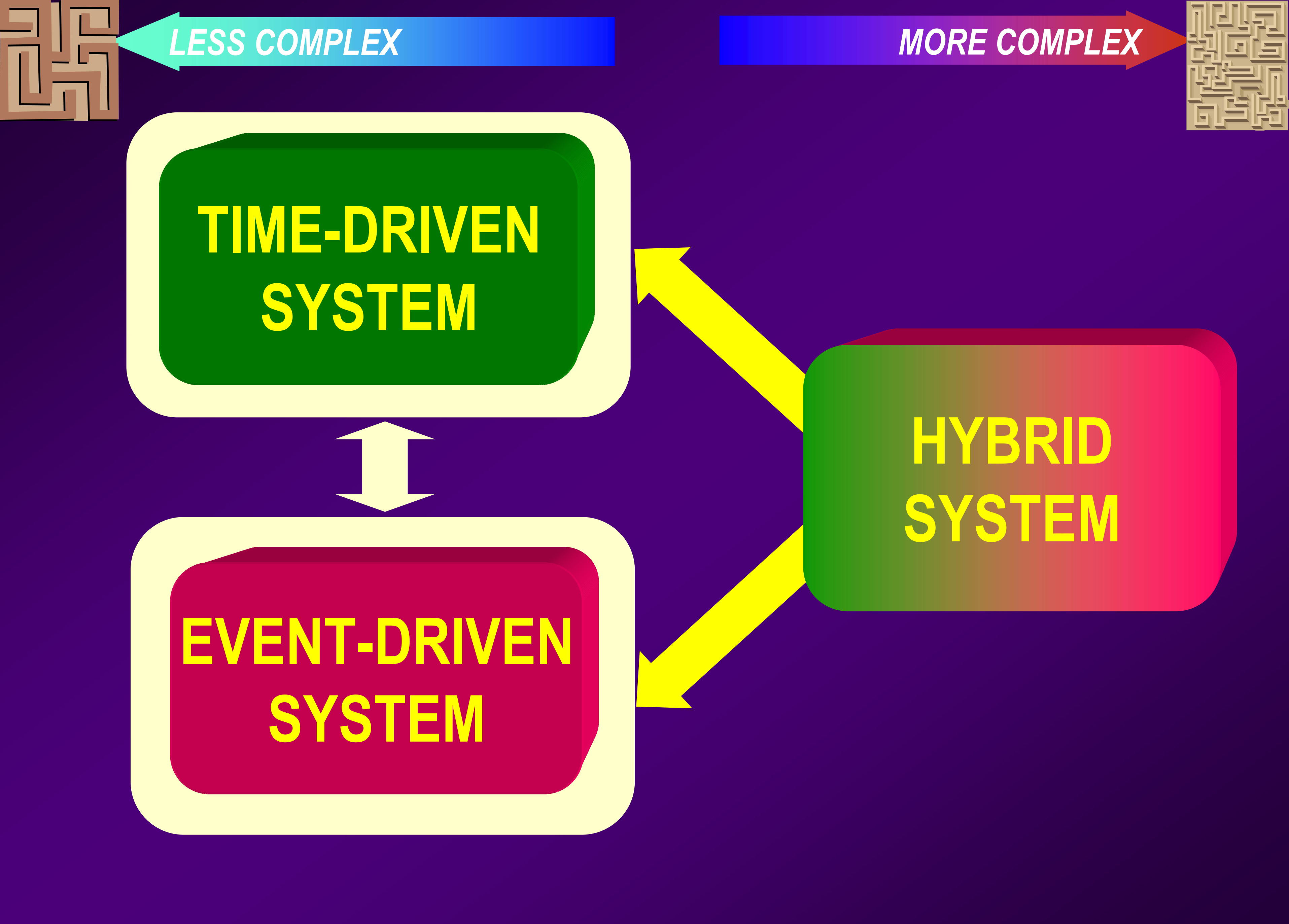
MODELING OF HYBRID SYSTEMS

C. G. Cassandras
Dept. of Manufacturing Engineering
and
***Center for Information and Systems Engineering* (CISE)**
Boston University
cgc@bu.edu
http://vita.bu.edu/cgc

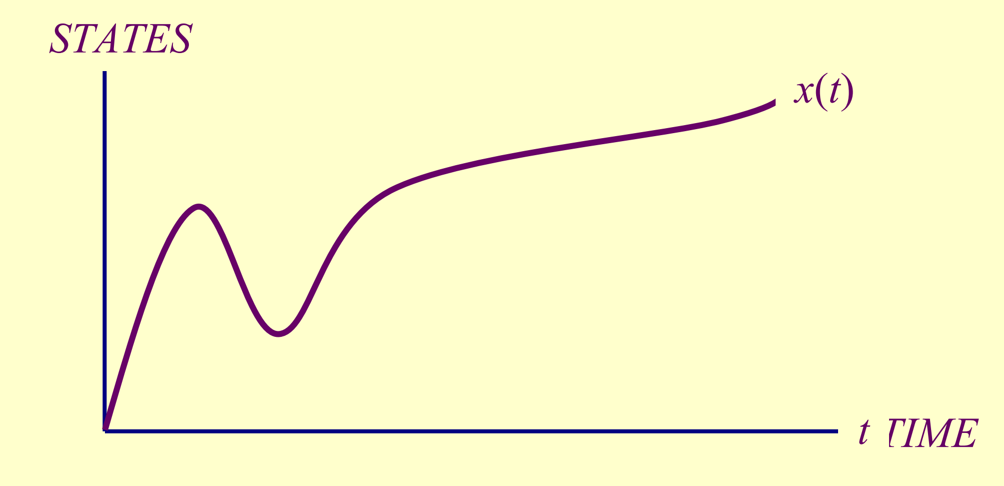
OUTLINE

- TIME-DRIVEN vs EVENT-DRIVEN SYSTEMS
- *DES*: Automata, Petri Nets, Max-Plus Algebra
- DISCRETE EVENT SIMULATION
  → HYBRID SYSTEM SIMULATION
- *HYBRID SYSTEMS*: Hybrid Automata, MLD Systems
- MODELS FOR SWITCH *TIMING* CONTROL


