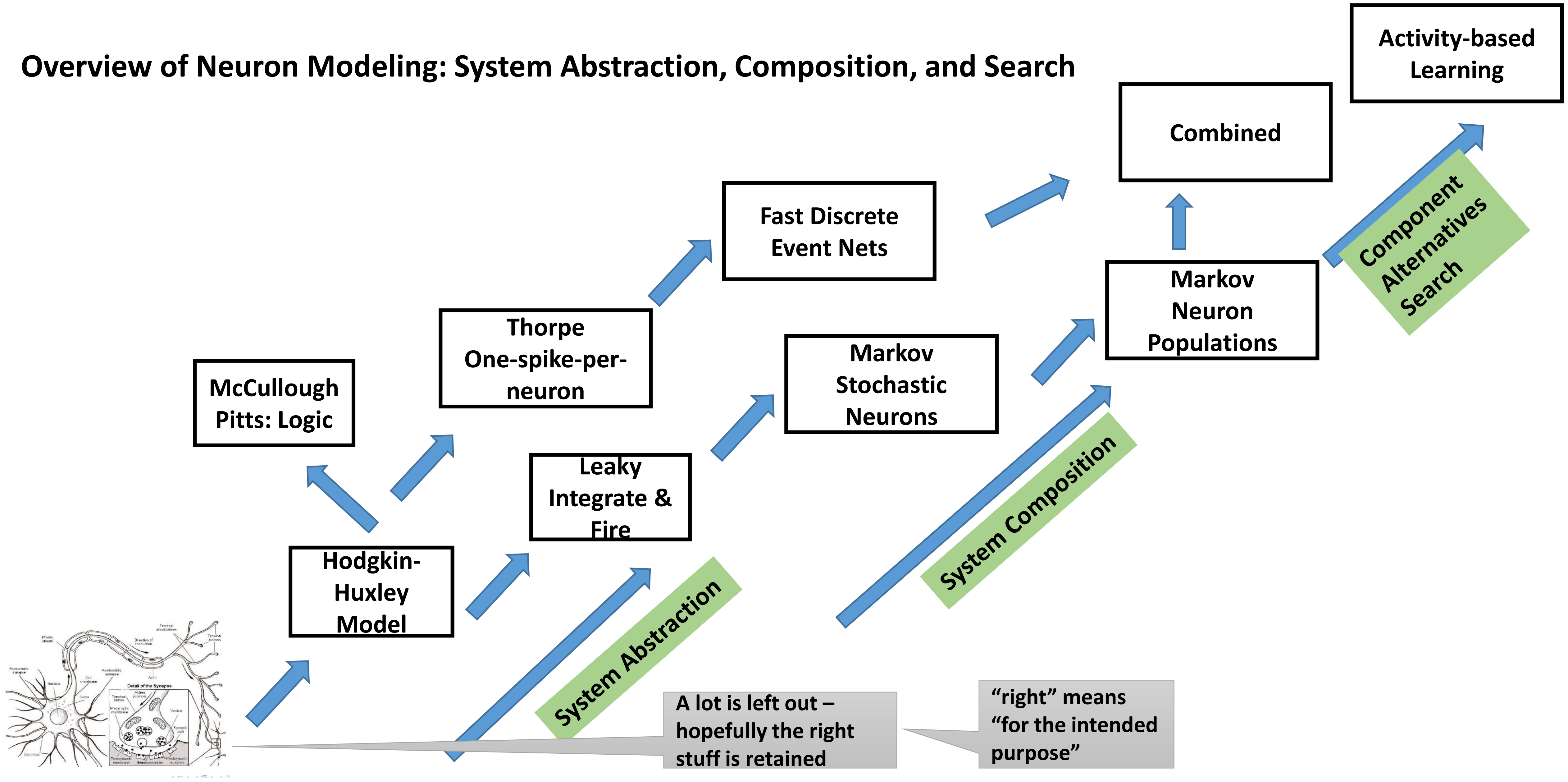
Lecture 2

Overview of Neuron Modeling: System Abstraction, Composition, and Search



A lot is left out – hopefully the right stuff is retained

“right” means “for the intended purpose”

Taxonomy of Levels and types of DEVS Neuron Models

- Small circuit oriented

  Discrete Event abstraction, One spike per neuron, fast non-deterministic processing within available time

  - Neuron Basic models = atomic models of different types: decay neuron, simple threshold neuron
  - "Neural Circuits" = coupled model with Neuron models as components specific coupling,
    e.g. decision of arrival order of inputs

- Population oriented

  Provide numbers needed for statistical confidence – deterministic aggregation of probabilistic input

  - Neuron models = atomic models of different types: fire once, refractory spike generator, etc
  - Cell Assembly = coupled model with Neuron models as components and all to all coupling,
    e.g. net of fire once neurons
  - Cell AssemblyComposite = coupled model with Cell Assembly as components and specific coupling, e.g., AndOr Ne

- CombinedMultiType

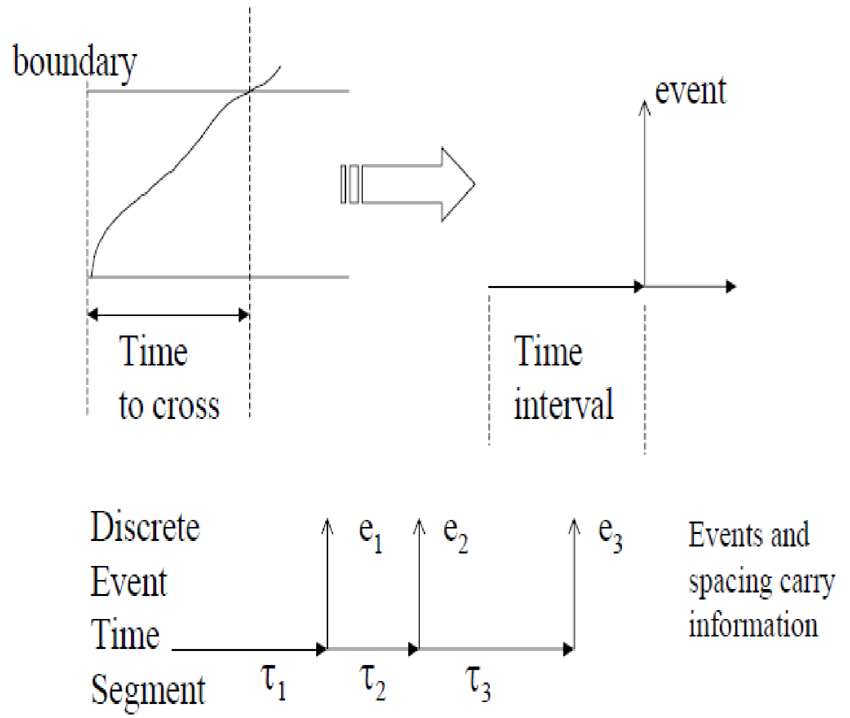
  Combine fast probabilistic and slower deterministic processing

  - Neuron models = from both families
  - "Brain" Models - coupled model with Neuron models, Neural circuits and Cell Assemblies as components

DEVS Formalism

- The **DEVS (Discrete Event Systems Specification)** formalism provides a way of expressing discrete event models
- DEVS is universal for discrete event dynamic systems and is capable of representing a wide class of other dynamic systems
- Universality for discrete event systems is defined as the ability to represent the behavior of any discrete event model where “represent” and “behavior” are appropriately defined
- Concerning other dynamic system classes, DEVS can exactly simulate discrete time systems such as cellular automata and approximate, as closely as desired, differential equation systems
- DEVS closure under coupling supports hierarchical modular construction and composition methodology
- Bottom-up methodology keeps incremental complexity bounded and permits stage-wise verification since each coupled model “build” can be independently tested

