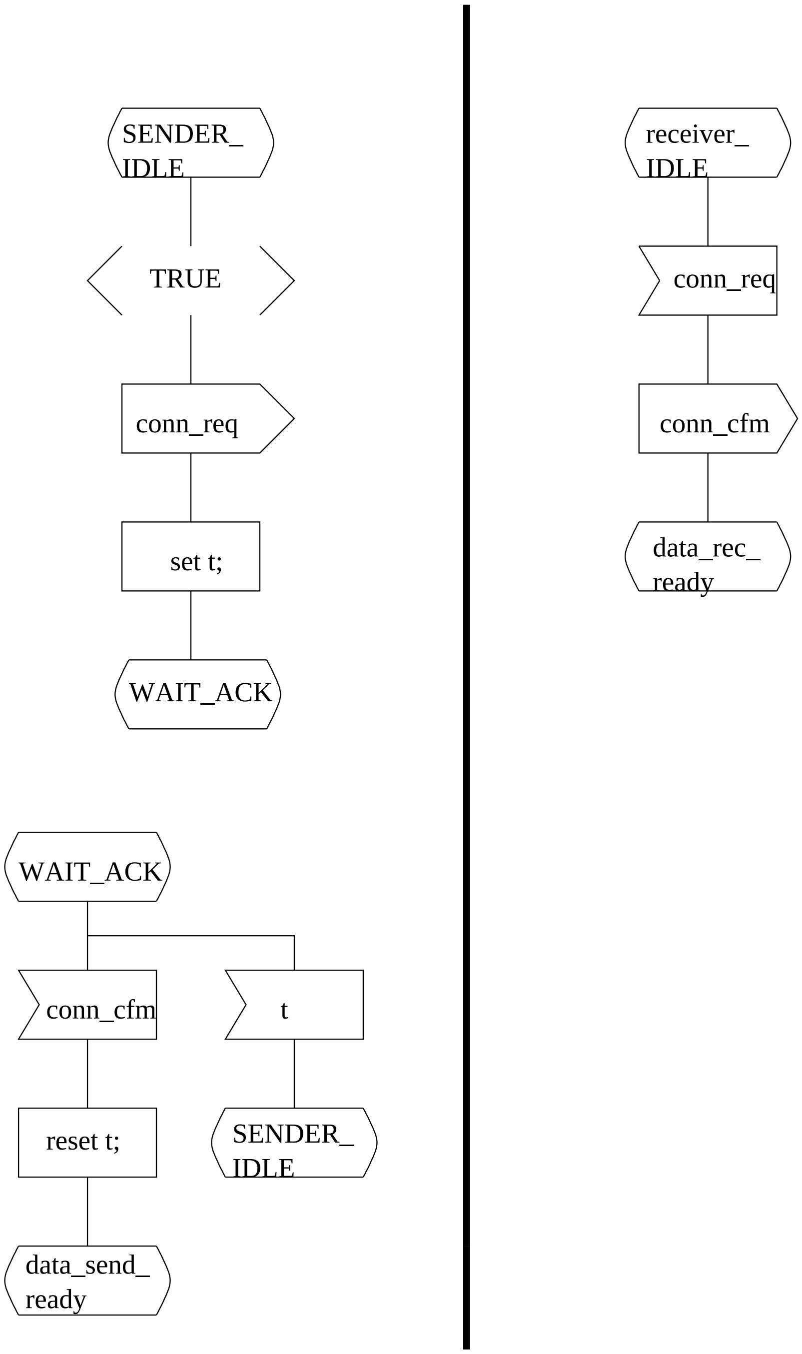
How to convert an SDL specification to CCS Process Algebra



A simple communication protocol

The CCS description is as follows:

$$
\begin{aligned}
Sender &= \overline{CONN\_REQ\_S}.Sender\_wait\_ack \\
Sender\_wait\_ack &= Sender + CONN\_CFM.Sender\_data\_send\_ready \\
Receiver &= CONN\_REQ.\overline{CONN\_CFM\_R}.Receiver\_data\_send\_ready \\
Forward\_q &= CONN\_REQ\_S.Forward\_q + \\
& CONN\_REQ\_S.\overline{CONN\_REQ}.Forward\_q \\
Ack\_q &= CONN\_CFM\_R.Ack\_q + \\
& CONN\_CFM\_R.\overline{CONN\_CFM}.Ack\_q \\
Protocol &= (Sender \parallel Forward\_q \parallel Receiver \parallel Ack\_q)
\end{aligned}
$$

The signals are written in *UPPER CASE*. Output signals have a line drawn over them, as in $\overline{CONN\_REQ\_S}$. The lossy channel is modelled by the agents *Forward_ q* and *Ack_ q*.

A few abstractions have been made in the model. The timer is not explicitly presented. The possibility of timer expiring is modelled as choice in the agent *Sender* where the agent can either return to its initial state (Sender) or to state Sender_data_send_ready.

The agent *Forward_ q* represents the queue of the Receiver process and the *Ack_ q* the queue of the Sender process. The processes themselves are represented as agents *Sender* and *Receiver*. When a signal arrives to a queue, it will either pass it to the corresponding process or lose it.

The agents *Sender* and *Receiver* model the processes. The conversion is in this case quite straightforward. The elements of SDL transitions are modelled as actions. The SDL states are not explicitly modelled, but the agent modelling the Sender process is defined in two parts, first the actions leading to SDL state Sender_wait_ack and then the actions possible from that state. The Receiver process is so simple that no such division was necessary for it.

If the SDL description had variables, the conversion would become more complex. We would model the variables as agents, but possibly the values of variables would have to be restricted very heavily.