Introduction to Neuromorphic Computing
Insights and Challenges

Todd Hylton

Brain Corporation

hylton@braincorporation.com

Outline

- What is a neuromorphic computer?
- Why is neuromorphic computing confusing?
- What about building a brain?
  - Todd's Top 10 list of challenges
- Alternative ways of thinking about a building a brain
- Closing thoughts

What is a Neuromorphic Computer?

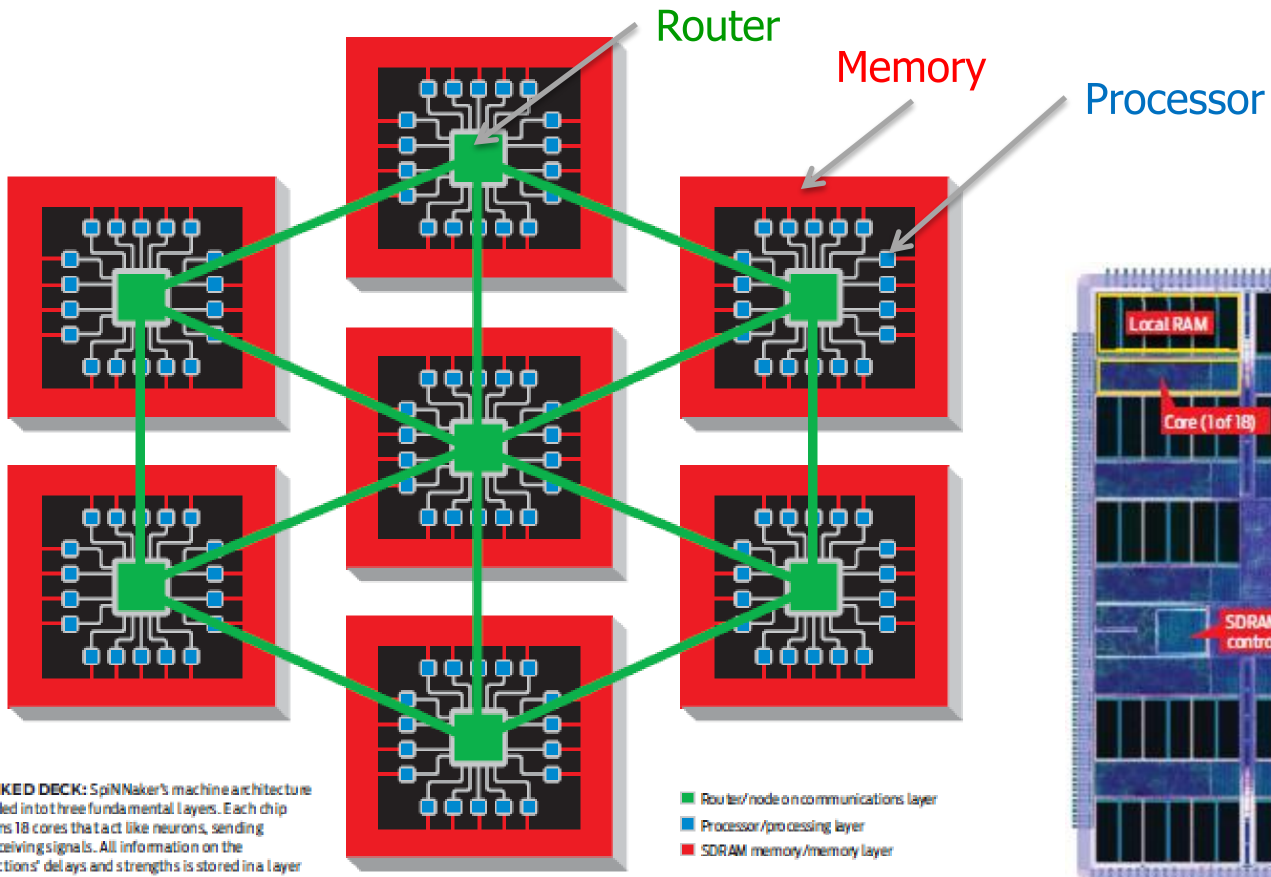
What is a Neuromorphic Computer?

