NEURAL SYSTEM SIMULATION – IMPLEMENTATION PLAN

AGENT SYSTEMS

GENERAL IDEA

- Representation of a **group of interconnected neurons**
- **Neurons** modeled as **agents**
- **Connections** between neurons are formed **dynamically**
- Additional agent types include **Sensors** and **ReceptoryFields**
- Technology: **MASON**

REMAINDER: MASON UNDERSTANDING OF AGENTS

In MASON:

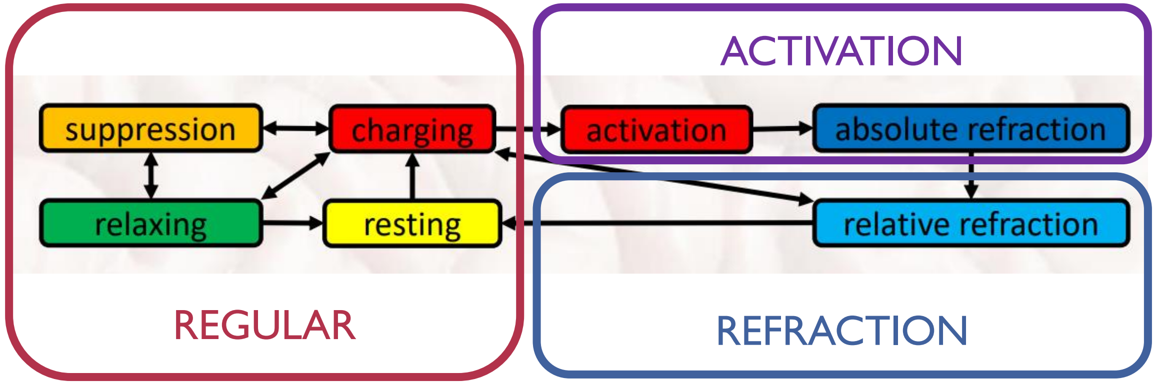
- An agent is an object that contains the **step(SimState)** method and is (can be) **scheduled** for execution
- MASON holds a **global schedule** of upcoming agents' activations
- At each step MASON fires the agent that's sheduled to execute at the **earliest** time (multiple agents if ties occure)
- Upon execution, agent's **step(SimState)** method is called; an agent has access to global simulation state and consequently to all other agents

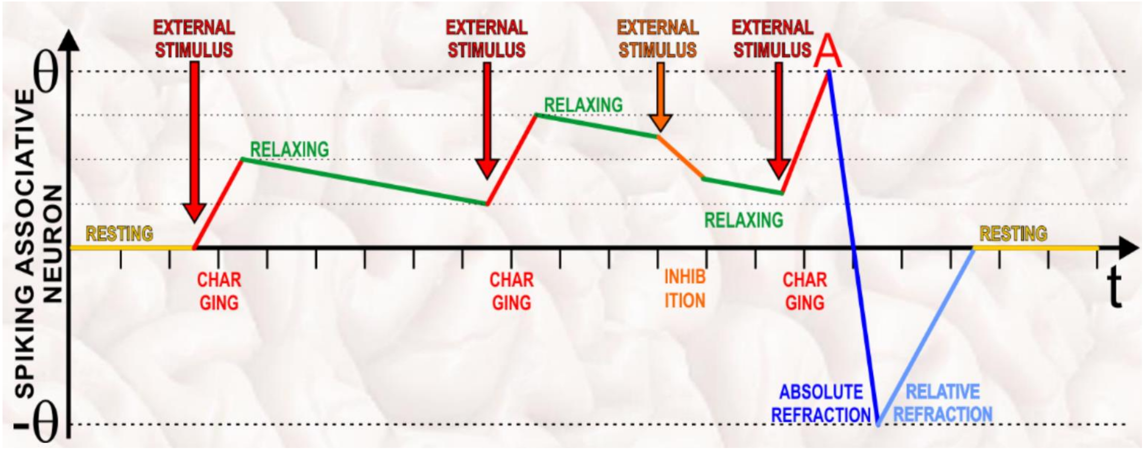
MAIN AGENT TYPE

ASSOCIATIVE SPIKING NEURON

ASSOCIATIVE SPIKING NEURON

- Neuron maintains internal **excitation** level, influenced by external stimulation and passage of time; it can be described as a state machine (proposed model involves a simplified verison of states space, with states grouped into three distinct groups, marked on figure below).

