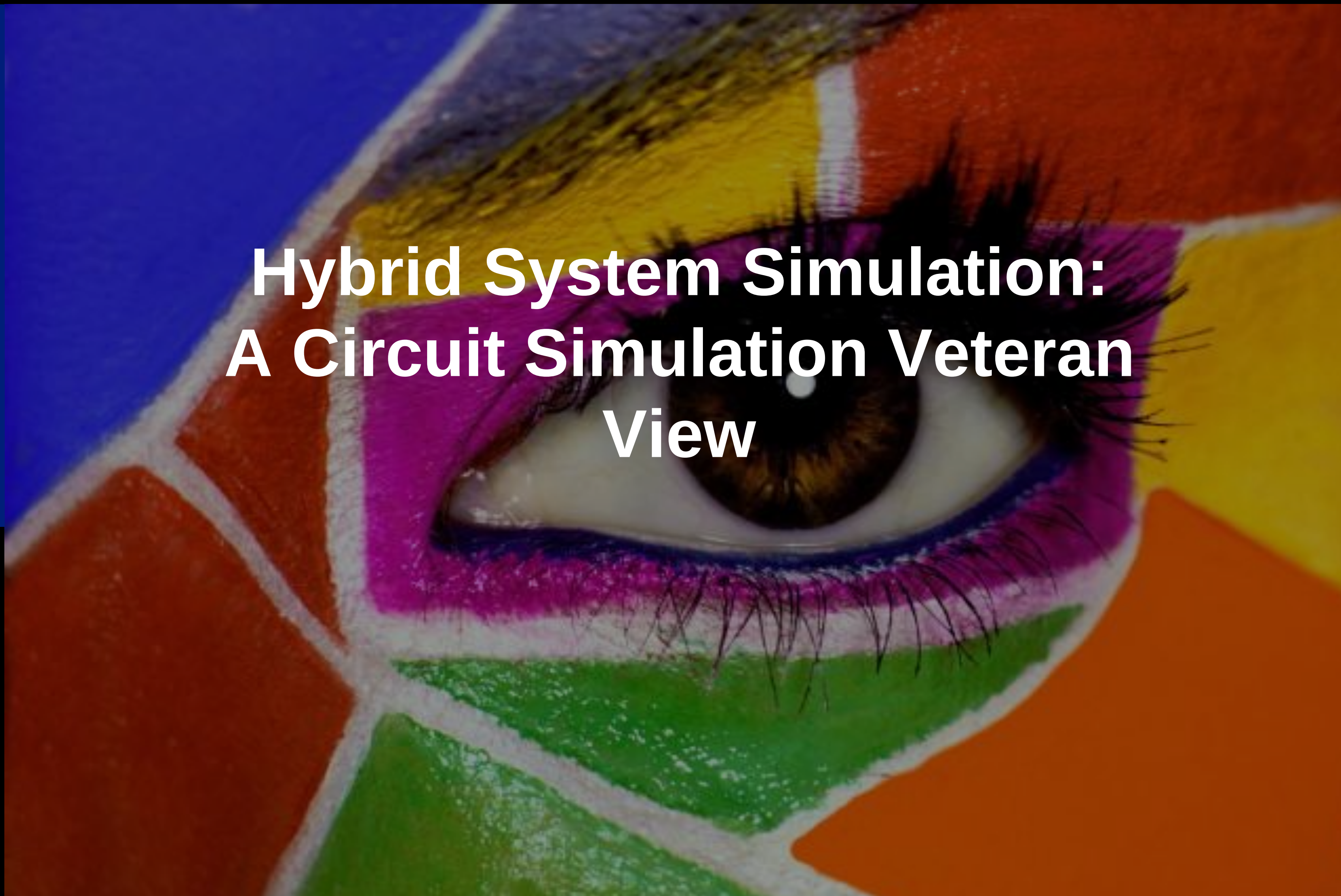
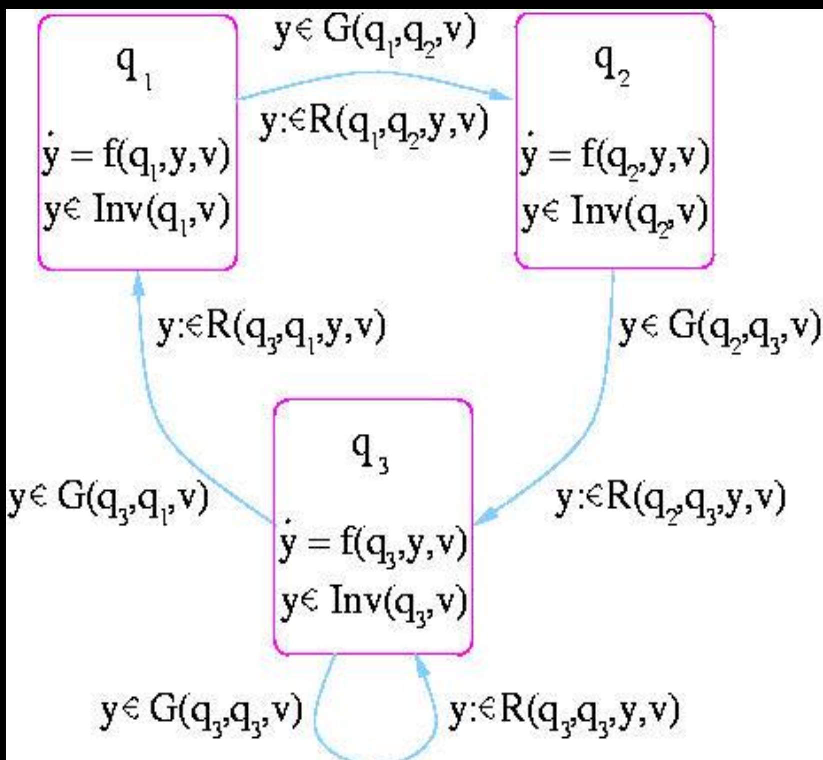
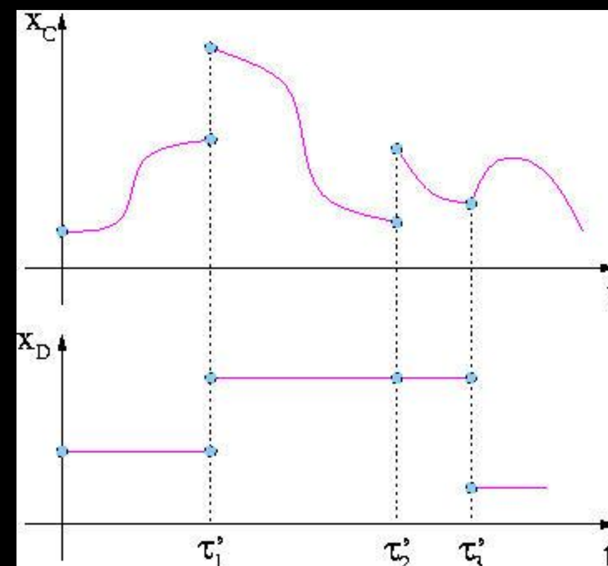
Hybrid System Simulation: A Circuit Simulation Veteran View