LESS COMPLEX
MORE COMPLEX
TIME-DRIVEN SYSTEM
HYBRID SYSTEM
EVENT-DRIVEN SYSTEM

TIME-DRIVEN vs *EVENT-DRIVEN* SYSTEMS

TIME-DRIVEN SYSTEM



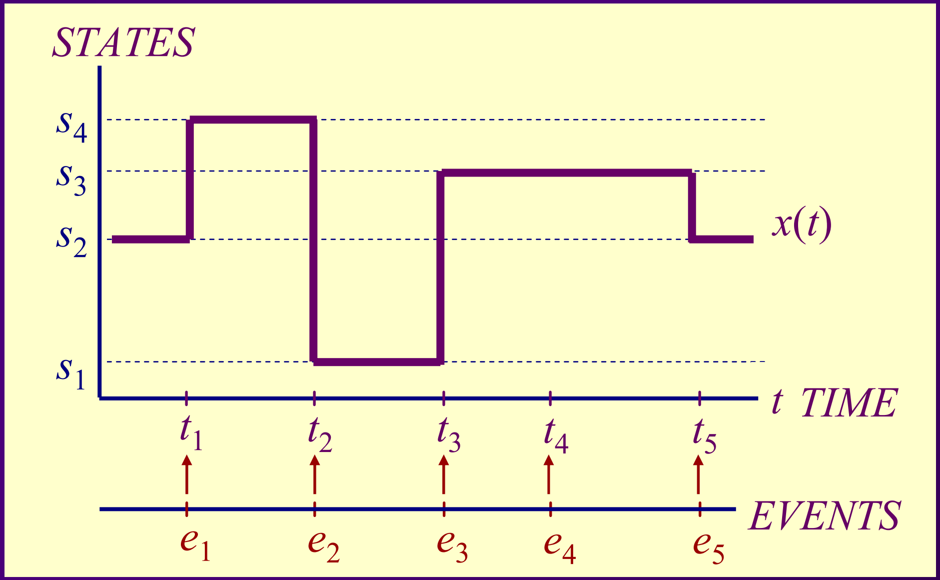
STATE SPACE:

$$X = \Re$$

DYNAMICS:

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

EVENT-DRIVEN SYSTEM



STATE SPACE:

$$X = \{s_1, s_2, s_3, s_4\}$$

DYNAMICS:

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

AUTOMATA

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 events $e \notin \Gamma(x)$)

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

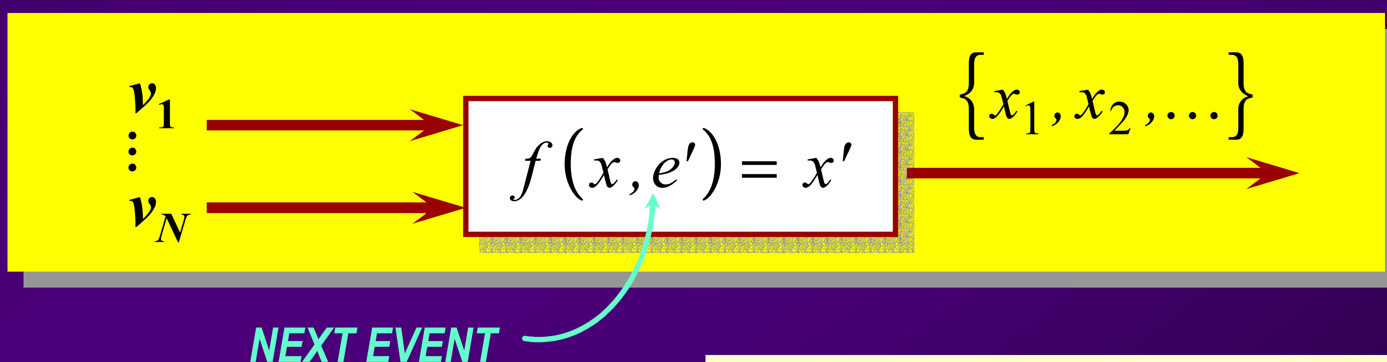
Add a ***Clock Structure V*** to the automaton: $(E, X, \Gamma, f, x_0, V)$

where:

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

and $\boldsymbol{v}_i$ is a ***Clock or Lifetime sequence***: $\boldsymbol{v}_i = \{v_{i1}, v_{i2}, \ldots\}$