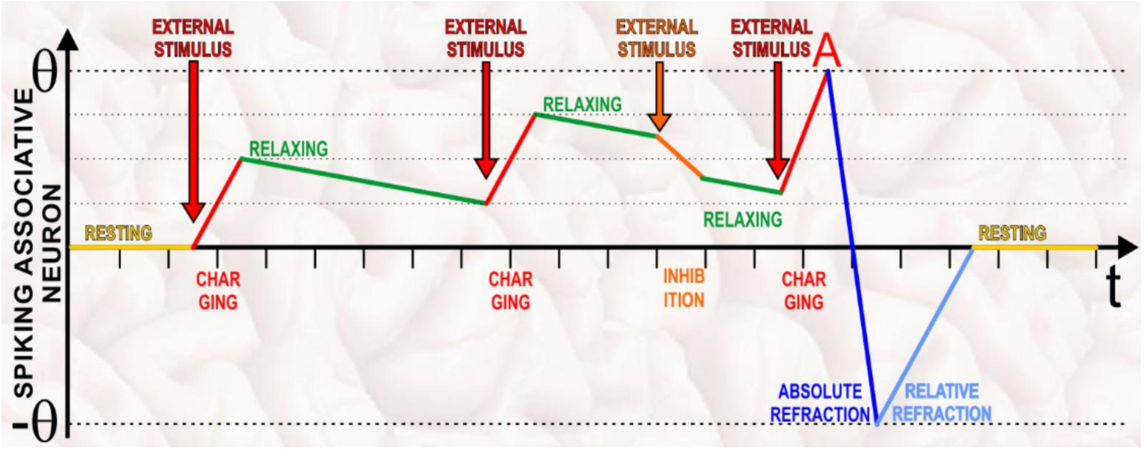


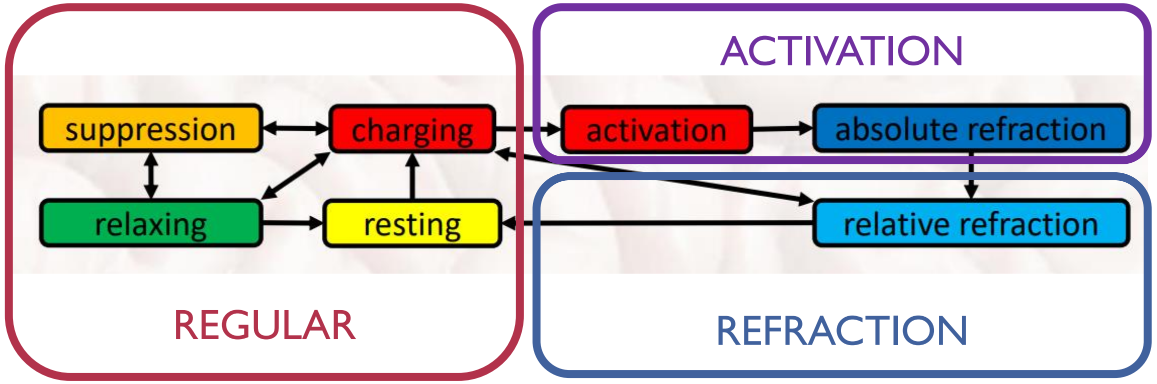


Images source: https://home.agh.edu.pl/~horzyk/lectures/ci/CI-AssociativeNeuralGraphs.pdf, slide 73

ASSOCIATIVE SPIKING NEURON

- Neuron **starts** simulation in **REGULAR** state with no stimuli on inputs – it's predicted activation time equals infinity
- An **external stimuli** (from other agents – neurons and sensors – or simulation scenario) may cause neuron to **expect activation**
- When neuron's input state changes – one of it's neighbours is either activated or it's activation ends – several things happen:
  - Neuron updates it's **excitation** level based on past input state and time passed
  - Neuron's predicted activation time is **recomputed** (with the assumption that the current input state will be constant). If it turns out to be **finite**, neuron is **scheduled** for **activation** after computed time.
  - If neuron was already scheduled for activation (based on past, outdated input state) this old event is **descheduled**.

