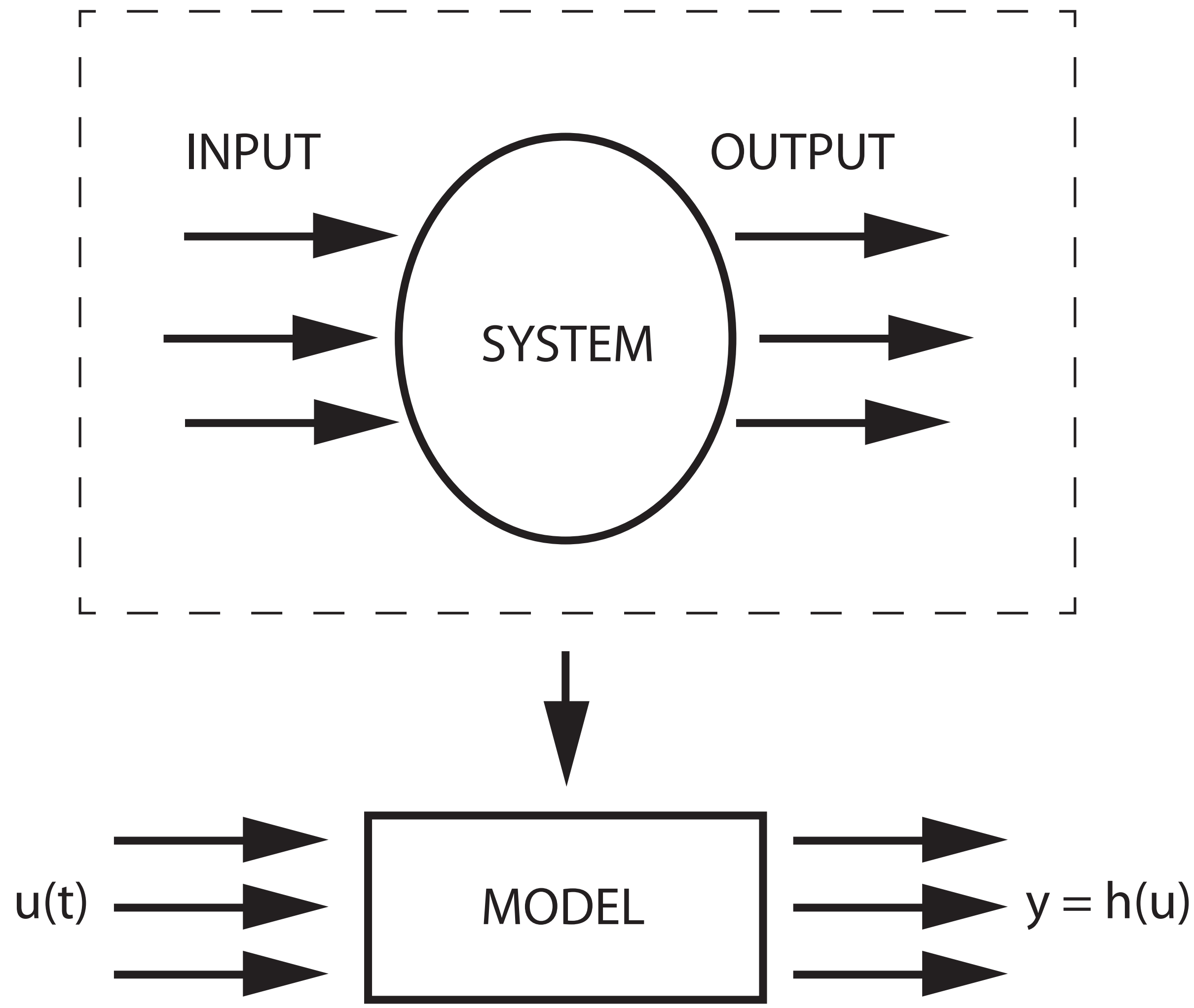
LECTURE 2: DISCRETE EVENT SYSTEMS I

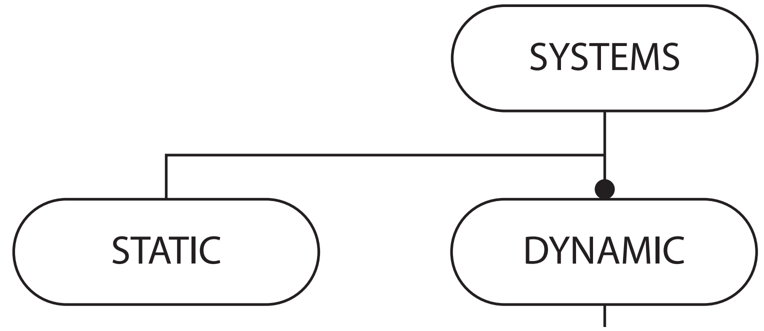
Modeling and Simulation 2

Daniel Georgiev

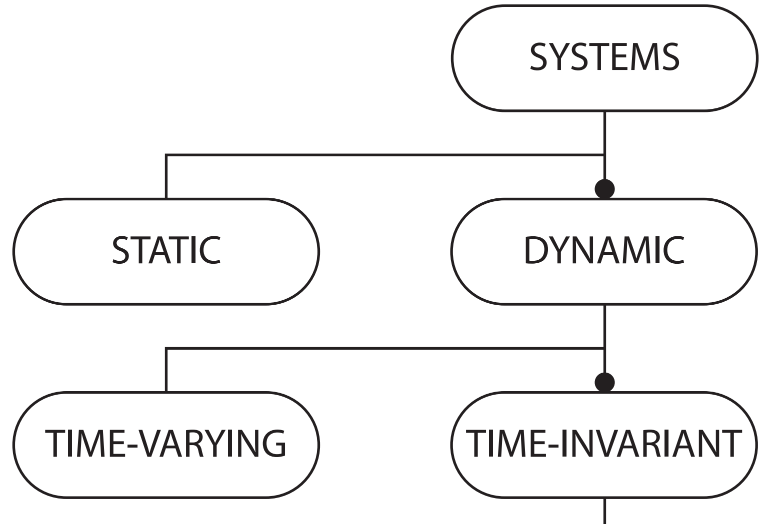
Winter 2015



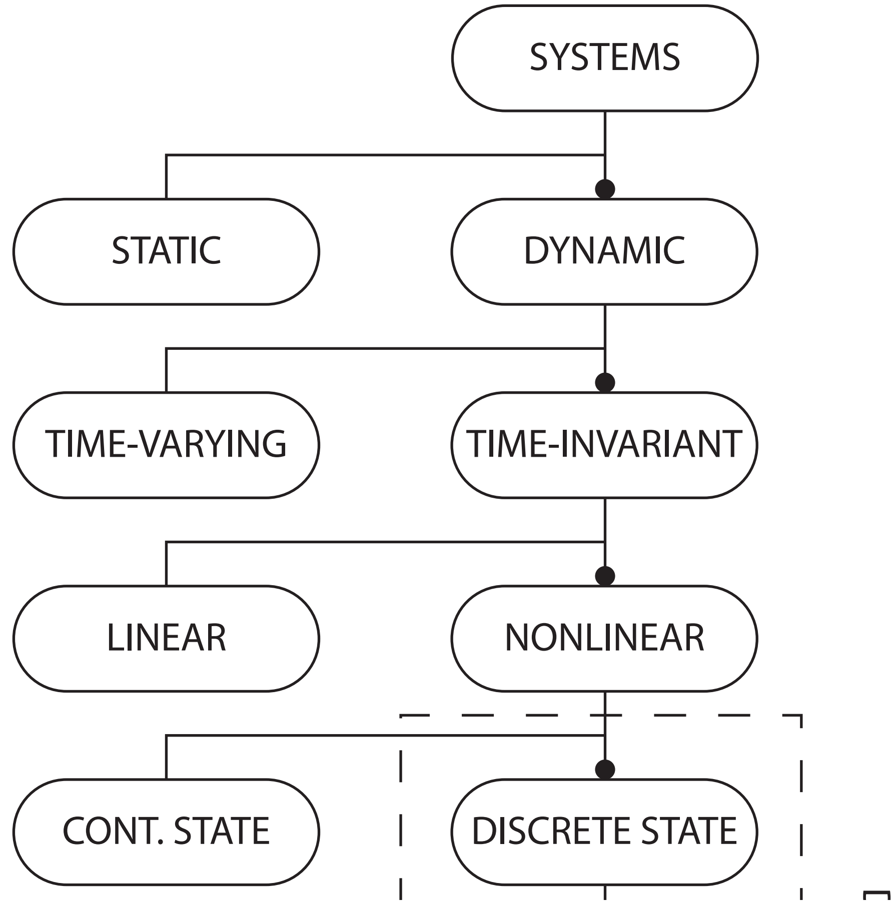
INPUT-OUTPUT MODELS



STATIC VS DYNAMIC MODELS



