

Supervisory Control of Discrete Event Systems

Part 3: Supervisory Control Theory

Dr Andrea Paoli

Associate Professor in Robotics and Automation

Loughborough University

Wolfson School of Mechanical, Electrical and Manufacturing Engineering

Fall semester 2026-2027

University of Ferrara

Outline

- 1 The Feedback Control Architecture & Event Classification
- 2 Formal Specifications & Controllability of Language
- 3 Supremal Controllable Sublanguage
- 4 Supervisory Control Under Partial Observation
- 5 Modular and Decentralized Architectures
- 6 Conclusions
- 7 Review Questions & Self-Assessment

Part 3: Intended Learning Outcomes (ILOs)

1 Model Closed-Loop Supervisory Architectures:

Formalize feedback control using parallel composition ($S/G := G \parallel S$) and define admissible enabling maps under actuator limits ($\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$).

2 Verify Language Controllability:

Test whether a specification $K \subseteq L_m(G)$ satisfies the controllability condition $(\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K})$, diagnosing paths where uncontrollable events force exit from safe behavior.

3 Synthesize Optimal Supervisors (supC):

Compute the supremal controllable sublanguage $\text{supC}(K)$ by iteratively pruning illegal states and uncontrollable back-propagation paths, enforcing nonblocking via Trim.

4 Characterize Sensory Limitations:

Analyze the impact of partial observation ($\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$) on state disambiguation and supervisory decision-making.

Why Closed-Loop Control in DES?

Limitations of Open-Loop Operation

Open-loop execution relies strictly on the autonomous dynamics of the plant:

- **Coordination Hazards:** As seen in Part 2, concurrent subsystems interacting via asynchronous composition ($G_1 \parallel G_2$) can enter uncoordinated **deadlocks** or **livelocks**.
- **Forbidden State Regions:** Physical systems inevitably contain unsafe configurations, such as buffer overflows, collision points, or hazardous machine states.
- **Specification Drift:** Without dynamic intervention, spontaneous trace generation can lead the system out of the desired language specification.

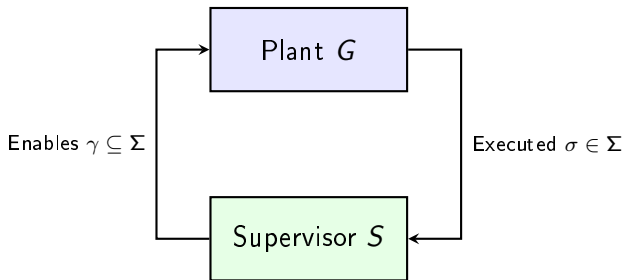
The Supervisory Objective

Introduce an external controlling agent—the **Supervisor** (S)—that dynamically restricts plant behavior by disabling transitions, ensuring:

$$L(S/G) \subseteq L_{\text{safe}} \quad \text{and} \quad S/G \text{ is nonblocking}$$

The Supervisory Feedback Loop

The Standard Supervisory Architecture



Core Operational Principle

The plant G acts as the spontaneous event generator. The supervisor S acts strictly as an **enabling/disabling filter**; it cannot force an event to fire.

Closed-Loop Representation

When S is realized as an automaton over alphabet Σ , the closed-loop system is mathematically characterized via parallel composition:

$$S/G := G \parallel S$$

Event Partitioning: Actuator Limits

Controllable vs. Uncontrollable Events

The plant alphabet Σ is partitioned into two disjoint subsets reflecting actuator capabilities:

$$\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$$

Controllable Events (Σ_c)

Transitions that the supervisor can dynamically **enable** or **disable**:

- Actuator commands (e.g., motor start)
- Task release (e.g., dispatch AGV, load part)
- Network transmissions (e.g., send packet)

Uncontrollable Events (Σ_{uc})

Transitions that **cannot be prevented** once the state is reached:

- Sensor triggers (e.g., limit switch, part arrival)
- System faults (e.g., tool break, power trip)
- Human actions (e.g., emergency button press)

Core Design Takeaway

Because uncontrollable events cannot be disabled directly, safety can only be enforced by disabling controllable transitions that precede them.

The Supervisory Map & Physical Admissibility

The Supervisory Control Map γ

A supervisor is formally formalized as a feedback mapping:

$$\gamma : L(G) \rightarrow 2^\Sigma$$

where $\gamma(s) \subseteq \Sigma$ represents the set of events **enabled** after string $s \in L(G)$:

- If $\sigma \in \gamma(s)$, the plant is permitted to execute σ .
- If $\sigma \notin \gamma(s)$, event σ is disabled (inhibited by the supervisor).

Physical Admissibility Constraint

Because the supervisor cannot physically block uncontrollable events, γ is **admissible** if and only if:

$$\forall s \in L(G) : \quad \Sigma_{uc} \cap \Gamma_G(f(x_0, s)) \subseteq \gamma(s)$$

Operational Meaning: If an uncontrollable event is active in the current plant state ($\sigma \in \Gamma_G(x)$), the supervisor **must enable it**. Disabling authority applies strictly to Σ_c .

Formulating Control Specifications

Linguistic Representation of Legal Behavior

The primary objective of supervisory control is to confine the closed-loop plant behavior within a designated **legal language**:

$$K \subseteq L_m(G), \quad \overline{K} \subseteq L(G)$$

- $\overline{K}$ defines the set of all allowable prefixes (safe trajectories).
- K defines the set of completed tasks or target operational cycles.

State-Based Representation: Specification Automaton H

In practical synthesis, K is modeled as a deterministic finite-state automaton:

$$H = (X_H, \Sigma, f_H, x_{0,H}, X_{m,H})$$

such that the generated and marked behaviors precisely match the specification:

$$L(H) = \overline{K} \quad \text{and} \quad L_m(H) = K$$

Note: H acts as a reference model encoding what the closed-loop system is permitted to do.