- A neuromorphic computer is a machine comprising many simple processors / memory structures (e.g. neurons and synapses) communicating using simple messages (e.g. spikes).
- Neuromorphic algorithms emphasize the *temporal* interaction among the processing and the memory.
  - Every message has a time stamp (explicit or implicit)
  - Computation is often largely event-driven
- Neuromorphic computing systems excel at computing complex dynamics using a small set of computational primitives (neurons, synapses, spikes).

I think of neuromorphic computers as a kind of "dynamical" computer in which the algorithms create complex spatio-temporal dynamics on the computing hardware

Neuromorphic Computing Hardware Architecture

SpiNNaker (“Spiking Neural Network Architecture”)

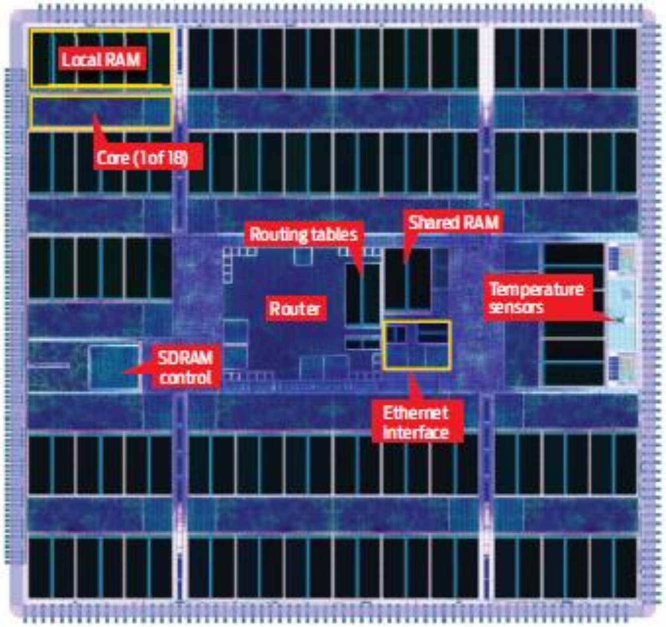


STACKED DECK: SpiNNaker's machine architecture is divided into three fundamental layers. Each chip contains 18 cores that act like neurons, sending and receiving signals. All information on the connections' delays and strengths is stored in a layer of synchronous dynamic RAM (SDRAM) on each chip, and all signals pass through a separate router layer.