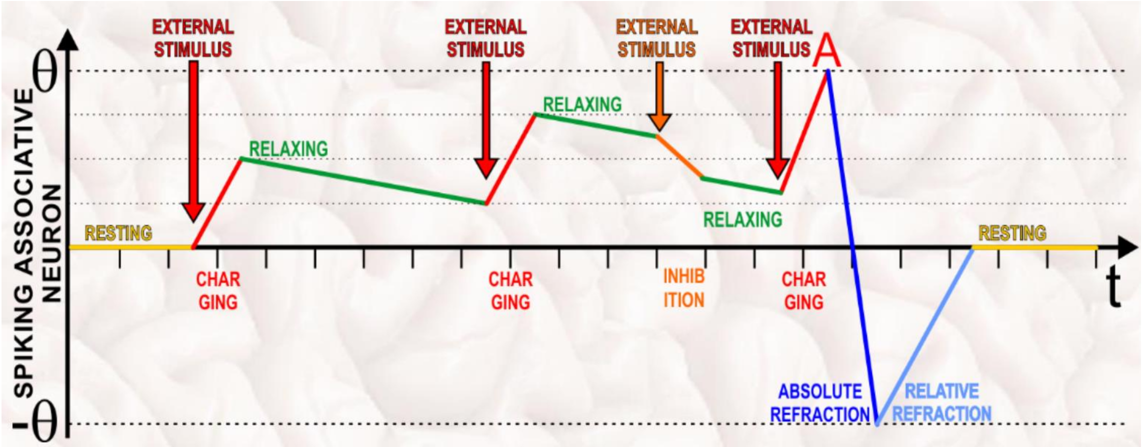


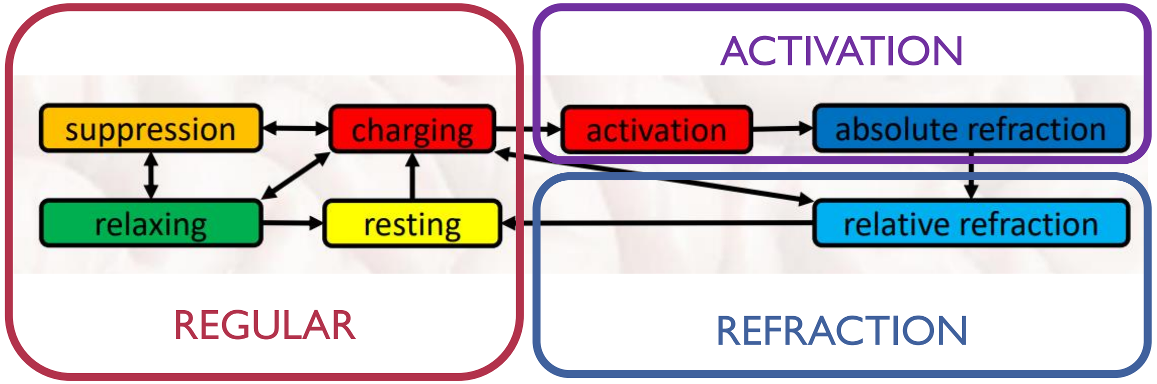


Images source: https://home.agh.edu.pl/~horzyk/lectures/ci/CI-AssociativeNeuralGraphs.pdf, slide 73

ASSOCIATIVE SPIKING NEURON

- After neuron reaches activation, it **stimulates** all it's **neighbours** and **schedule** an **end of activation** event (it enters the **ACTIVATION** state)
- When **end of activation** event occures, neuron notifies all it's neighbours (they are forced to recompute their estimated activation time) and schedules itself for **end of refraction** (it enters the **REFRACTION** state)
- When **end of refraction** event occures, neuron resets it's internal **excitation** level and enters the **REGULAR** state
- In **ACTIVATION** and **REFRACTION** states neuron mostly **ignores stimuli changes** – it does update it's input state, but doesn't update excitement nor schedule new activations





Images source: https://home.agh.edu.pl/~horzyk/lectures/ci/CI-AssociativeNeuralGraphs.pdf, slide 73