Standard Classes of Control Specifications

1. Safety & Forbidden Behavior (State / Transition Avoidance)

Prevents the plant from entering hazardous configurations:

- **Mutual Exclusion:** Preventing simultaneous occupancy of a shared physical zone or workstation.
- **Buffer Limits:** Disabling feeder transitions when downstream capacity is exhausted.

2. Sequencing & Operational Ordering

Enforces strict execution precedence across independent plant components:

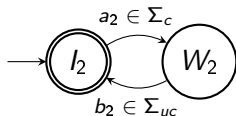
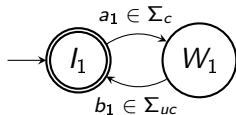
- Forcing event α to precede β (e.g., an AGV must dock before loading begins).
- Alternating tasks between parallel processing units.

3. Liveness & Nonblocking (Task Completion)

Ensures that nominal operations can always reach a designated marked state: $\overline{L_m(S/G)} = L(S/G)$ and guarantees that enforcing safety and ordering constraints does not induce deadlock or livelock.

Concrete Example: Shared-Buffer Workstation

Plant Models: Two Decoupled Machines



$$G = G_1 \parallel G_2 \text{ over } \Sigma = \{a_1, a_2\} \cup \{b_1, b_2\}.$$

Workcell Physical Description

- **Machine 1 (G_1):** Starts job ($a_1 \in \Sigma_c$), finishes and drops part into buffer ($b_1 \in \Sigma_{uc}$).
- **Machine 2 (G_2):** Takes part from buffer and starts processing ($a_2 \in \Sigma_c$), finishes job ($b_2 \in \Sigma_{uc}$).

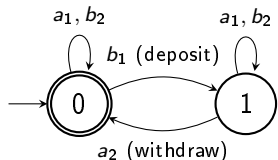
The Engineering Problem

The buffer has **capacity 1**. In open-loop ($G_1 \parallel G_2$), M_1 can complete two consecutive parts before M_2 picks up:

$$s = a_1 b_1 a_1 b_1 \in L(G) \implies \text{Buffer Overflow!}$$

Example: Deriving the Specification Automaton H

Specification Automaton H



Alphabet Scope

$\Sigma_H = \{a_1, b_1, a_2, b_2\} = \Sigma$. Unconstrained events (a_1, b_2) self-loop at every state.

Translating Rules into Formal Constraints

- 1 No Overflow:** A part can be deposited (b_1) only if buffer state is 0:

b_1 is disabled in state 1

- 2 No Underflow:** A part can be extracted (a_2) only if buffer state is 1:

a_2 is disabled in state 0

Target Behavior

The legal marked language is $K = L_m(H) \cap L_m(G)$, and safe prefixes are given by $\bar{K} = L(H) \cap L(G)$.

Example: The Control Feasibility Dilemma

Key Question for Control Synthesis

In state 1, the specification H requires disabling b_1 to avoid overflow. However: $b_1 \in \Sigma_{uc}$ (Machine 1 part completion is uncontrollable). Can the supervisor physically disable b_1 once Machine 1 is working?

The Control Dilemma & Upstream Intervention

- **Physical Impossibility:** If state $(W_1, 1)$ is reached, the plant can autonomously fire b_1 , violating the buffer specification H .
- **Admissible Strategy:** Because the supervisor cannot disable $b_1 \in \Sigma_{uc}$, it **must intervene upstream** by disabling the controllable event $a_1 \in \Sigma_c$ whenever the buffer already holds a part (state 1).
- **Motivation for Theory:** The desired language K may demand disabling uncontrollable events. We need a mathematical test to determine if K is achievable under admissible feedback.

Language Controllability

This brings us directly to the concept of **Language Controllability**: does the specification allow uncontrollable transitions that the plant can physically execute?

Language Controllability: Formal Definition

The Core Problem

Given a plant G over $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$ and a desired legal language $K \subseteq L_m(G)$, can we construct a physically admissible supervisor S such that $L(S/G) = \overline{K}$?

Definition: Controllability (Ramadge & Wonham, 1987)

A language $K \subseteq L_m(G)$ is **controllable** with respect to G and Σ_{uc} if and only if:

$$\overline{K} \Sigma_{uc} \cap L(G) \subseteq \overline{K}$$

Deconstruction of the Invariance Condition

- $\overline{K}$: Any valid sequence of safe events executed so far.
- $\overline{K} \Sigma_{uc}$: The set of candidate extensions formed by appending an uncontrollable event $u \in \Sigma_{uc}$.
- $\cap L(G)$: Only those uncontrollable extensions that the plant G is physically capable of firing.
- $\subseteq \overline{K}$: All such physically possible uncontrollable continuations **must stay inside** the legal language.

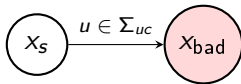
Diagnosing Controllability: The Violation Condition

When Does Controllability Fail?

A specification language K is **uncontrollable** if and only if there exist a string $s \in \overline{K}$ and an uncontrollable event $u \in \Sigma_{uc}$ such that:

$$s \in \overline{K}, \quad su \in L(G), \quad \text{but} \quad su \notin \overline{K}$$

The Control Contradiction



The plant can autonomously fire u at state $x_s = f_G(x_0, s)$, while transition (x_s, u) is forbidden in the specification ($su \notin \overline{K}$).

The Physical Dead-End

- Once the plant completes trace s , it enters state x_s .
- State x_s permits transition $u \in \Sigma_{uc}$ by plant dynamics.
- By physical admissibility, a supervisor **cannot inhibit** u .
- Departure from legal behavior is entirely inevitable once x_s is reached.

