Examples of Supervisory Control

Example 38. In a transmitter you have (at least) four events:

r. Reception of the message to be transmitted
s. Start of transmission
t. Timeout
a. Acknowledgement

Which events do you think are controllable and which are not? Build an automaton model for a transmitter. New messages received during transmission are discarded, and messages should be sent again after completion of a message.
Automaton model for transmitter $i$
The process T consists of two transmitters, T1 and T2. The transmitters send on the same channel that can be used by only one transmitter at a time. We have three different specifications given below, that try to supervise the use of the channel.

Can you build an non-blocking and controllable supervisor based on the specifications? Are all three ok?
Spec1 does not allow restart of transmissions, so it will cause blocking if we get a timeout. Spec3 allows start of another transmission if the other get a timeout, which might not be fair, and it could in principle cause a message to jam forever.
Example 39. A manufacturing cell consists of two machines $M_1$ and $M_2$ with a buffer $B$ in between them. Output from $M_1$ goes to $B$, while $M_2$ takes work parts from $B$. Raw-material comes from an infinite resource and output from $M_2$ goes to an infinite storage. Model the machines and the buffer, and take into account that the machines can break down and get repaired. Which events do you think are controllable and which are not? Furthermore, we have the following specifications

1. The buffer can only store one work piece. How would the model change if the buffer could store $n$ pieces?

2. Only one machine can be repaired at a time.

3. If both machines are broken, $M_2$ is repaired first.

4. $M_1$ may not start if $M_2$ is broken.
We can neither control when the machines break down, when the job is done, nor when the repair is done. Start of machines and start of repairs are controllable. For each additional space in the buffer we need a separate state; it is a weak spot of automata.
Specifications:
One repair at a time
Repair $M_2$ first
Start $M_1$ only if $M_2$ works

The parallel composition of the models and the specifications is directly a non-blocking and controllable supervisor, with 30 states.

If we allow breakdown when the machines are idle it does not change the above statement.

This is so if buffer can stop d1 when full. But if it is not so the buffer will be a specification and not a plant, and you will get 4 uncontrollable states where machine 1 is working and buffer is full. These states can be forbidden and removed, which means that you do not allow s1 before buffer is empty.

If machine breakdown is not allowed at idle state you will, as a result of removing the uncontrollable states, get 7 non-reachable states. It will be impossible to reach machine 1 broken and buffer full.