Algorithms for Supervisory Control

We basically have four requirements for the supervisor:

1. The specifications

2. The uncontrollable states must be forbidden

3. It must be non-blocking

4. It must be controllable

There is no known way of satisfying the last two in one step, we need to iterate.

Making the supervisor non-blocking can make it uncontrollable

Making the supervisor controllable can make it blocking.
Algorithm for supervisor synthesis

1. $S_0 = \textit{Compare}(G||Sp, G); i = 0$, forbid uncontrollable states.

2. $S_{i+1} = \textit{NonBlock}(S_i)$, search for blocking states and forbid them.

3. $S_{i+2} = \textit{Contrl}(S_{i+1})$, search for states with uncontrollable transitions to forbidden states and forbid them.

4. If $S_{i+1} \neq S_{i+2}$ then $i = i + 2$ and go to 2

5. $S = S_{i+1}$

Forbidding overrides marking in all steps.

$S'$ is non-blocking, controllable, fulfills the specification $Sp$, with minimal restrictions.

$S'$ can be an empty automaton $\Rightarrow$ too restrictive specs, or incorrect model of plant, or incorrectly modeled specs.
Algorithm for non-blocking, \( NonBlock(A) \)

**What we assume given:**
- \( Q_A \) = set of states in \( A \)
- \( M_A \) = set of marked states
- \( X_A \) = set of forbidden states
- \( \delta_A(q, \sigma) \) = transition function, depends on state \( q \) and event \( \sigma \)

**What we get:**
- \( Q_{co} \) = set of co-accessible states,
- \( Q_x \) = set of blocking + previously forbidden states

**What we do:** We search for all states from where we get to a marked state, states that are co-accessible. All the other states are blocking.

1. \( Q_{co} = M_A \); \( Q_x = X_A \); \( Q = \emptyset \)

2. For each \( q \in Q_A \setminus (Q_{co} \cup Q_x) \)
   - If \( \exists \sigma \in \Sigma_A : \delta(q, \sigma) \in Q_{co} \) then \( Q = Q \cup \{q\} \)
   - else if \( \forall \sigma \in \Sigma_A : \delta(q, \sigma) \in Q_X \) or \( \delta(q, \sigma) \) undefined then \( Q_X = Q_X \cup \{q\} \)

3. If \( Q \neq \emptyset \) then \( Q_{co} = Q_{co} \cup Q \); \( Q = \emptyset \); go to 2.

4. \( X_A = Q_A \setminus Q_{co} \)

Increases only \( X_A \), the not forbidden part of \( A \) is non-blocking, terminates.
Example 35. Apply NonBlock on the two automata given below

\[ A_1 \]

\[ A_2 \]
Solution to Example 35

$A_1$

step 1: $Q_{co} = \{q3\}$, $Q_x = \emptyset$ and $Q = \emptyset$

Step 2, if: $Q_{A1} \setminus (Q_{co} \cup Q_x) = \{q0, q1, q2, q3\}$ are tested for reaching into $Q_{co}$, $q1$ and $q2$ pass this test, event $c$ does the job, $Q = \{q1, q2\}$

Step 2 else if: $\{q0, q4\}$ are tested for if there exist any executable event that do not take us to a forbidden state. At $q0$ we can execute $a$ and $b$, but at $q4$ we are stuck. $Q_x = \{q4\}$

Step 3 $Q_{co} = Q_{co} \cup Q = \{q1, q2, q3\}$, $Q = \emptyset$

Step 2, if: $Q_{A1} \setminus (Q_{co} \cup Q_x) = \{q0\}$ for reaching into $Q_{co}$, $q0$ pass this test, both and $b$ does the job, $Q = \{q0\}$ Step 2, else if: nothing more to do

Step 3: $Q_{co} = Q_{co} \cup Q = \{q0, q1, q2, q3\}$, $Q = \emptyset$

Step 2: $Q_{A1} \setminus (Q_{co} \cup Q_x) = \emptyset$, nothing more to do

Step 3: We are done

Step 4: $X_{A1} = Q_{A1} \setminus Q_{co} = \{q4\}$

$A_2$

Very much similar to $A_1$, but we never have to test $q2$ as it is forbidden from the start... $X_{A2} = \{q2, q4\}$
**Algorithm for controllability,** \( Contrl(A) \)

**What we assume given:**
- \( Q_A \) = set of states in \( A \)
- \( M_A \) = set of marked states
- \( X_A \) = set of forbidden states
- \( \delta_A(q, \sigma) \) = transition function, depends on state \( q \) and event \( \sigma \)
- \( \Sigma_{uc} \) = set of uncontrollable events

**What we get:**
- \( Q_x \) = set of states with uncontrollable transitions to forbidden states + previously forb. states.