What Are Hybrid Systems?

- Dynamical systems with interacting continuous and discrete dynamics

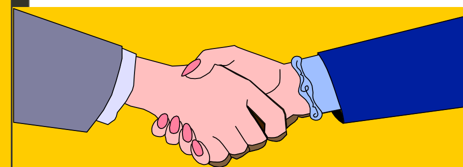




Proposed Framework

Control Theory

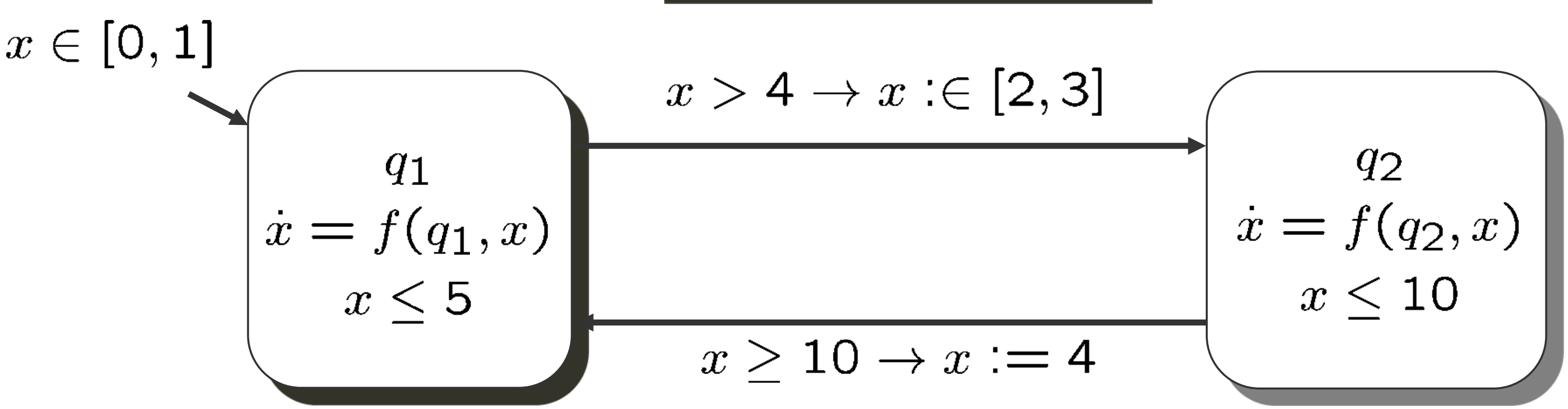
- Control of individual agents
- Continuous models
- Differential equations



