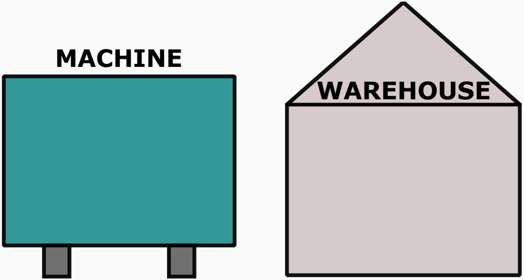
Systems Design Laboratory

A Supervisory Control Example

Matteo Zavatteri

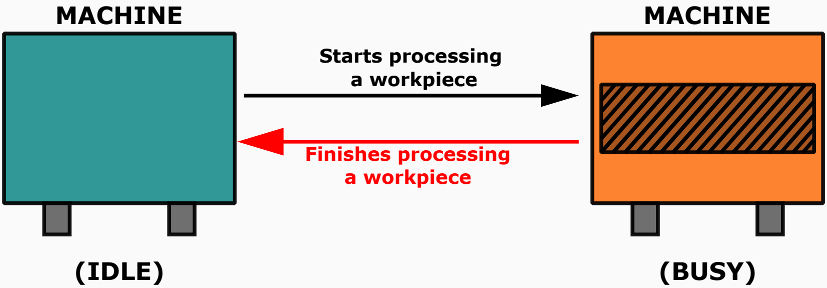
Department of Computer Science, University of Verona, ITALY

Machine-Warehouse Example



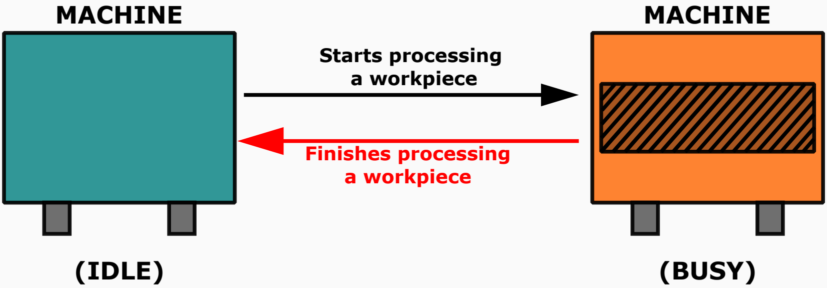
- A machine processing workpieces
- A warehouse storing finished workpieces

Machine



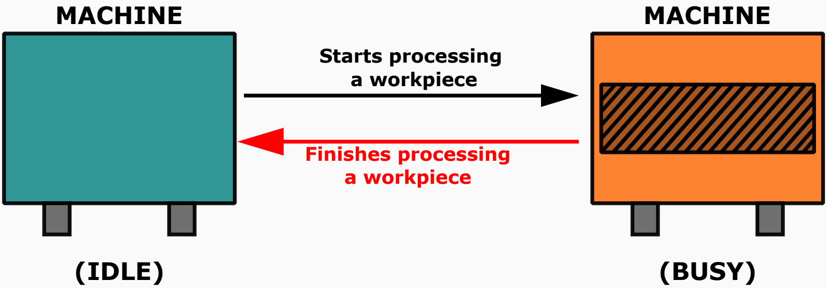
- Initially, the machine is IDLE
- Once it starts processing a workpiece it is BUSY
- Once it is BUSY it can finish processing a workpiece (this event must always be possible)
- The machine can process an infinite number of workpieces

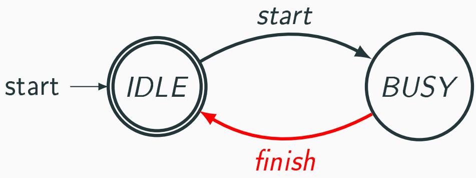
Automaton for Machine



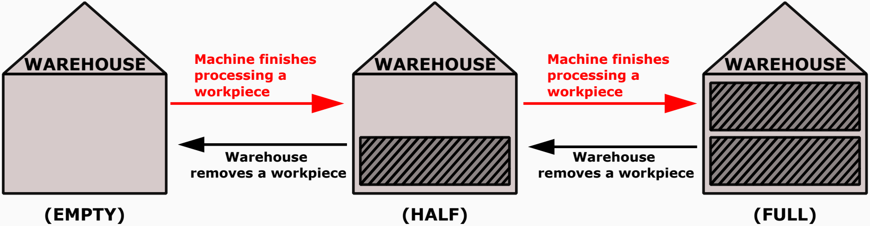
- States?
- Events and transitions?

Automaton for Machine - States



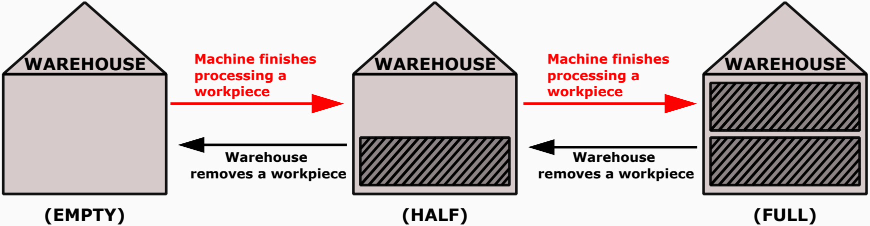


Warehouse



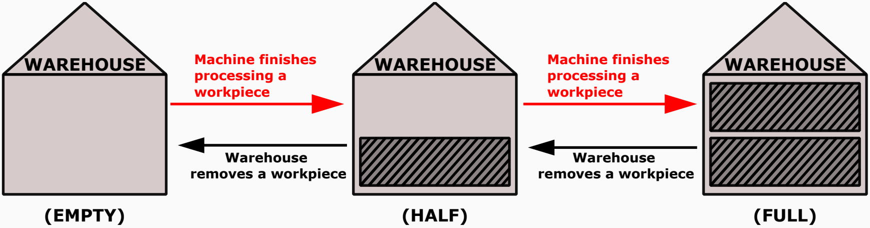
- The warehouse has a capacity of two workpieces
- Initially, the warehouse is EMPTY
- Synchronization: when the machine finishes processing a workpiece, the workpiece is stored in the warehouse
- At any time, the warehouse can remove a workpiece (if any)

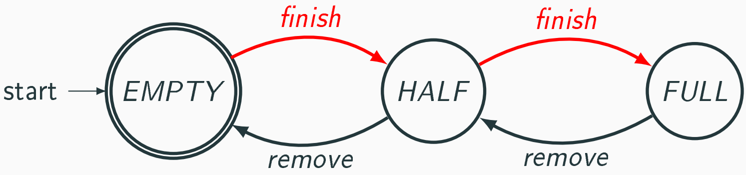
Warehouse



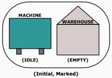
- States?
- Events and transitions?

