FROM PROGRAMMABLE LOGIC CONTROL (PLC) TO DISCRETE EVENT SYSTEMS (DES)

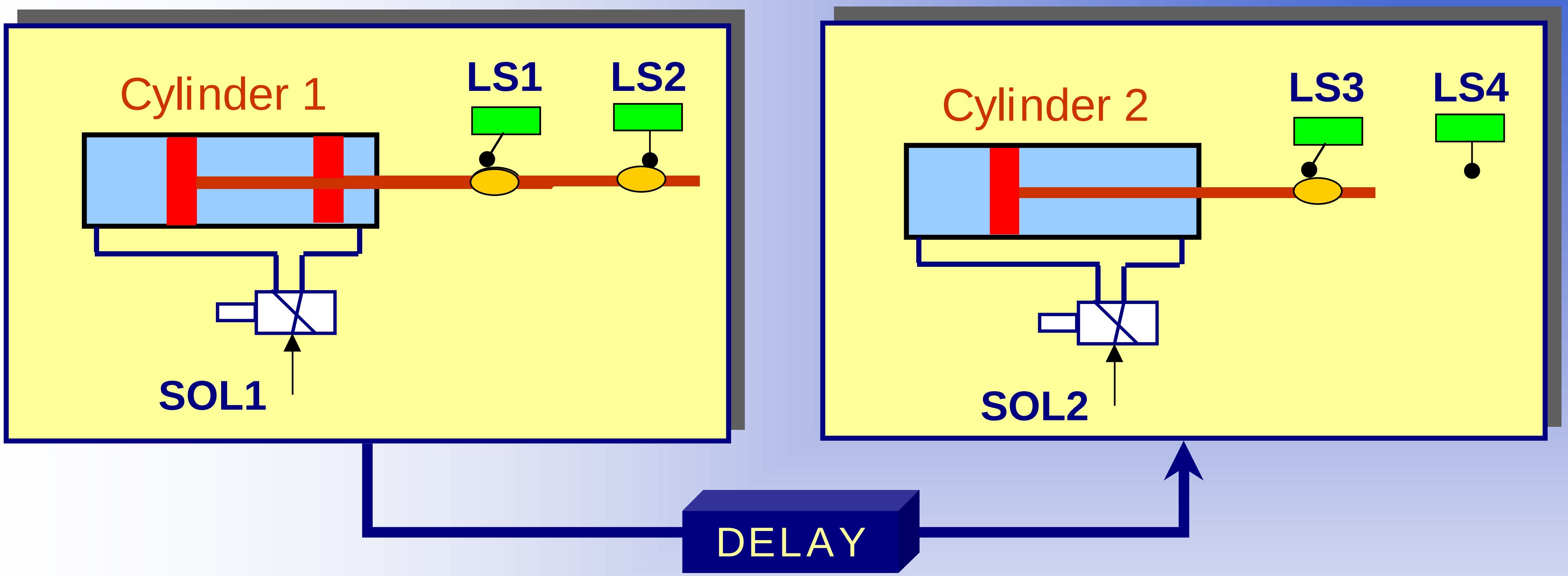
C. G. Cassandras
Dept. of Manufacturing Engineering
Boston University
Boston, MA 02215
cgc@bu.edu
http://vita.bu.edu/cgc

OUTLINE

- A MOTIVATING EXAMPLE
- DES MODELING:

  STATE AUTOMATA
  (STATE MACHINES)
- SUPERVISORY CONTROL
- 'RAPID LEARNING'
- RAPID RECONFIGURATION TECHNIQUES

A MOTIVATING EXAMPLE



-**START** initiates cycle with **LS1** and **LS3** ON

- **SOL1** activated, piston 1 moves right

- **LS2** triggered, **SOL1** deactivated, piston 1 moves left

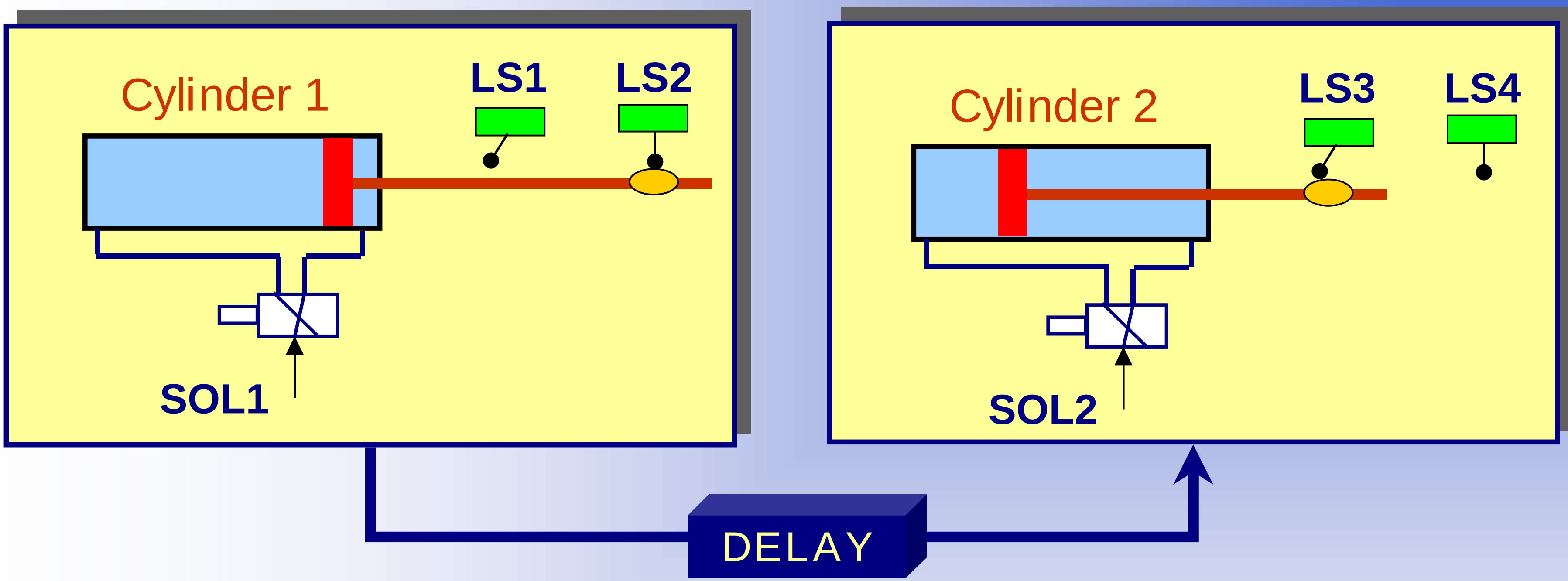
- Time delay of 1 sec.

- **SOL2** activated, piston 2 moves right

- **LS2** triggered, **SOL1** deactivated, piston 2 moves left

- **STOP** resets everything to rest

EXAMPLE – PLC APPROACH



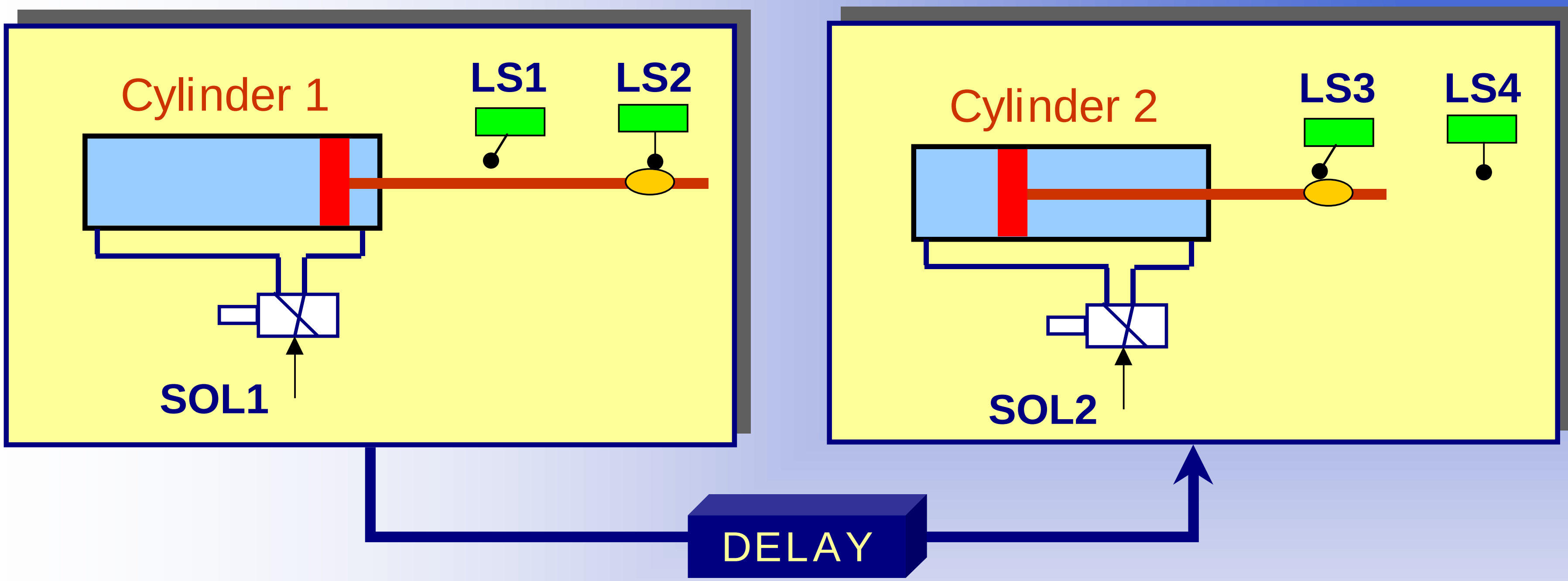
Cylinder 1 control: $SOL1 = (SOL1 + START \cdot LS1 \cdot LS3) \cdot \overline{LS2} \cdot \overline{STOP}$

ACTIVATE

DEACTIVAT

NOTE: Upon deactivation of **SOL1**, the system looks IDENTICAL as at rest, yet it must know to activate the **DELAY** function.

