

Supervisory Control of Discrete Event Systems

Part 1: Foundations and Linguistic Models

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

Loughborough University

Key Facts & Academic Excellence

- **Engineering Heritage:** Renowned globally for excellence in engineering education and applied research.
- **Wolfson School:** a leading center for automation, cyber-physical systems, and Industry 4.0.
- **Rankings & Accreditations:** Consistently ranked among the UK top 10 universities with top-tier student satisfaction and employer partnerships. Ranked first in the world for sports-related subjects.



Module Aim & Learning Objectives

Module Aim

This module provides a rigorous mathematical and algorithmic foundation for the modeling, qualitative analysis, and control of **Discrete Event Systems (DES)**. By shifting from continuous-time differential dynamics to logical, event-driven abstractions, students will master the **Ramadge-Wonham Supervisory Control Theory (SCT)**. The ultimate goal is to enable the synthesis of **correct-by-construction**, nonblocking supervisory controllers capable of enforcing formal safety, liveness, and coordination specifications in complex automated, robotic, and cyber-physical systems.

Module Aim & Learning Objectives

Theoretical Objectives:

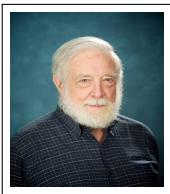
- **Formal Modeling:** Master formal languages, prefix closures, and finite-state automata (G).
- **System Composition:** Model concurrent interactions via synchronous parallel product ($G_1 \parallel G_2$).
- **Controllability Theory:** Analyze language controllability and nonblocking conditions ($\overline{L_m} = L$).

Applied & Synthesis Objectives:

- **Optimal Synthesis:** Implement the fixpoint algorithm to compute the supremal controllable sublanguage $\sup \mathcal{C}(E)$.
- **Safety & Liveness:** Guarantee dead/livelock avoidance and mutual exclusion in industrial processes.
- **Scalability:** Understand modular, decentralized, and hierarchical control architectures.

Pioneers & Mentors of Supervisory Control Theory

Foundational contributors to the Ramadge-Wonham (RW) framework and modern DES



W. Murray Wonham

Univ. of Toronto

Framework Origin

Co-founder of SCT, controllable languages, sup C , and modular/hierarchical control.



Peter J. Ramadge

Princeton University

Framework Origin

Seminal RW framework (1987–1989), feedback loop modeling, and state-feedback logic.



Stéphane Lafortune

Univ. of Michigan

Diagnosis & Opacity

Partial observation, fault diagnosers, opacity & cyber-security, and industrial applications.



Feng Lin

Wayne State Univ.

Decentralized Control

Pioneer of observability & co-observability, distributed SCT, and industrial applications.

Core References & Reading List

Primary Textbook (Required Reading)

Cassandras, C.G. and Lafortune, S., 2021. *Introduction to Discrete Event Systems*. 3rd ed. Cham: Springer Nature Switzerland. Chapters 1-3.

Secondary & Advanced References

- Wonham, W.M. and Cai, K., 2018. *Supervisory Control of Discrete-Event Systems*. Monograph. Toronto: Systems Control Group, University of Toronto. Available from: https://www.researchgate.net/profile/Walter-Wonham/publication/319434005_Supervisory_Control_of_Discrete-Event_Systems_v20170901
- Lin, F. and Wonham, W.M., 1988. Decentralized supervisory control of discrete-event systems. *Information Sciences*, 44(3), pp. 199–224.

Outline

- 1 Introduction
- 2 Continuous-Variable vs. Discrete-Event Dynamical Systems (CVDS vs. DEDS)
- 3 Formal Characterization of DES
- 4 Linguistic Models of DES
- 5 Conclusions
- 6 Review Questions & Self-Assessment

Part 1: Intended Learning Outcomes (ILOs)

- 1 Define and understand DES dynamics:** Define discrete event systems and explain how event-driven dynamics fundamentally differ from time-driven CVDS.
- 2 Classify DES modeling abstractions:** Identify and differentiate between the three primary modeling layers (*Logical*, *Timed*, and *Stochastic*) and the specific behavioral questions each layer addresses.
- 3 Understand linguistic models:** Formalize system behavior using alphabets, traces, and prefix-closed execution languages, including the physical partition of events.
- 4 Apply linguistic operations:** Compute and manipulate language closures, intersections, and natural projections to represent system trajectories, safety specifications, and sensory uncertainty.
- 5 Understand limitations & motivate language generators:** Analyze the theoretical and computational limits of pure string languages for infinite/cyclic behaviors and justify the necessity of state-based generators (automata).