Warehouse

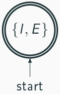

WAREHOUSE
Machine finishes processing a workpiece
WAREHOUSE
Machine finishes processing a workpiece
WAREHOUSE
Warehouse removes a workpiece
Warehouse removes a workpiece
(EMPTY)
(HALF)
(FULL)


finish
finish
start
EMPTY
HALF
FULL
remove
remove

Graphical Description

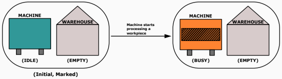


Parallel composition

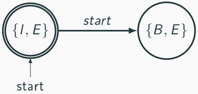


Concurrency and Synchronization

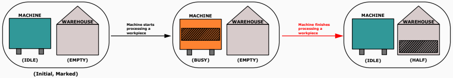
Graphical Description



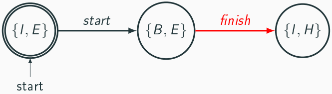
Parallel composition



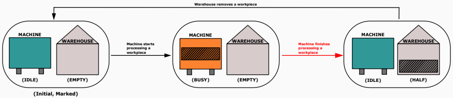
Graphical Description



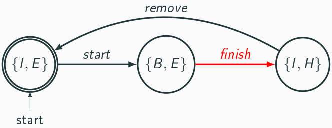
Parallel composition



Graphical Description

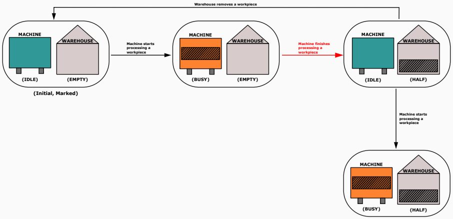


Parallel composition

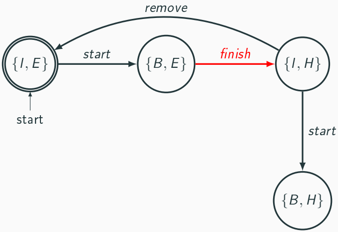


Concurrency and Synchronization

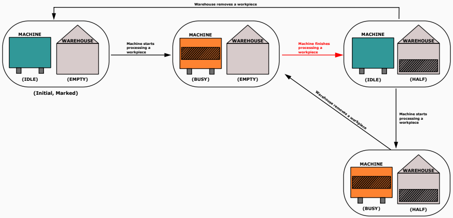
Graphical Description



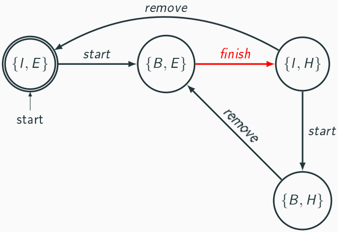
Parallel composition



Graphical Description

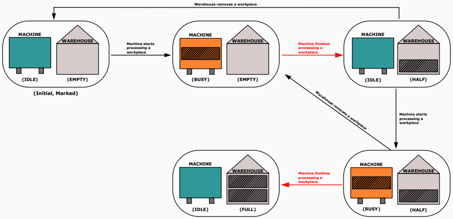


Parallel composition

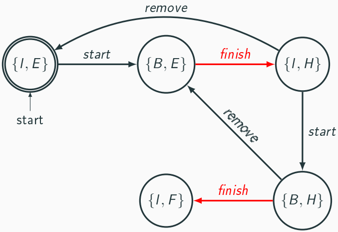


Concurrency and Synchronization

Graphical Description

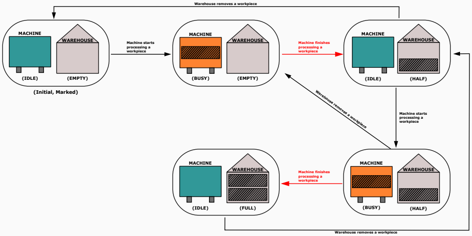


Parallel composition

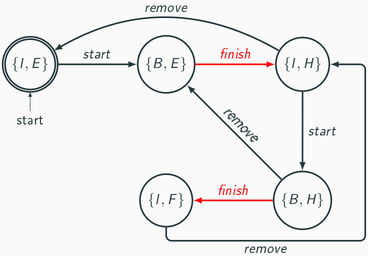


Concurrency and Synchronization

Graphical Description

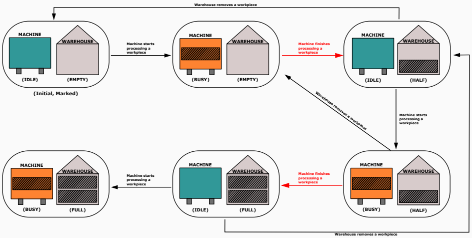


Parallel composition

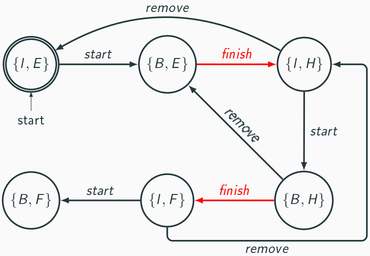


Concurrency and Synchronization

Graphical Description

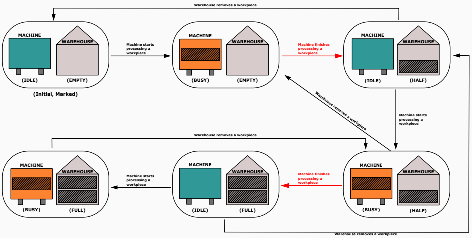


Parallel composition

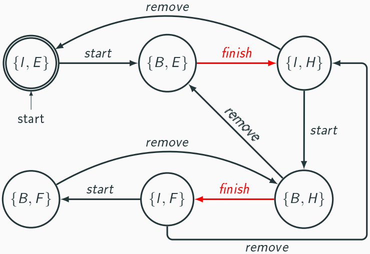


Concurrency and Synchronization - Plant

Graphical Description



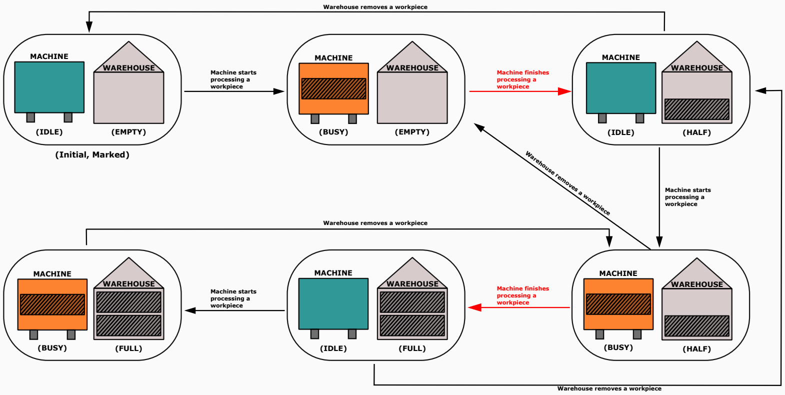
Parallel composition



Note: When the machine is BUSY and the warehouse is FULL, despite *finish* is uncontrollable, the machine cannot execute it since *finish* is not executable by the warehouse.