EXAMPLE – PLC APPROACH *CONTINUED*



Solution: Introduce some memory…

$$DONE = (DONE + LS2)\cdot\overline{LS4}\cdot\overline{STOP}$$

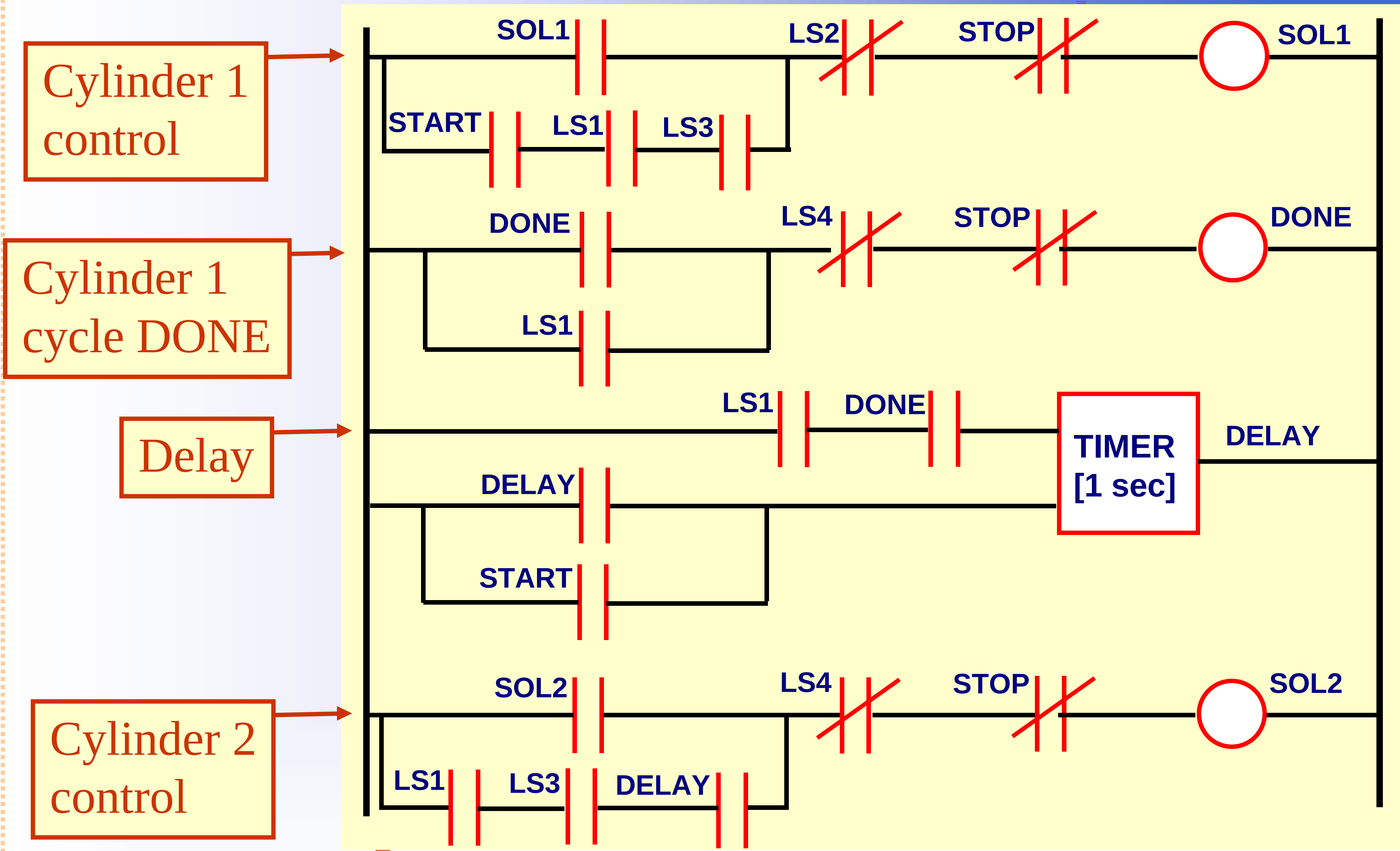
INDICATES CYLINDER1 CYCLE DONE

$$TIMER = LS1\cdot DONE$$

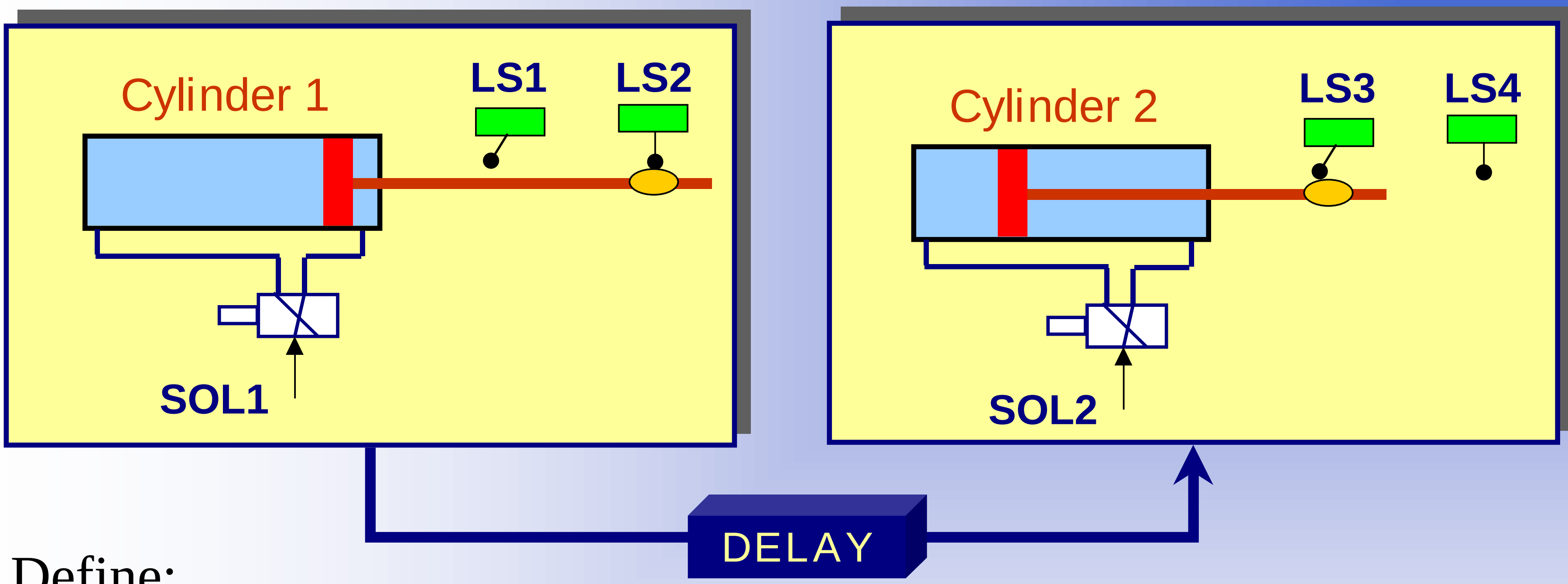
CONDITION TO ACTIVATE TIMER

$$SOL2 = (SOL2 + LS1\cdot LS3\cdot DELAY)\cdot\overline{LS4}\cdot\overline{STOP}$$

EXAMPLE – LADDER DIAGRAM



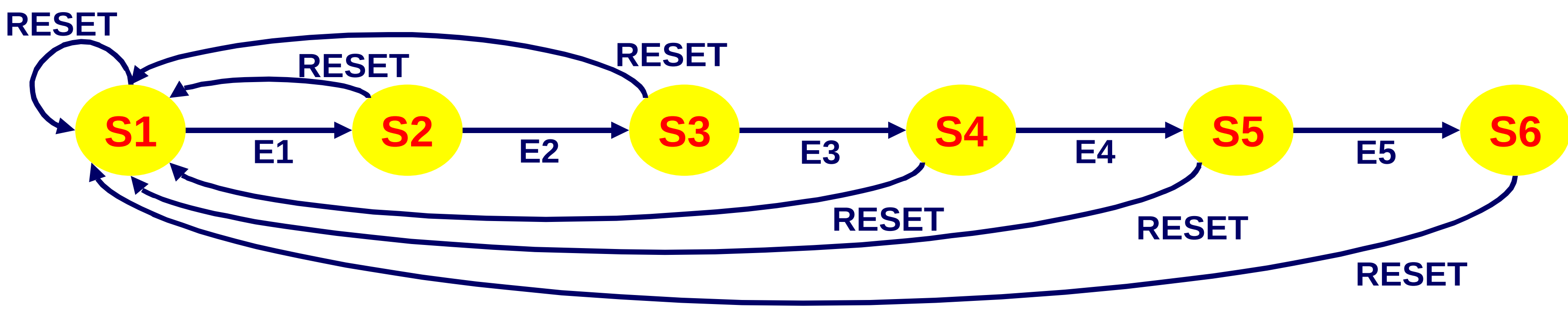
EXAMPLE – DES APPROACH



Define:

- System **STATES**
  (e.g., "System at rest", "Piston 1 moving right")
- **EVENTS** causing state transitions
  (e.g., "LS2 triggered", "Timer activated")

EXAMPLE – DES APPROACH *CONTINUED*



STATES

S1	Rest (**LS1**, **LS3** ON)
S2	Piston 1 moves right
S3	Piston 1 moves left
S4	Timer ON
S5	Piston 2 moves right
S6	Piston 2 moves left

EVENTS

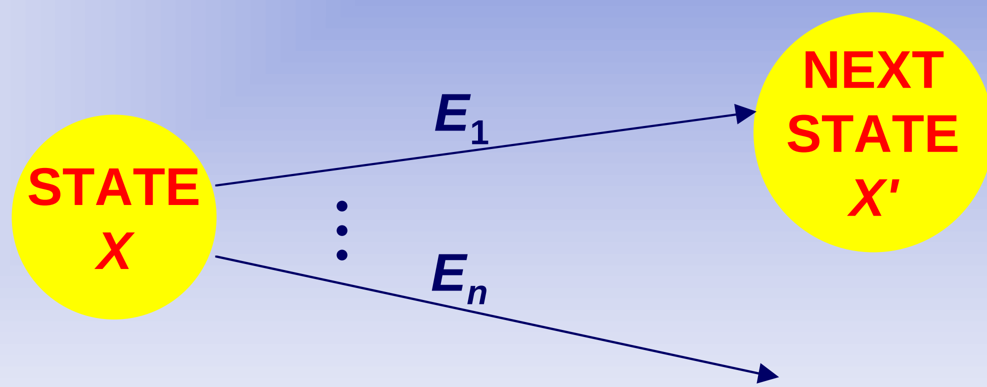
E1	**START**
E2	**LS2** triggered
E3	**LS1** triggered
E4	**TIMER** activated
E5	**LS4** triggered
E6	**LS3** triggered
RESET	**STOP**

STATE TRANSITION LOGIC

Two main elements in a *Discrete Event System*:

1. **STATE - the status of some system component**
2. **EVENT - instantaneous action causing a state transition**

STATE TRANSITION DIAGRAM:



STATE TRANSITION FUNCTION: $X' = f(X, E)$

NOTE: Only *FEASIBLE* events at state ***X*** are considered

DES MODELING: STATE AUTOMATON

AUTOMATON: (E, X, Γ, f, x_0)

E : Event Set

X : State Space

$\Gamma(x)$: Set of *feasible* or *enabled* events at state x

f : State Transition Function $f : X \times E \to X$
(undefined for $e \notin \Gamma(x)$)

x_0 : Initial State $x_0 \in X$

$$\{e_1, e_2, \ldots\} \longrightarrow \boxed{f(x,e) = x'} \longrightarrow \{x_1, x_2, \ldots\}$$

TIMED AUTOMATON

Add a ***Clock Structure V*** to the Automaton: $(E, X, \Gamma, f, x_0, \mathbf{V})$

where: $\mathbf{V} = \{\mathbf{v}_i : i \in E\}$

and $\mathbf{v}_i$ is a ***Clock or Lifetime*** Sequence: $\mathbf{v}_i = \{v_{i1}, v_{i2}, \ldots\}$
one for each event i

$$\{\mathbf{v}_1, \mathbf{v}_2, \ldots\} \longrightarrow \boxed{f(x, e') = x'} \longrightarrow \{x_1, x_2, \ldots\}$$

NEXT EVENT

Need an *internal mechanism* to determine
NEXT EVENT e' and hence
NEXT STATE $x' = f(x, e')$

HOW THE TIMED AUTOMATON WORKS...

CURRENT STATE

$x \in X$ with feasible event set: $\Gamma(x)$

CURRENT EVENT

e that caused transition into x

CURRENT EVENT TIME

t associated with e

Associate a
CLOCK VALUE/RESIDUAL LIFETIME y_i
with each feasible event $i \in \Gamma(x)$

HOW THE TIMED AUTOMATON WORKS...

CONTINUED

NEXT/TRIGGERING EVENT *e′* :

$$e' = arg \min_{i \in \Gamma(x)} \{y_i\}$$

NEXT EVENT TIME *t′* :

$$t' = t + y^*$$

$$\text{where:} \quad y^* = \min_{i \in \Gamma(x)} \{y_i\}$$

NEXT STATE *x′* :

$$x' = f(x, e')$$

HOW THE TIMED AUTOMATON WORKS...

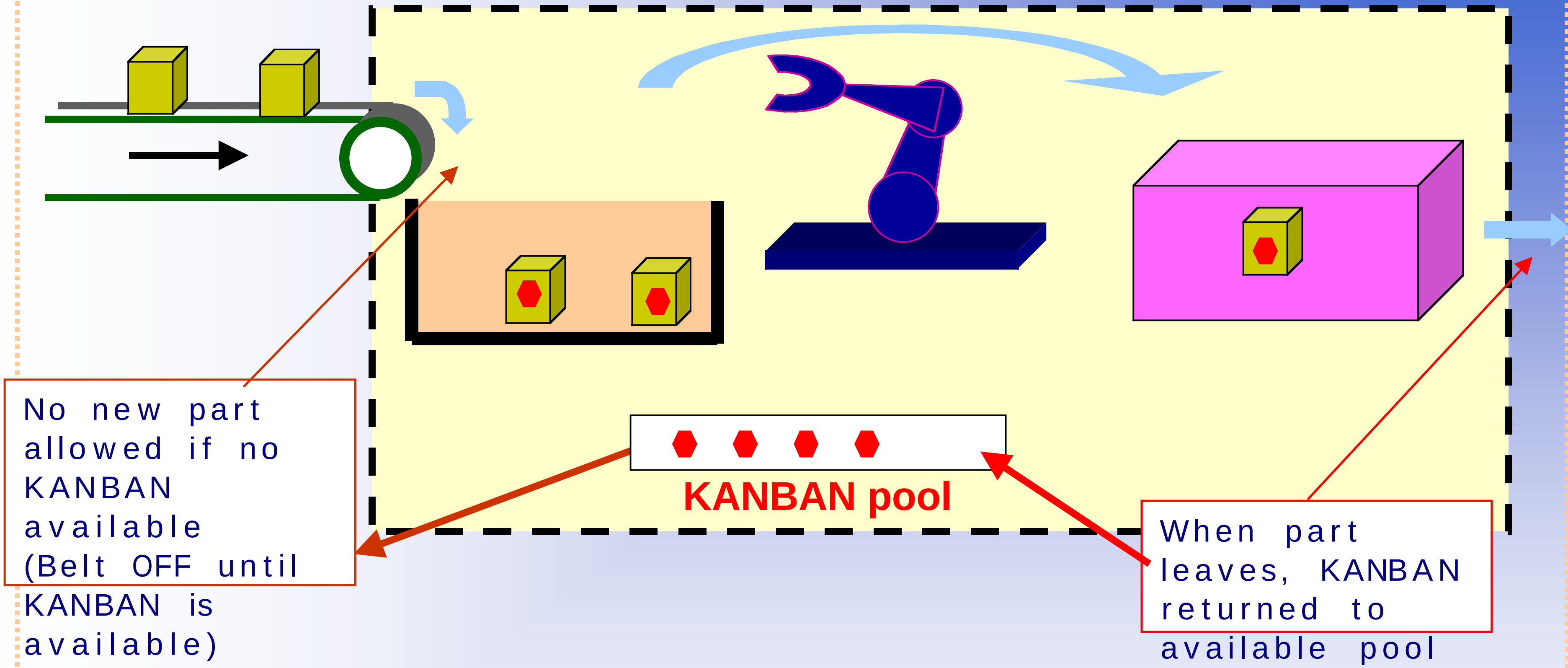
CONTINUED

Detemine new *CLOCK VALUES* y_i'
for every event $i \in \Gamma(x)$

$$y_i' = \begin{cases} y_i - y^* & i \in \Gamma(x'), i \in \Gamma(x), i \neq e' \\ v_{ij} & i \in \Gamma(x') - \{\Gamma(x) - e'\} \\ 0 & otherwise \end{cases}$$

where : v_{ij} = new lifetime for event i

TIMED AUTOMATON - AN EXAMPLE



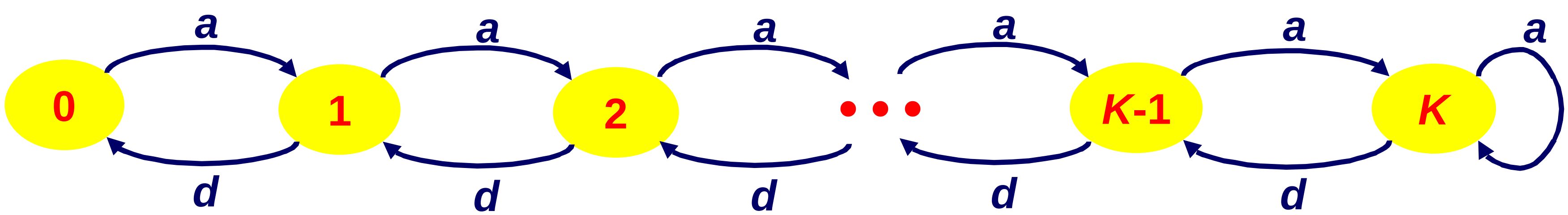
STATE : No. of parts in workcenter {0,1,…,*K*}

EVENTS : - part ARRIVALS (from conveyor belt)