🟩 Router/node on communications layer

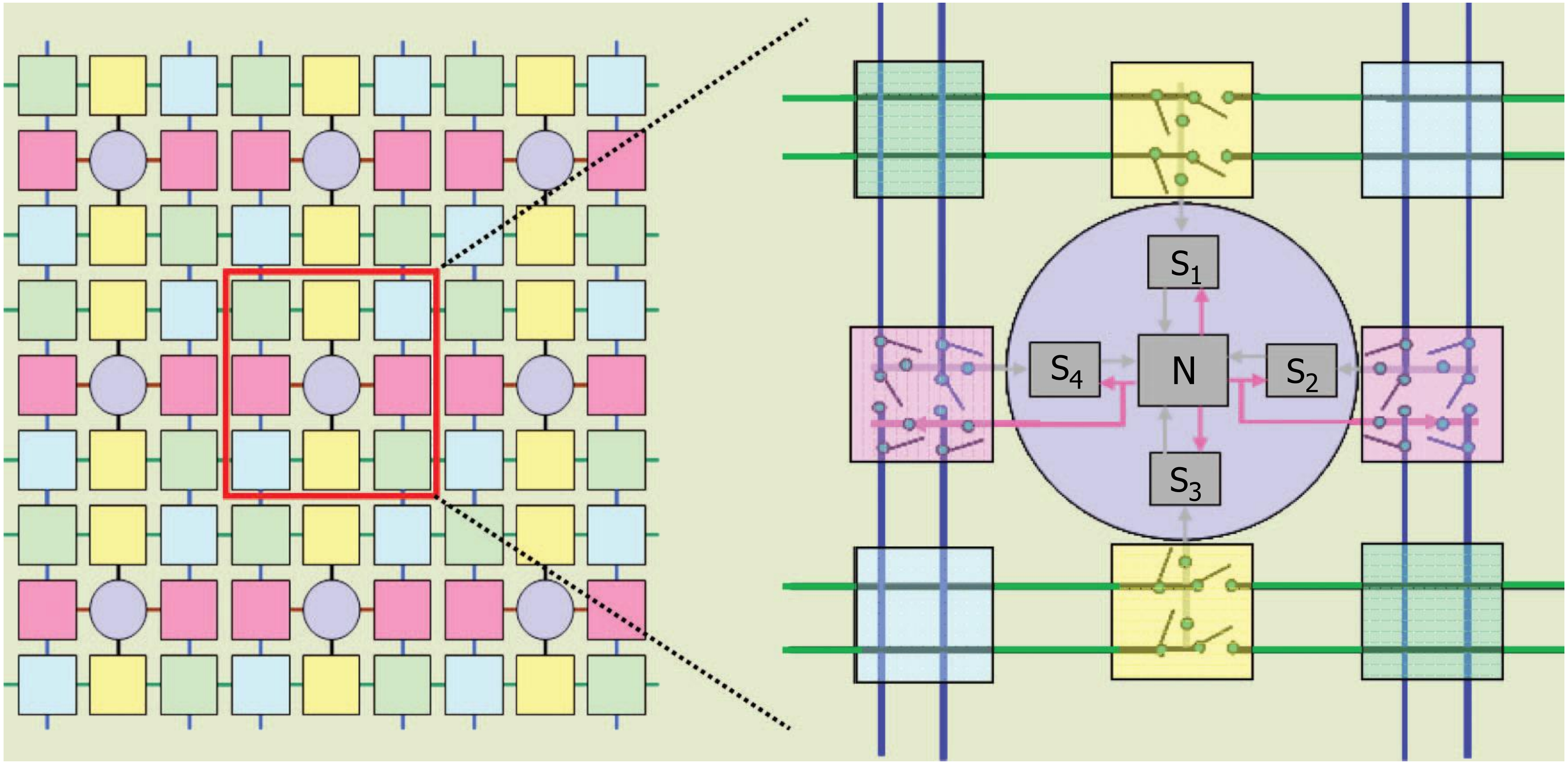
🟦 Processor/processing layer

🟥 SDRAM memory/memory layer



Steve Furber, “To Build a Brain” , *IEEE Spectrum*, August 2012

HRL Labs – Neuromorphic Architecture



Narayan Srinivasa and Jose M. Cruz-Albrecht, “Neuromorphic Adaptive Plastic Scalable Electronics”, **IEEE PULSE,** JANUARY/FEBRUARY 2012