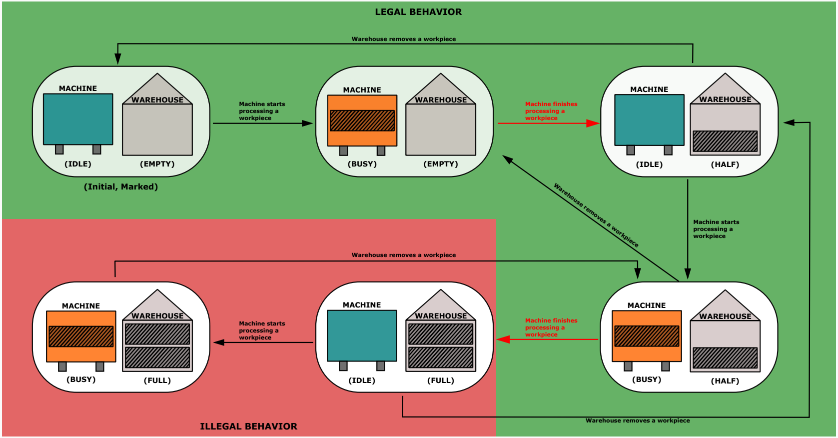
This means that, by exploiting synchronization, **and at plant level only**, we can prevent uncontrollable events from executing (by composing the "right" automata).

Requirement 1: The warehouse stores at most one workpiece



Example 1: Supervisory Control - Desired Behavior

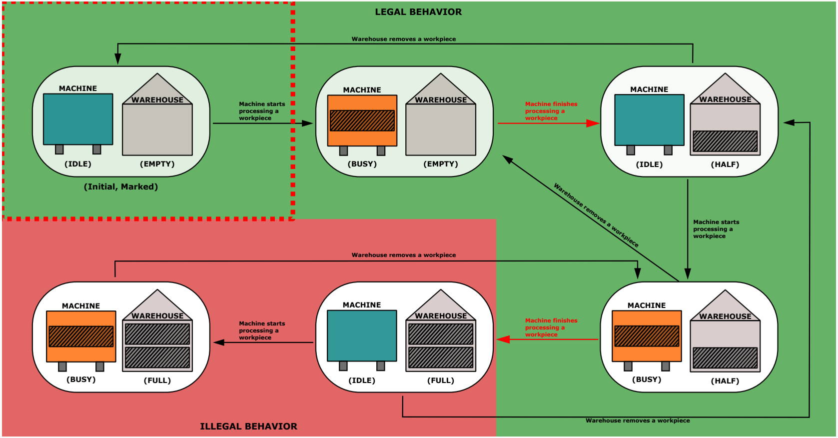
Requirement 1: The warehouse stores at most one workpiece



Can we control the plant in order to enforce such behavior? Can you spot any problem?

Example 1: Supervisory Control - Desired Behavior - Problem

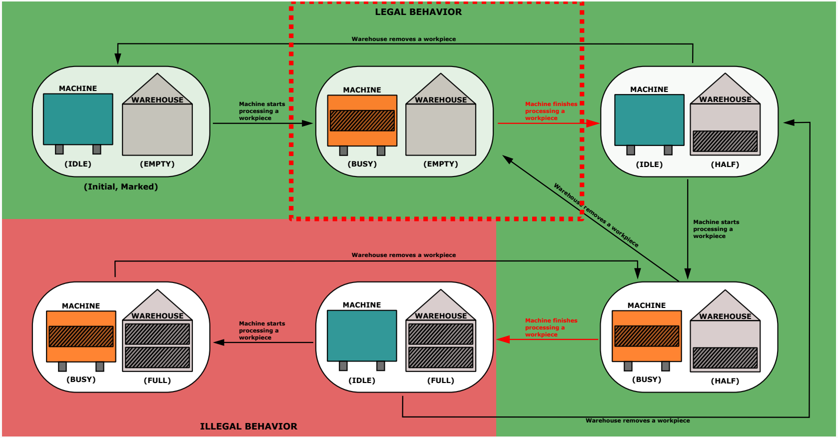
Requirement 1: The warehouse stores at most one workpiece



No problem. When the machine is IDLE and the warehouse is EMPTY, the machine can start processing a workpiece. This leads to machine BUSY and warehouse EMPTY which is still a desired behavior.

Example 1: Supervisory Control - Desired Behavior - Problem

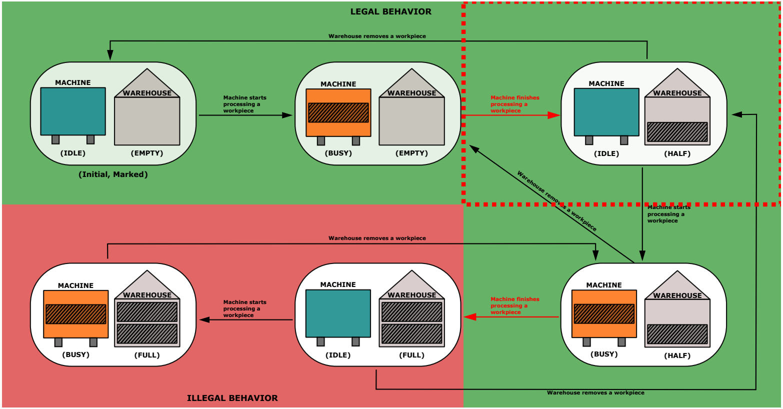
Requirement 1: The warehouse stores at most one workpiece



No problem. When the machine is BUSY and the warehouse is EMPTY, the machine can finish processing the workpiece. This leads to machine IDLE and warehouse HALF which is still a desired behavior.

Example 1: Supervisory Control - Desired Behavior - Problem

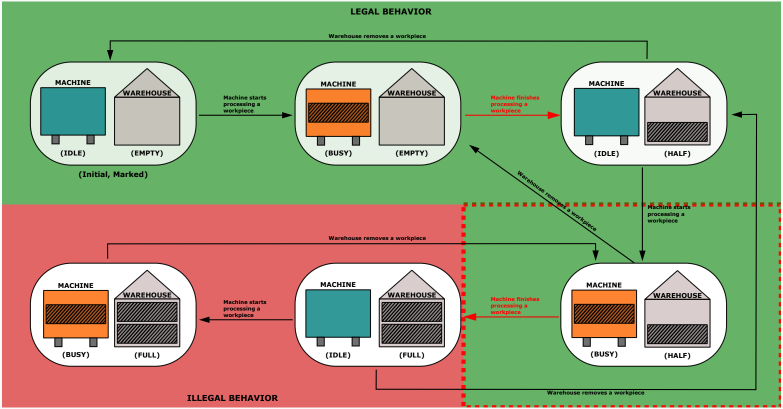
Requirement 1: The warehouse stores at most one workpiece



No problem. When the machine is IDLE and the warehouse is HALF, either the machine starts processing a workpiece (leading to machine BUSY, warehouse EMPTY) or the warehouse removes a workpiece from its storage (leading to machine IDLE, warehouse EMPTY). Either way, they are both desired behaviors.