one for each event i



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*

Determine 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

EVENT CLOCKS ARE STATE VARIABLES

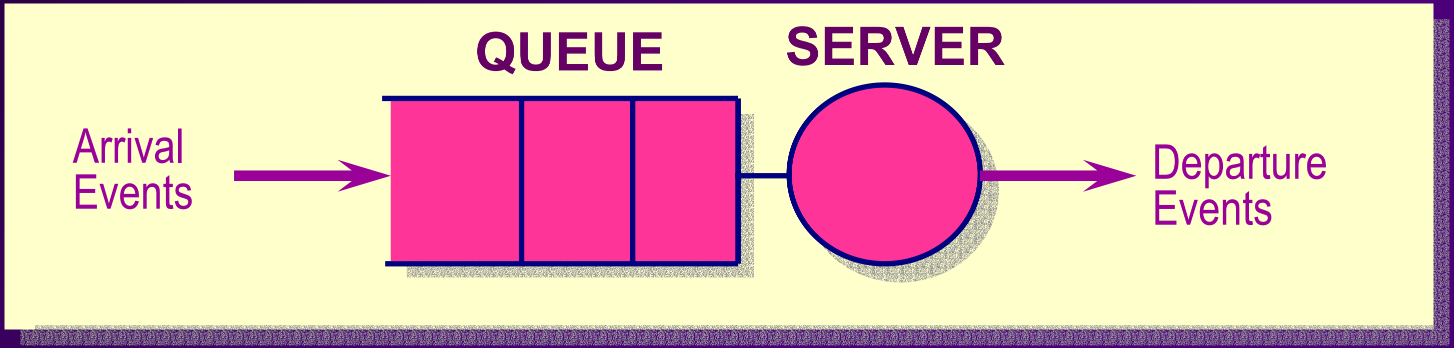
v_1 ⋮ v_N →

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

$$\mathbf{y}' = \mathbf{g}(\mathbf{y}, x, V)$$

→ $\{x_1, x_2, \ldots\}$

TIMED AUTOMATON - AN EXAMPLE



$$E = \{a, d\}$$
$$X = \{0, 1, 2, \ldots\}$$

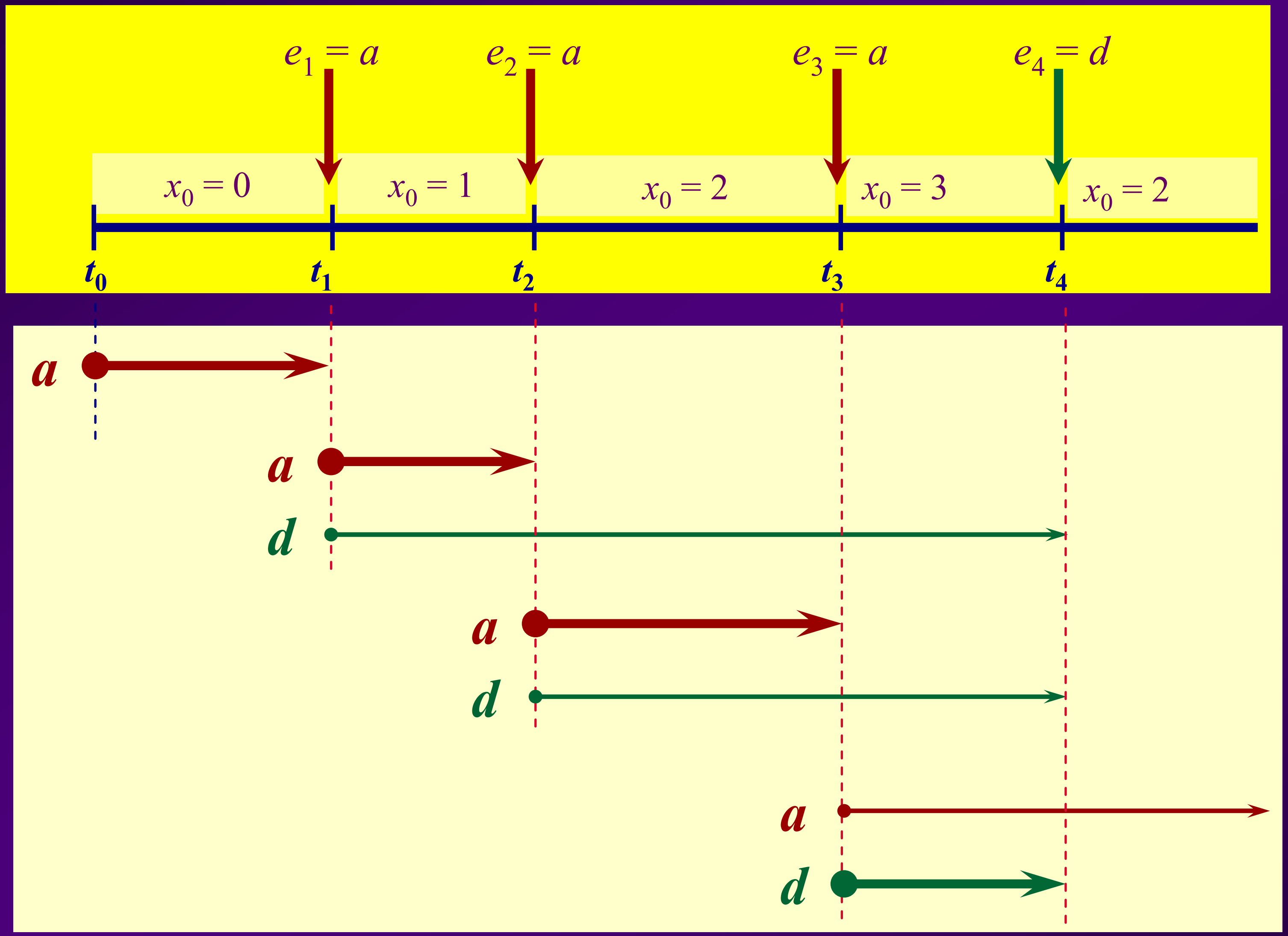
$$\Gamma(x) = \{a, d\}, \quad \text{for all } x > 0$$
$$\Gamma(0) = \{a\}$$