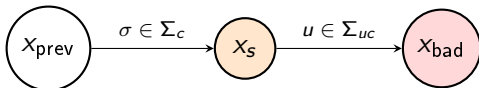
Diagnosing Controllability: Upstream Prevention

The Supervisory Dilemma

When an uncontrollable violation occurs at state x_s , the supervisor cannot resolve the problem at the point of failure:

$$u \in \Sigma_{uc} \implies u \in \gamma(s) \text{ must hold by admissibility}$$

Backward Inevitability



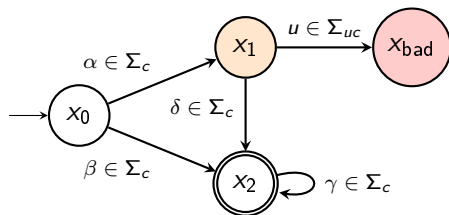
Reaching x_s is fatal. Prevention must happen upstream by disabling $\sigma \in \Sigma_c$.

The Control Remedy

- **Mark State as Unsafe:** Since x_s inevitably leaks into illegal behavior via Σ_{uc} , state x_s must be treated as forbidden.
- **Upstream Actuation:** The supervisor must disable an earlier controllable event $\sigma \in \Sigma_c$ entering x_s .
- **Foundation of Synthesis:** This iterative back-propagation principle forms the core engine of the supC algorithm (Section 3.3).

Example: The Mechanism of Upstream Prevention

Plant Automaton G



Legal States: $X_{\text{safe}} = \{x_0, x_1, x_2\}$. State x_1 carries useful work via $\delta \rightarrow x_2$.

The Conflict at State x_1

- We **would like to keep** x_1 : executing $\delta \in \Sigma_c$ reaches the marked state x_2 .
- However, transition $u \in \Sigma_{uc}$ is concurrently active at x_1 and leads to x_{bad} .
- By admissibility, the supervisor **cannot disable** u . Enabling δ does not prevent u from firing first.

The Inevitable Sacrifice

- Even though $x_1 \in X_{\text{safe}}$, reaching it risks an uncontrollable leak into x_{bad} .
- State x_1 becomes **co-illegal** (weakly illegal).
- The supervisor must intervene at x_0 by disabling $\alpha \in \Sigma_c$, sacrificing path $\alpha\delta$ to guarantee safety.

$L_m(G)$ -Relative Closure: Preventing Deadlock

The Marking Conflict

Controllability guarantees that the plant can be kept within prefix boundaries $\overline{K}$. However, it does not automatically guarantee that tasks can be completed ($K \subseteq L_m(G)$).

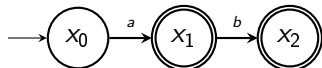
Definition: Relative Closure

A language $K \subseteq L_m(G)$ is said to be $L_m(G)$ -**relatively closed** if and only if: $K = \overline{K} \cap L_m(G)$

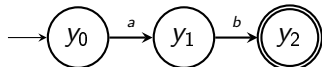
Physical Meaning

- If a string $s \in \overline{K}$ reaches a state that is marked in the plant ($s \in L_m(G)$), it **must also be marked** in the specification ($s \in K$).
- Avoids scenarios where the supervisor restricts marked plant terminations without providing alternative marked paths, creating artificial blocking traps.
- If G is nonblocking and K is marked naturally by completed operational cycles, relative closure aligns task completion between plant and supervisor.

Example: Violation of $L_m(G)$ -Relative Closure

Physical Plant G 

$$L(G) = \{\varepsilon, a, ab\}, \quad L_m(G) = \{a, ab\}$$

Specification Automaton H (Desired)

$$L(H) = \overline{K} = \{\varepsilon, a, ab\}, \quad K = L_m(H) = \{ab\}$$

Mathematical Violation Test

Since $\overline{K} = \{\varepsilon, a, ab\}$:

$$\overline{K} \cap L_m(G) = \{a, ab\} \neq K = \{ab\}$$

$\Rightarrow$ Marking mismatch on prefix string a : state $x_1 \in X_m(G)$, while $y_1 \notin X_m(H)$.

The Supervisory Impossibility

- Marking in closed loop is dictated by plant hardware: $L_m(S/G) = L(S/G) \cap L_m(G)$.
- Enabling a to reach target ab inevitably lets the plant enter marked state x_1 .
- The supervisor **cannot strip the marked status** from string a to satisfy $a \notin K$.
- Enforcing $L_m(S/G) = K$ and $L(S/G) = \overline{K}$ simultaneously is impossible.

The Fundamental Theorem of Supervisory Control

Theorem (Ramadge & Wonham, 1987)

Let G be a plant automaton over alphabet $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$, and let $K \subseteq L_m(G)$ be a non-empty specification language ($K \neq \emptyset$). There exists a **nonblocking**, physically **admissible** supervisor S s.t.:

$$L(S/G) = \overline{K} \quad \text{and} \quad L_m(S/G) = K$$

if and only if both of the following conditions hold simultaneously:

- 1 **Controllability:** $\overline{K} \Sigma_{uc} \cap L(G) \subseteq \overline{K}$
- 2 **$L_m(G)$ -Relative Closure:** $K = \overline{K} \cap L_m(G)$

Engineering Takeaway

- **Safety without intervention leakage:** Condition 1 ensures autonomous plant actions cannot force the system outside safe prefixes $\overline{K}$.
- **No artificial blocking:** Condition 2 guarantees that every completed physical cycle corresponds to a legitimate marked task in K .

Supervisor Feedback Realization

Canonical Feedback Map γ

When the conditions of the Fundamental Theorem hold, the supervisor enabling map $\gamma : \overline{K} \rightarrow 2^\Sigma$ realizes exact language equivalence: $\gamma(s) = \Sigma_{uc} \cup \{\sigma \in \Sigma_c : s\sigma \in \overline{K}\}$, $\forall s \in \overline{K}$

Operational Mechanism & Automaton Implementation

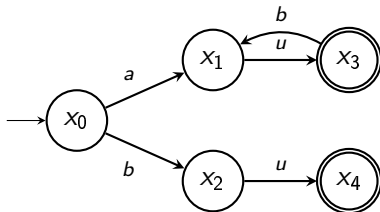
- **Admissibility Guaranteed:** $\Sigma_{uc} \subseteq \gamma(s)$ is enforced unconditionally for every trace $s \in \overline{K}$.
- **Active Control:** An actuator event $\sigma \in \Sigma_c$ is enabled if and only if $s\sigma \in \overline{K}$.
- **Automaton Realization:** For regular K , S is implemented as a deterministic automaton and the closed-loop system evolves via synchronous product: $S/G := G \parallel S$.