Example 1: Supervisory Control - Controllable Behavior

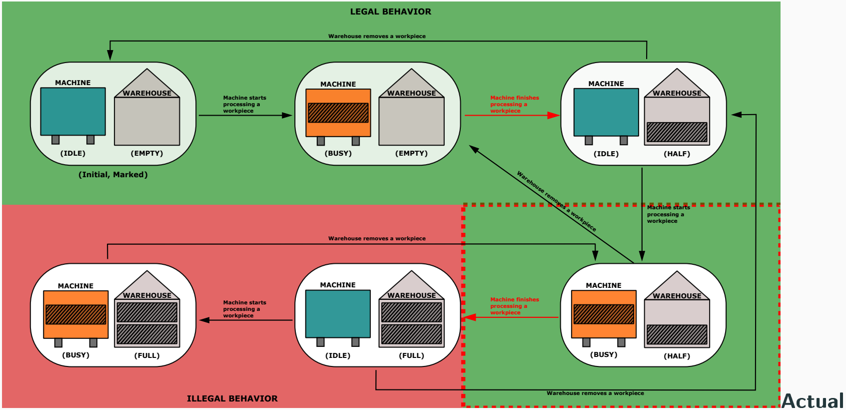
Requirement 1: The warehouse stores at most one workpiece



Problem. When the machine is BUSY and the warehouse is HALF, either the the warehouse removes a workpiece from its storage (leading to machine BUSY, warehouse EMPTY) or the machine finishes processing the workpiece (leading to machine IDLE, warehouse FULL). The second case is not a desired behavior.

Example 1: Supervisory Control - Desired Behavior

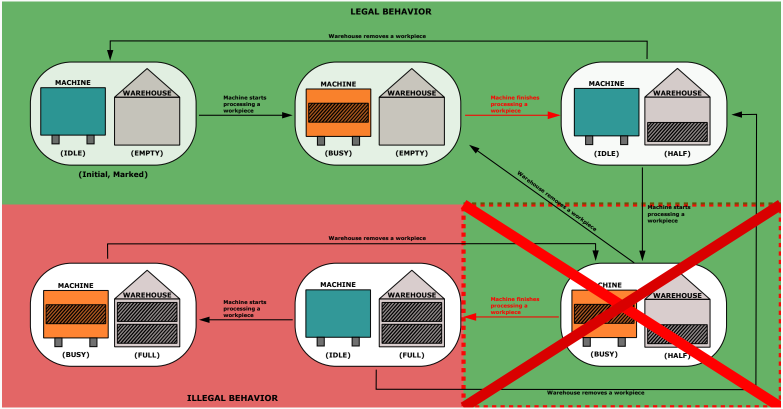
Requirement 1: The warehouse stores at most one workpiece



Problem. When the machine is BUSY and the warehouse is HALF we cannot prevent machine from finishing processing the workpiece because *finish* is uncontrollable and it is executable in the plant.

Example 1: Supervisory Control - Desired Behavior - Solution

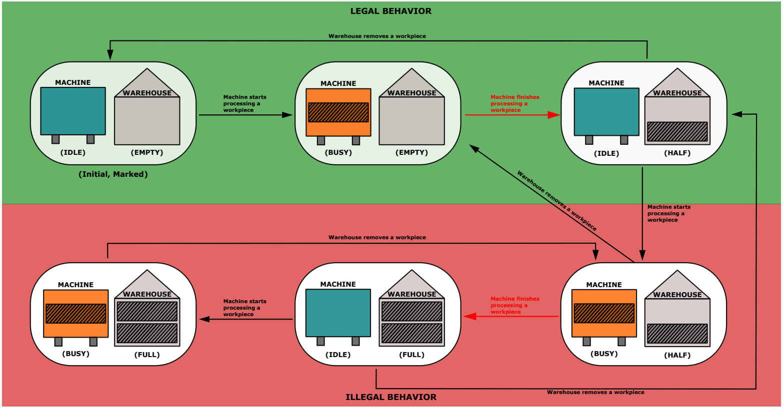
Requirement 1: The warehouse stores at most one workpiece



Solution. We need to prevent the plant to get to that state. That is, we prevent machine to start processing a workpiece when machine is IDLE and the warehouse is HALF.

Example 1: Supervisory Control - Controllable Behavior

Requirement 1: The warehouse stores at most one workpiece

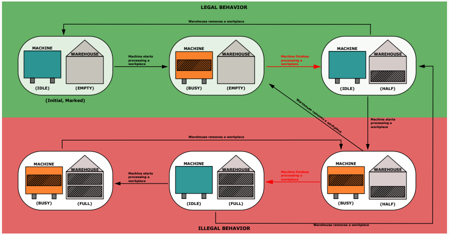


Actual controllable behavior (supremal controllable sublanguage).

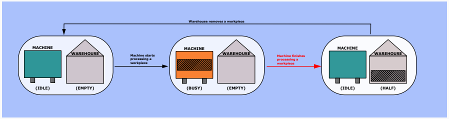
Example 1: Supervisory Control - Supervisor

Requirement 1: The warehouse stores at most one workpiece

Plant



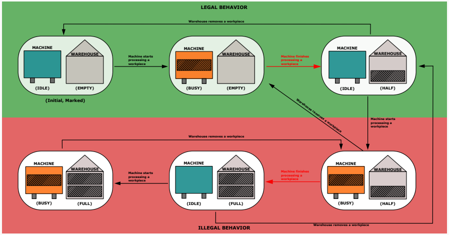
Supervisor

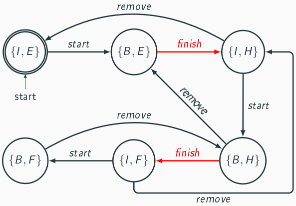


Example 1: Supervisory Control - Supervisor

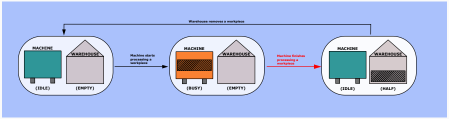
Requirement 1: The warehouse stores at most one workpiece

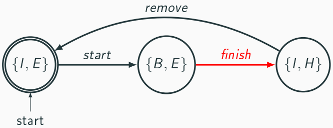
Plant





Supervisor



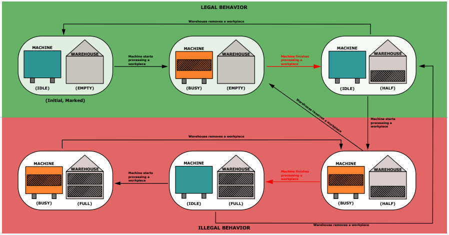


How can we synthesize this supervisor automatically?