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

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

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 v_i is generated

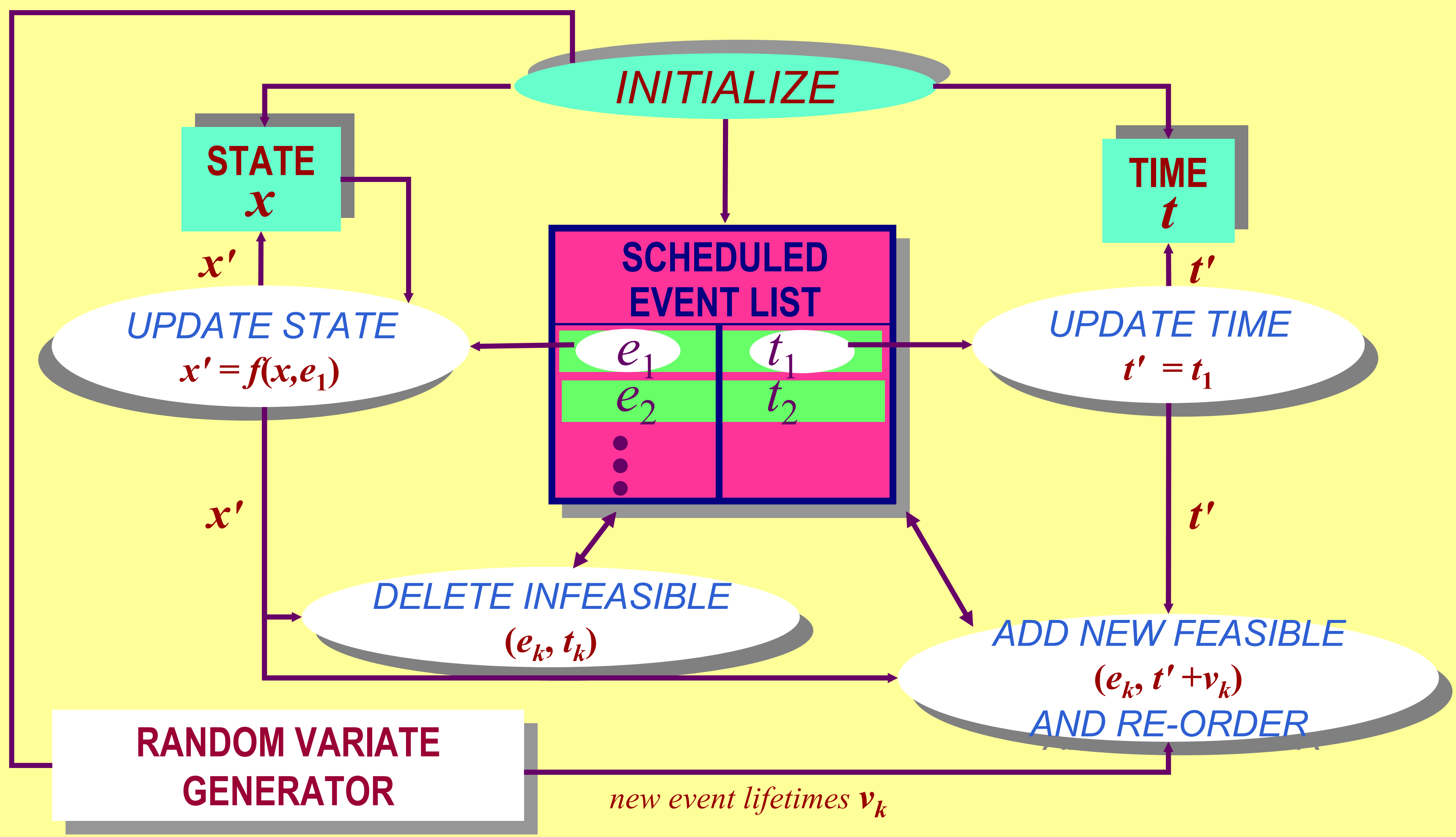
↓

Generalized Semi-Markov Process
(GSMP)

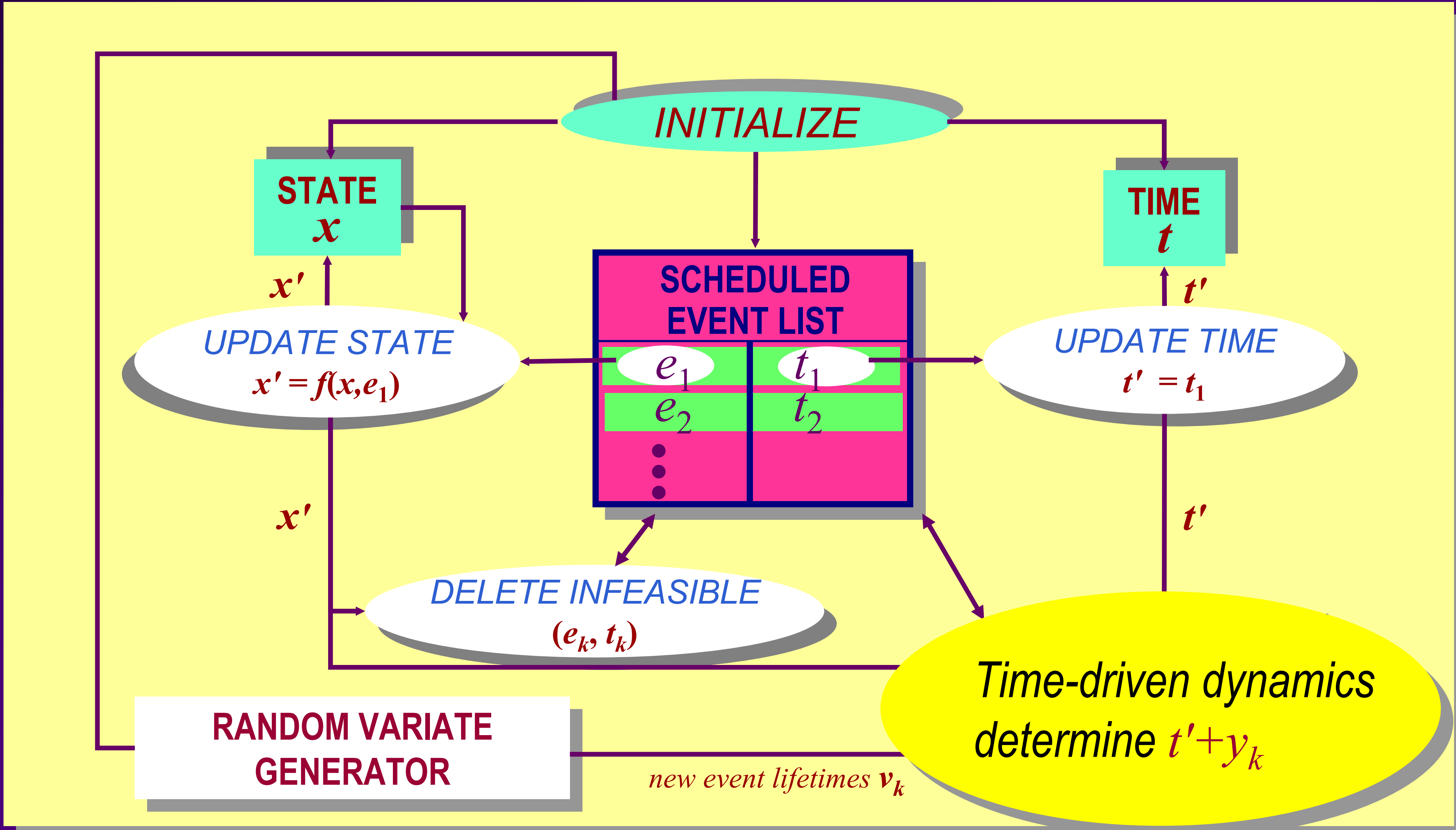
In a simulator, v_i is generated through a pseudorandom number generator