System Modeling: Continuous-Variable Dynamical Systems (CVDS)

Core Concept

Systems where the state changes **continuously** with respect to physical time t . The behavior is primarily governed by the laws of physics (mechanics, thermodynamics, electromagnetism).

Mathematical Framework:

- **State Space:** Typically $\mathbb{R}^n$.
- **Evolution:** Differential equations (CT) or Difference equations (DT):

$$\dot{x}(t) = f(x(t), u(t), t)$$

- **Dynamics:** **Time-driven** dynamics.

Typical Examples

- Robotic arm trajectories
- Chemical reactor temp.
- DC Motor speed control
- Voltage in a circuit

CVDS Example: Electrical RC Circuit

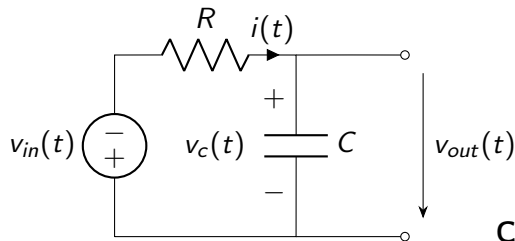


Figure: First-order RC Low-pass Filter

Governing Equation:

$$R \cdot i(t) + v_c(t) = v_{in}(t) \quad i(t) = C \frac{dv_c(t)}{dt}$$

$$RC \frac{dv_c(t)}{dt} + v_c(t) = v_{in}(t)$$

CVDS Characteristics:

- **State Variable:** $v_c(t) \in \mathbb{R}$ defined for any $t \in [0, \infty)$ (infinite resolution).
- **Forced Response:** The evolution is driven by the continuous input $v_{in}(t)$ over time.

System Modeling: Discrete-Event Dynamical Systems (DEDS)

Core Concept

Systems where the state changes **instantaneously** at asynchronous points in time due to the occurrence of specific events. The state space is symbolic rather than numerical.

Mathematical Framework:

- **State Space:** Discrete set
 $X = \{x_1, x_2, \dots\}$ or symbolic (e.g., {Idle, Busy, Failed}).
- **Evolution:** State transition functions:

$$x_{k+1} = f(x_k, e_{k+1}), \quad e \in \Sigma$$

- **Dynamics:** **Event-driven** dynamics.

Typical Examples

- Manufacturing assembly
- Network protocols
- Automated warehouses
- Software logic / OS

DEDS Example: Automated Manufacturing Cell

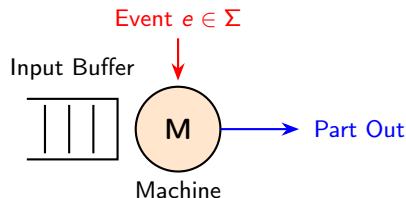


Figure: Simple Workcell Abstraction

Formal Representation:

- **Discrete State Space:** $X = \{\text{Idle}, \text{Busy}, \text{Down}\}$
- **Events:** $\Sigma = \{a, s, c, f, r\}$
 a : Arrival; s : Start; c : Complete;
 f : Failure; r : Repair.

Why is this a DES?

- **Event-Driven:** The state remains "Idle" indefinitely until event ' a ' occurs.
- **Asynchronous:** Events do not follow a clock; a failure (f) can happen at any time.
- **Logical State:** We care *if* the machine is working, not the exact RPM of its motor.

State Trajectories: Time-Driven vs. Event-Driven

CVDS Trajectory:

- Smooth, continuous signals.
- Time $t \in \mathbb{R}^+$ is the independent variable.

DES Trajectory:

- Piecewise-constant state trajectory.
- Transitions triggered by events $(e_1, e_2, \dots)$ occurring at discrete instances.
- Duration between events is irrelevant to the logical state update.