- part DEPARTURES (from machine)

TIMED AUTOMATON - AN EXAMPLE

CONTINUED



$$E = \{a, d\} \qquad X = \{0, 1, \ldots, K\}$$

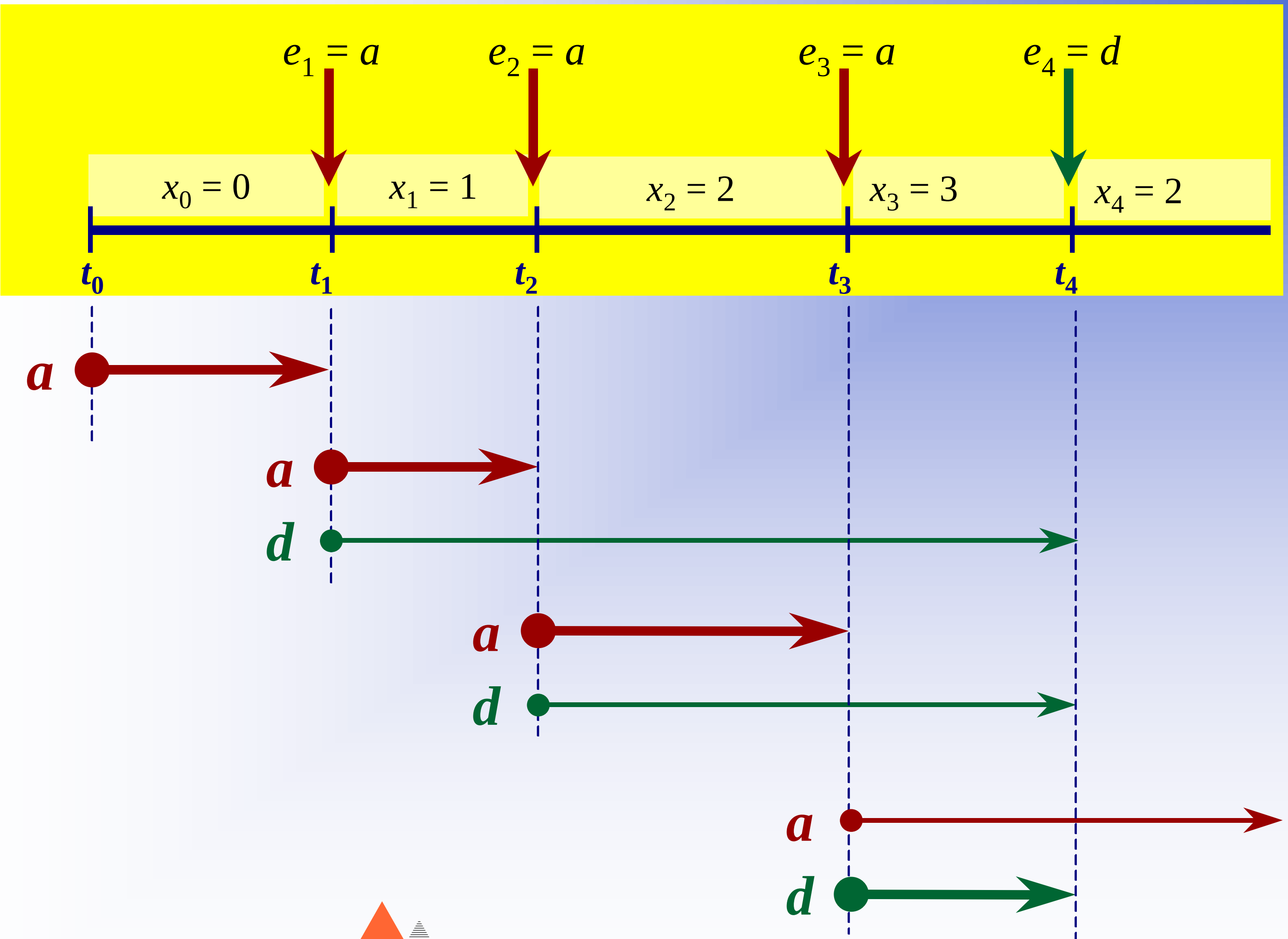
$$\Gamma(x) = \{a, d\} \quad \text{for all } x > 0$$
$$\Gamma(0) = \{a\}$$

$$f(x, e') = \begin{cases} x+1 & e' = a, x < K \\ x-1 & e' = d, x > 0 \end{cases}$$

Given input: $\mathbf{v}_a = \{v_{a1}, v_{a2}, \ldots\}, \ \mathbf{v}_d = \{v_{d1}, v_{d2}, \ldots\}$

TIMED AUTOMATON
- A TYPICAL SAMPLE PATH

CONTINUED



STOCHASTIC TIMED AUTOMATON

- Same idea with the Clock Structure consisting of ***Stochastic Processes***

- Associate with each event *i a* ***Lifetime Distribution*** based on which $\boldsymbol{v}_i$ is generated

Generalized Semi-Markov Process
(GSMP)

- In a simulator, $\boldsymbol{v}_i$ is generated through a pseudorandom number generator

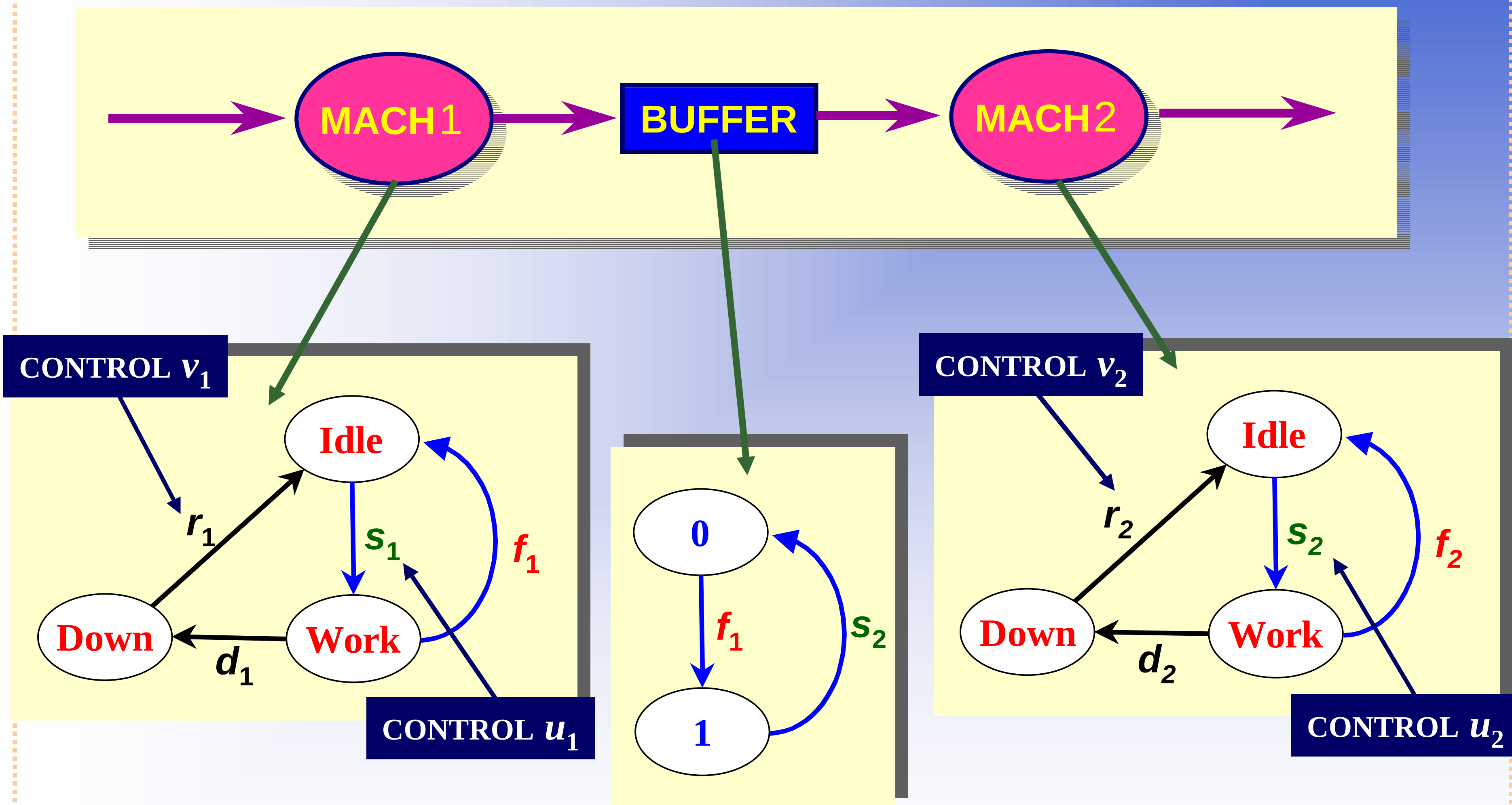
UNTIMED DES: SUPERVISORY CONTROL

- Events may be CONTROLLABLE or UNCONTROLLABLE
- ***Supervisory control***:

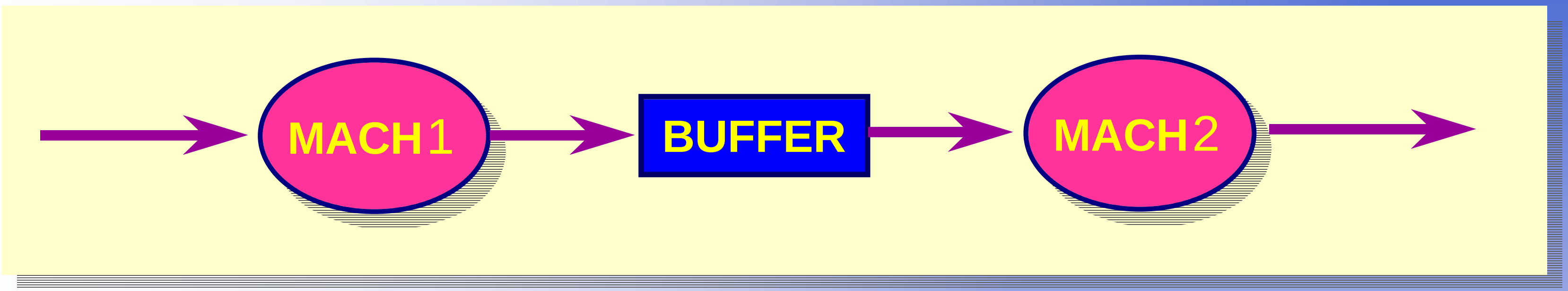
  ENABLE/DISABLE controllable events so as to meet desired specifications (avoid deadlock, illegal states, etc.)