TIME-VARYING VS TIME-INVARIANT MODELS



DISCRETE VS CONTINUOUS STATE MODELS

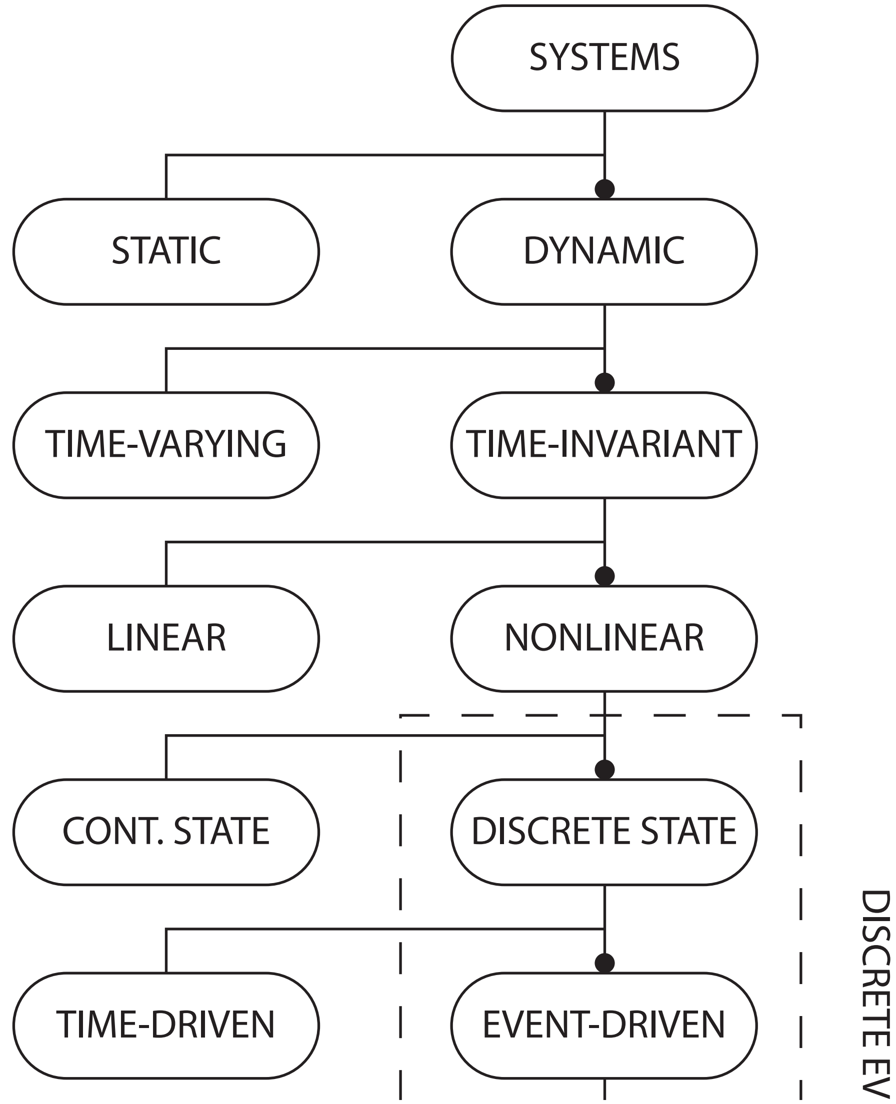
Definition: *The state of a system at time t0 is the information required at t0 such that the output y(t), for all t≥t0, is uniquely determined from this information and from u(t), t≥t0.*