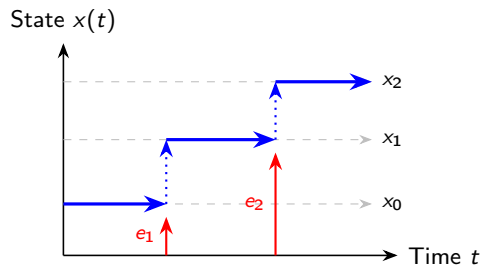
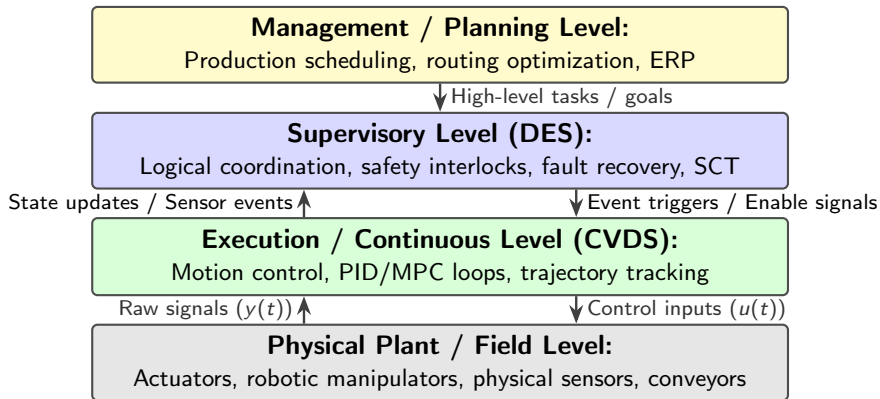


Figure: Piecewise-constant state evolution in a DES

Control Hierarchy in Complex Systems



Control Hierarchy: Key Takeaways

1. Complementary Paradigms (CVDS + DES)

- DES **does not replace** low-level continuous control (CVDS).
- It **complements** the continuous layer by orchestrating when continuous control loops are triggered, switched, or paused.
- CVDS guarantees local stability/tracking; DES ensures global logical coordination.

2. New Discrete-Level Challenges

The discrete abstraction introduces problems differential equations cannot solve:

- **Supervisory Synthesis & Controllability:** Restricting behavior under uncontrollable events (Σ_{uc}).
- **Safety & Liveness:** Avoiding deadlocks, livelocks, and unsafe states.
- **Resource Allocation:** Guaranteeing mutual exclusion without starvation.
- **Diagnosis & Opacity:** Fault detection, partial observability, and cyber-security/privacy.

How CVDS and DES Interact in Physical Systems

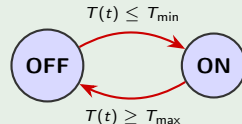
CVDS Domain ($x \in \mathbb{R}^n$):

- Continuous physical dynamics ($\dot{x} = f(x, u, t)$).
- Physical variables: temperature $T(t) \in \mathbb{R}$.

DES Supervisor ($x \in X$):

- Discrete operating modes: $X = \{\text{OFF}, \text{ON}\}$.
- **Event-Driven:** Updates state only when threshold crossings occur.

Thermostat Abstraction

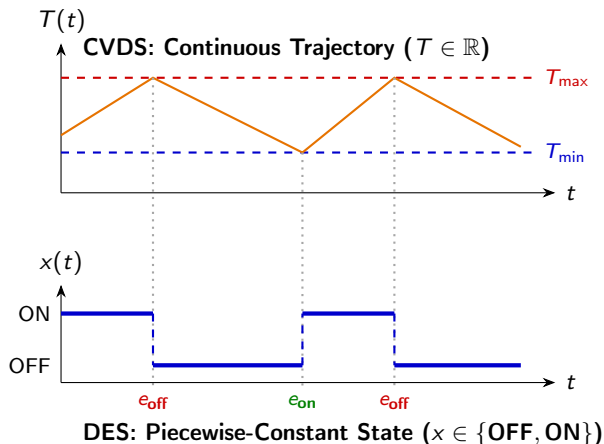


Threshold crossings generate discrete events that trigger supervisor mode transitions.

Interface Mechanism

The continuous trajectory $T(t)$ crosses $T_{\min}/T_{\max} \implies$ triggers event $e \in \Sigma \implies$ DES supervisor switches state.