  In the simplest case, all events are assumed ***observable***.

SUPERVISORY CONTROL -- EXAMPLE



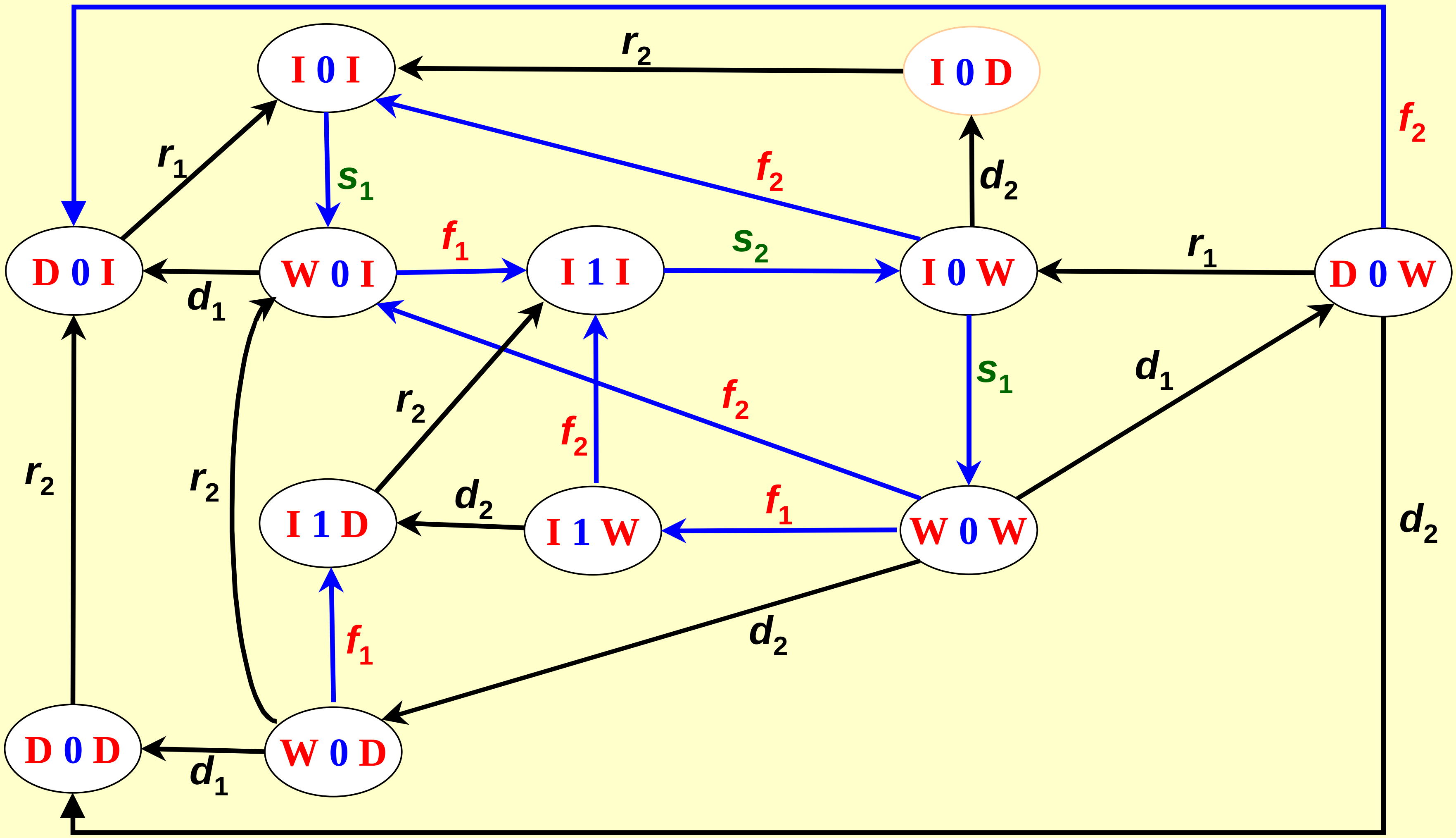
OPERATING RULES (SPECIFICATIONS)



- MACH1 can only start when BUFFER is empty.
- MACH2 can only start when BUFFER is full.
- MACH1 cannot start when MACH2 is down.
- If both MACH1 and MACH2 are down, then MACH2 is repaired first

STATE TRANSITION DIAGRAM

CONTINUED



SUPERVISOR DESIGN

STATE → / CONTROL ↓	I0I	W0I	I1I	I0W	I0D	W0W	I1W	I1D	W0D	D0D	D0I	D0W
u_1	1	1	0	1	0	1	0	0	0	0	-	-
v_1	-	-	-	-	-	-	-	-	-	0	1	1
u_2	0	0	1	0	0	0	1	-	0	0	0	0
v_2	-	-	-	-	1	-	-	1	1	1	-	-
RED. STATE	0	0	1	0	3	0	1	2	3	4	5	5

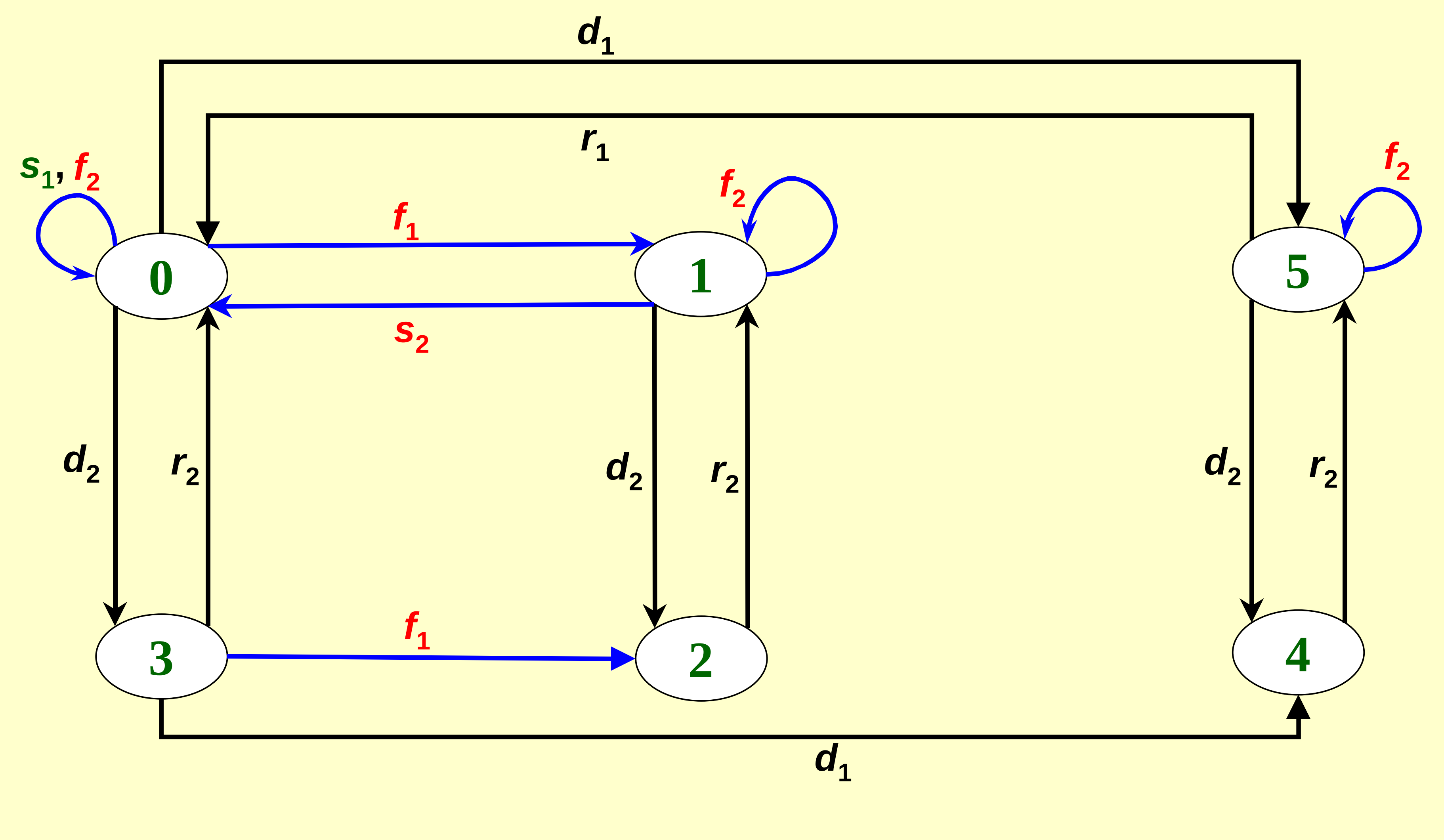
1 = ENABLE event

0 = DISABLE event

- = DON'T CARE

From a SUPERVISOR perspective, model is reduced to 6 states !