CONCEPT OF A STATE

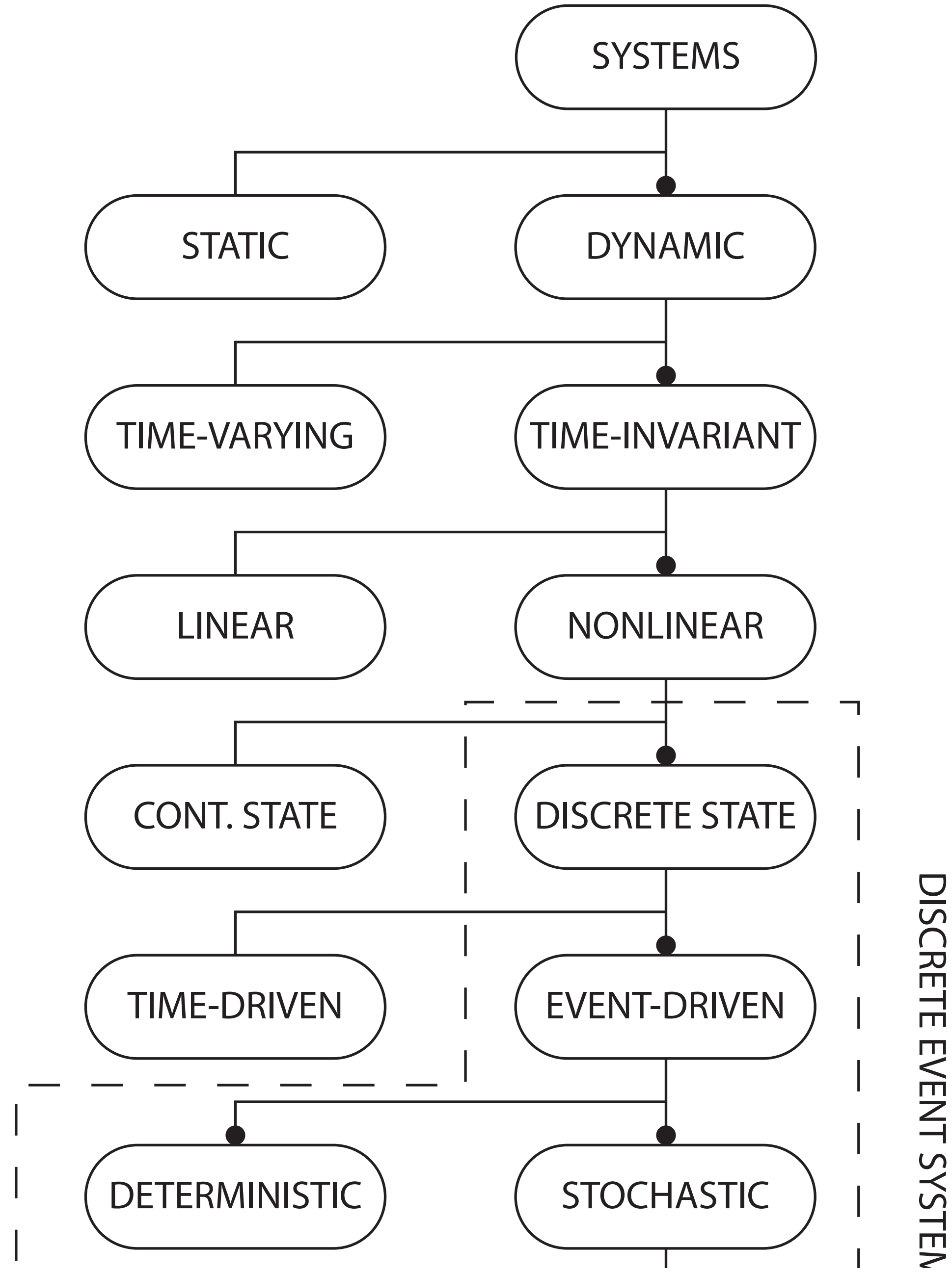
Definition: *The state of a system at time t0 is the information required at t0 such that the output y(t), for all t≥t0, is uniquely determined from this information and from u(t), t≥t0.*

In other words … *If God yesterday assembled yesterday's state of the state of the world … we would not know the difference.*