How CVDS and DES Interact in Physical Systems



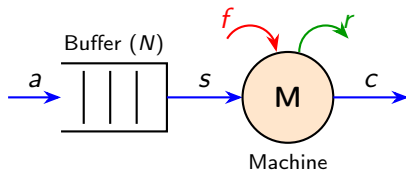
Formal Characterization of DES

Defining Characteristics

A system is classified as a **Discrete Event System (DES)** if it possesses the following three fundamental properties:

- 1 **Discrete State Space:** The state space X is a discrete set (finite or countable). Changes are qualitative, not quantitative.
- 2 **Asynchronous Transitions:** State transitions do not occur at fixed clock ticks but are triggered by **events** that can occur at arbitrary time instants.
- 3 **Instantaneous Transitions:** The duration of the event itself is ignored. The system "jumps" from x to x' at the exact moment the event occurs.

Bridging the Gap: How to Model this System?



System Overview

Capacity N , initial state: empty buffer, idle machine.

Class Discussion

If we want to build a purely logical/discrete mathematical model of this cell:

- 1 **What are the Events?**
What represents an atomic action or change?
- 2 **What is an Execution Trajectory?**
How do we record what happened over time?
- 3 **What is the System Model?**
What distinguishes valid executions from impossible ones?

The Formal Paradigm: Languages as DES Models

1. Events $\rightarrow$ Alphabet (Σ)

Events are symbols / characters:

$$\Sigma = \{a, s, c, f, r\}$$

2. Trajectory $\rightarrow$ String / Trace (w)

A sequence of observed events is a word:

$$w = a s c a \in \Sigma^*$$

Σ^* = all possible finite combinations; ε = empty trace.

3. System Model $\rightarrow$ Language (L)

Physical rules act as a **grammar**:

- $a s c \in L$ (valid physical trace)
- $s c \notin L$ (cannot start with empty buffer)

$$L \subseteq \Sigma^*$$

The DES model is the **set of all allowable strings** generated by the system grammar.

Core Takeaway

In DES, **modeling a system means defining its Language L** . System verification and supervisory control consist in guaranteeing that the system never executes forbidden words ($w \notin L_{\text{safe}}$).

Mathematical Foundation: Events and Languages

The Event Alphabet (Σ)

The set of all possible events that can occur in the system is called the **Alphabet**:

$$\Sigma = \{e_1, e_2, \dots, e_m\}$$

Strings and Sequences:

- A **string** (or trace) s is a sequence of events: $s = e_{i_1} e_{i_2} \dots e_{i_k}$.
- The set of all possible finite strings over Σ is denoted as Σ^* (Kleene Closure).
- The empty string (no events occurred) is denoted by ε .

Definition of a Language

A **Language** L over Σ is a subset of Σ^* ($L \subseteq \Sigma^*$). In DES theory, the language represents the **observed behavior** of the system.

Levels of Abstraction in DES Modeling

Depending on the control and analysis objectives, a DES can be modeled at three progressive levels of detail:

1 Logical / Untimed Level (Event-Driven):

- **Focus:** Event ordering, qualitative state transitions, logic.
- **Mathematical Tools:** Formal Languages, Automata (FSA), Petri Nets.
- **Typical Questions:** Safety, deadlock avoidance, controllability, SCT.

2 Timed Level (Deterministic Time):

- **Focus:** Exact event timestamps t_k , event clocks, delays.
- **Mathematical Tools:** Timed Automata, Max-Plus Algebra, Timed Petri Nets.
- **Typical Questions:** Worst-case execution time, real-time deadlines, throughput.

3 Stochastic / Probabilistic Level:

- **Focus:** Random event arrivals and service durations (distributions).
- **Mathematical Tools:** Markov Chains, Generalized Semi-Markov Schemes (GSMP).
- **Typical Questions:** Average waiting times, system utilization, bottleneck analysis.

Three Levels of Abstraction: Formal Examples

Running Example: Buffer-Machine Processing

Events: part arrival (a), start (s), completion (c), failure (f), repair (r).