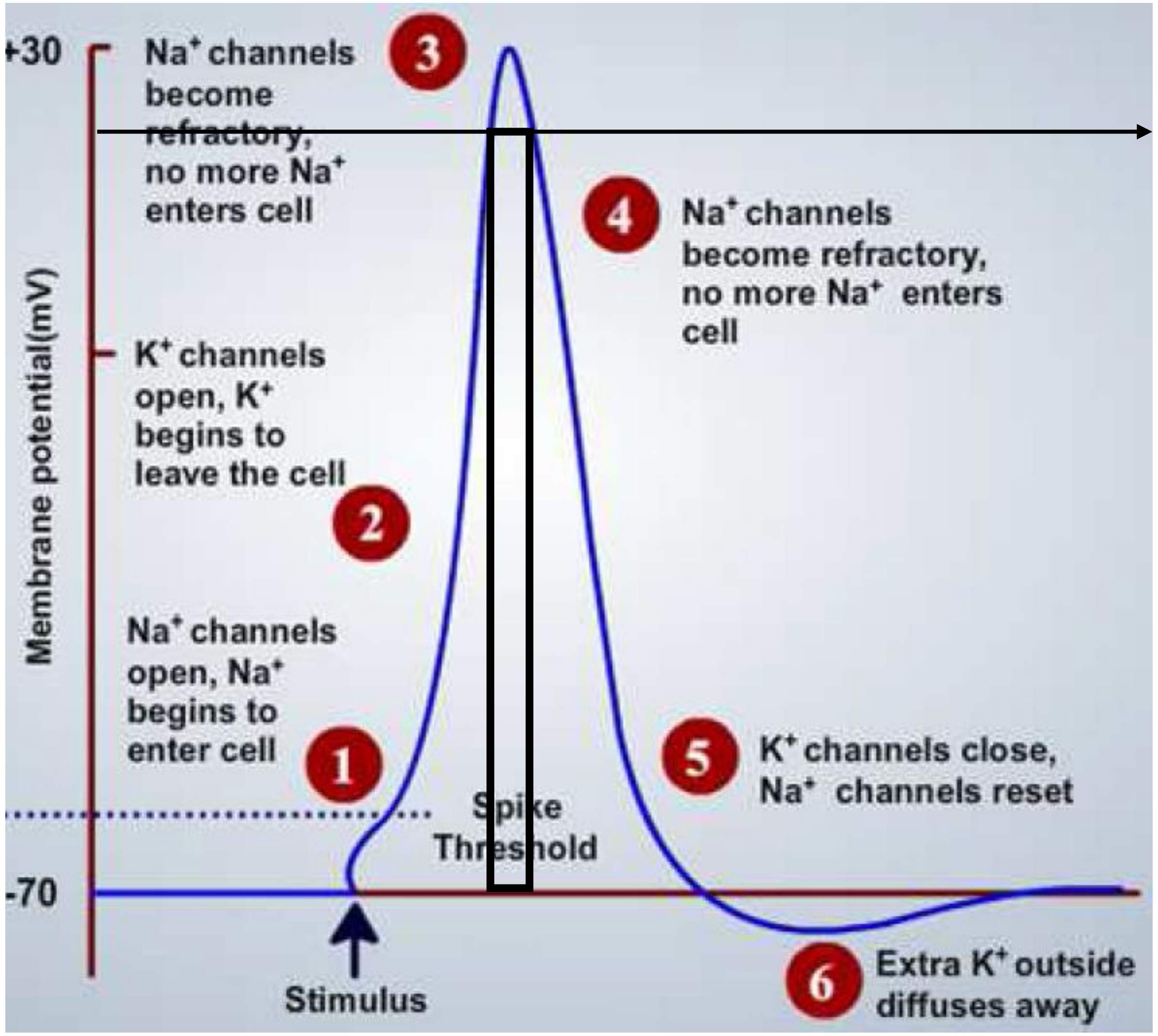
Discrete Event Abstractions

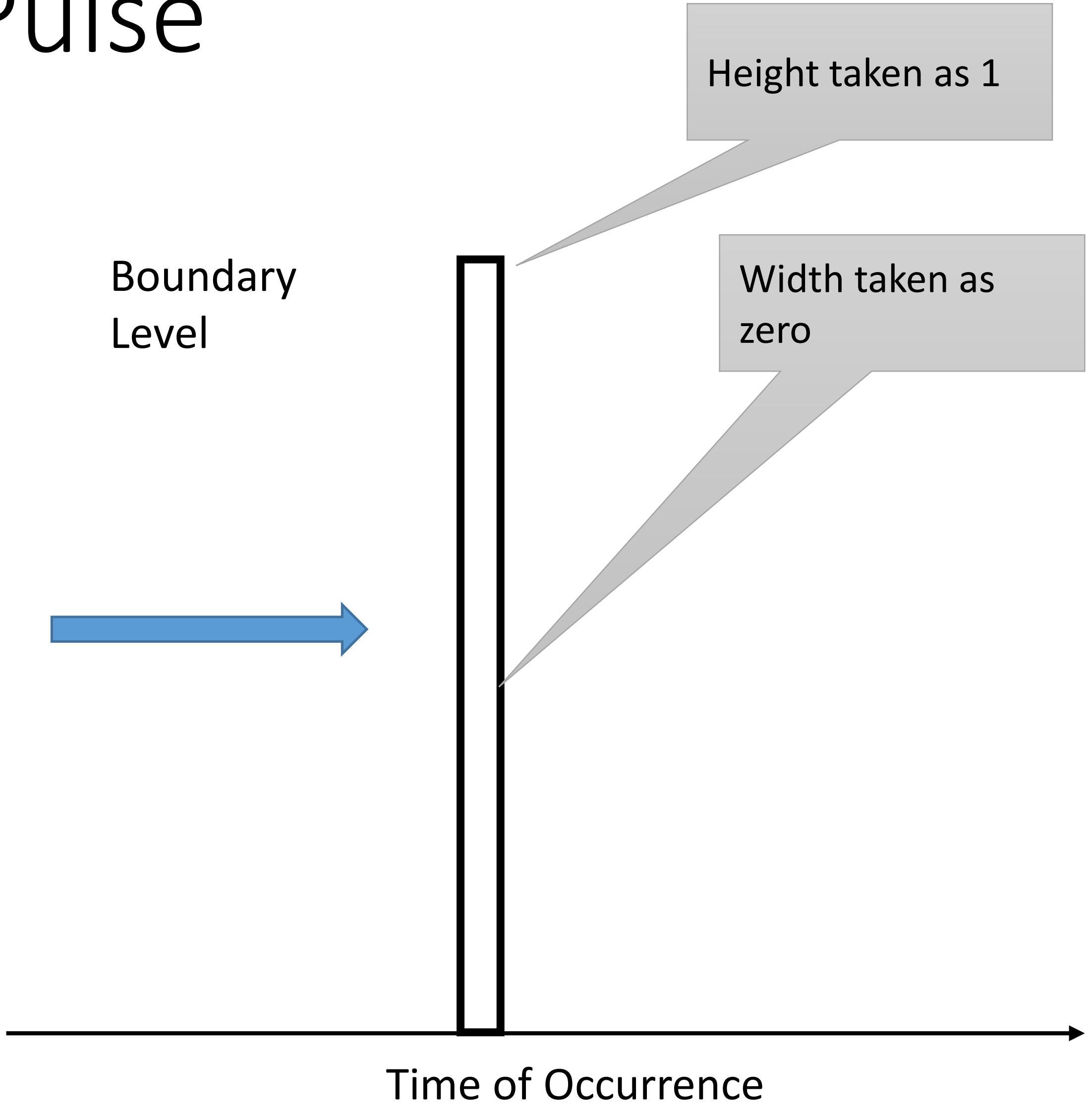


How continuous trajectories are abstracted into time-indexed events

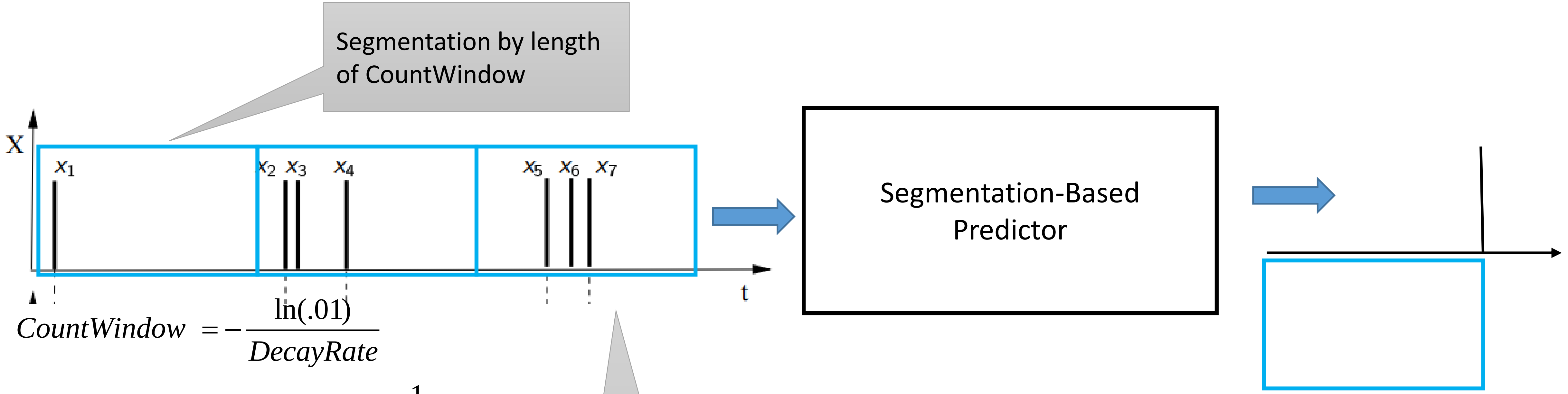
- An abstraction is a formalism that attempts to capture the essence of a complex phenomenon relative to a set of behaviors of interest to a modeler
- A discrete event abstraction represents dynamic systems through two basic elements: discretely occurring *events* and the *time intervals* that separate them
- It is the information carried in events and their temporal separations that DEVS employs to approximate arbitrary systems
- In the quantized systems approach next events are boundary crossings and the details of the trajectories from one crossing to another are glossed over with only the time between crossings preserved