DISCRETE EVENT SIMULATION

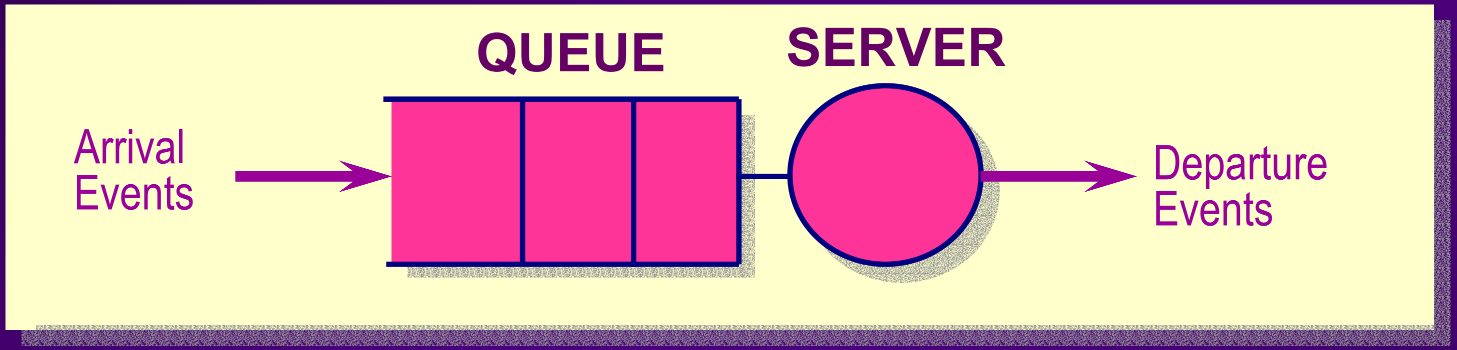
…is simply a computer-based implementation of the DES sample path generation mechanism described so far



Timing of NEW FEASIBLE EVENTS is now determined by time-driven dynamics

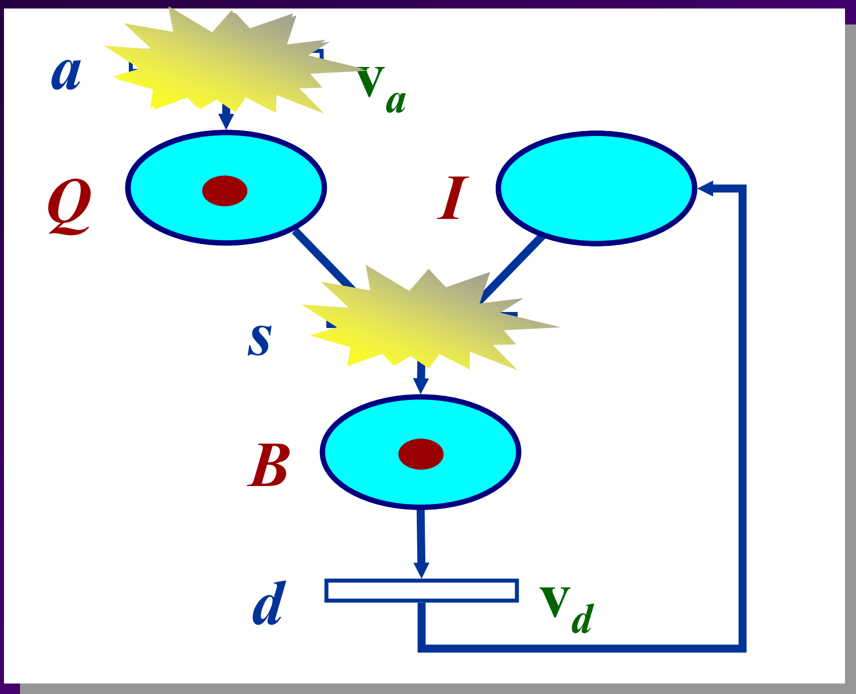


Proceed by example and contrast to *Timed Automaton* model...