CONCEPT OF A STATE



TIME-DRIVEN VS EVENT-DRIVEN MODELS



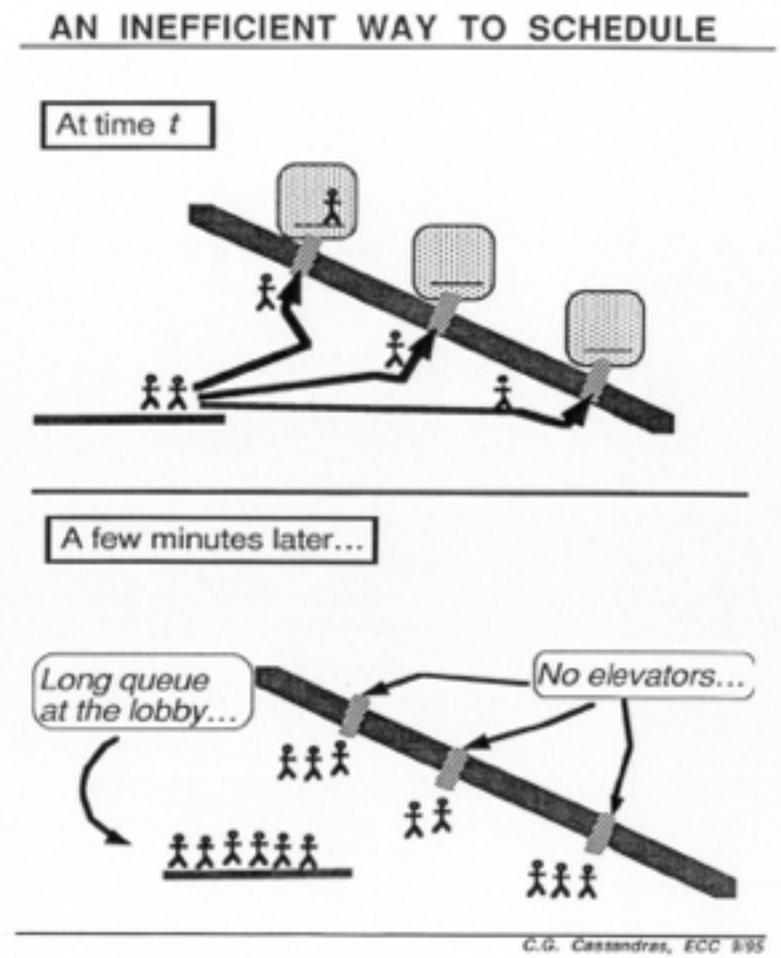
DETERMINISTIC VS STOCHASTIC MODELS

Dispatching Control in an Elevator System[2]

- Events: *hall_call, car_call, car_arrives_at_floor_i,* etc.
- States: position of car k, number of passengers waiting at floor i, etc. (very large state space!)
- Control problem: *which car to send where so as to achieve "satisfactory" performance?*
- Performance measures: *average* waiting time (until car comes), *average* service time (until car delivers to desired floor), fraction of passengers waiting more (on average) than one minute, etc.
- Probabilistic formulation: passenger arrival rates at floors, probability distribution for destination floors, load times and travel times, etc.
- Common solution: threshold-based control, i.e., hold a car until a *threshold* is reached.
  → The issue is then to determine this threshold and "automatically" adjust it in real-time, based on observed passenger arrival rates.