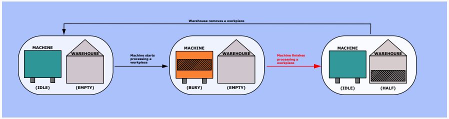
Example 1: Supervisory Control - Supervisor

Requirement 1: The warehouse stores at most one workpiece

Plant



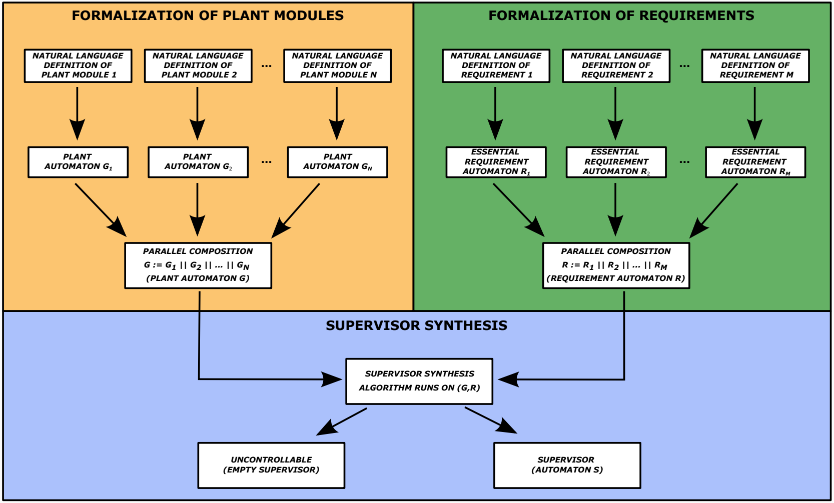
Supervisor

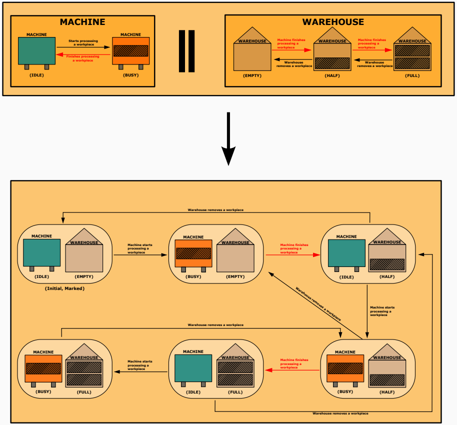


- Basically the supervisor here is the "intended part" of the system
- We would like to avoid computing it "by hand" (this case study is simple, but what about a real one?)

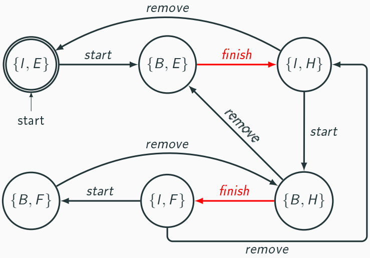
We need formal methods!

Supervisor Synthesis: Workflow



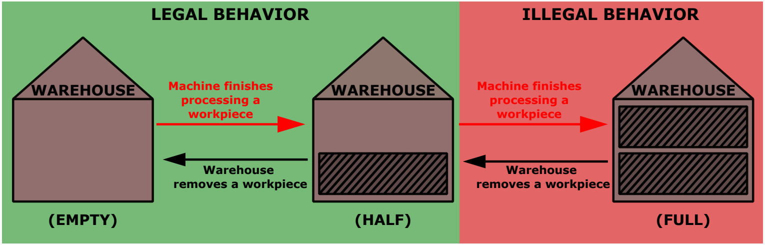


Plant Automaton G



Example 1: Supervisor Synthesis - Essential Requirement

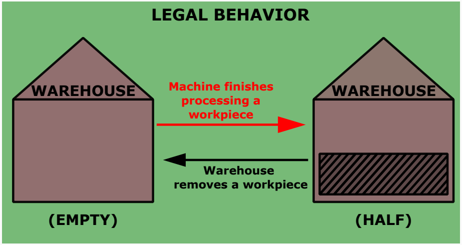
Requirement 1: The warehouse stores at most one workpiece

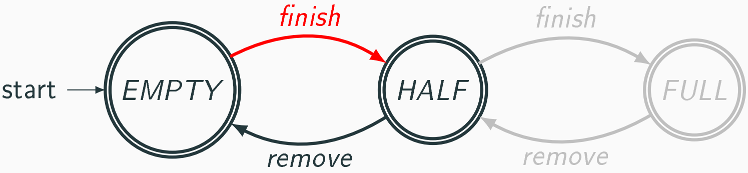


- States?
- Transitions?

Example 1: Supervisor Synthesis - Essential Requirement

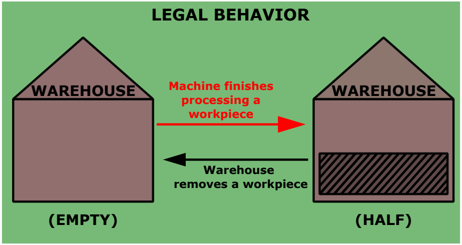
Requirement 1: The warehouse stores at most one workpiece

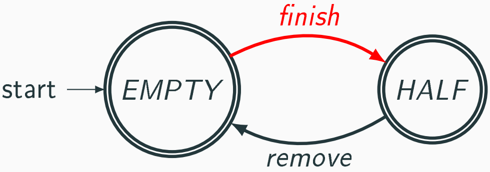


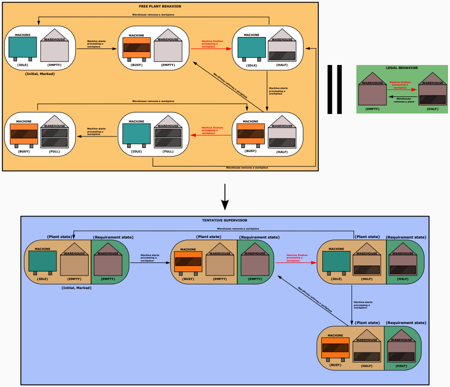


Example 1: Supervisor Synthesis - Essential Requirement

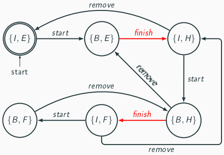
Requirement 1: The warehouse stores at most one workpiece



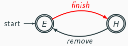




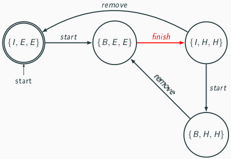
Plant Automaton G



Requirement R_1

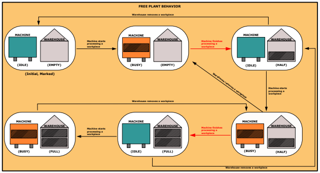


Parallel composition $G||R_1$

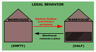


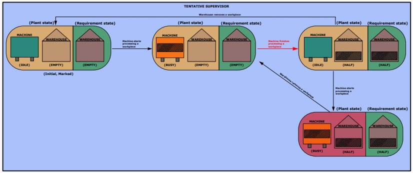
(Tentative Supervisor)