EVENTS: - **Arrivals (*a*)**
- **Departures (*d*)**

STATES: Number of customers in queue or in service, {0,1,2,…}



TRANSITIONS (EVENTS):

- ***a*** : Customer arrives
- ***s*** : Service starts
- ***d*** : Customer departs

PLACES (CONDITIONS):

- ***Q*** : Queue not empty
- ***I*** : Server idle
- ***B*** : Server busy

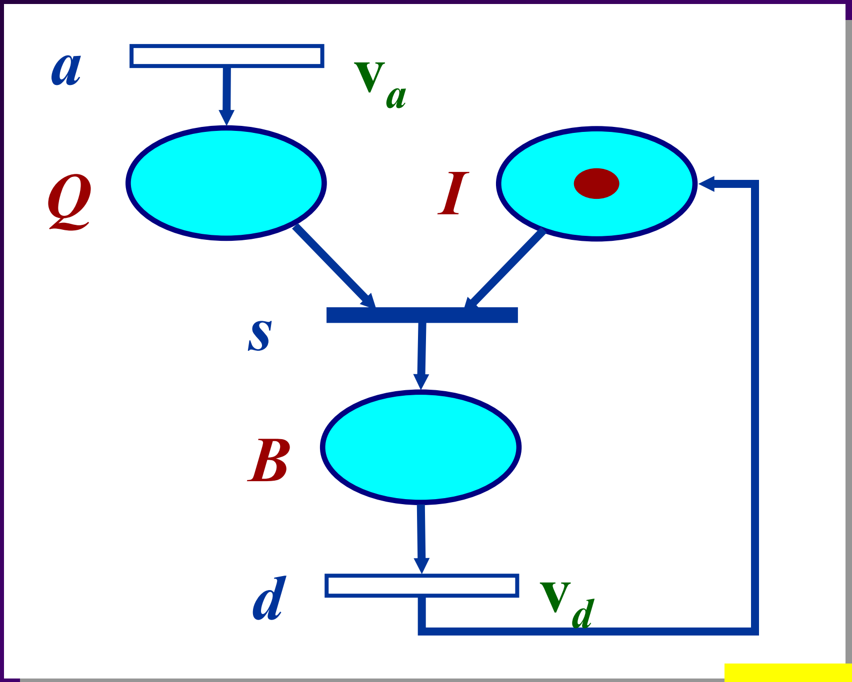
TRANSITION TIMING:

- ***a*** $\longrightarrow \quad \mathbf{v}_a = \{v_{a,1}, v_{a,2}, \ldots\}$
- ***s*** $\longrightarrow \quad \mathbf{v}_s = 0$ (no delay)
- ***d*** $\longrightarrow \quad \mathbf{v}_{\mathrm{d}} = \{v_{d,1}, v_{d,2}, \ldots\}$

Transition *fires* after specified delay

PETRI NETS

CONTINUED



a_k : kth arrival time
s_k : kth service start time
d_k : kth departure time

$\pi_{Q,k}$: kth time Q gets token
$\pi_{I,k}$: kth time I gets token
$\pi_{B,k}$: kth time B gets token

$$a_k = a_{k-1} + v_{a,k}$$
$$s_k = max\left[\pi_{Q,k}, \pi_{I,k}\right]$$
$$d_k = \pi_{B,k} + v_{d.k}$$

$$\pi_{Q,k} = a_k$$
$$\pi_{I,k} = d_{k-1}$$
$$\pi_{B,k} = s_k$$

$$d_k = max\left[a_k, d_{k-1}\right] + v_{d.k}, \qquad k = 1,2\ldots$$

MAX-PLUS ALGEBRA

ADDITION: $a \oplus b = max[a, b]$

MULTIPLICATION: $a \otimes b = a + b$

- From Petri net model:

$$a_k = a_{k-1} + v_{a,k} \qquad a_0 = 0$$

$$d_k = max[a_{k-1} + v_{a,k}, d_{k-1}] + v_{d.k} \qquad d_0 = 0$$

- Fix: $v_{a,k} = C_a, \quad v_{d,k} = C_d \quad \text{for all } k = 1,2,\ldots$

- Equations become:

$$a_{k+1} = (a_k \otimes C_a) \oplus (d_{k\text{-}1} \otimes -L) \qquad -L = -\infty$$

$$d_k = (a_k \otimes C_d) \oplus (d_{k\text{-}1} \otimes C_d)$$

- In matrix form:

$$\begin{bmatrix} a_{k+1} \\ d_k \end{bmatrix} = \begin{bmatrix} C_a & -L \\ C_d & C_d \end{bmatrix} \begin{bmatrix} a_k \\ d_{k-1} \end{bmatrix}$$

- Define:

$$\mathbf{x}_k = \begin{bmatrix} a_{k+1} \\ d_k \end{bmatrix}, \qquad \mathbf{A} = \begin{bmatrix} C_a & -L \\ C_d & C_d \end{bmatrix}$$

- Get a "*linear*" system - in the max-plus sense:

$$\mathbf{x}_{k+1} = \mathbf{A}\,\mathbf{x}_k, \qquad \mathbf{x}_0 = \begin{bmatrix} C_a \\ 0 \end{bmatrix}$$

REFERENCES ON DES MODELING

Cassandras and Lafortune, *Introduction to Discrete Event Systems*, Kluwer, 1999

David and Alla, *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*, Prentice-Hall, 1992.