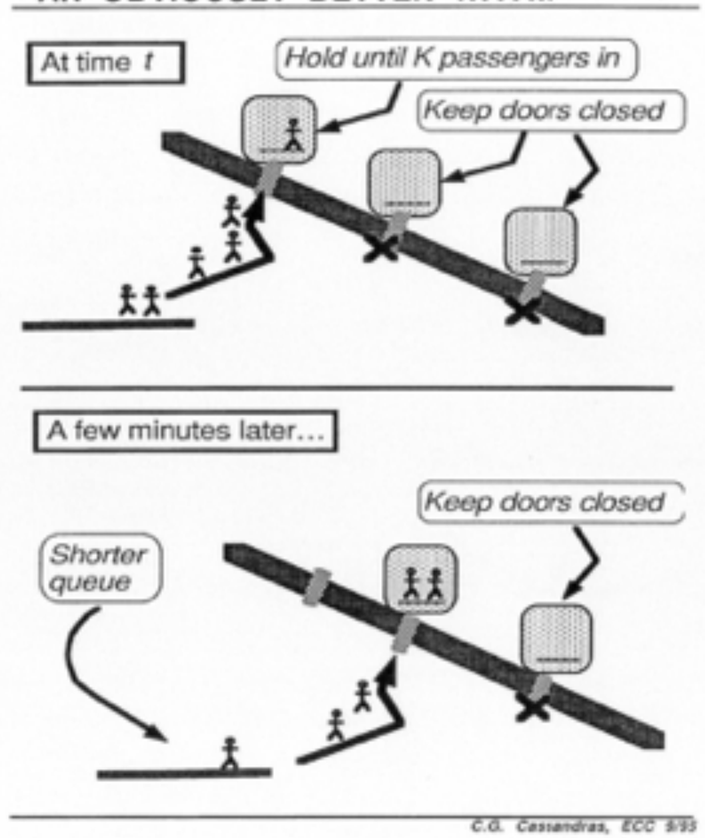
[2]Example due to C. Cassandras

DISCRETE EVENT SYSTEMS

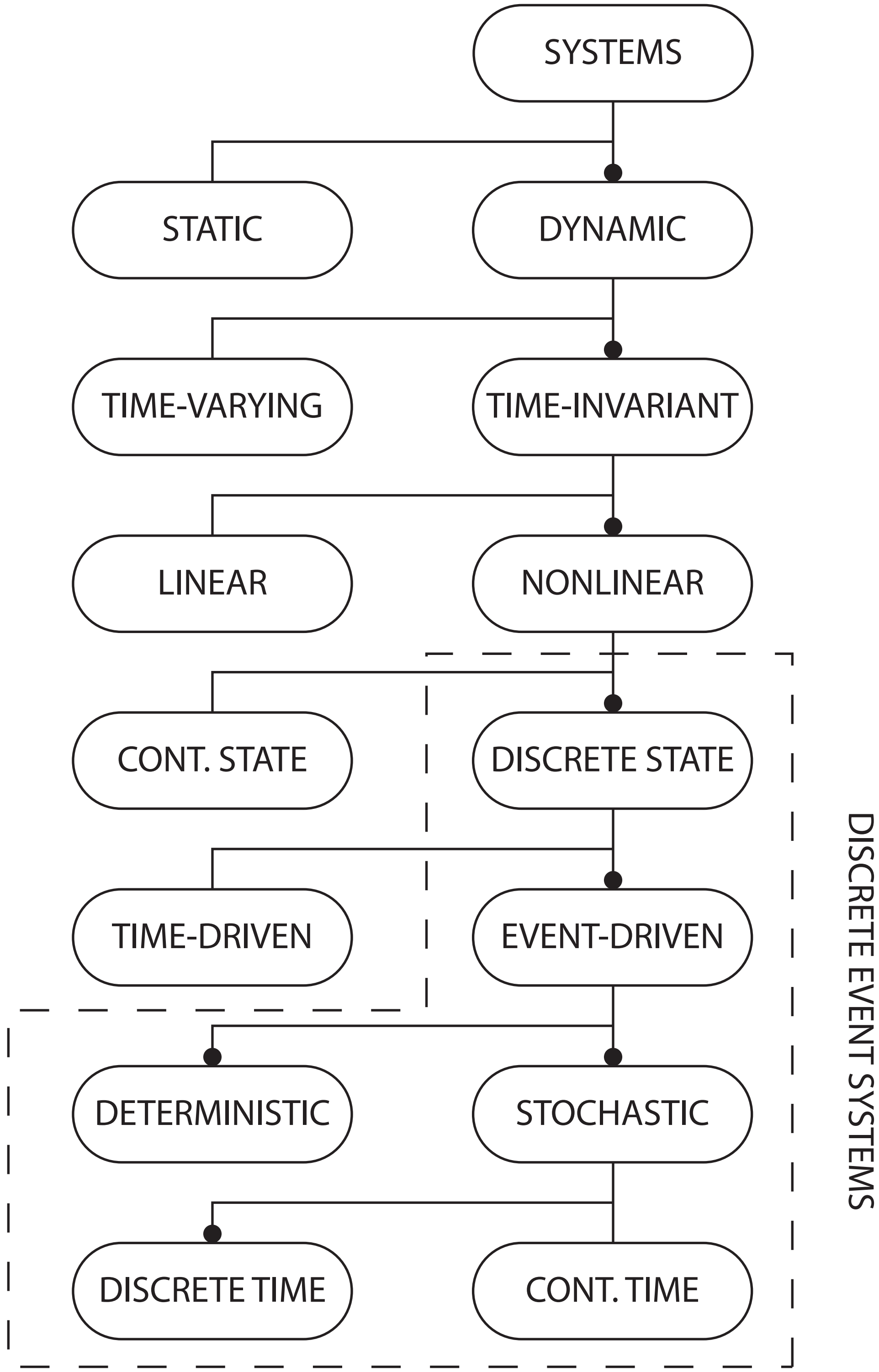


DISCRETE EVENT SYSTEMS

AN OBVIOUSLY BETTER WAY...



DISCRETE EVENT SYSTEMS



A set of events E = {e1,e2,e3,…},

LANGUAGE:
e1e2e3…

TIMED LANGUAGE:
(e1,t1)(e2,t2)(e3,t3)…

STOCHASTIC TIMED LANGUAGE:
P(s1):(e1,t1)(e2,t2)(e3,t3)…

THREE LAYERS OF ABSTRACTION

Definition: *A* ***string*** *is a sequence of events.*

Definition: *A* ***language*** *over an event set E = {e1,e2,e3,…} is a set of finite-length strings formed from events in E.*

Language operations: *concatenation, prefix closure, Kleene closure, complement*

WHAT IS A LANGUAGE IN DES THEORY?

Example: (server repair), E = {s,c,b,r}. In a queuing system, a server may start an operation ’s’, complete an operation ‘c’, break down ‘b’, and be repaired ‘r’.

Corresponding language: The language that describes this simple process $L = \{(s(cs)^n br)^m, n \geq 0, m \geq 0\}$. This language is infinite.

Corresponding automaton: Such a language has a well defined finte description.

REPRESENTING LANGUAGES USING AUTOMATA

Definition: (Deterministic automaton)
A deterministic automaton, denoted by G, is a six-tuple

$$A = (Q, E, g, q_0, \Gamma, Q_m)$$

Definition: (Languages generated and marked)
The language generated by G is

$$\mathcal{L}\left(A\right) := \{s \in E^* : f(q_0, s) \text{ is defined}\}$$

The language marked by G is

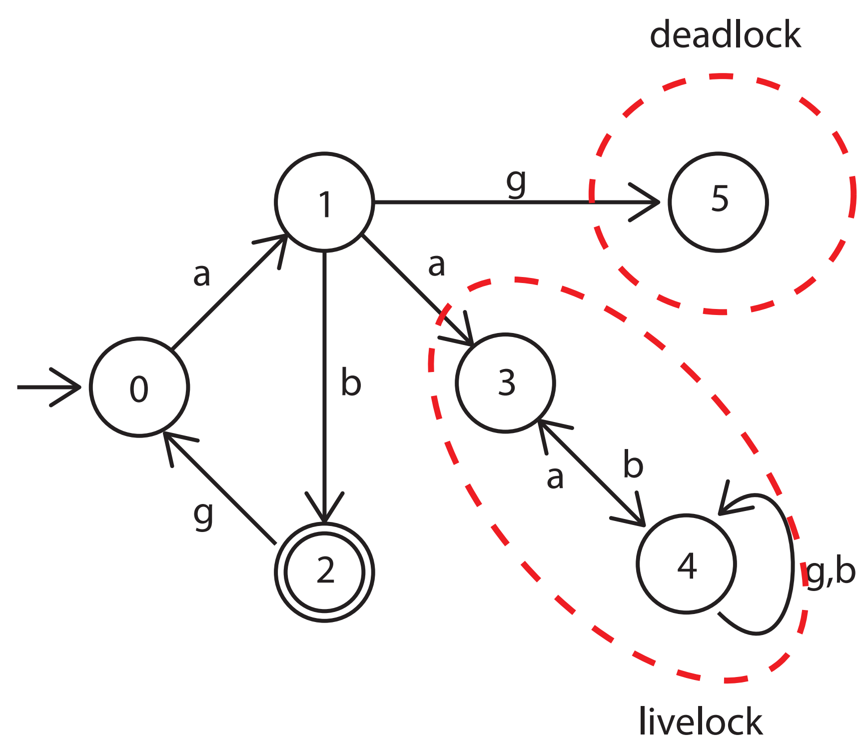
$$\mathcal{L}_m\left(A\right) := \{s \in E^* : f(q_0, s) \in Q_m\}$$

FORMAL DEFINITION

Definition: (Blocking)
Automaton A is said to be blocking if

$$\overline{\mathcal{L}_m(A)} \subset \mathcal{L}(A)$$

Example:



DEADLOCK AND LIVELOCK

Definition: (Accessible Part)
A state is called accessible if it can be reached from the initial state.

Definition: (Coaccessible Part)
A state q is called coaccessible if there is a string that goes through q before it goes through a state in Qm.

Computation of accessible and coaccessible parts is an important model checking tool.