ASSOCIATIVE SPIKING NEURON - PSEUDOCODE

```
enum NeuronState:
        REGULAR,      # encompasses RESTING, CHARGING, RELAXING and SUPPRESSION
        ACTIVATED,
        REFRACTED


base agent Neuron:
        neighbours = []        # neurons that are OUTPUTS of current neuron (that react to it's activation)
        current_input_excitations = {}        # { neighbour: (weight, start_time) }
        state = REGULAR
        excitation = 0.0
        last_update_time = None
        nearest_activation = None

        … # Neuron's methods are presented on the following slides
```

ASSOCIATIVE SPIKING NEURON - PSEUDOCODE

```
# base agent Neuron:
        step():
                # step() is called in three cases: when neuron is activated, when it activation ends (refraction starts) and when refraction period ends

                if state == REGULAR:        # neuron was in regular state and has just been activated
                        for neigh in neighbours:
                                neigh.start_input_excitation(self)
                        state = ACTIVATED
                        self.schedule(ACTIVATION_TIME)          # activation time is constant, it may be one of experiment parameters

                elif state == ACTIVATED:  # activation has just ended and neuron should start refraction
                        for neigh in neighbours:
                                neigh.stop_input_excitation(self)
                        state = REFRACTED
                        self.schedule(REFRACTION_TIME)          # refraction time is constant, it may be one of experiment parameters

                else:   # state == REFRACTED; refraction has ended and neuron has just returned to regular state
                        excitation = 0.0
                        for input in current_input_excitations:
                                weight, _ = current_input_excitations[input]
                                current_input_excitations[input] = (weight, curr_time())
                        state = REGULAR
                        reschedule_activation()
```

ASSOCIATIVE SPIKING NEURON - PSEUDOCODE

```
# base agent Neuron:

    start_input_excitation(neighbour):
        # called by neighbour neuron when it's activation starts

        current_input_excitations.put(neighbour, (WEIGHTS[neigh, self], curr_time())
            # WEIGHTS are read from global simulation state (it's a property of connection between neurons)

        if state == REGULAR:
            excitation = new_excitation()
            reschedule_activation()


    stop_input_excitation(neighbour):
        # called by neigbour neuron when it's activation ends

        current_input_excitations.remove(neighbour)
        if state == REGULAR:
            excitation = new_excitation()
            reschedule_activation()
```

ASSOCIATIVE SPIKING NEURON - PSEUDOCODE

```
# base agent Neuron:

    new_excitation():
        # returns current excitation of neuron based on last known excitation value, current inputs and time passed
        # inputs must not change between subsequent calls to new_excitation()
        # function sets last_update_time to curr_time each time it is called
        # the exact formula for return value is the following:
```

$$excitation_{curr} = \begin{cases} excitation_{last} + \sum w_i * (t_{curr} - t_{last})\,, & curr_{inputs} \neq \emptyset \\ excitation_{last} - signum(excitation_{last}) * relaxation(excitation_{last}, t_{curr} - t_{last}), & w.p.p \end{cases}$$

```
    compute_activation_time():
        # returns time at which activation would occure if all the inputs were steady
        # the exact formula for return value is the following:
```

$$t_{rel} = \frac{threshold - excitation_{curr}}{\sum w_i}$$

```
    reschedule_activation():
        if nearest_activation is not None:
            nearest_activation.deschedule()

        activation_time = compute_activation_time()
        self.schedule(activation_time)
```