Abstraction of Spike to Pulse





Predictor Objective: Verify that LIF Neuron response to pulse input trajectory can be approximated by segmentation and counting of pulses within segments



Segmentation by length of CountWindow
X
x_1
x_2
x_3
x_4
x_5
x_6
x_7
t
Segmentation-Based Predictor

$$CountWindow = -\frac{\ln(.01)}{DecayRate}$$

$$EffectiveFraction = \frac{1}{CountWindow * DecayRate}$$

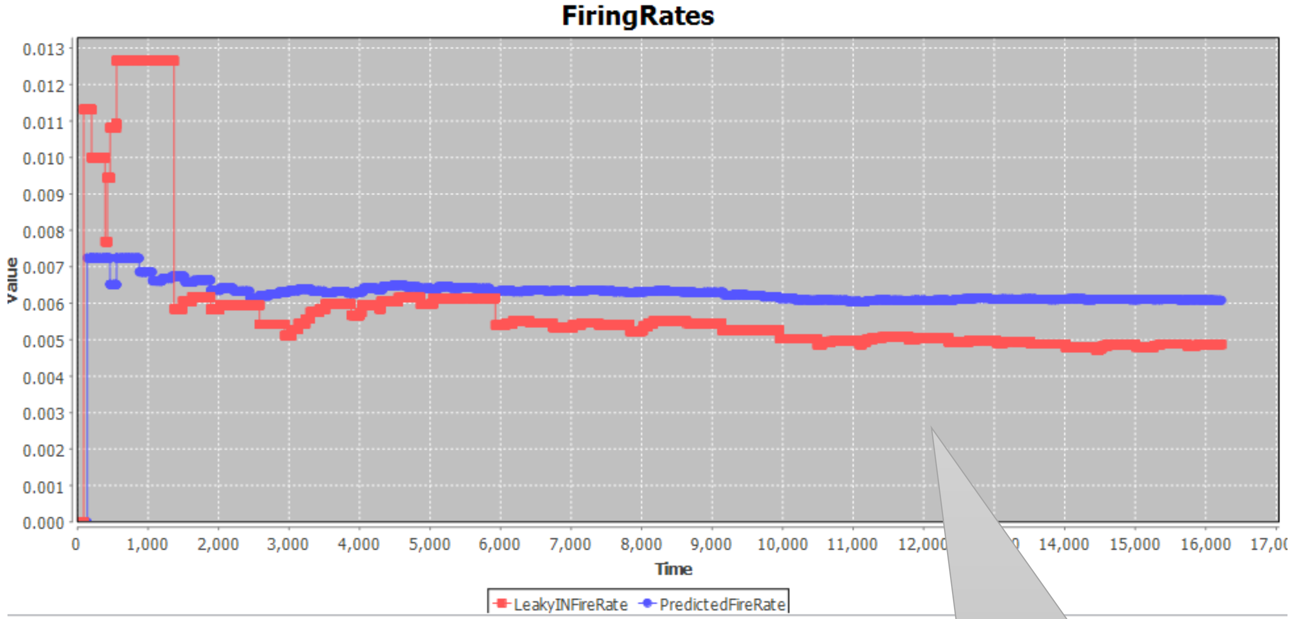
$$EffectiveThreshold = \frac{Threshold}{EffectiveFraction}$$

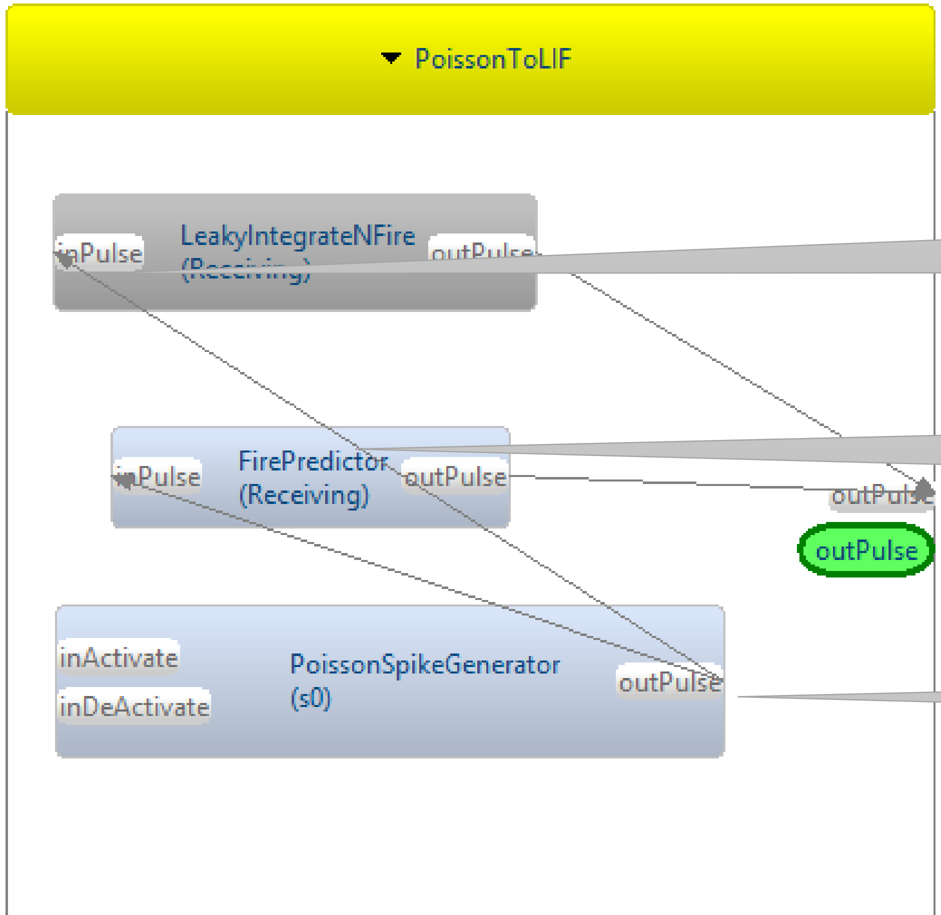
Count the number of pulses in window; if it is bigger than threshold output pulse (take into account that decay reduces the

Predict pulse at end of segment if

$$\# Pulses > EffectiveThreshold$$

Segmentation verification





Leaky Integrate and FIre

Segmentation-based Predictor

Posson Spike Generator

Input pulse rate = .1;
Decay rate = .1;
Output rates differ by .002

Derivation => Proof of Mapping (Behavior Morphism)

Derivation:

$$e^{-CountWindow*DecayRate} = .01$$

$$\Rightarrow CountWindow*DecayRate$$

$$= -\ln(.01)$$

$$EffectiveFraction$$

$$= \frac{AreaUnderDecayCurve}{AreaOfNoDecay} = \frac{\int_0^{CountWindow} e^{-t*DecayRate}}{1*CountWindow}$$

$$= \frac{\left(\frac{1-e^{-CountWindow*DecayRate}}{DecayRate}\right)}{CountWindow} \approx \frac{1}{CountWindow*DecayRate}$$