Definition: (Complement)
The complement automaton generates and marks the complement languages.

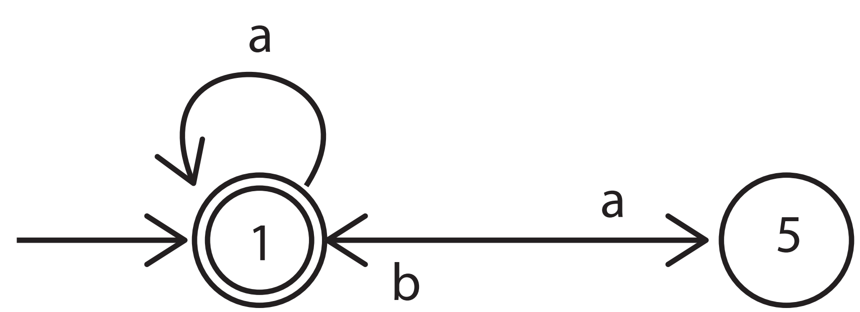
Exercise: Perform all these operations on the automaton in the previous slide.

OPERATIONS ON SINGLE AUTOMATA

Definition: (Nondeterministic automaton)
A nondeterministic automaton is a sixtuple

$$A_{nd} = (Q, E, g_{nd}, q_0, \Gamma, Q_m)$$

Example:



Theorem:
Any language generated by a nondeterministic automaton can be generated by a deterministic automaton.

NONDETERMINISTIC AUTOMATON

Example: (Non-regular language)
The following language cannot be generated by a finite state automaton.

$$L = \{\epsilon, ab, aabb, aaabbb, ...\} = \{a^n b^n : n \geq 0\}$$

Definition: (Regular language)
A language is said to be regular if it can be marked by a finite-state automaton.

Theorem: The following are language operations that preserve regularity: prefix closure, Kleene closure, complement, union, concatenation, intersection.

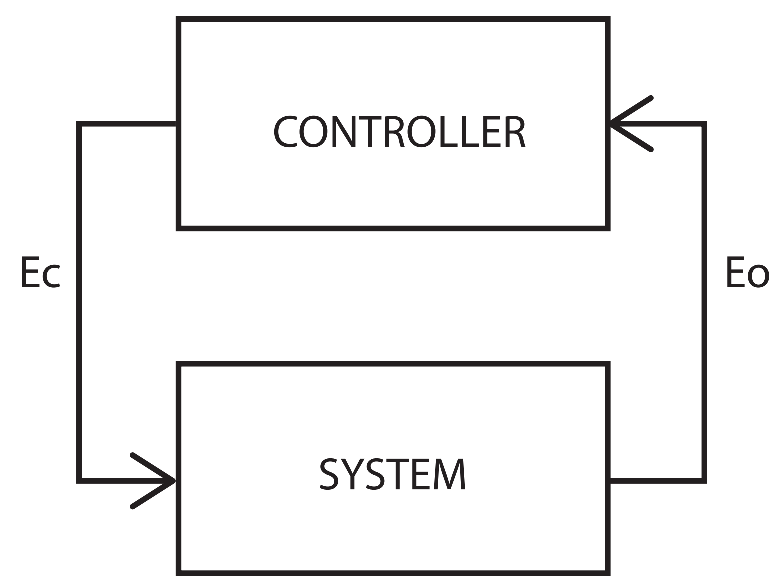
FINITE STATE AUTOMATON

Theorem: (Regular language construction)
For an event set E = {e1,e2,…en}, consider the basic languages: {},1, {ei}. Then any regular language can be constructed by repeated application of concatenation, union, and Kleene closure.

REGULAR LANGUAGES

Definition: (Controllable and Observable events)
The event set can be generally divided into controllable and observable sets.