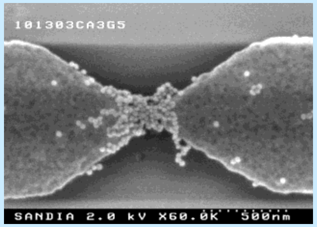
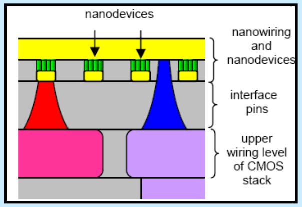
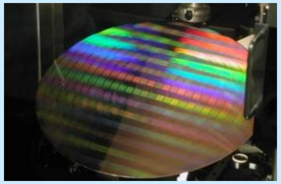
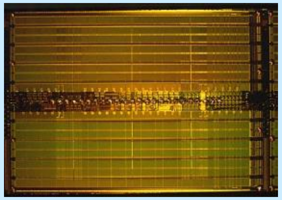
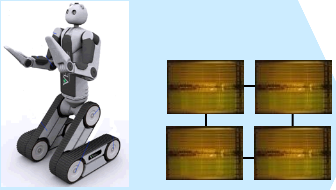
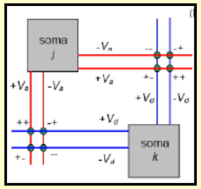
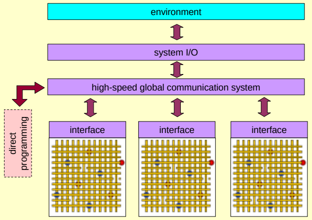
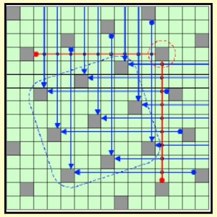
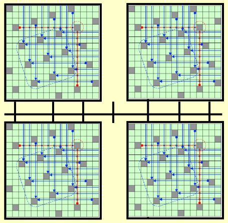
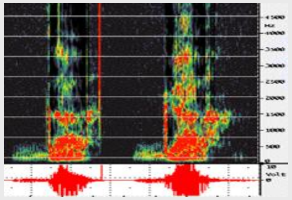
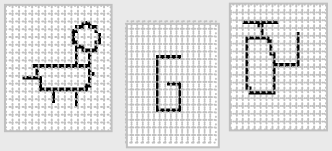
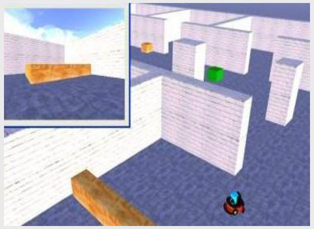
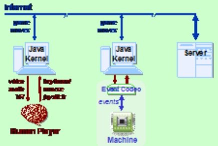
Messaging

- Spike
    - Simplest possible temporal message
    - Facilitates algorithms inspired by biological neural systems
    - Supports time and rate based algorithms
- Information "packet"
    - Generalization of spike time message
    - A "spike" that carries additional information
    - Facilitates other dynamical computing architectures using different primitives
- Routing of spikes / packets
    - Messages can be packaged with an address and routed over a network (e.g. IBM, SpiNNaker)
    - Messages can be delivered over a switching fabric (e.g. HRL)
    - Networks can be multiscale – e.g. on core, on chip, off chip

Key Technology Issues / Choices

- Distributing large amounts of memory (synapses) among many processors (neurons) on a single chip.
  - Off-chip memory burns power and taxes memory bandwidth
  - DRAM needs large array sizes to be space efficient and does not integrate into most logic processes
  - Back end memory technologies (e.g. memristors, PCM) are immature and not available in SOA CMOS
- Developing a scalable messaging (spiking) architecture.
- Selection of computational primitives (e.g. neuron and synapse models)
- Engineering for scale, space and power efficiency
- Creating a large-scale simulation capability that accurately models the neuromorphic hardware
- Creating tools to develop and debug neural algorithms on the simulator and the neuromorphic hardware
- Writing the algorithms (including those that learn)

SyNAPSE Program Plan

	Phase 0	Phase 1	Phase 2	Phase 3	Phase 4
Hardware	 Component Synapse Development	 nanodevices nanowiring and nanodevices interface pins upper wiring level of CMOS stack Process and Component Circuit Development	 CMOS Process Integration	 ~10^6 neuron single chip implementation	 ~10^8 neuron multi-chip robot
Architecture & Tools	 Microcircuit Architecture Development	 environment system I/O high-speed global communication system interface interface interface direct programming System Level Architecture Development	 10^6 Neuron Design for Simulation and Hardware Layout	 10^8 neuron design for simulation and hardware layout	DARPA-DESIGN HANDBOOKS Electronic Cortex Design Handbook First Edition Complete synapse-to-systems approach Detailed algorithmic descriptions Design automation toolsets Quantitative evaluation framework Real-world applications Comprehensive Design Capability
Emulation & Simulation		 Simulate Large Neural Subsystem Dynamics	 ~10^6 neuron level Benchmark	 ~10^8 neuron level Benchmark	 “Human” level Design (~10^{10} neuron)
Environment		 Internet Java Kernel Java Kernel Server Event Codes events Human Player Machine Build	Expand & Refine	Expand & Sustain	Sustain

SyNAPSE – Miscellaneous Lessons Learned

- There are many, many ways to build a neuromorphic computer
- Although much can be leveraged from conventional computing technologies, building a neuromorphic computer requires a large investment in development tools
- Neuromorphic computers can be applied as “control” systems for agents (e.g. robots) embedded in a dynamic environment.
- Neuromorphic algorithms can be replicated on a conventional computer, but with much lower efficiency.
- Biological scale networks are not only possible, but inevitable.
- The technology issues are challenging but surmountable.
- The time scale for developing a new memory technology and integrating it into SOA CMOS process is much longer than that needed to build a neuromorphic computer.
- The biggest current challenge in neuromorphic computing is defining the algorithms – i.e. the structure and dynamics of the network.