1. Logical / Untimed

- **Trace (Order only):**
 $w = asfrc \in \Sigma^*$
- **Model Dynamics:**
 $x_{k+1} = f(x_k, e_{k+1})$
- **Key Question:**
Is recovery reachable?
(Safety / Logic)

2. Timed (Deterministic)

- **Timed Trace:**
 $(a, 0.0), (s, 1.2), (c, 4.5)$
- **Model Dynamics:**
 $t_{k+1} = t_k + d_e(x_k)$
(e.g., duration $d_s = 3.3$ s)
- **Key Question:**
Is deadline $D \leq 5.0$ s met?
(Throughput)

3. Stochastic

- **Random Variables:**
 $T_{\text{serv}} \sim \text{Exp}(\mu),$
 $T_{\text{fail}} \sim \text{Exp}(\lambda)$
- **Model Dynamics:**
 $P(x_{k+1} = j \mid x_k = i)$
- **Key Question:**
What is average queue
length $\mathbb{E}[n]$?

Key Takeaway

Moving **Logical** $\rightarrow$ **Timed** $\rightarrow$ **Stochastic** adds quantitative metrics at the cost of higher modeling complexity.

Event Taxonomy: Controllability and Observability

Event Partitioning

In real-world engineering systems, not all events have the same physical nature. The alphabet Σ is partitioned based on control and measurement capabilities.

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

- **Controllable (Σ_c):** Can be enabled or disabled by the supervisor.
Examples: start service (s), start repair (r).
- **Uncontrollable (Σ_{uc}):** Occur spontaneously; cannot be prevented by control actions.
Examples: part arrival (a), machine failure (f).

Observability: $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$

- **Observable (Σ_o):** Monitored by sensors and reported to the supervisor.
Examples: optical switch trigger, motor stop.
- **Unobservable (Σ_{uo}):** Internal/silent events without direct sensory feedback.
Examples: unmeasured component wear, silent fault.

Note: Σ_c drives *Supervisory Control (SCT)*, while Σ_o drives *Fault Diagnosis & State Estimation*.

Mathematical Foundations: Operations on Strings

Strings and Traces

A string (trace) $s \in \Sigma^*$ is a finite sequence of events. ε denotes the **empty string** ($|\varepsilon| = 0$).

1. Basic Operations:

- **Length** ($|s|$): Number of events.
 $|a s c| = 3, \quad |\varepsilon| = 0.$
- **Concatenation**: If $s = ab$ and $t = cd$, then
 $st = abcd.$
- **Identity**: $s\varepsilon = \varepsilon s = s.$

2. Prefix Relation ($\leq$):

- s is a **prefix** of w (denoted $s \leq w$) if:

$$\exists t \in \Sigma^* \text{ such that } w = st$$

- *Example for $w = a s c$:*

$$\text{Prefixes}(w) = \{\varepsilon, a, a s, a s c\}$$

Causal Physical Meaning

A physical process **cannot** execute trace $w = a s c$ without sequentially traversing all its intermediate prefixes ($\varepsilon \rightarrow a \rightarrow a s \rightarrow a s c$).

Language Operations: Prefix-Closure (Theory)

Definition: Prefix-Closure ($\bar{L}$)

Given a language $L \subseteq \Sigma^*$, its **prefix-closure** $\bar{L}$ is the set of all prefixes of all strings in L :

$$\bar{L} = \{s \in \Sigma^* \mid \exists t \in \Sigma^* \text{ such that } st \in L\}$$

Fundamental Mathematical Properties:

- **Inclusion:** $L \subseteq \bar{L}$ (every string is a prefix of itself, choosing $t = \varepsilon$).
- **Idempotence:** $\bar{\bar{L}} = \bar{L}$.
- **Prefix-Closed Language:** A language L is said to be *prefix-closed* if and only if $L = \bar{L}$.

Physical Meaning in DES

$\bar{L}$ represents **causality** and the **step-by-step evolution** of a physical trajectory. A system cannot complete a target task sequence w without sequentially traversing all intermediate prefix states.

Prefix-Closure: Theoretical Examples

Example 1: Finite Language

Let $\Sigma = \{a, b, c\}$ and $L_1 = \{ab, ac\}$.

- Prefixes of ab : $\{\varepsilon, a, ab\}$; Prefixes of ac : $\{\varepsilon, a, ac\}$
- **Prefix-Closure:** $\overline{L_1} = \{\varepsilon, a, ab, ac\}$
- Note: $L_1 \neq \overline{L_1}$, therefore L_1 is **not** prefix-closed.