$$E = E_c \cup E_{uc}$$
$$E = E_o \cup E_{uo}$$



INPUT/OUTPUT/CONTROL

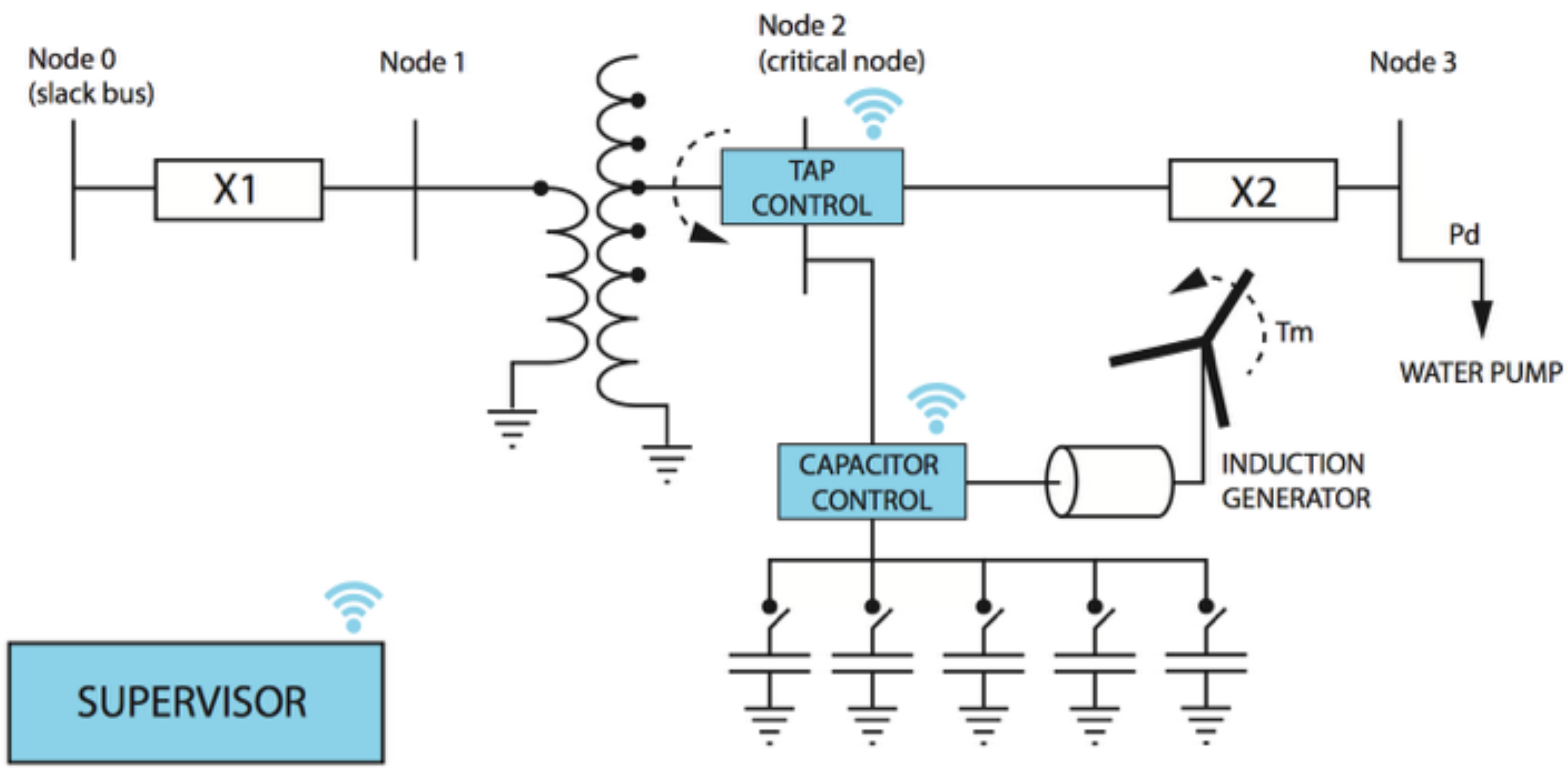
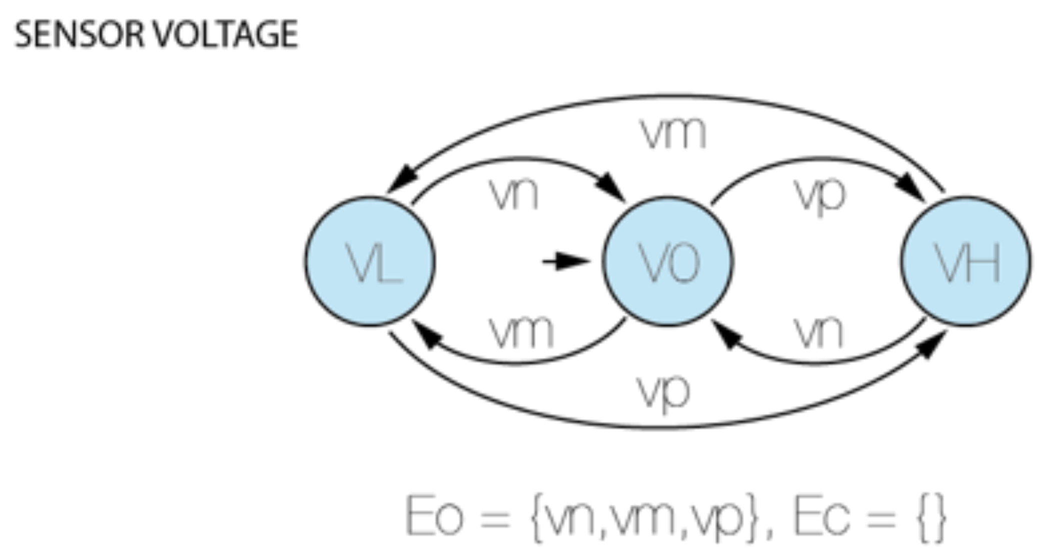
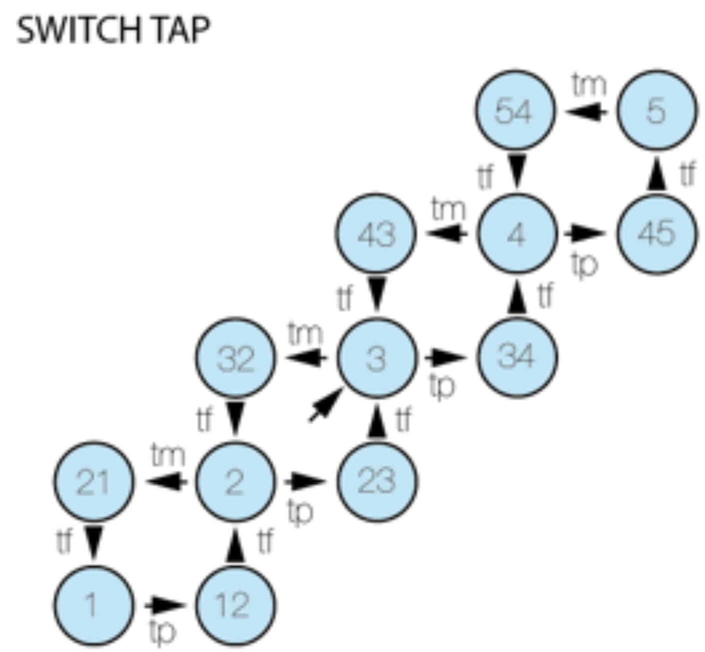


Figure 1: Network system schematic.