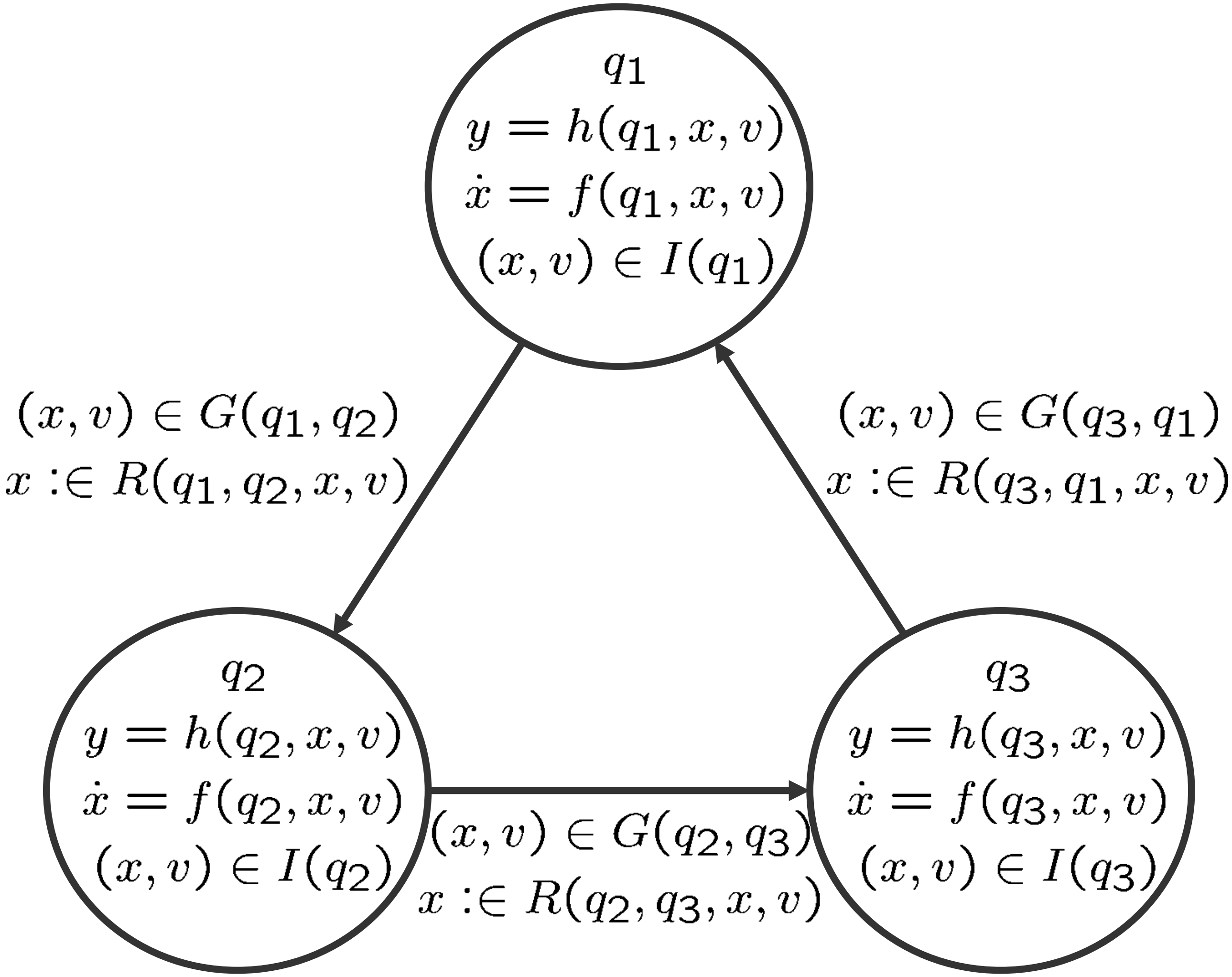
Computer Science

- Models of computation
- Communication models
- Discrete event systems

Hybrid Systems



Hybrid Systems