Example 1: Synthesis Algorithm - Removal of States

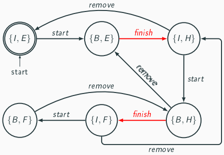


||

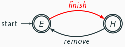




Plant Automaton G

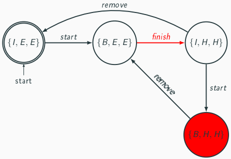


Requirement R_1



Parallel composition $G||R_1$

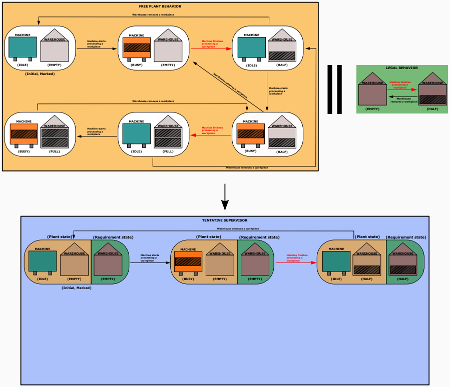
When



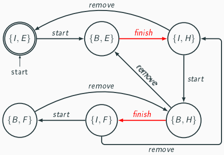
the machine is BUSY and the warehouse is HALF, the machine is prevented to finish. However, at plant level this is not permitted.

(Tentative Supervisor)

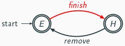
Example 1: Synthesis Algorithm - No more removals



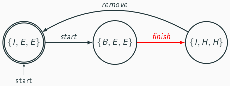
Plant Automaton G



Requirement R_1



Parallel composition $G||R_1$

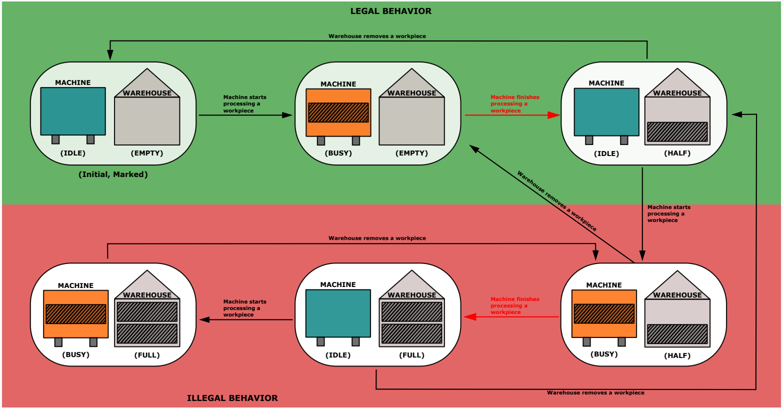


(Final Supervisor)

Besides “uncontrollable states”, we also need to remove non-accessible and non-coacessible states, if any (this example has none).

Example 2: Supervisory Control - Desired Behavior

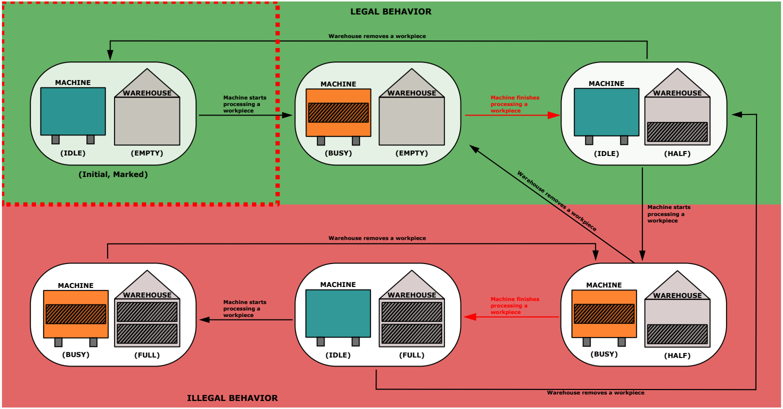
Requirement 2: The machine can start only if the warehouse is empty



Let's see if we spot any problems

Example 2: Supervisory Control - Desired Behavior

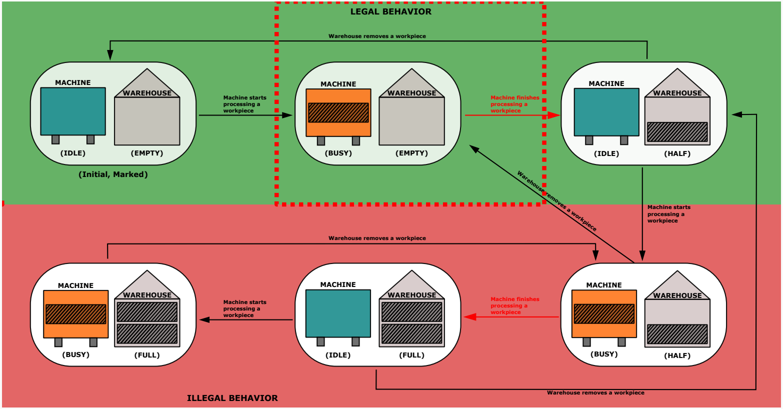
Requirement 2: The machine can start only if the warehouse is empty



No problem. When the machine is IDLE and the warehouse is EMPTY, the machine can start processing a workpiece. This leads to machine BUSY and warehouse EMPTY which is still a desired behavior.

Example 2: Supervisory Control - Desired Behavior

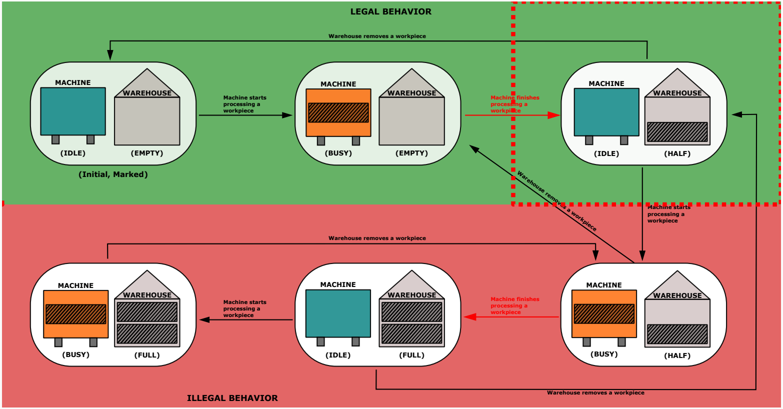
Requirement 2: The machine can start only if the warehouse is empty



No problem. When the machine is BUSY and the warehouse is EMPTY, the machine can finish processing the workpiece. This leads to machine IDLE and warehouse HALF which is still a desired behavior.