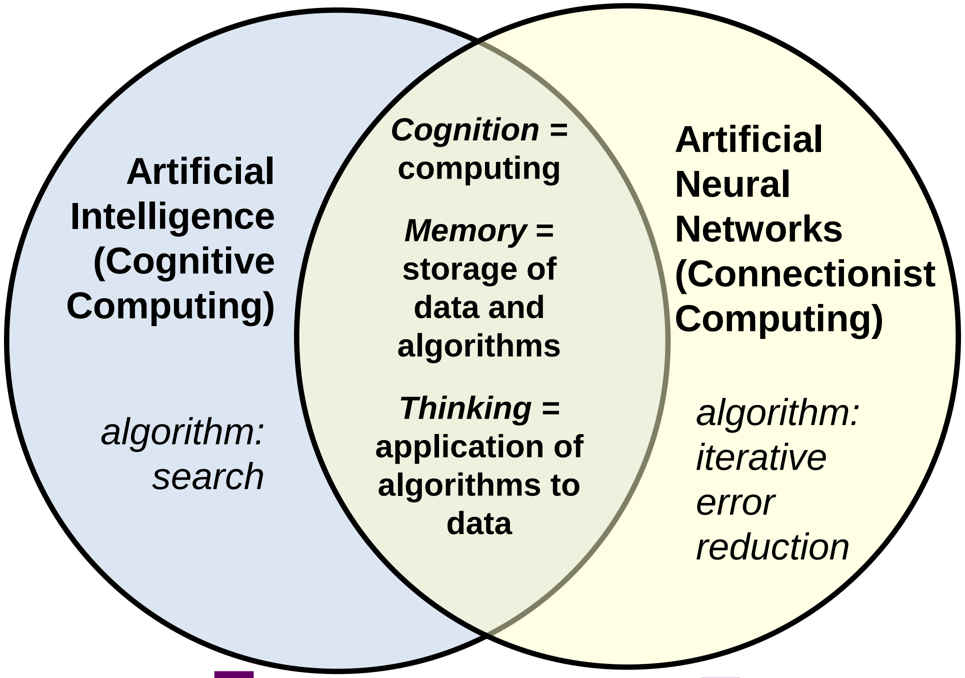
Why is Neuromorphic Computing Confusing?

Basic neuromorphic / cognitive computing proposition

Build computers that learn and generalize in a broad variety of tasks, much as human brains are able to do, in order to employ them in applications that require (too much) human effort.

- This idea is at least 40 years old, yet we still don’t have these kinds of computers.
- We have become disillusioned with these ideas in the past because the proposition was not fulfilled (AI and neural net “winters”)
- The proposition is (very) popular again because
  - Maturation of the computing industry
  - The successful application of some machine learning techniques
  - Interest and research on the brain

Neuromorphic / cognitive computing philosophy



Artificial Intelligence (Cognitive Computing)

algorithm: search

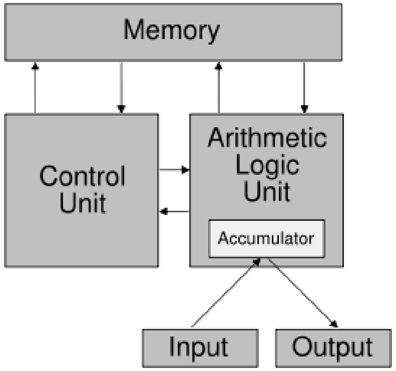
Cognition **= computing**

Memory **= storage of data and algorithms**

Thinking **= application of algorithms to data**

Artificial Neural Networks (Connectionist Computing)