Hybrid Systems

- **Hybrid Automata (Lygeros-Tomlin-Sastry, 2001)**

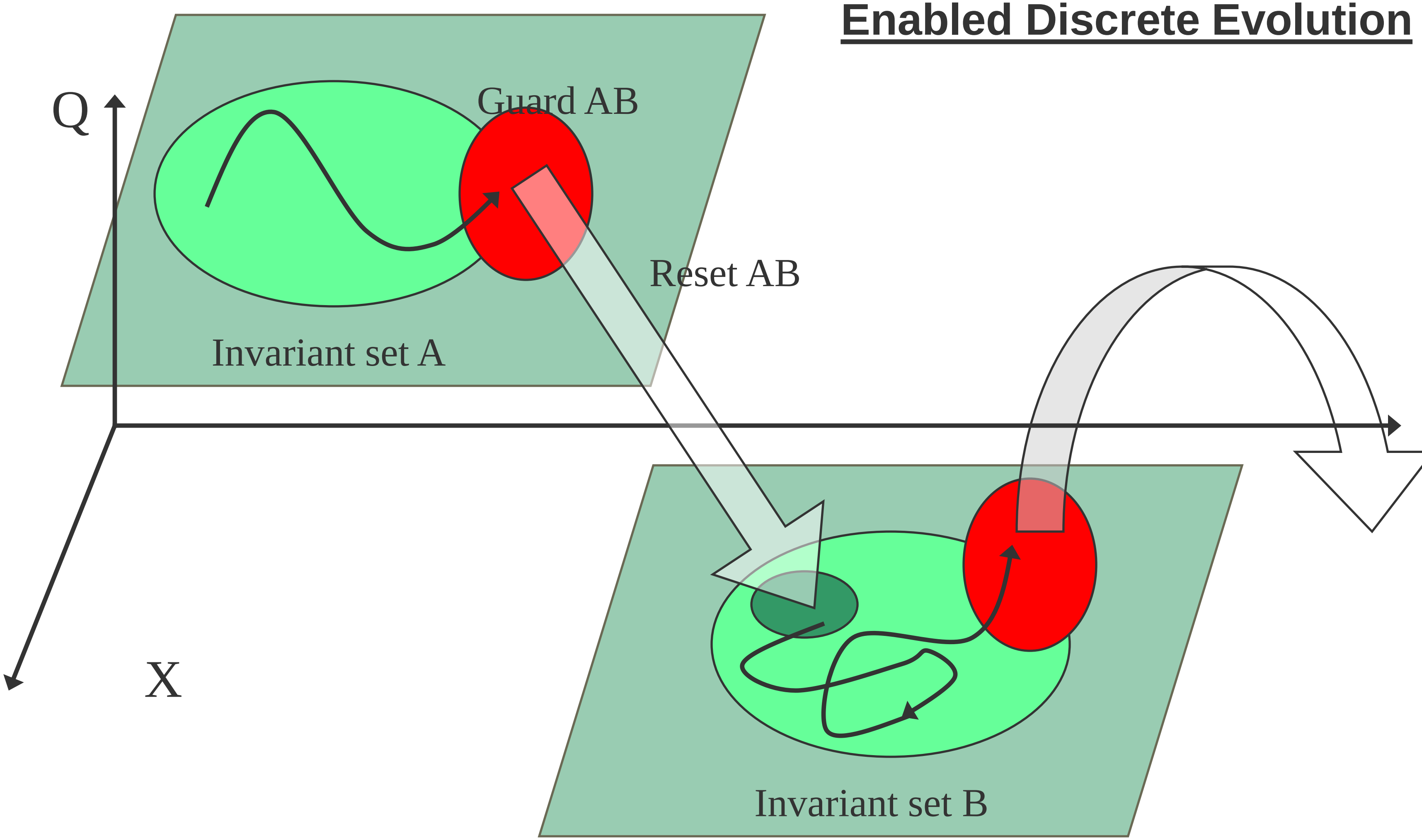
$H = (Q, X, V, Y, Init, f, h, I, E, G, R)$