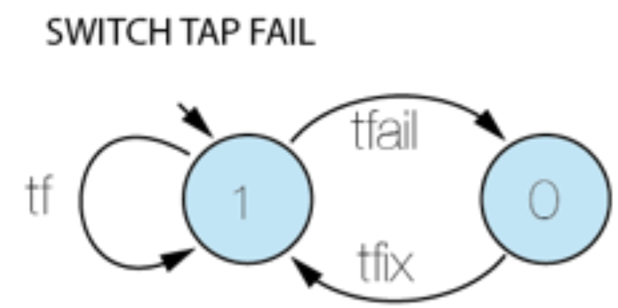
CASE STUDY



VOLTAGE SENSOR (SYSTEM)



Eo = {tf}, Ec = {tp,tm}



Eo = {}, Ec = {}

TAP CHANGER (SYSTEM)

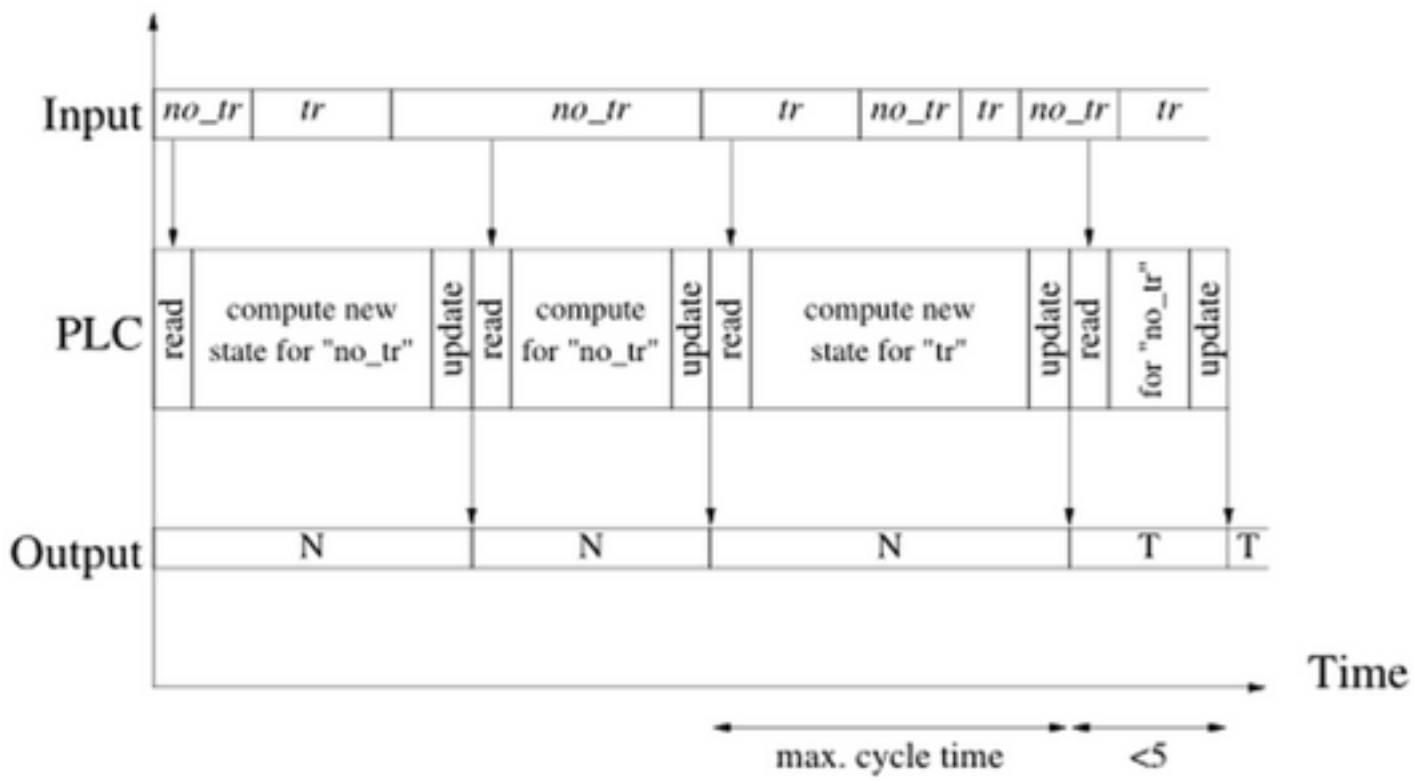


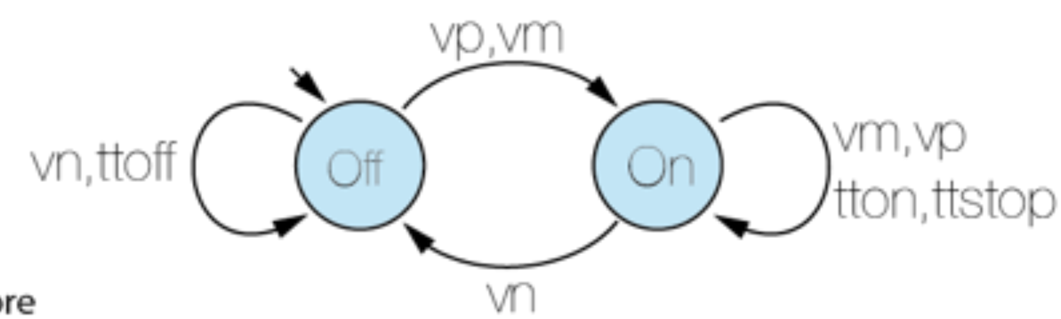
Fig. 2. Cyclic behaviour of a PLC.

GENERAL PLC SCHEMA

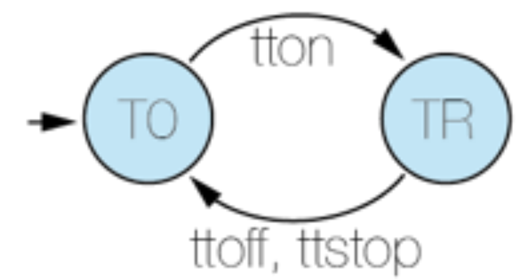
TAP PLC TIMER CONTROLLER

RULE 1: tton or ttstop can occur only if from the event set {vp,vm,vn} vp or vm occured last

RULE 2: ttoff can occur only following vn or before any events in the set {vp,vm,vn} ocur



TIMER TAP



Eo = {ttstop}, Ec = {tton, ttoff}

TAP PLC OUTPUT CONTROLLER

RULE 1: tp can occur only if vp is the last event in the string from the set {vm,vp}
RULE 2: tm can occur only if vm is the last event in the string from the set {vm,vp}
RULE 3: tp and tm can occur only if ttstop occured
RULE 4: an event from the set {tm,tp} can occur at most once between consecutive ttstop events or following the last ttstop event

PLC AUTOMATA MODEL