NEURONS ORGANIZATION

ACTIVE ASSOCIATIVE NEURAL GRAPH

ACTIVE ASSOCIATIVE NEURAL GRAPH

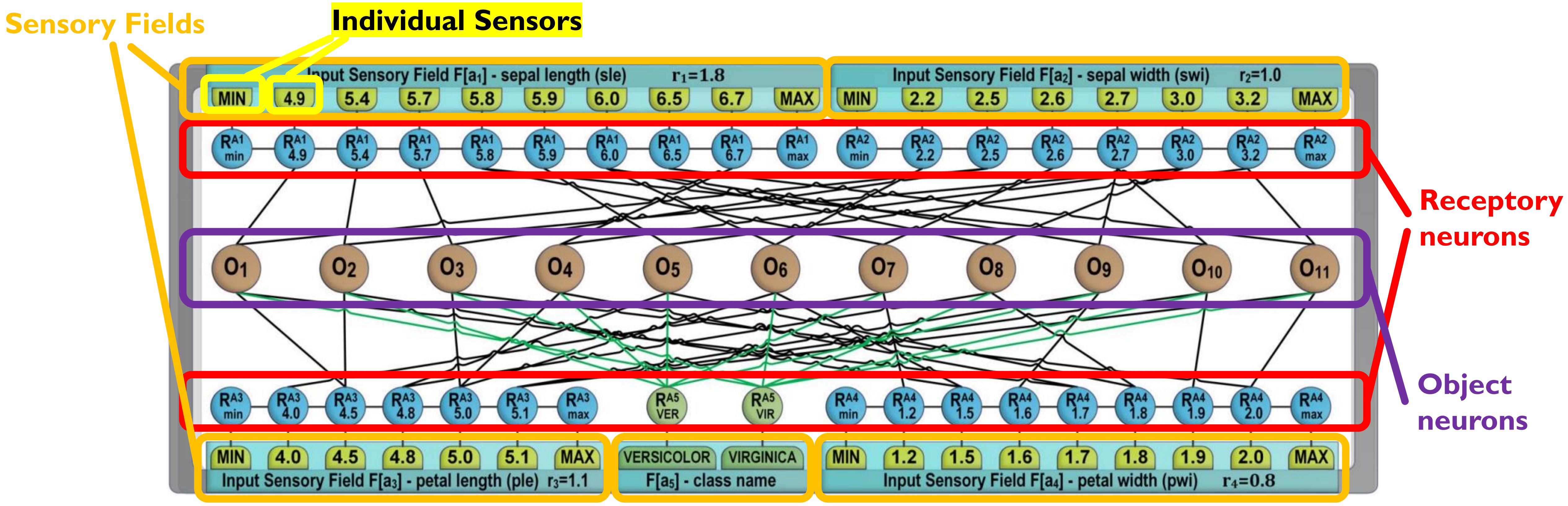


Image source: https://home.agh.edu.pl/~horzyk/lectures/ci/CI-AssociativeNeuralGraphs.pdf, slide 42

SPECIFIC NEURON TYPES

AGENTS DERIVED FROM BASE NEURON AGENT

RECEPTORY NEURON

- Connects to exactly one **Sensor** (in), two neighbouring **Receptory Neurons** (in/out) and one or more **Object Neuron** (in/out)
- Adheres to the **Plasticity Rule** (more details can be found further into the presentation)
- Weights of connections to other **Receptory Neurons** are computed after the formula:

$$w_{ij} = 1 - \frac{|v_i - v_j|}{r}$$

- Weights on connections to **Object Neurons** are computed after the following formula:

$$w_i^a = \frac{count(rn_i)}{count(a)}$$

OBJECT NEURON

- Connects to exactly n_{attr} **Receptory Neurons** (in/out)
- Weights of connections to **Receptory Neurons** are constantly equal to 1.0

OTHER AGENTS

AGENTS **NOT** DERIVED FROM BASE NEURON AGENT

RECEPTORY FIELD

- Represents a single receptor/sense
- Modelled as agent for convenience – it’s scheduled activations form the Simulation Scenario
- It’s job is to stimulate all it’s associated **Sensors** at given times with given values

SENSOR

- Represents specific value/range received from **Receptory Field**
- **Sensor's** excitement level when stimulated by sensory field is determined by difference between value represented by **Sensor** and value „observed" by **Receptory Field**:

$$x_{s_i} = |v_{rf} - v_i|$$

- Each **Sensor** is connected to a single **Receptory Neuron**
- Every sensor stimulates it's **Receptory Neuron** **immediately** (it does not have an activation threshold)

DYNAMIC STRUCTURE CHANGES

CREATING OF NEURONS AND CONNECTIONS

RULES

- **New patterns:** every time the network is presented with an unknown pattern (on it's **Receptory Fields**) that does not activate any **Object Neuron** within given time (simulation parameter), a new **Object Neuron** is created, connected to the first activated **Receptory Neuron** associated with each **Receptory Field**.
- **Plasticity Rule:** every time a **Receptory Field** „observes" value that is not represented by any existing **Sensor**, a new **Sensor** and **Receptory Neuron** are created to represent it. Then, when an existing **Receptory Neuron** detects that it is stimulated stronger by it's own **Sensor** than by it's neighbouring **Receptory Neuron**, it disconnects with it and connects with the newly created **Receptory Neuron**.