- Q is a finite collection of discrete state variables;
- X is a finite collection of continuous state variables;
- V is a finite collection of input variables, $V = V_D \cup V_C$;
- Y is a finite collection of output variables, $Y = Y_D \cup Y_C$;
- $Init \subseteq Q \times X$ is a set of initial states;
- $f : Q \times X \times V \to \mathbb{R}^n$ is a vector field;
- $h : Q \times X \to Y$ is an output map;
- $I : Q \to 2^{X \times V}$ assigns to each $q \in Q$ an invariant set;
- $E \subset Q \times Q$ is a collection of discrete transition;
- $G : E \to^{X \times V}$ assigns to each $e \in E$ a guard; and,
- $R : E \times X \times V \to 2^X$ defines a reset relation.

Ref: J. Lygeros, C. Tomlin, and S. Sastry,
***The Art of Hybrid Systems*, July 2001.**

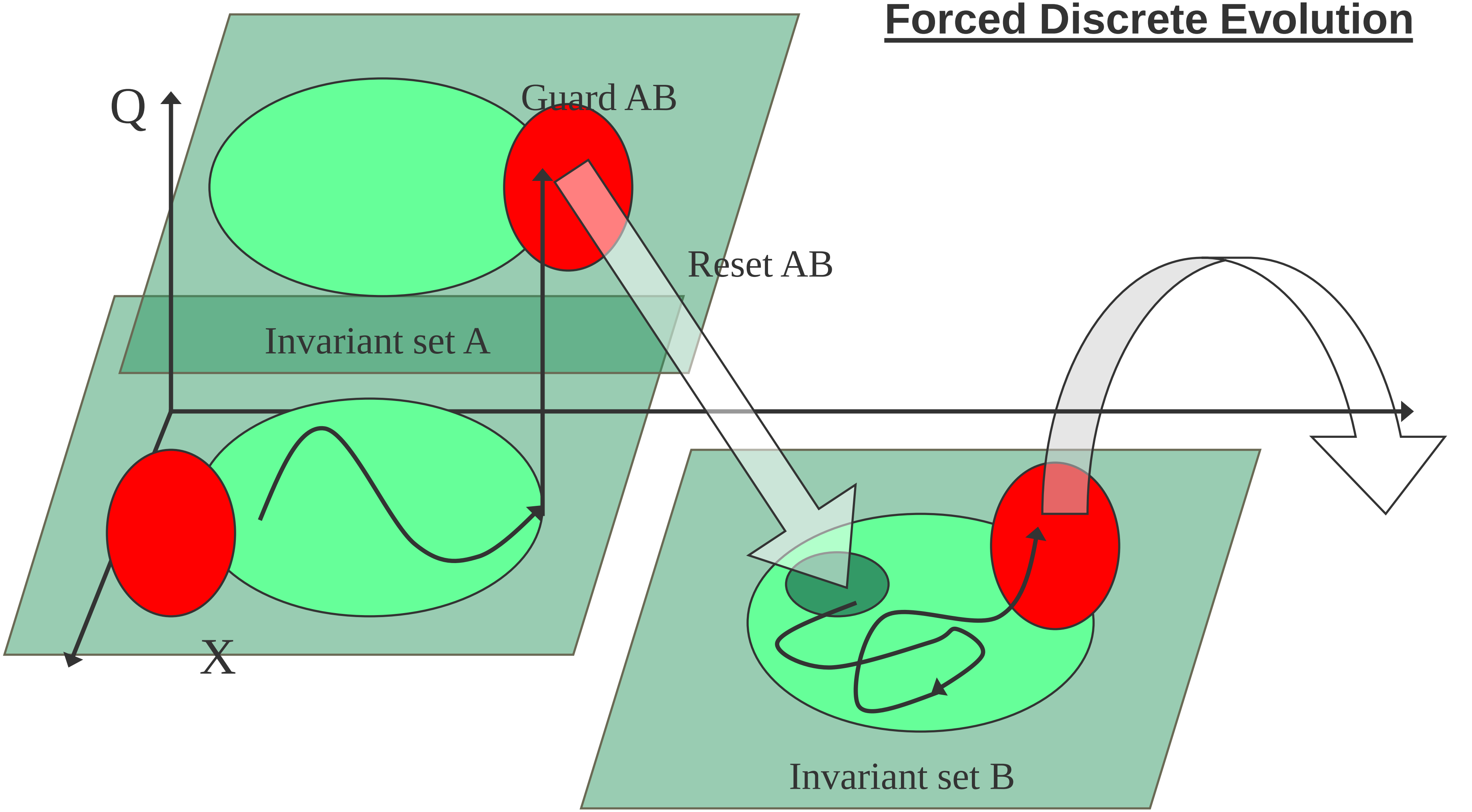
Hybrid Systems

Enabled Discrete Evolution



Hybrid Systems

Forced Discrete Evolution



Simulation

- Assess properties of the system
- Simulation must be accurate WITH RESPECT TO PROPERTY! (Accuracy means “almost” same results as the actual system)
- How do we make sure of this?

Back to the Future?

- In late 1970s, strong interest arose to simulate mixed analog-digital circuits
- Analog circuit simulation was in great demand and well suited to needs
- Digital circuit simulation used everywhere as well
- Mixed analysis raised all kind of interesting problems

Continuous Time (1970s)

- Numerical analysis techniques for the solution of ODEs
- State-of-the-art considered circuit simulators: SPICE(DOP, ARN, ASV) and derivatives, SPECTRE (K. Kundert, J White and ASV)
  - Hundred of thousands of variables
  - Nonlinear equations
  - Sophisticated heuristics (it takes about 5-9 years to build robustly)
  - Takes advantage of the peculiarities of circuit component models

Continuous Time

- Model of computation is DISCRETE TIME
  - All variables are computed at each time point
    - no run-time scheduling decisions on variable computation
  - Time interval can be
    - fixed (bad for stiff systems), but no run-time decision
    - variable (sophisticated solvers have this)
      - Variable time step algorithm predicts a time step that will satisfy accuracy criterion based on previous behavior
      - After actual computation, step may be rejected because constraints are violated
      - Run-time scheduling

Continuous Time

- Time-step prediction based on differentiability assumptions
- If discontinuity arises in input waveforms, all the bets are off: we must re-initialize at “break point” even if we are VERY close to it
- If discontinuity arises because of component models, time step may be reduced to a very small limit in the attempt of locating the discontinuity (“Internal time step too small”)
- Discontinuity may be located using interpolation techniques

Discrete Domain

- Two basic techniques:
  - Zero-time assumption:
    - Static scheduling of computation
    - Can be done off-line for maximum efficiency (cycle-based simulation)
  - Components modeled with delay. (Discrete Event Model).
    - All components evaluated at the same time-point always (wasteful)
    - Follow reaction to events: schedule components whose inputs have changed (assumes internal dynamics completely captured by pure delay) Selective-trace event-driven simulation.

Synchronization Problem

- “Synchronization” between domains:
  - sample the continuous time interface variables
  - integrate discrete event interface signals
  - detect guards and invariants (zero crossing detection)
- How do we advance time in a consistent way?