Example 2: Supervisory Control - Desired Behavior

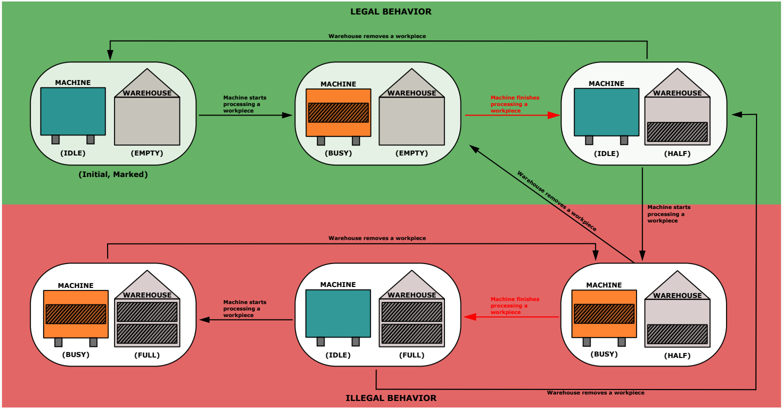
Requirement 2: The machine can start only if the warehouse is empty



No problem. When the machine is IDLE and the warehouse is HALF, either the warehouse removes a workpiece from its storage (leading to machine IDLE, warehouse EMPTY) or the machine starts processing a workpiece (leading to machine BUSY, warehouse EMPTY). The second is an undesired behavior.

Example 2: Supervisory Control - Controllable Behavior

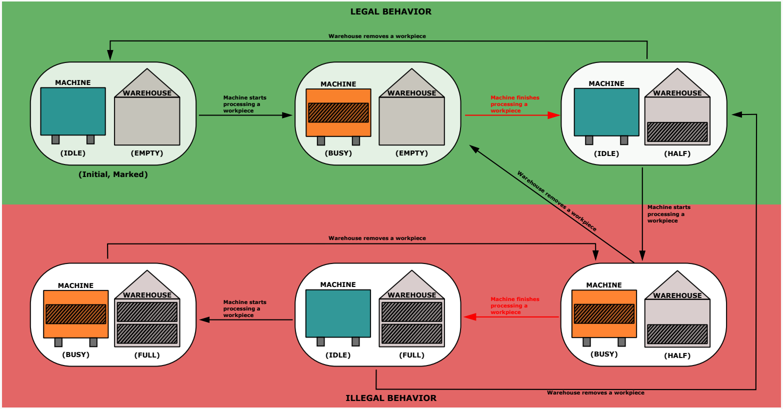
Requirement 2: The machine can start only if the warehouse is empty



However, in this state of the plant, since *start* is controllable, we can prevent the machine from starting working a workpiece. Therefore, the desired behavior is also controllable.

Example 2: Supervisory Control - Controllable Behavior

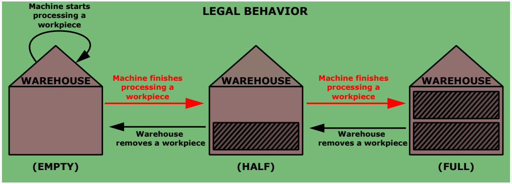
Requirement 2: The machine can start only if the warehouse is empty



However, in this state of the plant, since *start* is controllable, we can prevent the machine from starting working a workpiece. Therefore, the desired behavior is also controllable.

Example 2: Supervisor Synthesis - Essential Requirement

Requirement 2: The machine can start only if the warehouse is empty

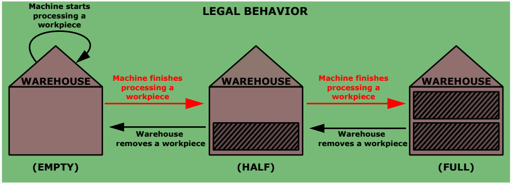


Essential Requirement

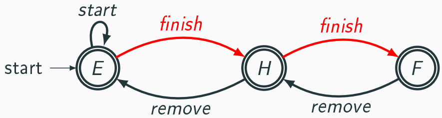
- States?
- Transitions?

Example 2: Supervisor Synthesis - Essential Requirement

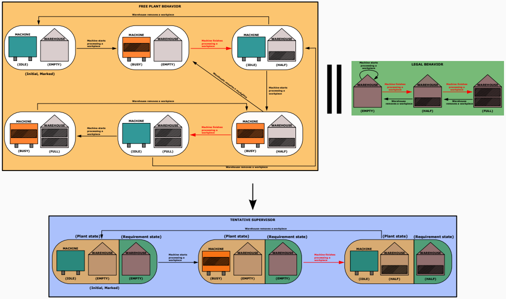
Requirement 2: The machine can start only if the warehouse is empty



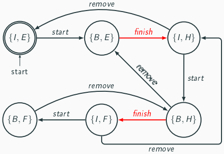
Essential Requirement



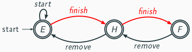
Example 2: Synthesis Algorithm - Tentative Supervisor



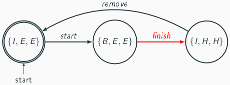
Plant Automaton G



Requirement R_2



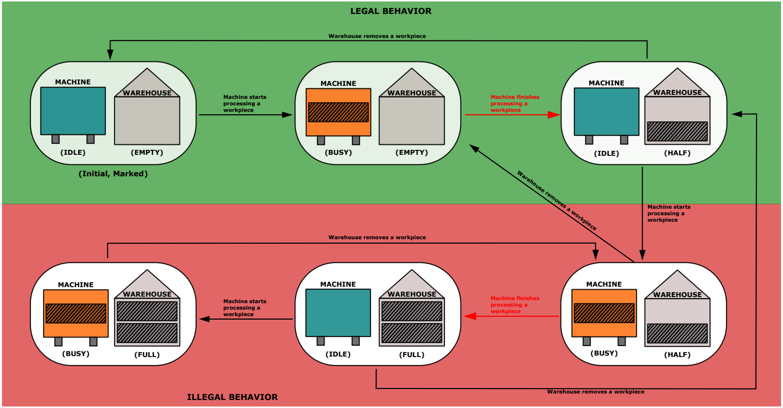
Parallel composition $G \| R_2$



(Tentative Supervisor, also final)

Example 3: Supervisory Control - Desired Behavior

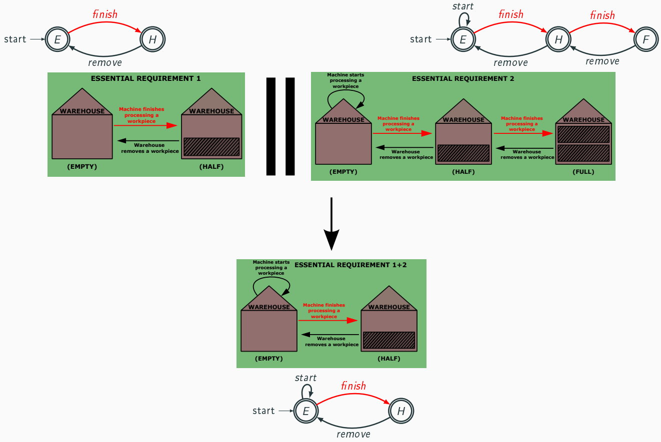
Requirement 1,2: The warehouse stores at most one workpiece AND the machine can start only if the warehouse is empty