Open Question for Section 3.3

What if K is **not** controllable? We cannot enforce K directly, but we must compute the **supremal controllable sublanguage** $\sup \mathcal{C}(K)$.

Example: Controllable & Nonblocking Synthesis — Setup

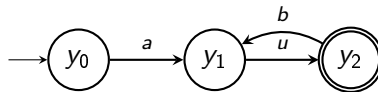
Plant Automaton G



$$\Sigma_c = \{a, b\}, \quad \Sigma_{uc} = \{u\}$$

$$L_m(G) = (a \cup b)(ub)^*u$$

Specification Automaton H



Target language:

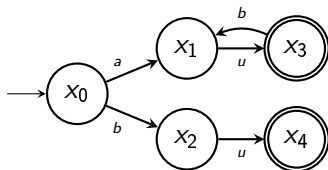
$$K = L_m(H) = a(ub)^*u \subset L_m(G)$$

Control Objective

- The plant offers two alternative operational tracks starting with a or b .
- The specification requires routing through the upper loop $a(ub)^*u$, strictly forbidding the lower path starting from b at state x_0 .
- Once inside the upper loop, the uncontrollable event u autonomously completes each processing stage

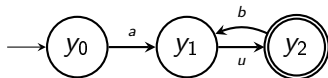
Example: Controllable & Nonblocking Synthesis — Setup

Plant Automaton G ($\Sigma_c = \{a, b\}$, $\Sigma_{uc} = \{u\}$)



$$L_m(G) = (a \cup b)(ub)^*u$$

Specification Automaton H



$$K = L_m(H) = a(ub)^*u \subset L_m(G)$$

Control Objective

- **Dual Tracks in Plant:** The unconstrained system G can start production along two tracks via a or b .
- **Strict Specification:** The customer requires only the upper processing loop $a(ub)^*u$, forbidding the lower track initiated by b at x_0 .
- **Autonomous Cycle:** Once on the upper track, the uncontrollable event $u \in \Sigma_{uc}$ processes parts autonomously.
- **Synthesizability:** The choice between the tracks is made by controllable events at x_0 , enabling exact enforcement without leaks.

Example: Controllable & Nonblocking Synthesis — Verification

Condition 1: Controllability ($\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$)

- The only uncontrollable event is $u \in \Sigma_{uc}$.
- Along any safe trace $s \in \overline{K}$, event u occurs only after strings ending in a or b (at state x_1).
- In the plant G , firing u from x_1 leads to x_3 , where $su \in \overline{K}$.
- The lower autonomous branch ($x_0 \xrightarrow{b} x_2 \xrightarrow{u} x_4$) is preemptively prevented at x_0 by disabling $b \in \Sigma_c$: $\gamma(\varepsilon) = \Sigma_{uc} \cup \{a\} = \{u, a\} \implies b$ is disabled at x_0
- Hence, no uncontrollable transition can escape $\overline{K} \implies$ **Controllability holds.**

Condition 2: $L_m(G)$ -Relative Closure & Nonblocking

- Every prefix $s \in \overline{K}$ can be extended to an accepting state via the autonomous event $u \in \Sigma_{uc}$ ($x_1 \xrightarrow{u} x_3$).
- Relative closure holds: $\overline{K} \cap L_m(G) = \overline{a(ub)^*u} \cap (a \cup b)(ub)^*u = a(ub)^*u = K$
- Plant physics and specification markings align perfectly $\implies$ **Nonblocking supervisor exists.**

Incontrollability and Approximation

The Engineering Dilemma

A desired specification $K \subseteq L_m(G)$ is frequently **not controllable**: $\overline{K} \Sigma_{uc} \cap L(G) \not\subseteq \overline{K}$
 Actuators cannot disable events in Σ_{uc} . Hence, exact realization $L(S/G) = \overline{K}$ is physically impossible.

The Synthesis Objective: Maximal Permissiveness

Rather than abandoning control, we search for a sub-behavior $M \subseteq K$ that is:

- **Safe:** M respects the user requirements ($M \subseteq K$).
- **Synthesizable:** M is controllable with respect to G and Σ_{uc} .
- **Minimally Restrictive:** M permits as much legal plant behavior as possible.

Central Questions

Does a unique “largest” controllable sublanguage always exist? And can we compute it algorithmically?

The Family of Controllable Sublanguages

The Sublanguage Family $\mathcal{C}(K)$

Given a plant G and specification $K \subseteq L_m(G)$, define the set of all controllable sublanguages:

$$\mathcal{C}(K) := \{L \subseteq K \mid \bar{L}\Sigma_{uc} \cap L(G) \subseteq \bar{L}\}$$

Note that $\emptyset \in \mathcal{C}(K)$, so $\mathcal{C}(K)$ is never empty.

Theorem: Closure Under Arbitrary Unions (Wonham & Ramadge, 1987)

Let $\{L_i\}_{i \in I}$ be an arbitrary family of languages such that $L_i \in \mathcal{C}(K)$ for all $i \in I$. Then: $\bigcup_{i \in I} L_i \in \mathcal{C}(K)$

Key Consequence: A Unique Best Solution

Because controllability is preserved under union, we never have to choose between behaviors:

- If two supervisors enforce controllable sublanguages L_1 and L_2 , their combination $L_1 \cup L_2$ is also controllable.
- Merging **all** controllable solutions yields a single, uniquely defined largest behavior: the **supremum** $\sup \mathcal{C}(K)$.