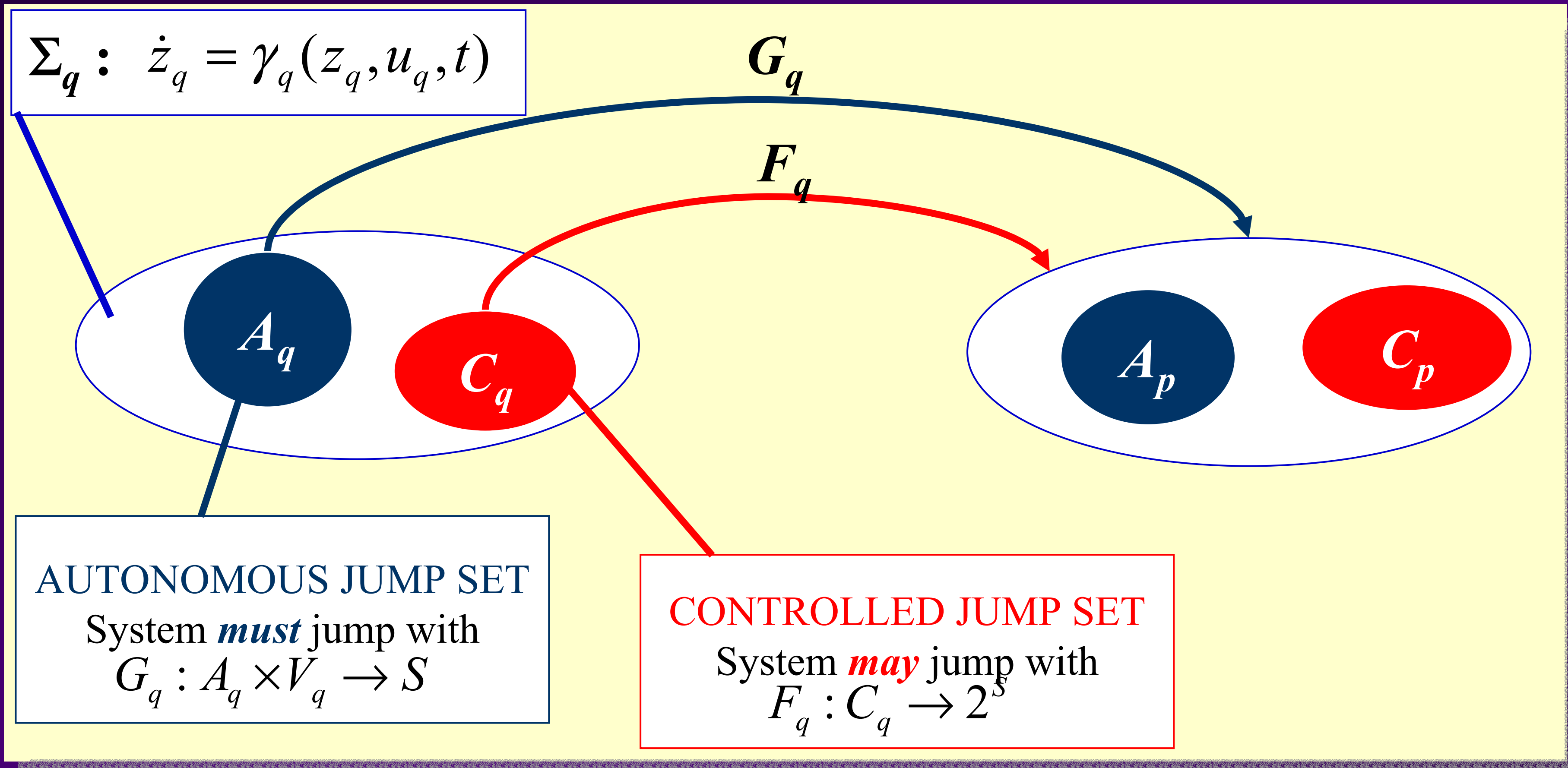
Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981

Glasserman and Yao, *Monotone Structure in Discrete-Event Systems*, Wiley, 1994

Baccelli, Cohen, Olsder, and Quadrat, *Synchronization and Linearity: An Algebra for Discrete Event Systems*, Wiley, 1992

HYBRID AUTOMATA

[*Branicky et al., 1998*]

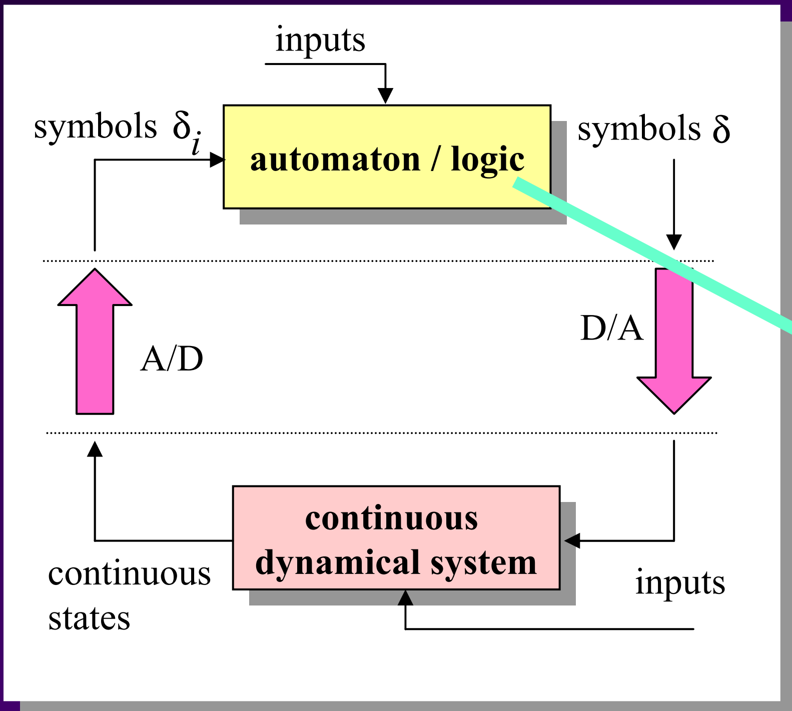


$$H = (Q, \Sigma, \mathbf{A}, \mathbf{G}, \mathbf{V}, \mathbf{C}, \mathbf{F})$$

Discrete state indices, q∈Q

MIXED LOGICAL DYNAMICAL (MLD) SYSTEMS

[*Bemporad and Morari, 1999*]