Mixed-Mode Simulators

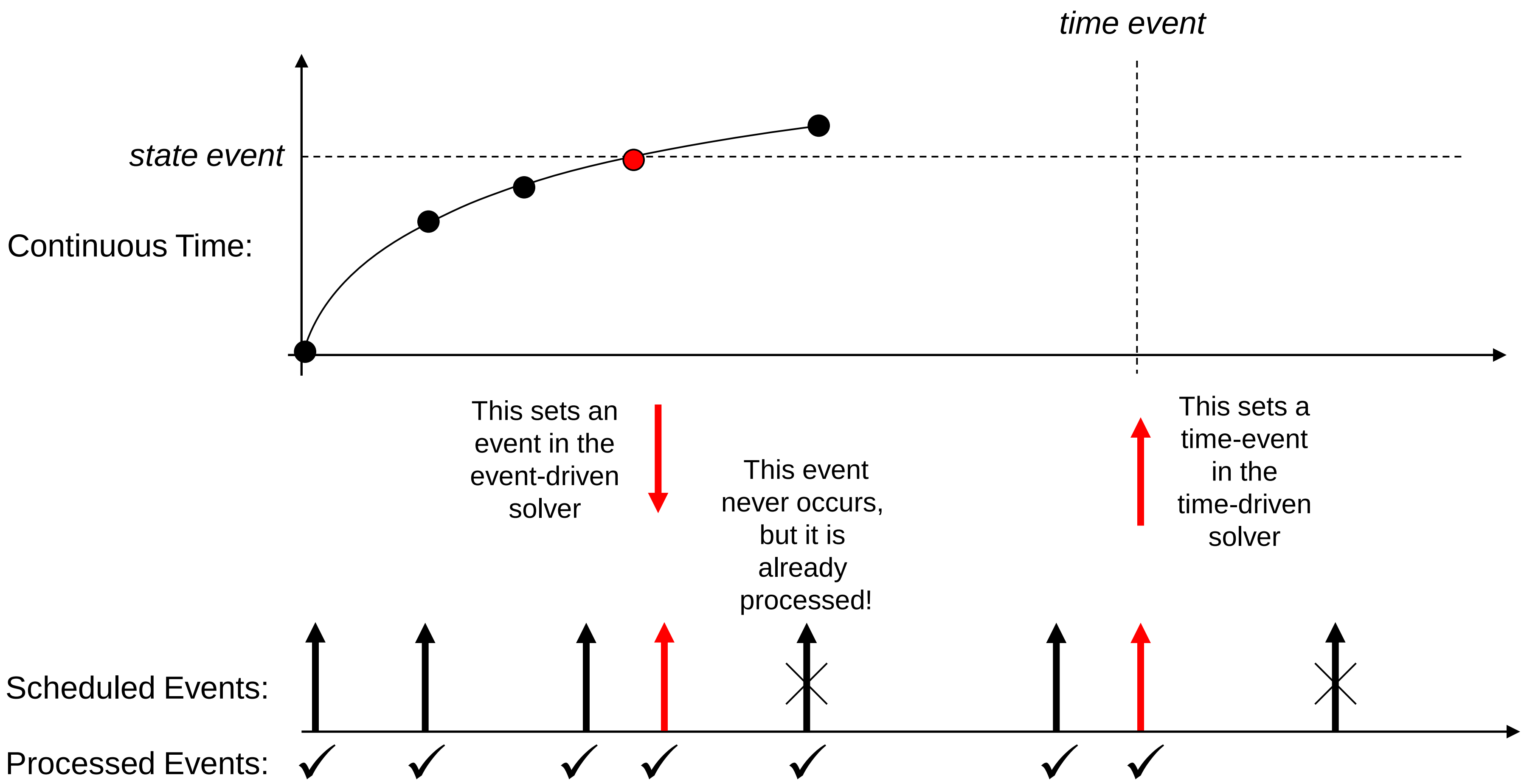
- SPLICE first in its kind (R. Newton’s Thesis, 1979):
  - Single simulator
  - Used event-driven selective trace technique
  - Using a unique time wheel where predicted evaluation time for continuous dynamics and actual evaluation time for discrete dynamics are stored and schedule accordingly

  PROBLEM: if predicted evaluation time is “wrong”, what can we claim about accuracy????
- SPLICE 2
  - If predicted evaluation time wrong, reset time wheel and unwind simulation