Same desired behavior of Requirement 2

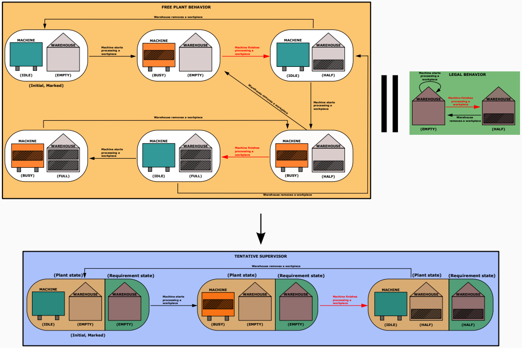
Example 3: Combining Requirements 1 and 2

Requirement 1,2: The warehouse stores at most one workpiece AND the machine can start only if the warehouse is empty

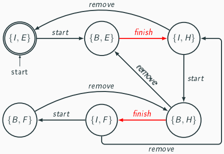


What about this new requirement?

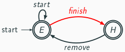
Example 3: Synthesis Algorithm - Tentative Supervisor



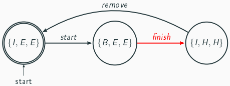
Plant Automaton G



Requirement $R_{1,2}$



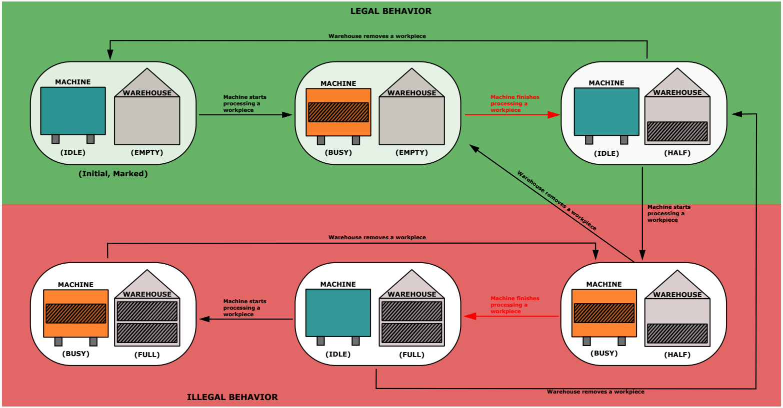
Parallel composition $G \| R_{1,2}$



(Tentative Supervisor, also final)

Example 4: Supervisory Control - Desired Behavior

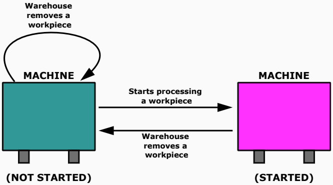
Requirement 4: If the Machine starts, then the warehouse must remove a workpiece before the machine can start again.



Same desired behavior of Requirement 2 and 1+2

Example 4: Supervisor Synthesis - Essential Requirement

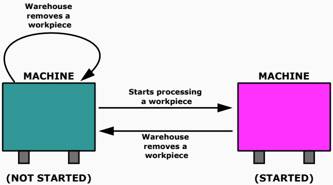
Requirement 4: If the Machine starts, then the warehouse must remove a workpiece before the machine can start again.



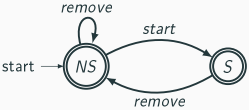
- States?
- Transitions?

Example 4: Supervisor Synthesis - Essential Requirement

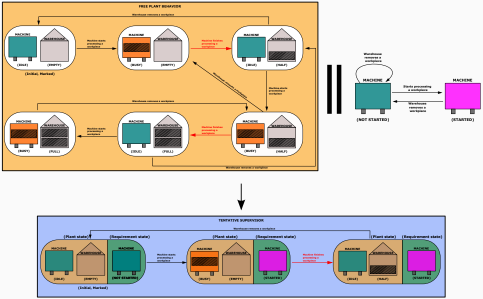
Requirement 4: If the Machine starts, then the warehouse must remove a workpiece before the machine can start again.



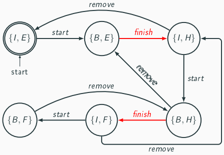
Essential Requirement



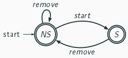
Example 4: Synthesis Algorithm - Tentative Supervisor



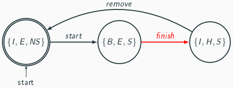
Plant Automaton G



Requirement R_4



Parallel composition $G||R_4$

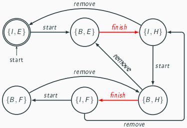


(Tentative Supervisor, also final)

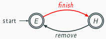
Comparing Different Requirements and Related Supervisors

Example 1

Plant Automaton G

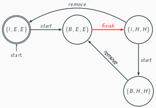


Requirement R_1

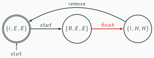


Parallel composition

$G || R_1$

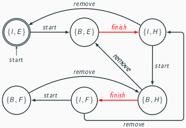


Supervisor

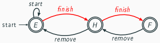


Example 2

Plant Automaton G

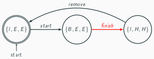


Requirement R_2

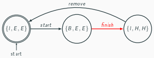


Parallel composition

$G || R_2$

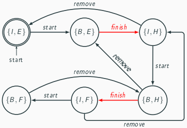


Supervisor

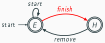


Example 3

Plant Automaton G

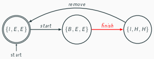


Requirement $R_{1,2}$

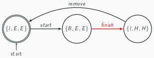


Parallel composition

$G || R_{1,2}$

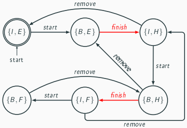


Supervisor

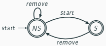


Example 4

Plant Automaton G

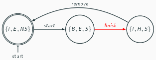


Requirement R_4

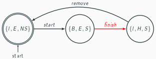


Parallel composition

$G || R_4$

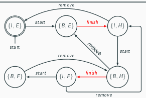
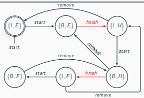
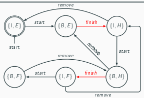
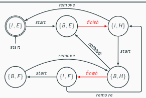
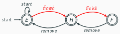
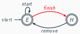
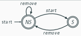
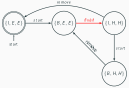
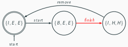
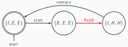
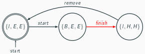
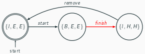
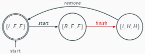
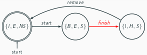


Supervisor



What can we say?

Comparing Different Requirements and Related Supervisors

Example 1	Example 2	Example 3	Example 4	What we can say
				**Same plant G**
				**Syntactically Different Requirements**
				**R_1 is semantically different from R_2, $R_{1,2}$, and R_4, whereas R_2, $R_{1,2}$, and R_4 are sematically equivalent.**
				**Same final supervisor (same supremal controllable sublanguage)**