THE “QUOTIENT” SUPERVISOR



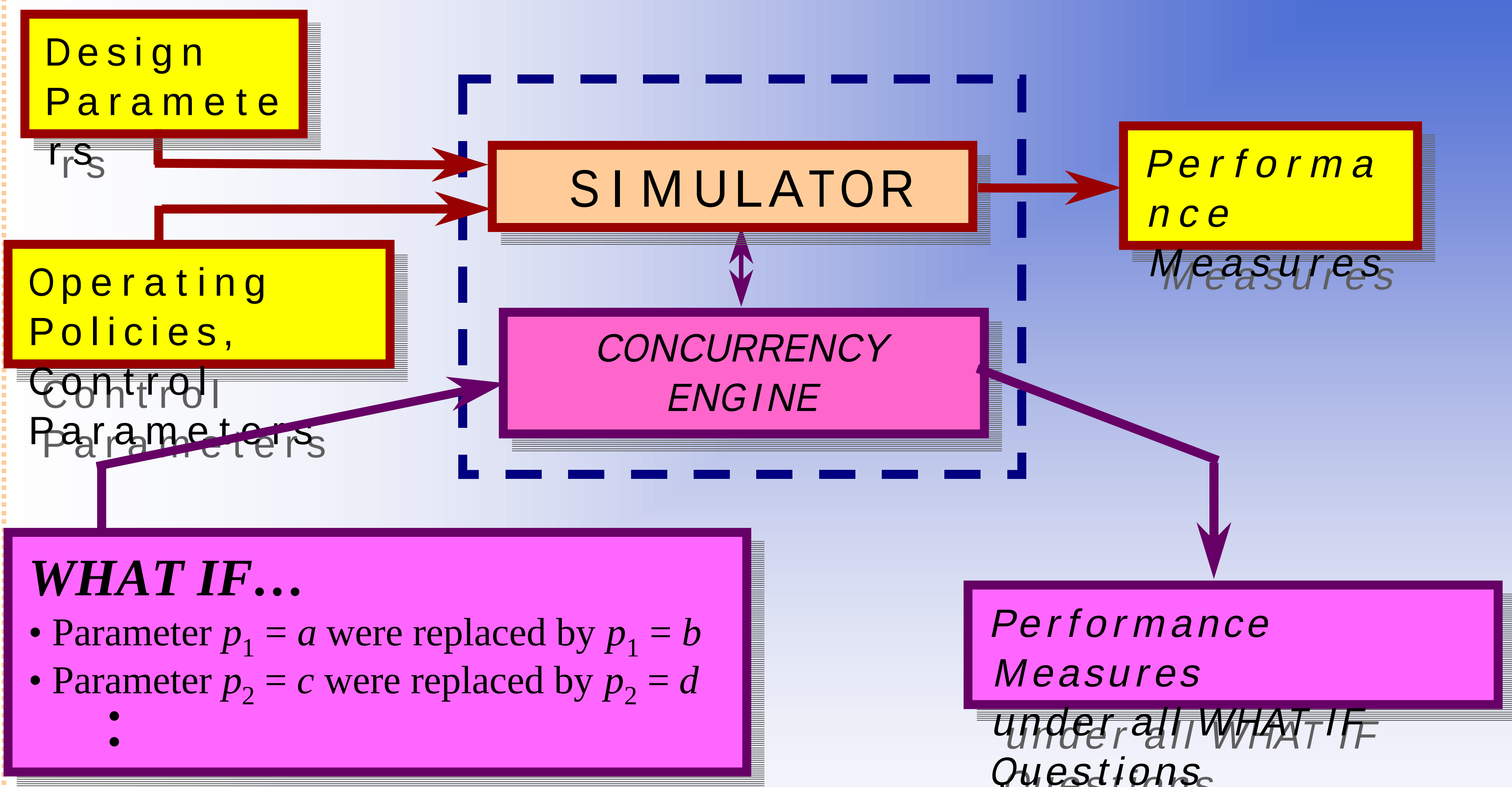
TIMED DES: SIMULATION AND ‘RAPID LEARNING’

CONVENTIONAL SIMULATION ANALYSIS

- Repeatedly change parameters/operating policies
- Test different conditions
- Answer multiple WHAT IF questions