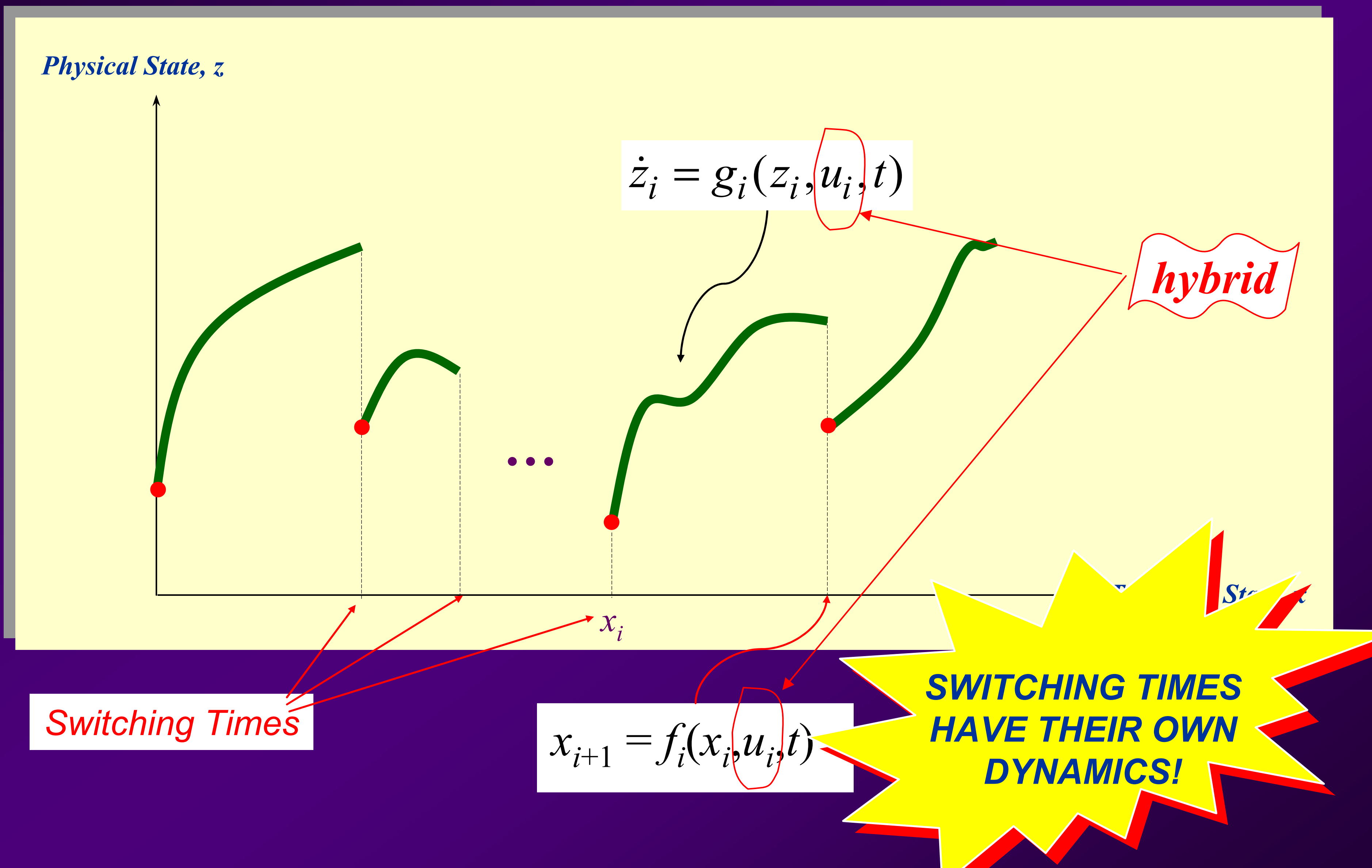


$$x(t+1) = Ax(t) + B_1u(t) + B_2\delta(t) + B_3z(t)$$
$$y(t) = Cx(t) + D_1u(t) + D_2\delta(t) + D_3z(t)$$
$$E_2\delta(t) + E_3z(t) \leq E_4x(t) + E_1u(t) + E_5$$

$X_1 \vee X_2$	$\delta_1 + \delta_2 \geq 1$
$X_1 \wedge X_2$	$\delta_1 + \delta_2 \geq 2$
$\neg X_1$	$\delta_1 \leq 0$
$X_1 \Rightarrow X_2$	$\delta_1 - \delta_2 \leq 0$
$X_1 \Leftrightarrow X_2$	$\delta_1 - \delta_2 = 0$
any logic proposition	$A\delta \leq B$, $\delta \in \{0,1\}^n$

- MLD systems are tailored to solving optimization problems involving hybrid dynamics (mixed-integer programming)

FOR *TIMING CONTROL* PURPOSES



TIMING CONTROL

CONTINUED

Time-Driven **Dynamics (*STATE* = *z*):**

$$\dot{z}(t) = g(z, u, t)$$

or:

$$z_{k+1} = g_k(z_k, u_k)$$

Event-Driven **Dynamics (*STATE* = Event Times $x_{k,i}$):**

$$x_{k+1,i} = \max_{j \in \Gamma_i} \{ x_{k,j} + \mathbf{a}_{k,j} \mathbf{u}_{k,j} \}$$

Event counter k = 1,2,...

***Event index i*∈ *E* = {1,…,*n*}**

TIMING CONTROL

CONTINUED

$$x_{k+1,i} = \max_{j \in \Gamma_i} \left\{ x_{k,j} + \mathbf{a}_{k,j} \mathbf{u}_{k,j} \right\}$$

Event Set: $E = \{1, \ldots, n\}$

*k*th Occurrence Time of Event *i*

Control vector