Example 2: Infinite / Repetitive Language

Let $L_2 = \{(ab)^n \mid n \geq 1\} = \{ab, abab, ababab, \dots\}$.

- Even prefixes (after full cycles): $(ab)^k$ for $k \geq 0$.
- Odd prefixes (stopped mid-cycle after event a): $(ab)^k a$ for $k \geq 0$.
- **Result:**

$$\overline{L_2} = \{\varepsilon\} \cup \{(ab)^k\}_{k \geq 1} \cup \{(ab)^k a\}_{k \geq 0}$$

Language Operations: Kleene Closure (Theory)

Definition: Kleene Star (L^*)

The **Kleene closure** (or star) of a language $L \subseteq \Sigma^*$ is the set of all finite concatenations of strings from L :

$$L^* = \bigcup_{k=0}^{\infty} L^k \quad \text{with} \quad L^0 = \{\varepsilon\}, \quad L^{k+1} = L^k L$$

Closures & Variants:

- **Empty String Inclusion:** $\varepsilon \in L^*$ always holds (since $L^0 = \{\varepsilon\}$).
- **Positive Closure (L^+)** excludes empty string: $L^+ = \bigcup_{k=1}^{\infty} L^k = LL^* \implies L^* = L^+ \cup \{\varepsilon\}$

Physical Meaning in DES

L^* models **cyclic, repetitive, and unbounded processes**. It describes the system's ability to repeat a set of operations or tasks an arbitrary number of times.

Kleene Closure: Theoretical Examples

Example 1: The Universal Language (Σ^*)

Let the alphabet be $\Sigma = \{a, b\}$.

- $\Sigma^0 = \{\varepsilon\}$
- $\Sigma^1 = \{a, b\}$
- $\Sigma^2 = \{aa, ab, ba, bb\}$
- **Universal Language:** $\Sigma^* = \bigcup_{k=0}^{\infty} \Sigma^k = \{\varepsilon, a, b, aa, ab, ba, bb, \dots\}$ represents the set of **all possible finite sequences** over Σ .

Example 2: Repetition of a Single Task

Let $L = \{ab\}$ (e.g., a = start, b = finish).

- $L^* = \{\varepsilon, ab, abab, ababab, \dots\} = \{(ab)^k \mid k \geq 0\}$
- $L^+ = \{ab, abab, ababab, \dots\} = \{(ab)^k \mid k \geq 1\}$

Language Operations: Natural Projection (Theory)

Definition: Natural Projection (P)

Let $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$ be partitioned into observable and unobservable events. The projection $P : \Sigma^* \rightarrow \Sigma_o^*$ is recursively defined as:

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

where $s \in \Sigma^*$ and $\sigma \in \Sigma$.

Extension to Languages:

- For any language $L \subseteq \Sigma^*$, the projected language is: $P(L) = \{P(s) \in \Sigma_o^* \mid s \in L\}$

Physical Meaning in DES

$P(L)$ models **sensor observation and filtering**: it represents the sequence of signals actually recorded by an external supervisor or monitoring unit.

Natural Projection: Theoretical Examples

Examples: String Evaluation and Language Projection

Setup: Let $\Sigma = \{a, b, u\}$ with $\Sigma_o = \{a, b\}$ and $\Sigma_{uo} = \{u\}$.

1. Single String Evaluation:

- For trace $s = a u u b u a$:

$$P(s) = P(a)P(u)P(u)P(b)P(u)P(a) = a\varepsilon\varepsilon b\varepsilon a = a b a$$

- The supervisor only detects the observable sub-trace aba .

2. Language with Unobservable Loops:

- For $L = \{a u^k b \mid k \geq 0\} = \{ab, aub, auub, \dots\}$:

$$P(L) = \{P(a u^k b) \mid k \geq 0\} = \{ab\}$$

- Silent events collapse to ε , rendering iterations indistinguishable.

Language Operations: Inverse Projection (Theory)

Definition: Inverse Projection (P^{-1})

Given a language over observable events $K \subseteq \Sigma_o^*$, the **inverse projection** $P^{-1} : 2^{\Sigma_o^*} \rightarrow 2^{\Sigma^*}$ is defined as:

$$P^{-1}(K) = \{s \in \Sigma^* \mid P(s) \in K\}$$

Fundamental Properties:

- For a single symbol $\sigma \in \Sigma_o$: $P^{-1}(\{\sigma\}) = \Sigma_{uo}^* \sigma \Sigma_{uo}^*$.
- For the empty string: $P^{-1}(\{\varepsilon\}) = \Sigma_{uo}^*$.
- In general, $L \subseteq P^{-1}(P(L))$.