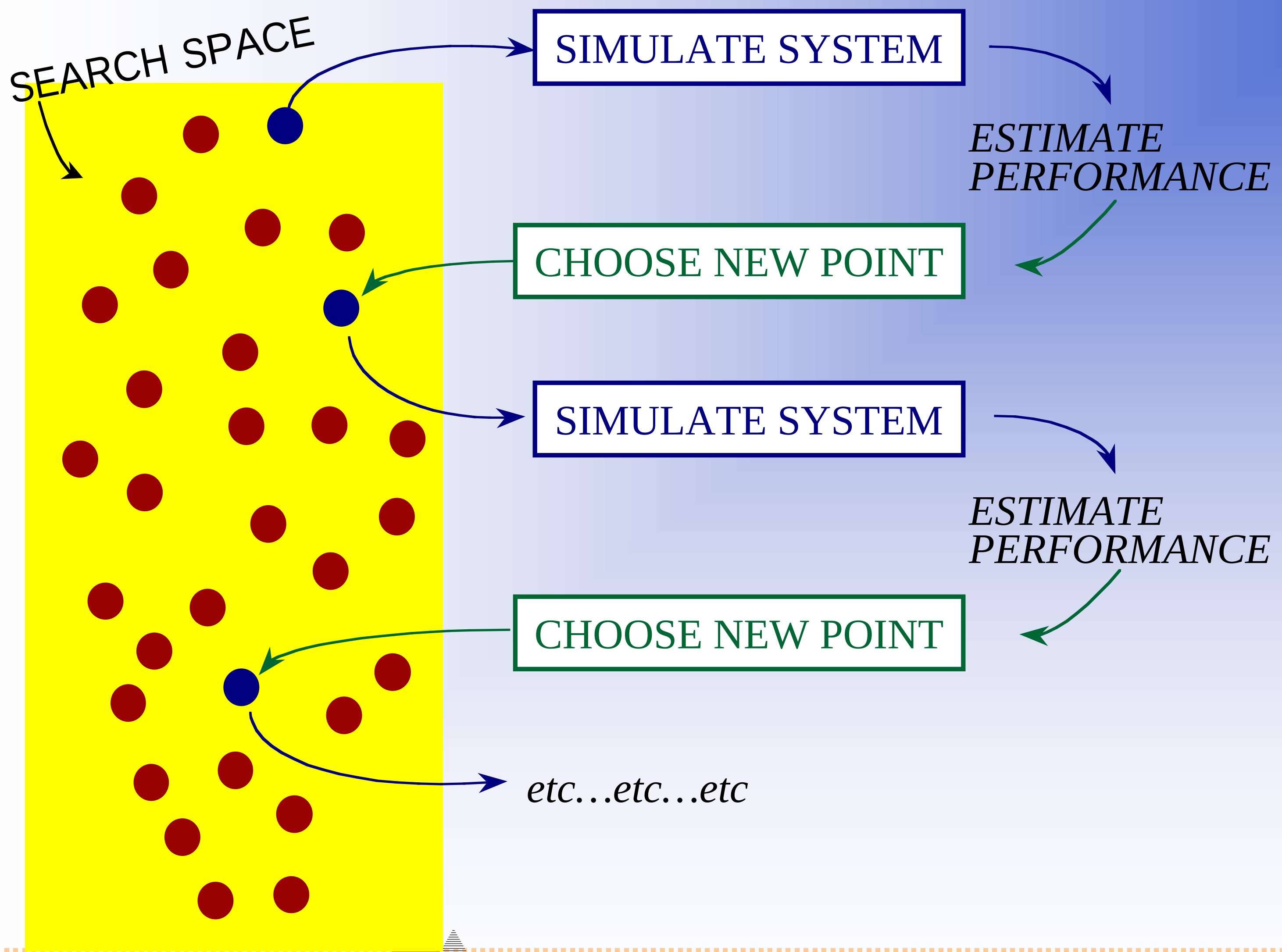
CONCURRENT SIMULATION

CONTINUED

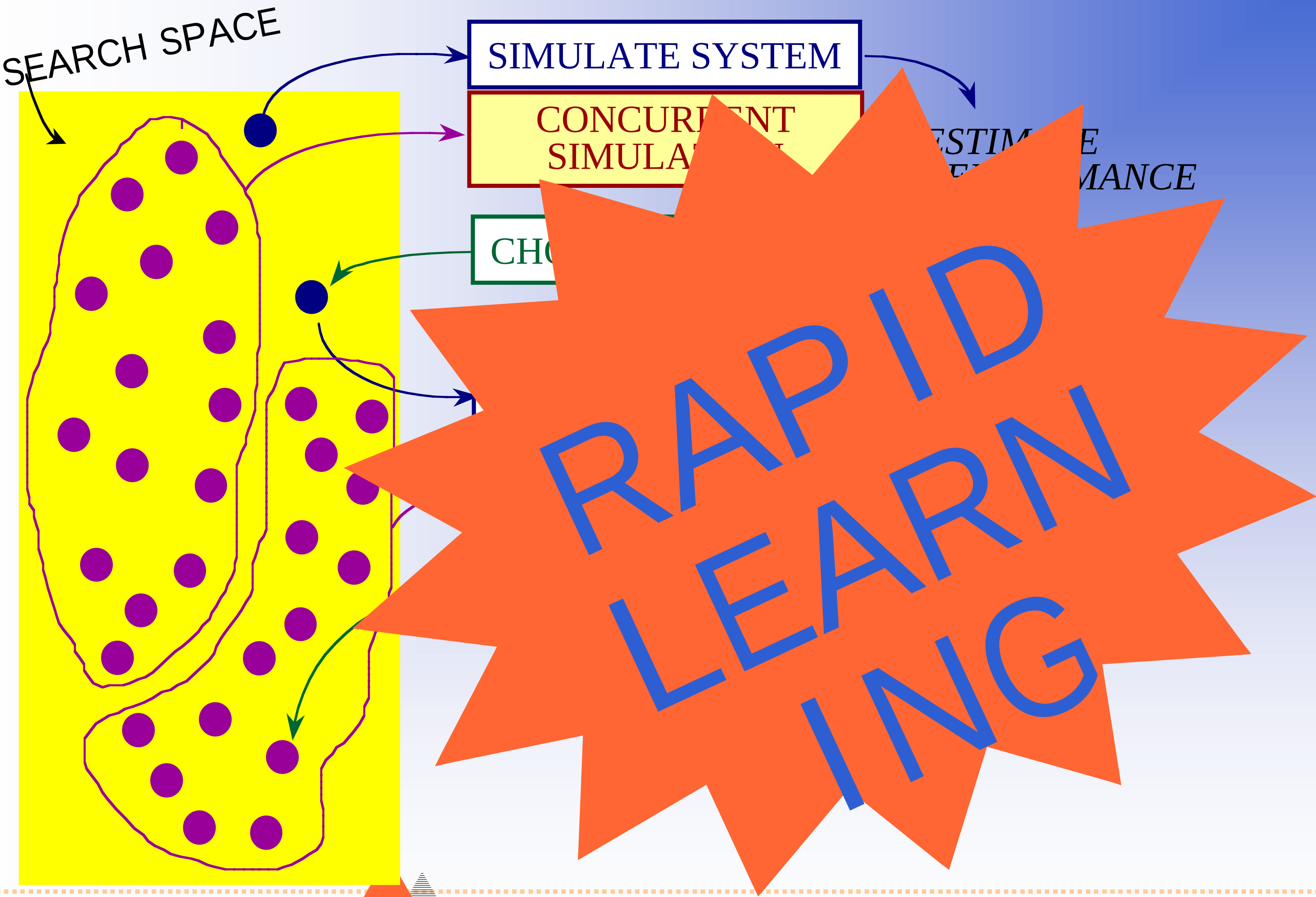


ANSWERS TO MULTIPLE "WHAT IF" QUESTIONS AUTOMATICALLY PROVIDED

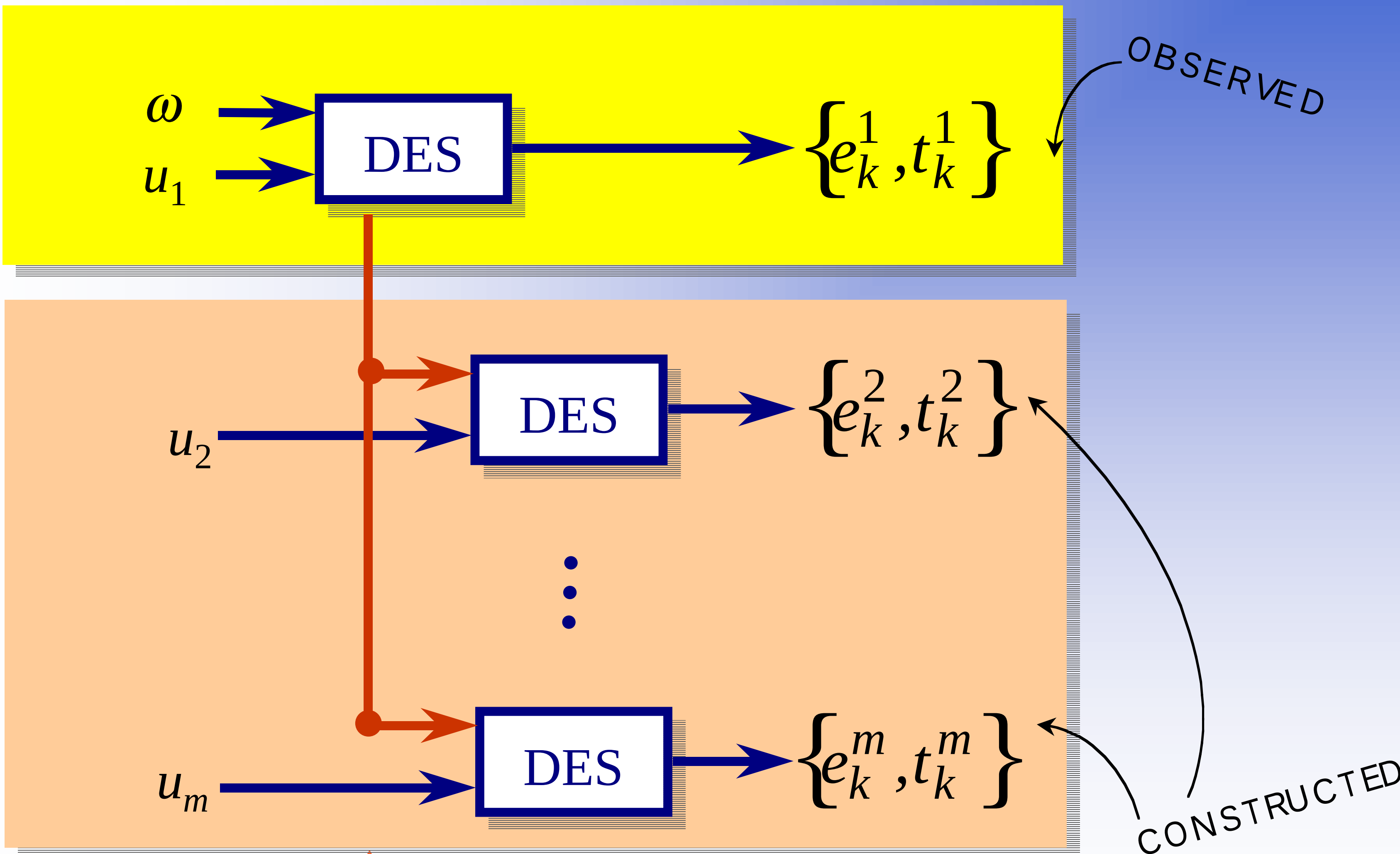
TYPICAL OPTIMIZATION OF COMPLEX SYSTEMS: *REPEATED TRIAL AND ERROR*



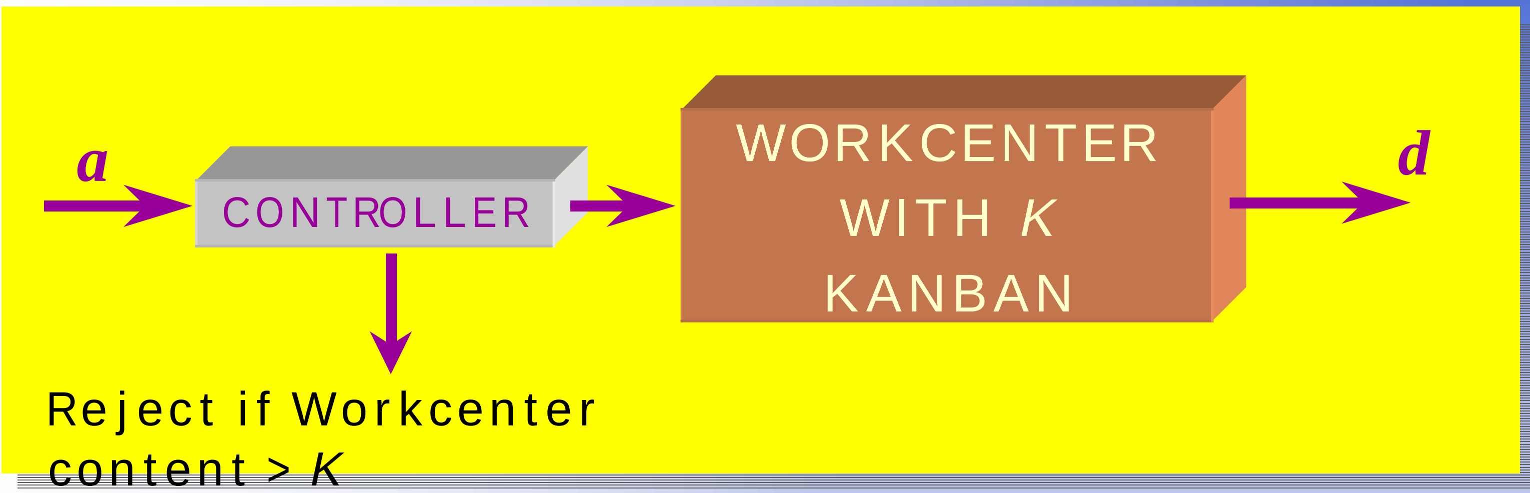
CONCURRENT SIMULATION FOR OPTIMIZATION



THE CONSTRUCTABILITY PROBLEM



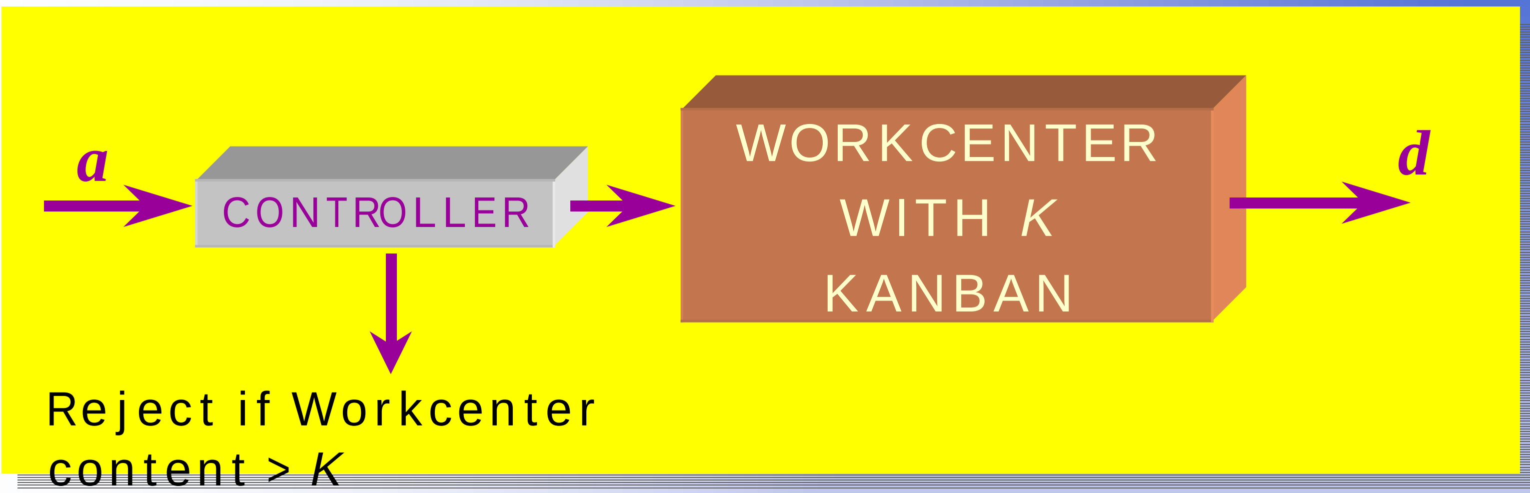
CONSTRUCTABILITY: AN EXAMPLE



PROBLEM:

- How does the system behave under different choices of *K*?
- What is the *optimal* *K*?

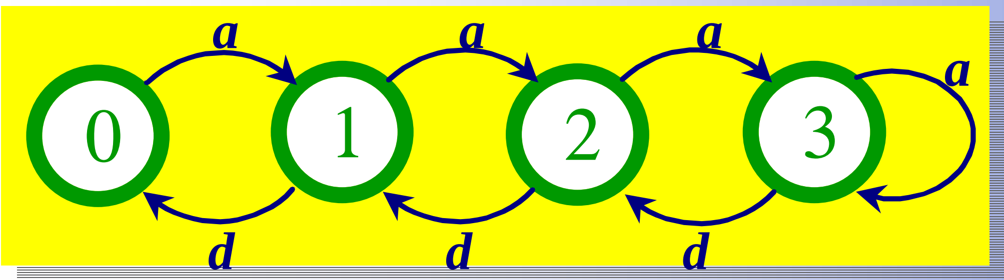
