Let $P$ and $Q$ be transition systems and $a$ an action. There are three ways to model non-determinism in transition systems:
In the case 1 the environment has a chance to interact with the system, if $a$ happens before the internal action $\tau$. In the case 2, the system itself decides, at the same time when the environment reacts, which one of $a$’s it chooses. The case 3 says that the system decides internally, if it behaves according to $P$ or $Q$.

For example when modelling a communication channel it is best to choose the alternative 3, if $P$ represents the delivery of a message and $Q$ the loss of a message. The environment, in this case the sender and receiver, cannot affect the behaviour of the channel. The delivery or loss of a packet is only dependent on the channel. If we used the alternative 1, the environment could in some cases affect the behaviour of the channel, what could not lead to realistic modelling.

The second example is the seat reservation system of airlines, whose one feature is described by the following transition system:
Thus the result of a seat reservation is totally unpredictable from the point of view of a customer, because a normal customer does not know, how the system is built or are there free seats available. The incapability of a customer to affect the system is modelled using internal transitions.
Communication in the computer networks is not usually synchronized. The parallel operator, however, demands synchronization.

It is possible to produce asynchronous communication by specifying a channel as a transition system. A process sends a message into the channel synchronously and continues then its computation. When it is ready, another process takes the message from the channel synchronously.

In this way there is an asynchronous communication between the two processes. For this asynchronous communication one has to pay a price: the size of the global state increases, sometimes considerably. This increase depends on the amount of packets which can travel simultaneously in the channel.
Sometimes it is also necessary to model the environment of a protocol. For example, it is possible to assume in the AB-protocol that the sender gets the data in get+ message from the environment, i.e. from the upper layer or other process, and that the receiver gives the data in give+ message to its own environment.

We model now the AB-protocol in a more orthodox way with the help of channels and environments. In addition, the timer is now a separate process. We make the following agreements:

- The sender sends the messages $d0$ and $d1$ into the channel.
- The receiver takes the messages $dd0$ and $dd1$ from the channel.
- The receiver sends the acknowledgements $a0$ and $a1$ into the channel.
The sender takes the acknowledgements aa0 and aa1 from the channel.

The sender and environment communicate synchronously with the message get+.

The receiver and environment communicate synchronously with a message give+.

The sender sets the timer synchronously with the help of a message setT+.

The timer informs the sender about the timeout synchronously with the help of a message timeout+.

The sender informs the timer synchronously with a message reset+ that the timer can return to its initial state.
Below the processes are represented as transition systems. It is assumed of the channel that it can contain only one message at a time. Furthermore, it is the task of the channel to lose or distort messages.
There are some new transitions in the sender process (states S2 and S7), where acknowledgements are received. This is because of the channel and the timer which may send timeout even if an acknowledgement is in the channel. It is necessary to take all the messages from the channel so that the channel can send new messages. If some message stayed in the channel, it would block all the traffic and the result would be a deadlock.
The timer can be represented with the help of two states.
The channel has been modelled in such a way that the environment cannot affect if messages disappear or not. It is thought that the channel takes the data messages $d_0$ and $d_1$ from the sender and delivers them as messages $dd_0$ and $dd_1$. Notice that it is not possible to take and deliver exactly the same message forms, because this would disturb the asynchronous communication. The same is true with the acknowledgements: $a_0$ and $a_1$ are taken by the channel and $aa_0$ and $aa_1$ are delivered.
Channels and Environments IX

\[
A'B = ((S; [[\text{timeout, reset, setT}]]; T)
\]

\[
[[d_0,d_1,aa_0,aa_1]]
\]

C)

\[
[[dd_0,dd_1,a_0,a_1]]
\]

R
The global state graph is now a combination of four processes. It is essentially more complicated than the earlier global state graph and that is why it is not wise to draw it completely manually. The start of the graph is seen below.