**What we do:** We search for all states that has an uncontrollable transition to a forbidden state.

1. \( Q_x = X_A ; Q = \emptyset \)

2. For each \( q \in Q_A \setminus Q_x \)
   - If \( \exists \sigma_{uc} \in \Sigma_{uc} : \delta(q, \sigma_{uc}) \in Q_x \) then \( Q = Q \cup \{q\} \)

3. If \( Q \neq \emptyset \) then \( Q_x = Q_x \cup Q ; Q = \emptyset \); go to 2.

4. \( X_A = Q_x \)

Increases only \( X_A \), the not forbidden part of \( A \) is controllable, terminates.
Example 36. Apply $Contrl$ on the automaton given below. The event $c$ is controllable and $uc$ is uncontrollable.
Solution to Example 36

step 1: $Q_x = \{q^7\}$ and $Q = \emptyset$

Step 2: $Q_A \setminus Q_x = \{q_0, q_1, q_2, q_3, q_4, q_5, q_6, q_8\}$ are tested for reaching through an uncontrollable event, uc in this case, into $Q_x$. This is so only for $q_5$, $Q = \{q_5\}$.

Step 3: $Q_x = \{q_5, q^7\}, Q = \emptyset$

Step 2: $Q_A \setminus Q_x = \{q_0, q_1, q_2, q_3, q_4, q_6, q_8\}$ are tested, $Q = \{q_3\}$.

Step 3: $Q_x = \{q_3, q_5, q^7\}, Q = \emptyset$

Step 2: $Q_A \setminus Q_x = \{q_0, q_1, q_2, q_4, q_6, q_8\}$ are tested, $Q = \emptyset$.

Step 3: $Q = \emptyset$, we are done.

Step 4: $X_A = Q_x = \{q_3, q_5, q^7\}$

All the other states are controllable.
Example 37. Synthesize a non-blocking and controllable supervisor for the two-machine + AGV system studied in Assignment 1. Note that the only specification we need in this case is the marking of initial states, we want to repeat the cycle over and over again.

We now also have the events unload1 and unload2 uncontrollable. We cannot decide how long the processing can take, and it could also be possible that machines make themselves requests for the AGV to come and unload.
1. $S_0 = Ac(M1 || M2 || AGV); i = 0$

2. $S_{i+1} = NonBlock(S_i)$.

3. $S_{i+2} = Contrl(S_{i+1})$.

4. If $S_{i+1} \neq S_{i+2}$ then $i = i + 2$ and go to 2

5. $S = S_{i+1}$
1. \( S_0 = Ac(M1 \parallel M2 \parallel AGV); \ i = 0 \)

2. \( S_{i+1} = NonBlock(S_i) \).

3. \( S_{i+2} = Contrl(S_{i+1}) \).

4. If \( S_{i+1} \neq S_{i+2} \) then \( i = i + 2 \) and go to 2

5. \( S = S_{i+1} \)
1. $S_0 = Ac(M1 \| M2 \| AGV); i = 0$

2. $S_{i+1} = NonBlock(S_i)$.

3. $S_{i+2} = Contl(S_{i+1})$.

4. If $S_{i+1} \neq S_{i+2}$ then $i = i + 2$ and go to 2

5. $S = S_{i+1}$
1. $S_0 = Ac(M1 \parallel M2 \parallel AGV)$; $i = 0$

2. $S_{i+1} = NonBlock(S_i)$.

3. $S_{i+2} = Contrl(S_{i+1})$.

4. If $S_{i+1} \neq S_{i+2}$ then $i = i + 2$ and go to 2

5. $S = S_{i+1}$

$Contrl(S_{i+1})$ does not find new forbidden states, so we end up in 5.
1. $S_0 = Ac(M1∥M2∥AGV); i = 0$

2. $S_{i+1} = NonBlock(S_i)$.

3. $S_{i+2} = Contrl(S_{i+1})$.

4. If $S_{i+1} \neq S_{i+2}$ then $i = i + 2$ and go to 2

5. $S = S_{i+1}$

We have removed the forbidden states and all transitions associated with them. This operation is denoted Purge in Supremica.