The Supremal Controllable Sublanguage $\sup \mathcal{C}(K)$

Definition

The **supremal controllable sublanguage** of K , denoted $\sup \mathcal{C}(K)$, is the union of all controllable sublanguages contained in K : $\sup \mathcal{C}(K) := \bigcup_{L \in \mathcal{C}(K)} L$

Fundamental Properties

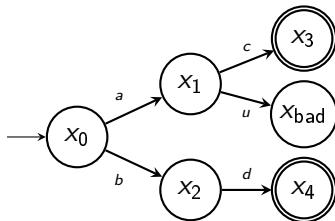
- $\sup \mathcal{C}(K) \in \mathcal{C}(K)$ (it is controllable).
- $\sup \mathcal{C}(K) \subseteq K$ (it respects the specification).
- If $L \in \mathcal{C}(K)$, then $L \subseteq \sup \mathcal{C}(K)$ (it contains all other controllable sublanguages).
- K is controllable if and only if $\sup \mathcal{C}(K) = K$.

Design Significance

$\sup \mathcal{C}(K)$ characterizes the **optimally permissive supervisor**: it disables a controllable event if and only if permitting it could lead to an inevitable violation via uncontrollable transitions.

Example: The Uncontrollable Trap (Setup)

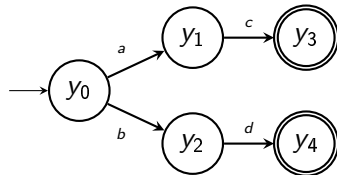
Plant Automaton G



$$\Sigma_c = \{a, b, c, d\}, \quad \Sigma_{uc} = \{u\}$$

u is an autonomous hazard from x_1 .

Specification H (Safe Behavior)



$$K = \{ac, bd\}$$

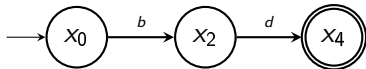
State x_{bad} is strictly forbidden.

Controllability Violation on Desired Specification K

- The desired safe prefixes are $\overline{K} = \{\varepsilon, a, b, ac, bd\}$.
- Firing $a \in \overline{K}$ is intended to lead toward goal ac . However, at state x_1 , plant G can spontaneously execute $u \in \Sigma_{uc}$: $a \in \overline{K}$, $u \in \Sigma_{uc}$, $au \in L(G)$ but $au \notin \overline{K}$.
- The supervisor **cannot disable u at state x_1** . Hence, K is uncontrollable.

Example: The Uncontrollable Trap ($\sup \mathcal{C}(K)$)

Optimal Controlled System S/G



$$\sup \mathcal{C}(K) = \{bd\} \subset K$$

$$\overline{\sup \mathcal{C}(K)} = \{\varepsilon, b, bd\}$$

Pruning via Backward Reachability

- State x_1 is an **uncontrollable hazard state**: once reached, the plant can autonomously enter forbidden state x_{bad} .
- Because u cannot be disabled, the only way to avoid x_{bad} is to **never enter** x_1 .
- The transition leading into x_1 is $a \in \Sigma_c$. The supervisor must preemptively disable a at initial state x_0 .

The Enabling Map γ

$$\gamma(\varepsilon) = \Sigma_{uc} \cup \{b\} = \{u, b\} \implies a \text{ is disabled at } x_0$$

The target sequence ac must be sacrificed to guarantee safety. The remaining language $\{bd\}$ is controllable, nonblocking, and maximal: $\sup \mathcal{C}(K) = \{bd\}$.

Algorithmic Computation of $\sup \mathcal{C}(K)$: Setup

State-Space Setup: Tracking Plant and Specification

Let $H = (X_H, \Sigma, \xi_H, x_{0,H}, X_{m,H})$ generate specification $K \subseteq L_m(G)$ over shared alphabet Σ .

- Construct the synchronous product automaton: $G_{cand} := G \parallel H = (X, \Sigma, \xi, x_0, X_m)$
- Each state $q = (x, y) \in X \subseteq X_G \times X_H$ tracks physical plant state x and logical specification state y simultaneously.
- G_{cand} restricts plant transitions strictly to those currently permitted by the specification:

$$L(G_{cand}) = L(G) \cap L(H) = \bar{K}$$

Origin of Controllability Conflicts

A combined state (x, y) is an **immediate violation state** if the physical hardware enables an autonomous event $u \in \Sigma_{uc}$ that the specification forbids: $\xi_G(x, u)!$ and $\xi_H(y, u)$ is undefined. Because u cannot be disabled by a supervisor, reaching (x, y) means safety cannot be guaranteed.

Algorithmic Computation of $\sup \mathcal{C}(K)$: State Pruning

Fixed-Point State-Removal Procedure

We compute the set of hazard states on G_{cand} through backward reachability along Σ_{uc} :

- 1 Base Set (Immediate Leaks):** Identify states with missing uncontrollable transitions:

$$X_{bad}^{(0)} = \{(x, y) \in X \mid \exists u \in \Sigma_{uc} \text{ s.t. } \xi_G(x, u) \text{! and } \xi((x, y), u) \text{ is undefined}\}$$

- 2 Uncontrollable Pre-Image (Backward Propagation):** Add states from which the plant can autonomously drift into a known bad state:

$$X_{bad}^{(k+1)} = X_{bad}^{(k)} \cup \left\{ q \in X \mid \exists u \in \Sigma_{uc} \text{ s.t. } \xi(q, u) \in X_{bad}^{(k)} \right\}$$

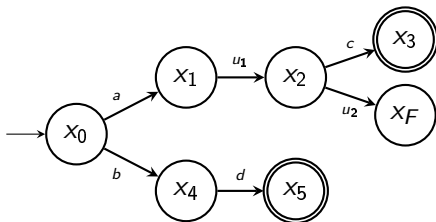
- 3 Convergence:** Repeat until $X_{bad}^{(k+1)} = X_{bad}^{(k)} =: X_{bad}^*$ (guaranteed in finite steps as $|X| < \infty$).