Fire *if:*

$$\#Pulses * EffectiveFraction > Threshold$$

$$\Leftrightarrow \#Pulses > \frac{Threshold}{EffectiveFraction}$$

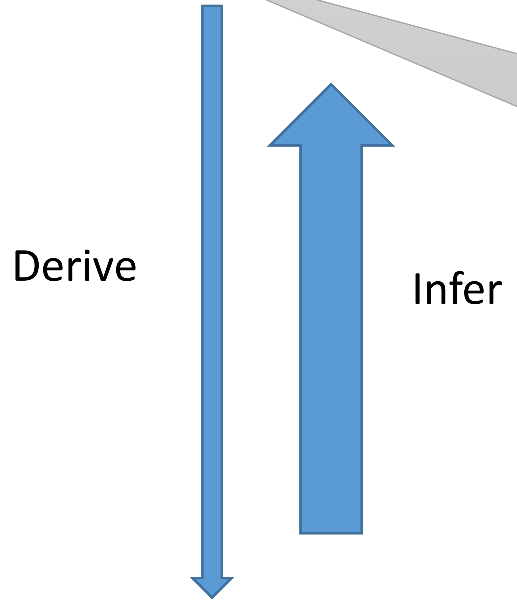
$$\Leftrightarrow \#Pulses > EffectiveThreshold$$

$$EffectiveThreshold = \frac{Threshold}{EffectiveFraction}$$

Propagating Parameter Values

$$Threshold = EffectiveFraction * EffectiveThreshold$$

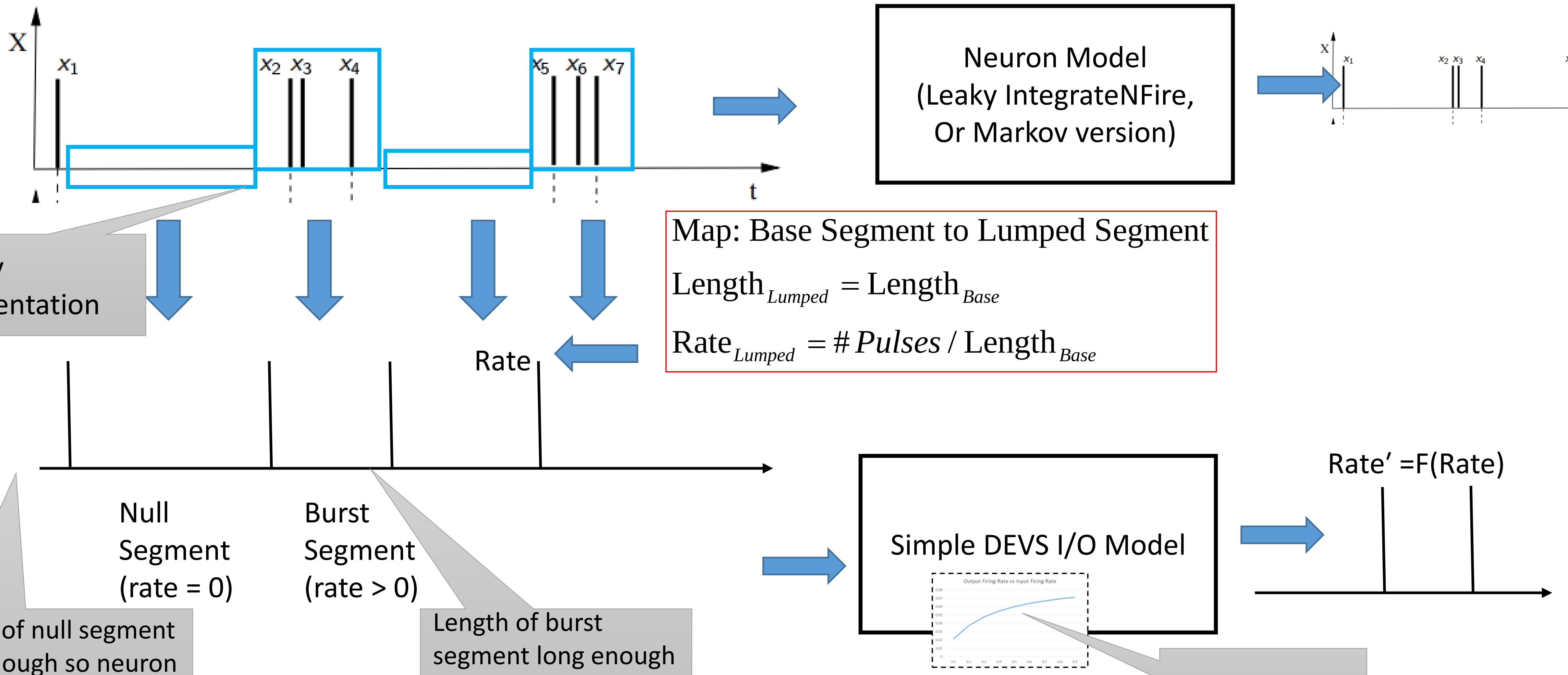
Compute Threshold, knowing Effective Fraction and Effective Threshold



$$EffectiveThreshold$$

Identify this value by fitting Lumped Model to data.

Morphism from Neuron with Bursty input segments to simple I/O DEVS model



Bursty Segmentation

Map: Base Segment to Lumped Segment

$$\text{Length}_{Lumped} = \text{Length}_{Base}$$

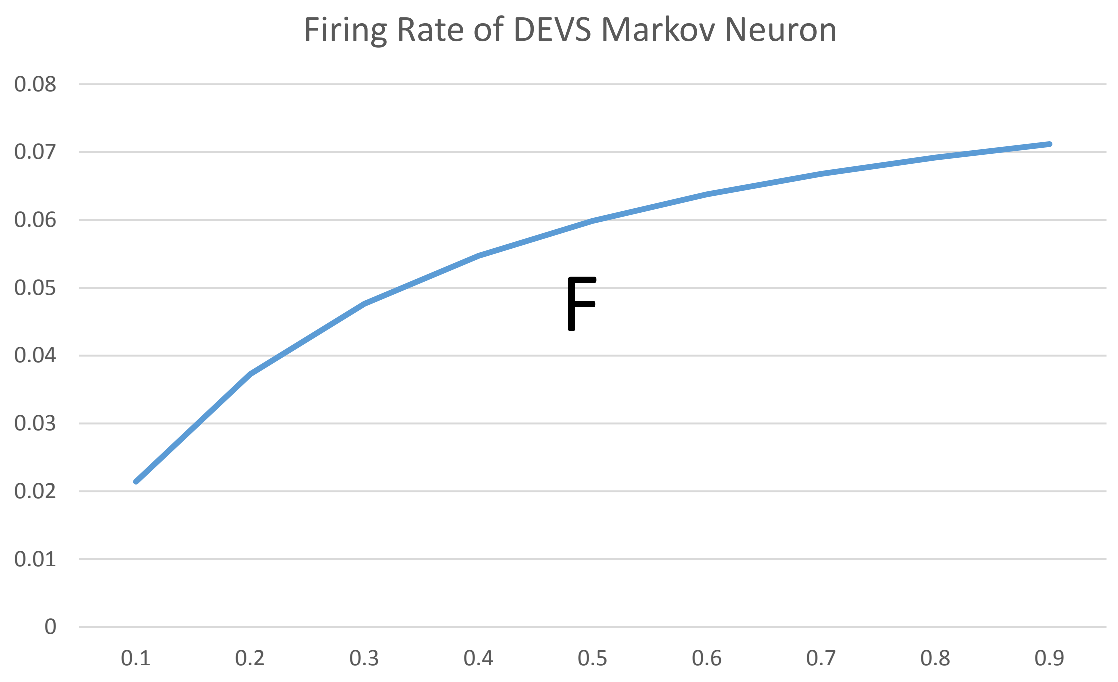
$$\text{Rate}_{Lumped} = \#\,Pulses\,/\,\text{Length}_{Base}$$

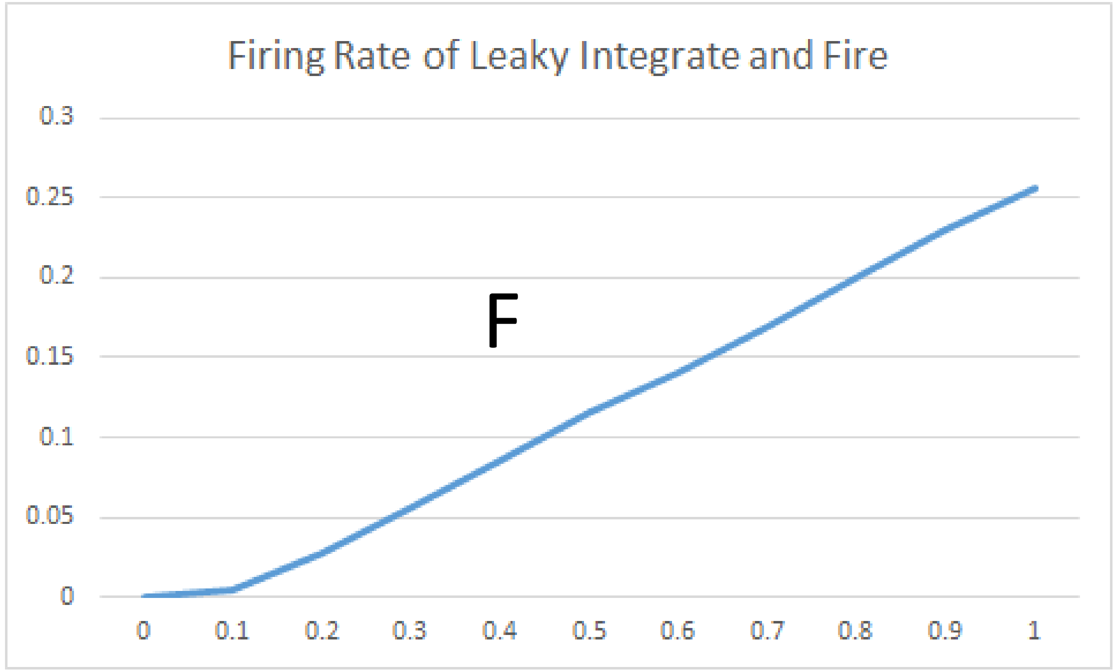
Length of null segment long enough so neuron Goes back to ground state