CONSTRUCTABILITY: AN EXAMPLE *CONTINUED*



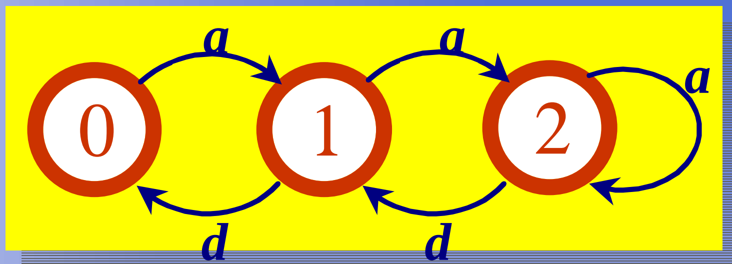
CONCURRENT SIMULATION APPROACH:

- Choose any K
- Simulate (or observe *actual system*) under K
- Apply Concurrent Simulation to *LEARN* effect of all other feasible K

CONSTRUCTABILITY: AN EXAMPLE *CONTINUED*

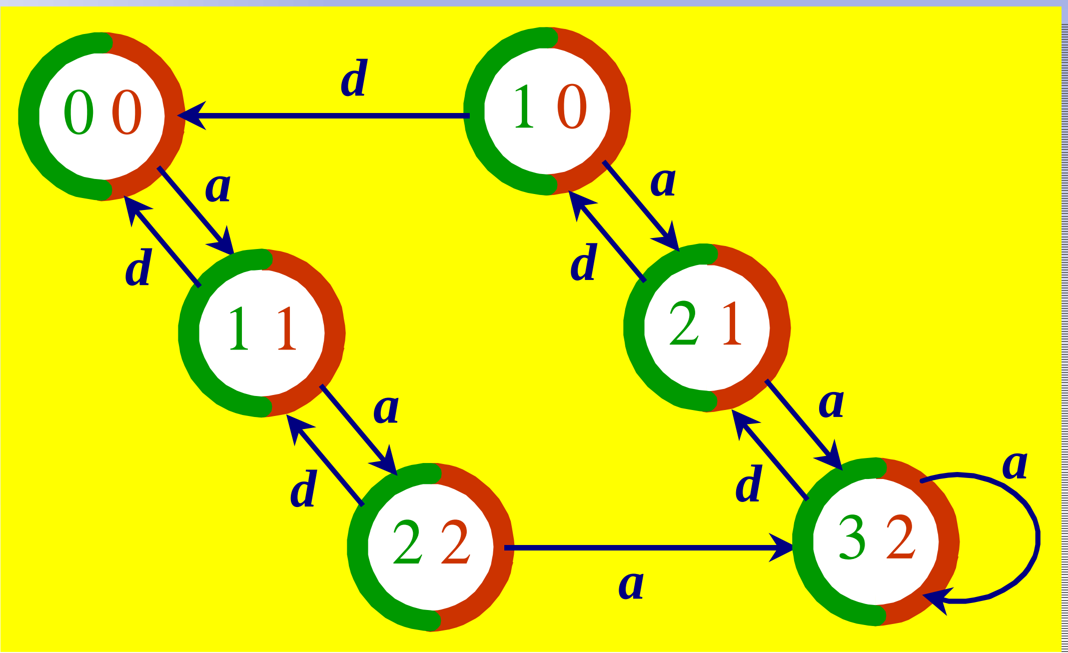


NOMINAL SYSTEM: $K = 3$

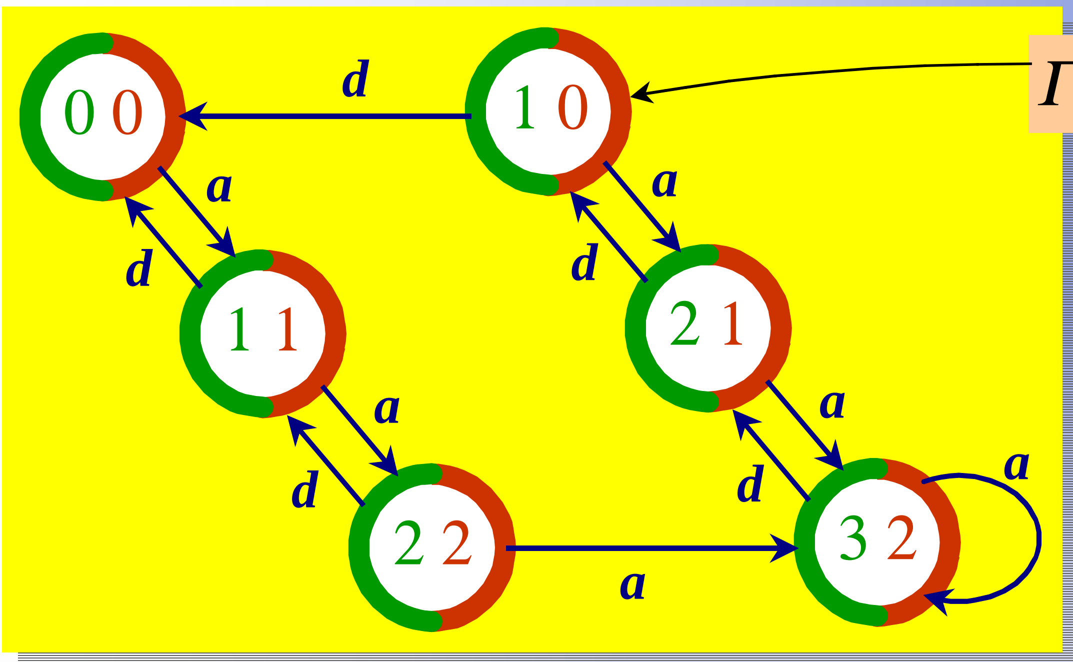


PERTURBED SYSTEM: $K = 2$

AUGMENTED SYSTEM



CONSTRUCTABILITY: AN EXAMPLE *CONTINUED*



$$\Gamma(0) = \{a\} \subset \Gamma(1) = \{a, d\}$$

OBSERVABILITY CONDITION

However, if roles of NOMINAL and PERTURBED are reversed, then things get a little trickier…