Final Clean-Up

Remove all states in X_{bad}^* and their incident edges. Then eliminate any states that become **unreachable** from x_0 or **blocking** (cannot reach X_m).

Worked Example: Multi-Step State Pruning (Setup)

Plant Automaton G



$$\Sigma_c = \{a, b, c, d\}, \quad \Sigma_{uc} = \{u_1, u_2\}$$

Safety Specification H

- State x_F represents a hazardous failure.
- The user requires reaching either good task x_3 (via c) or x_5 (via d).
- Desired language:

$$K = \{au_1c, bd\}$$

Controllability Conflict

Trace $au_1 \in \overline{K}$ leads to state x_2 . At x_2 , the plant can execute $u_2 \in \Sigma_{uc}$ into forbidden state x_F . Since u_2 cannot be disabled, K is not controllable.

Worked Example: Multi-Step State Pruning (Algorithm)

Step-by-Step Pruning of $G_{cand} = G \parallel H$

We compute the uncontrollable attractor X_{bad}^* backwards from forbidden paths:

- 1 Iteration $k = 0$ (Immediate Leak):** At state x_2 , plant transition $\xi_G(x_2, u_2) = x_F$ exists, but u_2 is disallowed in H : $X_{bad}^{(0)} = \{x_2\}$
- 2 Iteration $k = 1$ (1-Step Backward Reachability along Σ_{uc}):** Check states that can reach $X_{bad}^{(0)}$ via an uncontrollable transition:

$$x_1 \xrightarrow{u_1} x_2 \in X_{bad}^{(0)} \quad \text{with } u_1 \in \Sigma_{uc} \implies X_{bad}^{(1)} = \{x_2\} \cup \{x_1\} = \{x_1, x_2\}$$

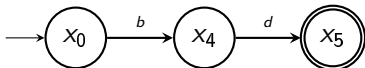
- 3 Iteration $k = 2$ (Fixed Point Evaluation):** The incoming edge to x_1 is $x_0 \xrightarrow{a} x_1$. Since $a \in \Sigma_c$ (controllable), x_0 does not enter X_{bad} : $X_{bad}^{(2)} = X_{bad}^{(1)} = \{x_1, x_2\} =: X_{bad}^*$

Key Insight

Although x_1 itself had no direct hazard, it is an **unavoidable pipeline**: entering x_1 permits the autonomous cascade $u_1 \rightarrow u_2$ into disaster.

Worked Example: Synthesis of $\sup \mathcal{C}(K)$

Optimal Controlled System S/G



$$\begin{aligned}\sup \mathcal{C}(K) &= \{bd\} \\ \overline{\sup \mathcal{C}(K)} &= \{\varepsilon, b, bd\}\end{aligned}$$

Removal & Clean-Up

- **Delete X_{bad}^* :** Remove states $\{x_1, x_2\}$ and incident edges.
- **Reachability Trim:** State x_3 becomes completely unreachable from x_0 ; remove x_3 .
- **Nonblocking Trim:** No dead-end states remain in the surviving sub-automaton.

Canonical Enabling Map γ

At root x_0 , the supervisor disables the controllable event a :

$$\gamma(\varepsilon) = \Sigma_{uc} \cup \{b\} = \{u_1, u_2, b\} \implies a \text{ is disabled}$$

The resulting system is nonblocking, strictly safe, and maximally permissive.

Partial Observation — Motivation

The Sensor Limitation

Previously, we assumed the supervisor possesses **full state awareness**:

$$\Sigma_o = \Sigma \implies \text{every event execution is instantly detected.}$$

In real industrial cyber-physical systems, sensors are limited, costly, or noisy:

- Internal slip, silent hardware faults, and remote network actions may produce no sensor trigger.
- The supervisor must make control decisions based solely on **observed records**.

Orthogonal Alphabet Partitions

The event set Σ is partitioned along two independent engineering axes:

- 1 **Actuation**: $\Sigma = \Sigma_c \dot{\cup} \Sigma_{uc}$ (can we disable it?)
- 2 **Sensing**: $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ (can we observe it?)

All four combinations ($\Sigma_c \cap \Sigma_o$, $\Sigma_c \cap \Sigma_{uo}$, $\Sigma_{uc} \cap \Sigma_o$, $\Sigma_{uc} \cap \Sigma_{uo}$) are physically possible.

The Natural Projection and Information Mask

Natural Projection $P : \Sigma^* \rightarrow \Sigma_o^*$

The observation mask is formalized by the string projection homomorphism P :

$$P(\varepsilon) = \varepsilon, \quad P(\sigma) = \begin{cases} \sigma & \text{if } \sigma \in \Sigma_o \\ \varepsilon & \text{if } \sigma \in \Sigma_{uo} \end{cases}, \quad P(s\sigma) = P(s)P(\sigma)$$

P simply erases all occurrences of unobservable events from the physical trace s .

The Control Consistency Constraint

Because the supervisor only perceives $P(s)$, its control decision map must act on observed traces:

$$\gamma : P(L(G)) \rightarrow 2^\Sigma$$

Fundamental consequence: If two distinct physical traces $s, s' \in L(G)$ produce the same observation record:

$$P(s) = P(s') \implies \gamma(P(s)) = \gamma(P(s'))$$

The supervisor is **forced to apply the exact same control action** to both situations!

The Observability Condition

Definition: Observability (Lin & Wonham, 1988)

A language $K \subseteq L_m(G)$ is **observable** with respect to G , P , and Σ_c if for all $s, s' \in \overline{K}$ and all $\sigma \in \Sigma_c$:

$$(P(s) = P(s') \wedge s\sigma \in \overline{K} \wedge s'\sigma \in L(G)) \implies s'\sigma \in \overline{K}$$

Deconstructing the Condition: The Ambiguity Trap

Observability forbids control confusion between indistinguishable histories:

- Traces s and s' look identical to the sensors ($P(s) = P(s')$).
- Along trace s , event $\sigma \in \Sigma_c$ is legal and desirable ($s\sigma \in \overline{K}$).
- Along trace s' , event σ is physically possible in the plant ($s'\sigma \in L(G)$), but leads to an illegal violation ($s'\sigma \notin \overline{K}$).