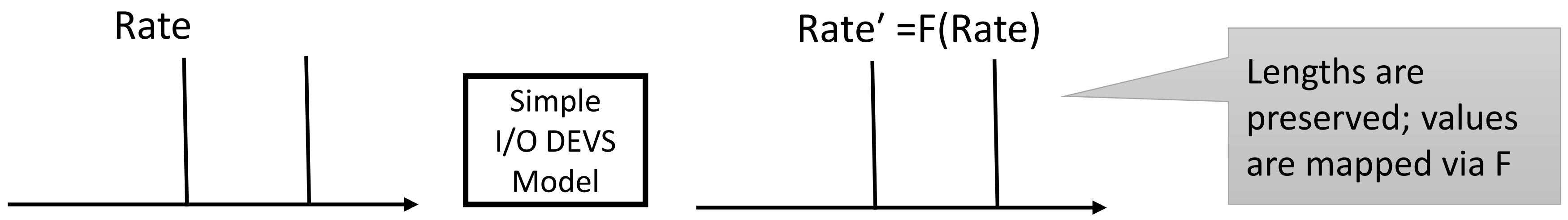
Length of burst segment long enough so neuron Reaches steady state

F: See next slide

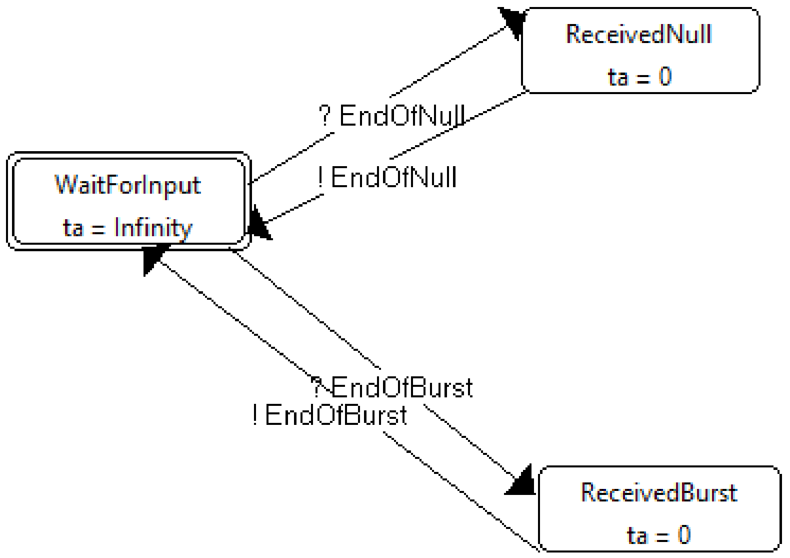
Simple I/O DEVS Model: Output Firing Rate = F(Input Firing Rate)

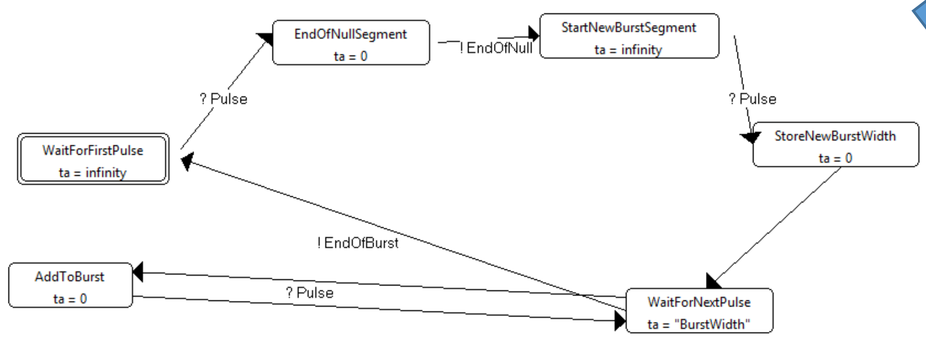


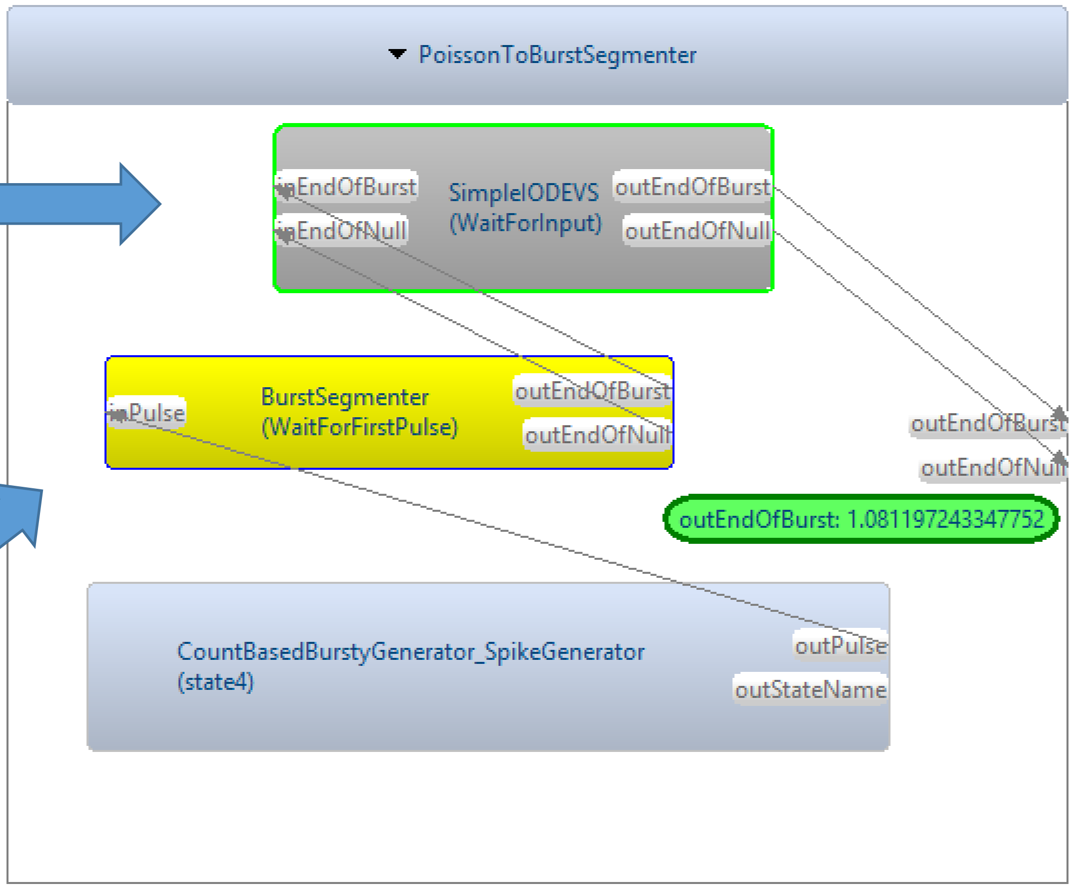




PoissonToBurstSegmenter (to SimpleIODEVS) Test



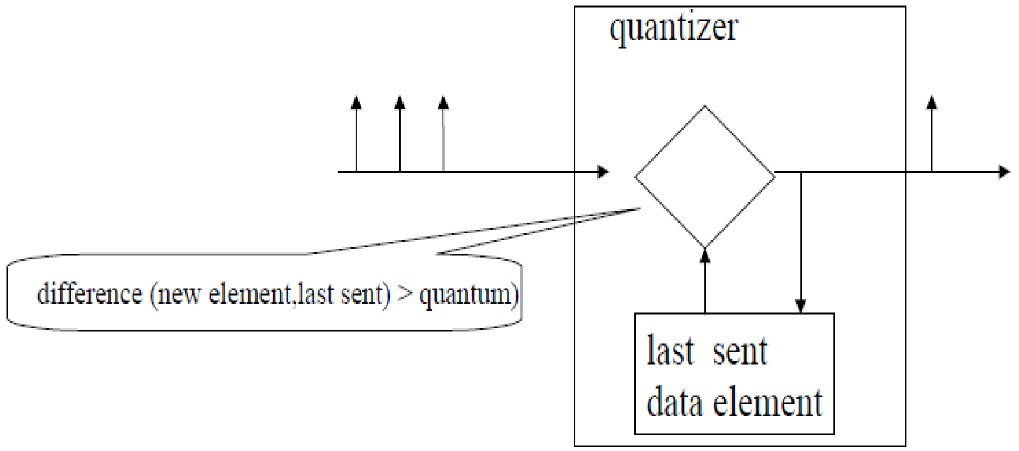




Quantization

Quantization Principle

Send new element only when significantly different from last sent element
•Difference: to measure change from one item to the next
•Quantum: to determine the minimum size of change for significance