Physical Meaning in DES

$P^{-1}(K)$ models **state estimation and diagnostic uncertainty**: it reconstructs all possible unobservable event sequences consistent with an observed trace.

Inverse Projection: Theoretical Examples

Inverse Projection Analysis

Setup: Let $\Sigma = \{a, b, u\}$ with $\Sigma_o = \{a, b\}$ and $\Sigma_{uo} = \{u\}$.

1. Reconstructing a Single Observed String:

- Let the observed sequence be $t = ab \in \Sigma_o^*$:

$$P^{-1}(\{ab\}) = \Sigma_{uo}^* a \Sigma_{uo}^* b \Sigma_{uo}^* = u^* a u^* b u^*$$

- Any number of silent events u could have occurred before, between, or after the observable events a and b .

2. Observation Uncertainty Set:

- If the system generates $s = a u b$, the sensor sees $P(s) = ab$.
- The supervisor's uncertainty set is $P^{-1}(P(\{s\})) = u^* a u^* b u^*$.
- This set includes all candidate traces compatible with the observation, such as $\{ab, uab, aub, aubu, \dots\}$.

The Need for Language Generators

Limits of Pure Language Models:

- **Infinite Cardinality:** Repetitive or cyclic behaviors generate infinite sets of strings ($|L| = \infty$).
- **No Finite Enumeration:** Cannot verify properties or synthesize supervisors string-by-string.
- **Implicit Memory:** A string describes the trace, but does not provide an explicit summary of current capacity.

State-Based Generators:

- **State as Equivalence Class:** The current state summarizes all past history needed to decide future events.
- **Finite Representation:** Compact structures generating potentially infinite languages ($L(M) \subseteq \Sigma^*$).
- **Two Main DES Formalisms:**
 - **Automata (FSA):** Explicit state enumeration.
 - **Petri Nets (PN):** Distributed/vectorial states (markings), ideal for concurrency.

From Denotative to Generative Modeling

We transition from **denotative descriptions** ($L \subseteq \Sigma^*$) to **generative state-space mechanisms** that produce strings through discrete transitions.

Generative Models: Deterministic Automaton

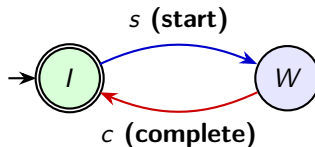
Formal Definition: The 6-Tuple (G)

A Deterministic Finite-State Automaton (DFA) is defined as:

$$G = (X, \Sigma, f, \Gamma, x_0, X_m)$$

- **State Space (X):** Finite set of discrete states.
- **Event Alphabet (Σ):** Finite set of discrete events.
- **Transition Function (f):** Partial function $f : X \times \Sigma \rightarrow X$.
 $f(x, e) = y$ means event e causes a transition from state x to state y .
- **Active Event Set (Γ):** $\Gamma(x) \subseteq \Sigma$ is the set of events feasible at state x .
- **Initial State (x_0):** The starting state ($x_0 \in X$).
- **Marked States (X_m):** $X_m \subseteq X$ denotes target, terminal, or completed states.

Deterministic Automaton: Minimal Example



6-Tuple Mapping & Generated Languages

- **States & Alphabet:** $X = \{I, W\}$, $\Sigma = \{s, c\}$
- **Initial & Marked:** $x_0 = I$, $X_m = \{I\}$ (double circle indicates idle/completed)
- **Active Sets & Transitions:** $\Gamma(I) = \{s\}$, $\Gamma(W) = \{c\}$; $f(I, s) = W$, $f(W, c) = I$
- **Generated Language $L(G)$ (All possible execution traces):**

$$L(G) = \overline{\{(sc)^n \mid n \geq 1\}} = \{\varepsilon, s, sc, scs, scsc, \dots\}$$

Key Takeaways: Foundations & Linguistic Models

1. Event-Driven Dynamics vs Continuous Systems

- DES evolution is governed by **asynchronous, discrete events** rather than the continuous flow of time ($t \in \mathbb{R}$).
- Abstraction hierarchy: **Logical** (order) $\rightarrow$ **Timed** (delays) $\rightarrow$ **Stochastic** (rates).