If this occurs, the supervisor faces an **unresolvable conflict**: enabling σ risks illegal behavior along s' , but disabling σ unnecessarily blocks safe progress along s .

The Fundamental Theorem with Partial Observation

Theorem (Lin & Wonham, 1988)

Let $K \subseteq L_m(G)$ with $K \neq \emptyset$. There exists a nonblocking supervisor S under projection P such that:

$$L(S/G) = \overline{K} \quad \text{and} \quad L_m(S/G) = K$$

if and only if the following three conditions hold simultaneously:

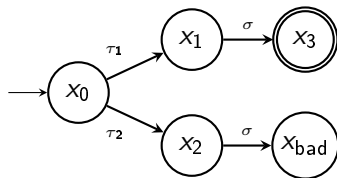
- 1 **Controllability:** $\overline{K} \Sigma_{uc} \cap L(G) \subseteq \overline{K}$
- 2 **$L_m(G)$ -Relative Closure:** $K = \overline{K} \cap L_m(G)$
- 3 **Observability:** K is observable with respect to G , P , and Σ_c .

Architectural Triad

- **Controllability:** Actuators are strong enough to prevent autonomous leaks.
- **Relative Closure:** Plant hardware does not declare false cycle completions.
- **Observability:** Sensors provide sufficient detail to disambiguate control actions.

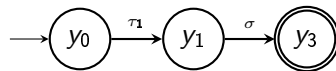
Example: The Sensor Ambiguity Dilemma (Setup)

Plant Automaton G



$$\Sigma_c = \Sigma_o = \{\sigma\}, \quad \Sigma_{uc} = \Sigma_{uo} = \{\tau_1, \tau_2\}$$

Desired Specification H



$$K = L_m(H) = \{\tau_1\sigma\}$$

Branch τ_2 must not execute σ .

Observability Breakdown

- Consider traces $s = \tau_1$ and $s' = \tau_2$ with controllable event σ : $P(\tau_1) = \varepsilon = P(\tau_2)$
- Desired: $s\sigma = \tau_1\sigma \in \overline{K}$. Illegal in plant: $s'\sigma = \tau_2\sigma \in L(G) \setminus \overline{K}$.
- The supervisor observes ε in both cases. It cannot distinguish whether the system is at x_1 (safe) or x_2 (danger).

The Non-Supremal Dilemma and Normality

Crucial Theoretical Obstacle: Observability Fails Union Closure

If $L_1, L_2 \subseteq K$ are observable, their union $L_1 \cup L_2$ is **not necessarily observable**:

- Consequently, the set of controllable and observable sublanguages **does not have a unique maximum element** in general!
- A unique $\sup \mathcal{CO}(K)$ does not exist; there are multiple, incomparable maximal sublanguages.

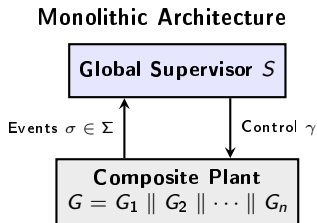
The Tractable Alternative: Normality

To recover union closure, Lin and Wonham introduced a stronger condition called **Normality**:

$$\overline{K} = P^{-1}(P(\overline{K})) \cap L(G)$$

- **Property 1:** Normality implies Observability (when controllable).
- **Property 2:** Normality is **closed under arbitrary unions**!
- A unique **supremal normal sublanguage** $\sup \mathcal{N}(K)$ always exists and can be computed via polynomial-time algorithms.

The Scalability Bottleneck: Monolithic Control



State-Space Explosion

If a plant has n subsystems with m states each:

$$|X_G| \approx m^n$$

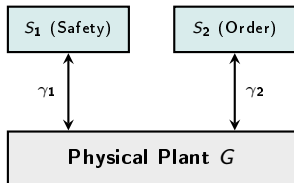
Ten 5-state robots $\Rightarrow 5^{10} \approx 10^7$ states!

Why Monolithic Fails in Practice

- **Intractable Synthesis:** Computing synchronous products and backward reachability exhausts memory.
- **Zero Modularity:** Modifying a single conveyor or sensor requires recomputing the entire global supervisor from scratch.
- **Wiring & Telemetry Cost:** Every sensor event must be routed back to a single centralized computer.
- **Single Point of Failure:** The entire manufacturing line halts if the central controller disconnects.

Modular Supervisory Control: Divide and Conquer

Modular Scheme (Specification-Driven)



Conjunctive Control Action

An actuator event is enabled if and only if **every supervisor agrees**:

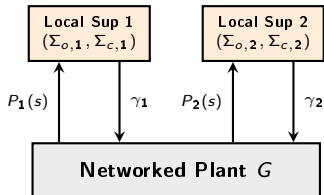
$$\gamma(s) = \gamma_1(s) \cap \gamma_2(s) \cap \cdots \cap \gamma_k(s)$$

Core Design Philosophy

- Instead of one monolithic requirement K , break down the goals into simple, independent rules:
 - S_1 : Mutual exclusion on a shared table.
 - S_2 : Buffer capacity limits.
 - S_3 : Alternating production sequence.
- Each supervisor S_i is tiny, inspects only the events it cares about, and is trivial to synthesize.
- **The Catch**: Individual supervisors may be nonblocking on their own, but their joint action can cause **deadlock** (conflicting disablements).

Decentralized Supervisory Control: Spatial Distribution

Decentralized Scheme (Locality)



Decision Fusion

How to combine commands without a central coordinator?

- **Conjunctively:** Default is enable; veto if anyone forbids.
- **Disjunctively:** Default is disable; fire if anyone authorizes.