For images: enough pixels have to differ in enough RGB color value

The generalized Quantization Principle

- Quantization is a general process for extracting information from a continuous stream of data
- Any differential equation system can be approximated as closely as desired using quantization- alternative to conventional numerical integration
- In distributed simulation, quantization is a basic filtering technique in which continuously changing state variables, such as positions and velocities, of one component are only transmitted to other "subscriber" components over the network, when their changes exceed a threshold level called a quantum
- Such quantum threshold crossings are the events and the intervals between them can be predicted so that the overall behavior can be produced by discrete event (and in particular DEVS) abstractions of the components
- The larger the quantum, the fewer the state updates that are "published," but also the greater the potential deleterious effect of the message reduction on simulation accuracy
- For many behaviors, the tradeoff of fidelity versus message reduction is very favorable – allowing available bandwidth to be utilized much more efficiently with minimal impact on fidelity

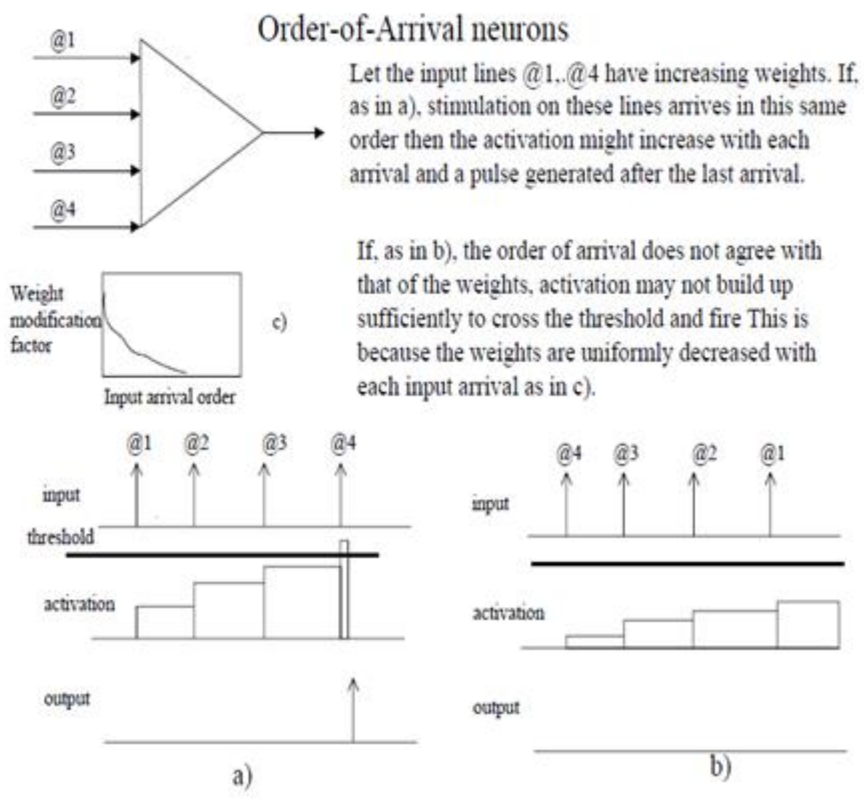
Fast Discrete Event Nervous System Architectures

- Many features of biological neurons are not represented in conventional artificial neural networks
- “One-spike-per-neuron” refers to information transmission from neuron to neuron by single pulses (spikes) rather than pulse trains or firing frequencies
- A face recognition multi-layered neural architecture based on the one-spike, discrete event principles has been demonstrated to
    - conform to the known time response constraints of human processing and
    - To execute computationally much faster than a comparable conventional artificial neural net
- The distinguishing feature of the one-spike neural architecture is that it relies on a temporal, rather than firing rate, code for propagating information through neural processing layers
- This means that an interneuron fires as soon as it has accumulated sufficient “evidence” and therefore the latency to the first spike codes the strength of this input
- Single spike information pulses are thus able to traverse a multi-layered hierarchy asynchronously and as fast as the evidential support allows
- Thorpe has shown that “act-as-soon-as-evidence-permits” behavior can be implemented by “order-of-arrival” neurons
    - have plausible real world implementations?
    - coding which exploits firing order is much more efficient than a firing-rate code which is based on neuron counts
    - is invariant with respect to overal input intensity level because latencies are uniformly affected by such changes

Strength-to-Latency Coding

- The basic concept that supports discrete event abstraction of neural behavior is strength-to-latency coding
- The strength of the input of an evidence gathering neuron (such as sensory neuron) is coded in the latency of its output response for downstream neurons
- The greater the stimulation of an input volley (evidence) the quicker the generation of a corresponding output spike
- Thus a neuron with lots of evidentiary support will be “heard” earlier by neurons in the next processing layer than one with low or no input strength

Order-of-Arrival Neurons



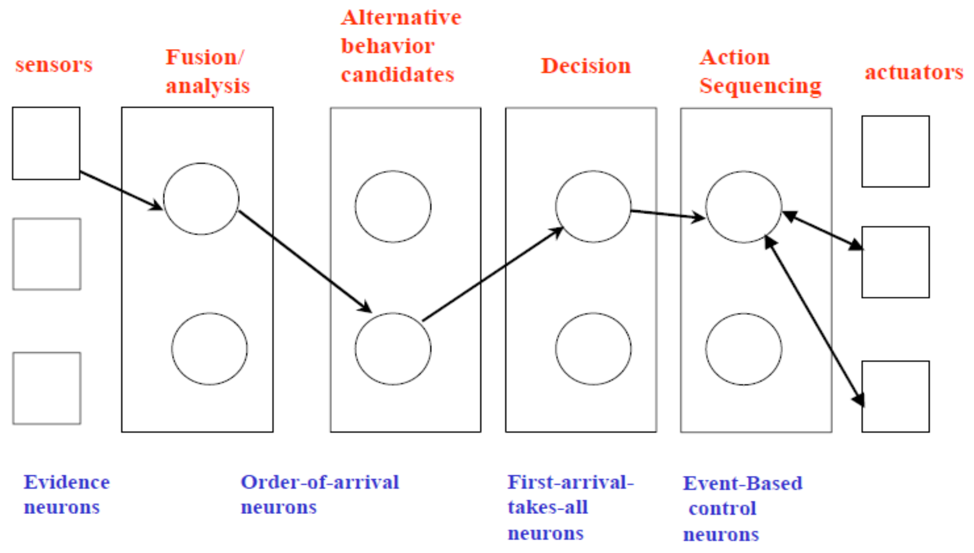
Order-of-Arrival neurons

Let the input lines @1..@4 have increasing weights. If, as in a), stimulation on these lines arrives in this same order then the activation might increase with each arrival and a pulse generated after the last arrival.

If, as in b), the order of arrival does not agree with that of the weights, activation may not build up sufficiently to cross the threshold and fire This is because the weights are uniformly decreased with each input arrival as in c).