2. The Linguistic Formalism

- **Alphabets & Traces:** Discrete actions Σ concatenate into execution traces $w \in \Sigma^*$.
- **Causality is Prefix-Closed:** Physical execution requires $L = \bar{L}$.
- **Control & Diagnosis:** Enabled via event partitioning ($\Sigma_c \cup \Sigma_{uc}$) and projection $P(L)$ ($\Sigma_o \cup \Sigma_{uo}$).

3. Why We Need Automata

- Pure languages cannot be computationally manipulated when $|L| = \infty$.
- State machines $G = (X, \Sigma, f, \Gamma, x_0, X_m)$ provide a **finite, compact generator** distinguishing ongoing physical behavior $L(G)$ from completed tasks $L_m(G)$.

Roadmap: What Comes Next?

Next Module: Deterministic Automata & Operations

Having established the linguistic and state-space foundations, we will develop the algebraic and algorithmic toolkit for system-level analysis:

1 Advanced Language Properties:

- Extended transition function $f(x, s)$ on arbitrary words $s \in \Sigma^*$.
- Formal definition of **Deadlock**, **Livelock**, and **Non-blocking Automata** ($\overline{L_m(G)} = L(G)$).

2 Automata Operations & Transformations:

- State reachability algorithms: $\text{Ac}(G)$, $\text{CoAc}(G)$, and $\text{Trim}(G)$.

3 Compositional Modeling (Interconnected Systems):

- **Product Composition** ($G_1 \times G_2$): Lock-step full synchronization.
- **Parallel Composition** ($G_1 \parallel G_2$): Concurrent execution with shared event synchronization (the foundation of SCT plant networks).

Final Objective

Moving from single components to **complex networked systems** ready for Supervisory Control Synthesis.

Part 1: Theoretical Review Questions

- 1 Definition of a DES:** Define a Discrete Event System (DES). What fundamentally distinguishes its state space and state transition mechanism from Continuous Variable Dynamic Systems (CVDS)?
- 2 Modeling Abstraction Levels:** List the three modeling levels of a DES. What qualitative questions can be answered strictly at the **logical level** without knowledge of time durations or transition probabilities?
- 3 Linguistic Representation:** Given an alphabet Σ , what is a formal language $L \subseteq \Sigma^*$? Explain why the physically generated language of a real system must be prefix-closed.
- 4 Prefix & Kleene Closures:** Define the **prefix-closure** ($\bar{L}$) and **Kleene star** (L^*). What distinct physical behaviors do they capture?
- 5 Language Operations (Projection):** Given the partition $\Sigma = \Sigma_o \dot{\cup} \Sigma_{uo}$, define the natural projection P and its inverse P^{-1} . What physical control and diagnostic concepts do $P(L)$ and $P^{-1}(P(w))$ capture?
- 6 Limitations of Pure Linguistic Models:** Discuss why pure language models ($L \subseteq \Sigma^*$) are insufficient for computational verification and synthesis, motivating the need for state-based generators (G).

Part 1: Practice Exercises

Exercise 1: System Modeling

Two AGVs (V_1, V_2) share an intersection. Events: e_i = enter intersection, x_i = exit intersection ($i \in \{1, 2\}$).

- Define the event alphabet Σ .
- Write a logical formal specification language $E_{safe} \subset \Sigma^*$ preventing collision.

Exercise 2: Prefix-Closure Computation

Let $\Sigma = \{a, b, c\}$ and consider the language: $L = \{a^n bc^n \mid n \geq 1\}$

- Find the prefix-closure $\bar{L}$.
- Is L prefix-closed? Why?

Exercise 3: Natural & Inverse Projection

Let $\Sigma = \{a, b, u\}$ with $\Sigma_o = \{a, b\}$ and $\Sigma_{uo} = \{u\}$. Given: $L = \{aub, auub, bua\}$

- Compute $P(L)$.
- Compute $P^{-1}(P(L))$ and show that $L \subset P^{-1}(P(L))$.