Local Information, Global Safety

- **Physical Reality:** Stations are geographically separated (e.g., automated guided vehicles, distributed assembly cells).
- **Partitioned Channels:** Each local controller S_i has its own private sensors ($\Sigma_{o,i}$) and actuators ($\Sigma_{c,i}$).
- Controllers do not exchange the full execution history: they operate using local feedback.
- **Core Challenge (Co-observability):** Can two local supervisors ensure a global safety rule when neither agent has the complete picture?

Architectures at a Glance: Comparative Overview

Dimension	Monolithic Control	Modular Control	Decentralized Control
Architecture & Topology	Single central controller with complete plant access.	Multiple specialized supervisors running in parallel on one platform.	Geographically distributed agents on local fieldbuses or subnets.
Synthesis & Computation	Intractable on large lines ($ X \approx m^n$ state explosion).	Highly tractable; each S_i is synthesized over small local alphabets.	Parallel local synthesis; complexity decoupled across plant stations.
Coordination Principle	Global enabling map $\gamma(s)$ generated centrally.	Conjunctive fusion $\gamma = \bigcap \gamma_i$; every module must grant consent.	Local maps $\gamma_i(P_i(s))$ fused via veto or permission rules.
Key Theoretical Challenge	Memory and runtime limits during state exploration.	Inter-module blocking: conflicting disablements cause deadlock.	Co-observability: avoiding conflicting actions under partial sensing.

Chapter 3: Key Takeaways

The Exact Synthesis Boundary

- **Controllability** ($\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$): Actuators must disable events before the plant can autonomously drift into forbidden behavior.
- **$L_m(G)$ -Relative Closure** ($K = \overline{K} \cap L_m(G)$): Eliminates blocking by ensuring marked physical states match valid specification completions.

Optimal Approximation via $\sup \mathcal{C}(K)$

- Union closure guarantees a **unique, maximally permissive** sublanguage.
- The fixed-point alg. iteratively prunes bad states finding the point where controllable events must step in.

From Monolithic Theory to Industrial Reality

- **Partial Observation**: Limited sensors introduce the risk of control ambiguity ($P(s) = P(s')$), solved by sensor placement or normality constraints.
- **Architectural Scaling**: Modular and decentralized schemes break the combinatorial explosion of $G_1 \parallel \dots \parallel G_n$, trading centralized computation for non-conflict verification.

Looking Ahead: Part 4 — Observation, Diagnosis & Opacity

What We Have Mastered So Far

- **Closed-loop synthesis:** How to restrict plant behavior to ensure safety and avoid blocking.
- **The complete-information baseline:** Controlling when state is fully known or coarsely projected.

The New Paradigm: Information-Inference under Masked Events

Instead of asking *"how do we control?"*, we now ask *"what can we reconstruct from partial traces?"*:

- **Asymmetric observation:** The system executes silent transitions (Σ_{uo}) while an external entity (supervisor, health monitor, or intruder) observes only $P(s) \in \Sigma_o^*$.
- **Set-valued inference:** Reconstructing current states and event histories from observation records.

The Three Pillars of Part 4

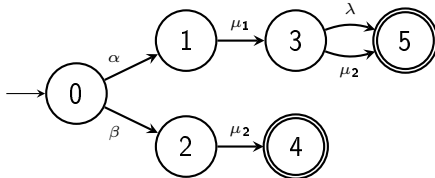
- 1 **State Estimation & Observers:** Deterministic observer automata and current-state uncertainty sets.
- 2 **Fault Diagnosis (Sampath's Diagnoser):** Detecting unobservable failure events ($f \in \Sigma_f \subseteq \Sigma_{uo}$).
- 3 **Security & Opacity:** Keeping secret states and secret behaviors hidden from an adversarial observer (current-state opacity, language opacity, and defense synthesis).

Part 3: Theoretical Review Questions

- 1 **Control Authority:** Why does the controllability condition $\overline{K}\Sigma_{uc} \cap L(G) \subseteq \overline{K}$ not impose any restriction on controllable events Σ_c ? How does the enabling map γ handle them?
- 2 **Blocking vs. Marking:** Explain the difference between an automaton being physically deadlock-free and the closed-loop system S/G being nonblocking ($L_m(S/G) = \overline{L_m(S/G)}$).
- 3 **Existence of the Supremum:** Why does the property of closure under arbitrary unions guarantee the existence of a unique $\sup \mathcal{C}(K)$? What breaks down when we move to partial observation ($\sup \mathcal{CO}(K)$)?
- 4 **Pruning Horizon:** In the state-pruning algorithm, why is it insufficient to delete only the states that directly enable forbidden transitions?
- 5 **Modular Deadlock:** If two modular supervisors S_1 and S_2 are individually nonblocking with respect to G , why can the joint system $(S_1 \parallel S_2)/G$ block?
- 6 **Observability Ambiguity:** Formulate in plain words what happens when two plant traces satisfy $P(s) = P(s')$ but require different control actions for some $\sigma \in \Sigma_c$.

Practice Exercise: Computation of $\sup \mathcal{C}(K)$

Plant Automaton G



$$\Sigma_c = \{\alpha, \beta, \lambda\}, \quad \Sigma_{uc} = \{\mu_1, \mu_2\}$$

$$X_m = \{4, 5\}$$

Specification Requirement K

The customer imposes two operational rules:

- Event β is completely forbidden.
- Transition $3 \xrightarrow{\mu_2} 5$ is an emergency shutdown and must **never** occur.

Desired target: $K = \{\alpha\mu_1\lambda\}$.

Tasks for the Student

- 1 Show that K is **not controllable**.
- 2 Apply the fixed-point algorithm on $G \parallel H$ to determine X_{bad}^* .
- 3 Compute $\sup \mathcal{C}(K)$ and the feedback map γ .
- 4 Is the resulting supervisor trivial (empty)? Justify.