- Dispersion in latencies sets the stage for neurons that are sensitive to the order of arrival of spikes
- An input train arrives on the input lines in the order of their weights accumulates maximum activation and may cause the neuron to fire if this exceeds the threshold
- Any other order of arrival will accumulate less activation and therefore, depending on the threshold level, may not generate an output spike
- Thus the neuron can discriminate among different order-of-arrivals of stimuli
- This ability to distinguish between N! Input patterns (where N is the number of input wires) thus supports a combinatorially more efficient information code than one based on the number of stimulated input wires rather than their order of stimulation

End-to-End Processing Layers



5 An "end-to-end" layered architecture to establish neuron behavior requirements

The one-spike concept provides the basic building block in an "end-to-end" processing system for small, fast reactive "nervous systems"

Have formulated the discrete event abstractions underlying the one-spike-per-neuron concept, and expressed them in DEVS

This fits the definition of discrete event abstraction:

- **events are threshold crossings which generate discrete spikes**
- **inter-event temporal separations include the latencies between input and output spikes**

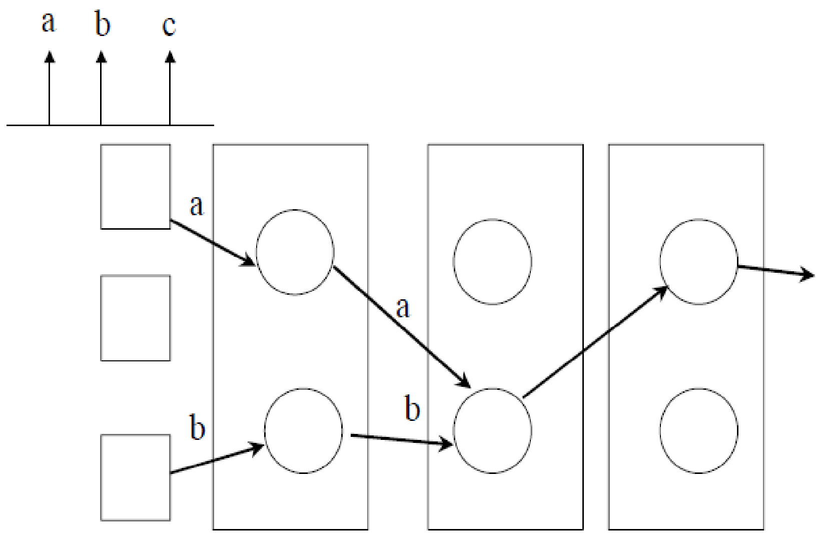
Illustrate a fast processing layered architecture,

- **including all its sensory, cognitive, actuator and communication related components,**
- **within real time processing, memory and energy constraints**
- **the kinds of neurons that are found in each layer**

Sensory layer neurons react directly to incoming energy (in various forms such as visual or infrared electromagnetic waves, sonar, etc.) These neurons perform the strength-to-latency coding.

- **Fusion/Analysis neurons fuse the data collected from the various sensors into some stereotyped situations that can be further related to reactive courses of action: operate on the order of-arrival principles above.**
- **Priming of alternative candidates for behavioral course of action is also done by order-of-arrival neurons.**
- **Decision, i.e., selection from the candidates, is performed the by winner-take-all neurons.**
- **Action sequencing plays out the memorized sequence of actions associated with a selected course of action and is done by event-based control neurons.**

Synchronizing Between Layers

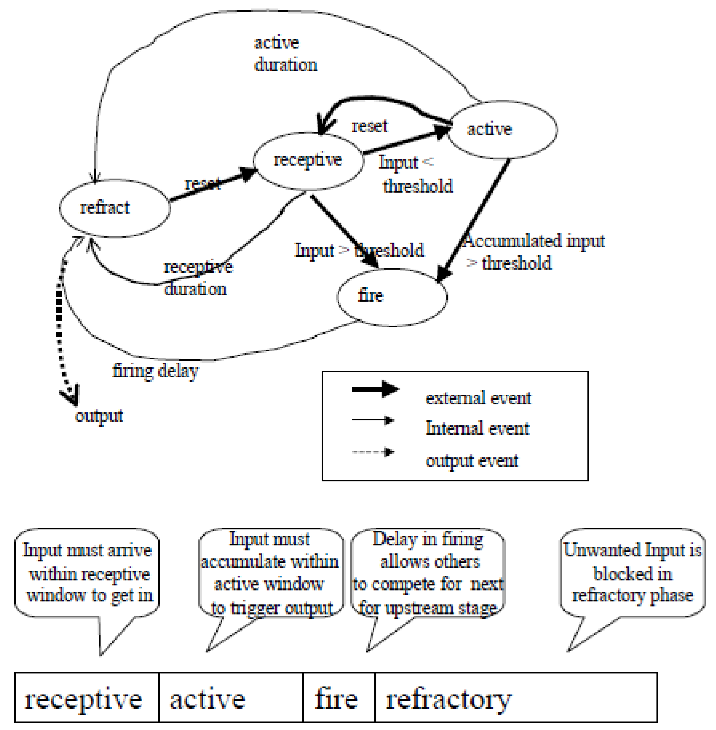


In temporal delay coding, laggard spikes from earlier frames may interfere with spikes from later frames, e.g **a** is still around when **b** is input

Laggard pulses in stength-to-latency information transmission

- Using strength-to-latency coding, operation is asynchronous.
- There is no global clock tick to synchronize states of neurons at a recurring time step.
- Laggard spikes from earlier percepts may interfere with spikes stimulated by later percepts.
- One solution is to place a time-out on the response of a neuron -- it is reset to its initial state after a given duration

DEVS Generic Neuron Model


active duration
reset
active
receptive
Input < threshold
reset
refract
Input > threshold
Accumulated input > threshold
receptive duration
fire
firing delay
output
external event
Internal event
output event
Input must arrive within receptive window to get in
Input must accumulate within active window to trigger output
Delay in firing allows others to compete for next for upstream stage
Unwanted Input is blocked in refractory phase
receptive
active
fire
refractory

DEVS model of neurons satisfying the behavioral requirements

- We develop requirements for basic behavioral properties of DEVS Neuron Models.
  - Inspired by the biological origins of discrete event neural abstractions,
  - Are logically required in implementing the architecture.
- DEVS neurons
  - Have the ability to respond to order of arrival of external events on their input ports
  - Controllable by passage of time, such as time windows and time outs
  - Delay firing to enable competition in sending output to next stage
  - Synchronizable through an external reset event.
  - The model has four main phases (control states): *receptive, refract, active* and *fire*, each with an associated time duration.
  - The actual durations are parameters of the model which range from 0 to infinity.
  - Their assignments produce different specialized behaviors that are derived from the generic model.

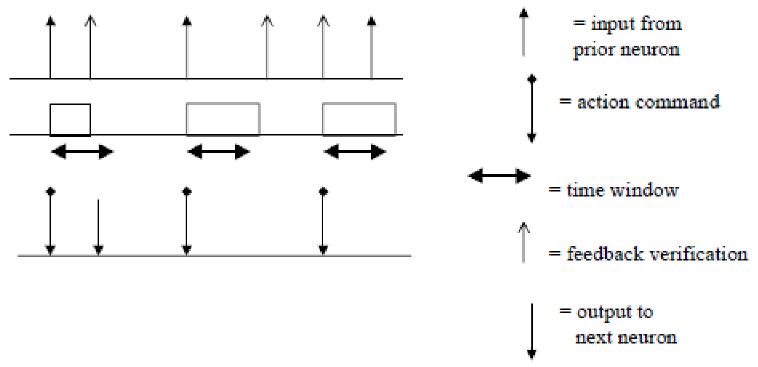
DEVS Generic Neuron Model Operation

- The model starts in the receptive phase.
- If an input arrives during the receptive period that is less than the threshold, then the neuron enters the active phase, where it waits for further inputs.
- If accumulated input exceeds the threshold before the active period has expired, then fire phase is entered.
- Also, if an above-threshold input arrives during receptive period, fire phase is entered directly.
- After a delay, an event (representing a pulse or spike) is produced on the output port.
- After firing, the model enters the refractory phase, where it is unresponsive to inputs.
- The active phase also times out to the refractory phase (if above threshold input is not accumulated).
- The reset input, occurring during the refractory period, puts the model back to the receptive phase.

DEVS Generic Neuron Model Behaviors

- The generic model can be specialized to realize the behaviors of the following:
- ***Evidence Neurons*** (at the sensory layer) – Physiologically, these have been identified as "integrate and-fire" neurons and represented as leaky integrators with threshold-based firing.
- With constant inputs arriving periodically, an output will be generated once the threshold has been reached.
- The output period is inversely related to the strength of the input thus implementing the analog-to-delay coding discussed earlier.
- ***Order-or-Arrival Neurons*** – these are implemented with appropriate weight settings as discussed earlier.
- ***Winner-Takes-All (First-Arrival-Takes-All) Neurons*** – these neurons implement winner-take-all behavior based on first arrival. Metaphorically, the neuron with the first spike to arrive from the previous processing stage, closes the door for pass through of later arrivals.
    - This approach works much faster than conventional winner-take-all circuits. Using the generic DEVS neuron, the lockout behavior can be accomplished by establishing mutual cross-inhibition (negative weights for inputs from competing predecessors).