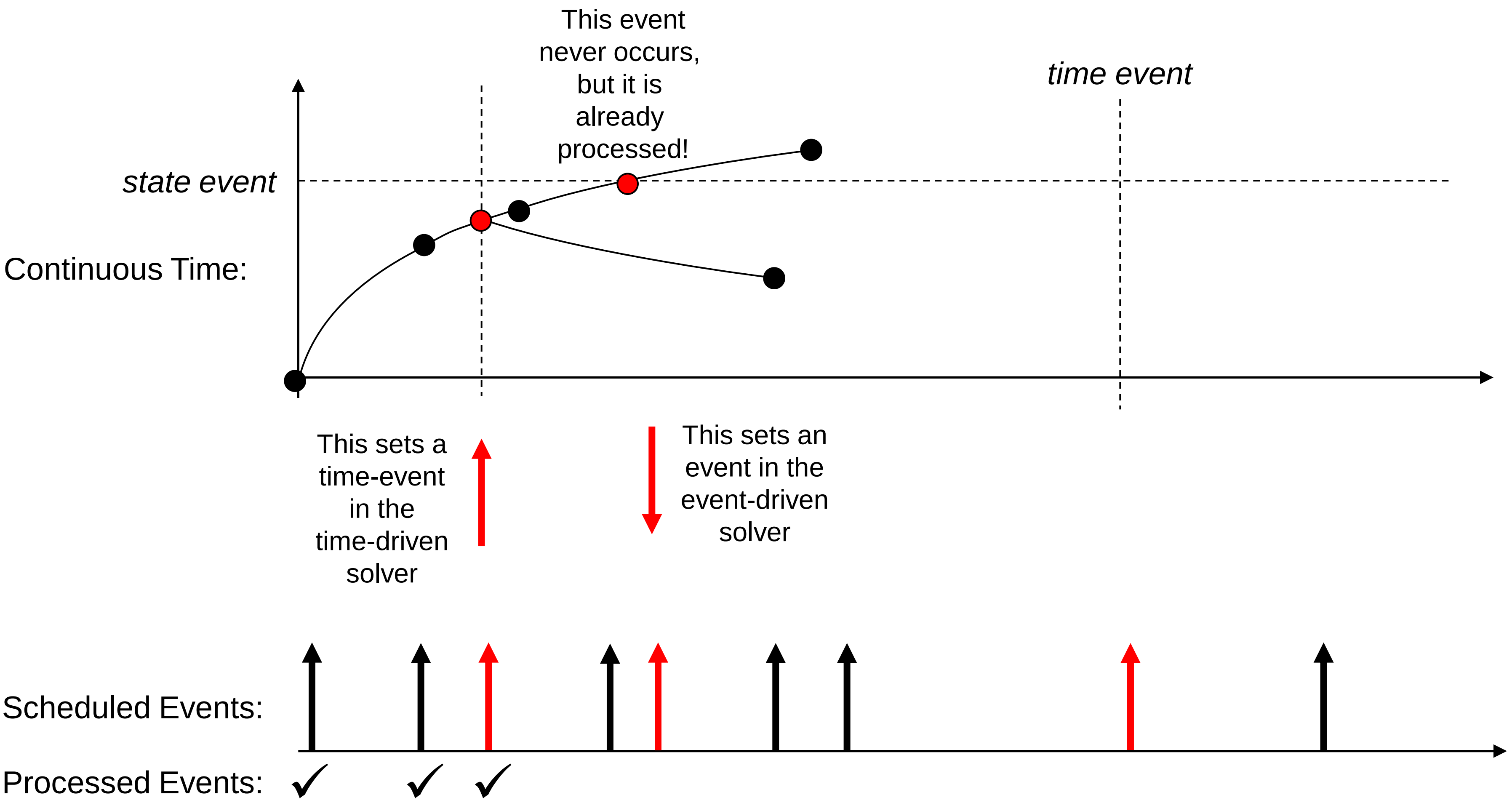
Problem Equivalent to Multi-rate Integration Methods

- Instead of using a single time step for the entire system, use different time steps per each subsystem (Petzold, Gear et al)
- Three heuristic methods were proposed:
    1. Schedule the fastest dynamics first
    2. Schedule the slowest dynamics first
    3. In-between
- Method 1 was the least efficient, while 2 was the most efficient

Event-driven leads



Time-driven leads



Questions

- Can we restrict the modeling constructs that are allowed?
  - For example, no state events allowed to trigger events in the event-driven solver
- What inaccuracy is acceptable?
  - Not do zero-crossing detection
  - Run event-driven part at a quick base rate
- What efficiency is necessary?
  - Lock step approach
  - Use one solver all together?
    - Do you lose the ability to handle a batch of discrete events independently?
  - Other techniques?
    - Model differently
- What is the preferred configuration?
  - Time-driven leads, store continuous-time state
  - Event-driven leads, store discrete event state
  - Alternatives?