Event-based Control Neurons



If a neuron receives input from predecessor it emits action command and waits for feedback verification If feedback from actuator is within the allowed time window, the action sequence continues to the successor.

DEVS neuron implementation of event based control for output effectors

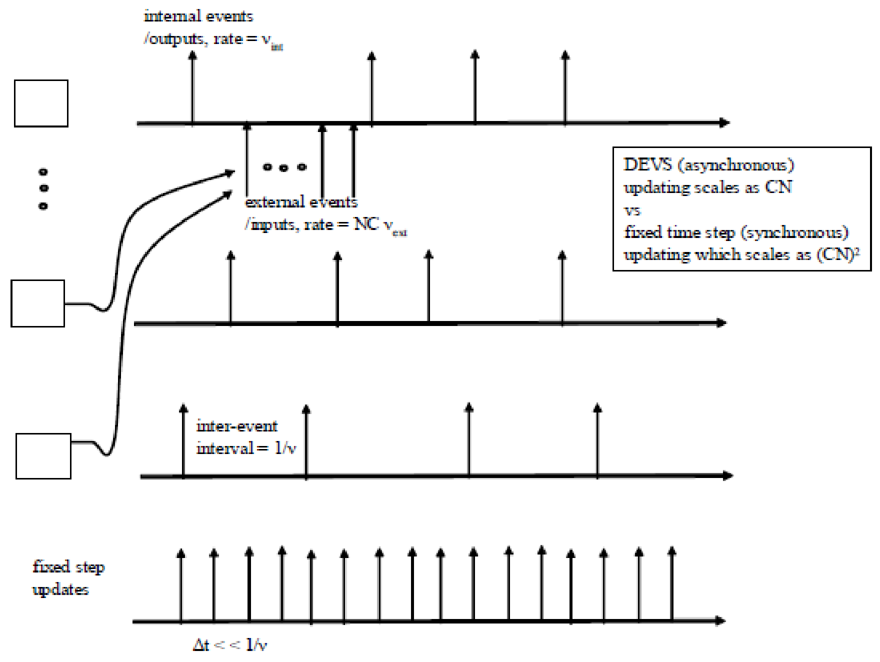
Neurons are connected in a series to control a sequence of discrete actuations forming a composite action.

- In event-based control, verification feedback from the current actuation (as in proprioceptive feedback) is required to fall within a time window before the next action can be activated.
- The realization by the generic DEVS neurons employs the time-out associated with the active phase.

DEVS Neurons: Time, Space, and Energy Constraints

- Space, time and resource constraints apply to real world information processing by neuron systems.
- Use of numbers of neurons (space), order-of-arrival coding is much more efficient than is rate-based coding.
- The "act-as-soon-as-evidence-permits" principle complies with the demands of quick reaction under time pressure.
- Further the time-outs associated with phases can be linked to limitations on the energy consumption necessary to maintain active states.
- Likewise the refractory phase can be associated with minimal energy consumption (or restoration of energy in the biological case).
- The underlying DEVS abstraction, concentrating on events and their timing, is efficient in both processing and communication.

Efficiency of Discrete Event Simulation



DEVS Network to illustrate efficiency of discrete event simulation

- An event-driven approach was taken for large-scale simulations of recurrent neural networks of spiking neurons and plastic synapses.
- The approach exploits the fact that the dynamic variables of both neurons and synapses evolve deterministically between any two successive spikes -- thus tracking the arrival of spikes enables unequivocal prediction of neural dynamics.
- Compared with conventional synchronous simulation. The result is a drastic reduction of the computational load per neuron which grows linearly with the number of neurons is only about 6% more with variable synapses.
- Place simulation complexity comparison in the context of the more fundamental discrete event conceptual framework.
- Discrete event abstraction can retain essential state and timing information while abstracting away details in underlying continuous trajectories.
- Let there be N components, each sending outputs to an average of C receivers.
- Such outputs, generated at internal state transitions, are assumed to be described by a Poisson process with rate, ν_{int}.
- Under random connectivity conditions, every component receives inputs at the rate, ν_{ext}.= CN* ν_{in}.
- A DEVS simulator only computes updates at internal and external events.
- At each component these occur at the combined rate ν = , ν_{in} * ν_{ext}. = (1+CN) , ν_{in}, a linear dependence on N (again assuming validity of the Poisson assumption).
-

Computation with Pulsed Neural Networks: Shortest Path Solvers

- Computational capabilities of hardware-based "pulsed" neural networks
- Does not take the full step toward discrete event abstraction
- A very simplified versions of the generic DEVS neurons provides shortest path solutions to directed graphs.
- The shortest path in such a graph, whose arcs represent time durations, can be regarded as an abstraction of the "act-as-as-soon-as-evidence-permits" processing in multilayer nets.
- In contrast, finding long paths cannot be done with the DEVS neuron.
- This suggests that an evolutionary explanation for the structure and behavior of biological neurons if indeed they operate on discrete event principles -- they are optimized for operation in a world requiring synthesis of quick multi-step reactions as dictated by shortest paths in graphs of dependent actions

Probabilistic Gene Network

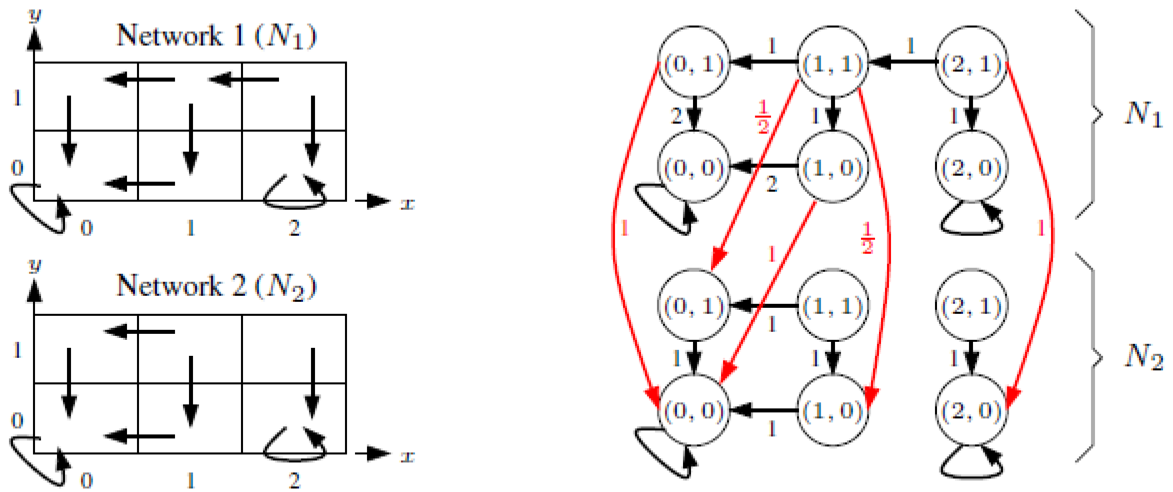
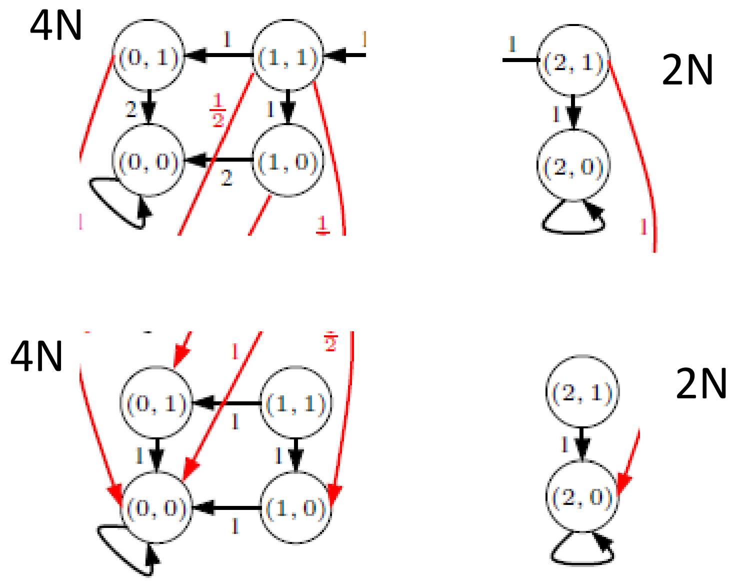


Figure 4: Two asynchronous state graphs (left) associated with interaction graph of Figure 2(a) on which we illustrate the construction of the probabilistic gene network (right). The numbers labelling the transitions (right) correspond to the numerator of Equation 2.


4N
2N
4N
2N
(0, 1)
(1, 1)
(0, 0)
(1, 0)
(2, 1)
(2, 0)
$\frac{1}{2}$