algorithm: iterative error reduction



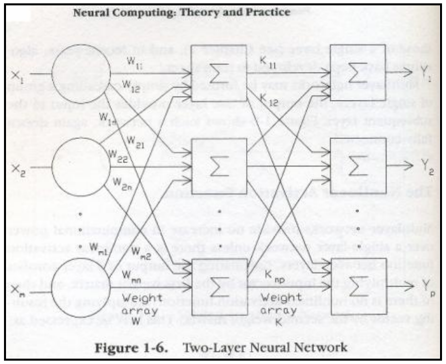
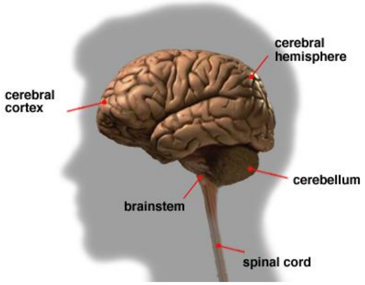


Figure 1-6. Two-Layer Neural Network

Cognitive computing views the brain as a computer and thinking as the execution of algorithms.



=

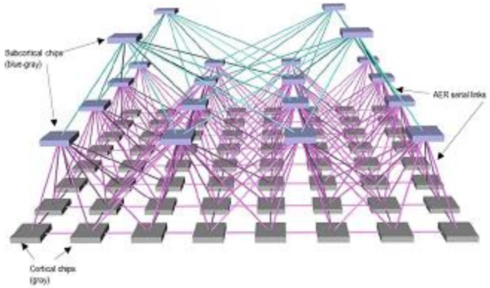
- Biological memory corresponds to a container holding data and algorithms. Learning fills the container with input-output rules defined on discrete (AI) or continuous (ANN) variables.
- Algorithms create input-output mappings using rules or weights stored in memory.
- AI focuses on search algorithms to select “production” rules.
- ANN focuses on iterative error reduction algorithms to determine “weights” yielding the desired input-output relationships.
- Algorithms are created by humans.

The Source of Confusion

The basic neuromorphic / cognitive computing proposition inappropriately mixes ideas and expectations from biological brains and computing.

SyNAPSE Architectural Concept

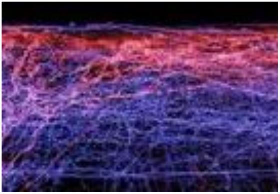
Neuromorphic Electronic System



Long-range interconnects

HIGH SPEED BUS

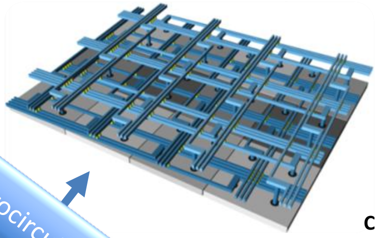
Multi-Gbit/sec digital comms

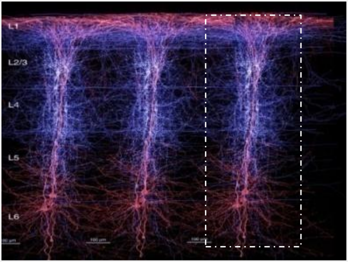


~5X10^8 long range axons @ 1 Hz

Microcircuit

LAMINAR CIRCUIT

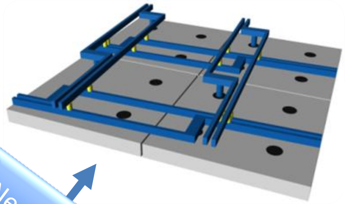




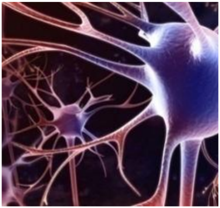
~10^4 Neurons / cortical column

Neurons

CMOS SUBSTRATE



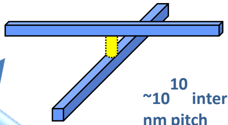
5X10^8 transistors/ cm^2 @ 500 transistors/ neuron



~10^6 Neurons /cm^2

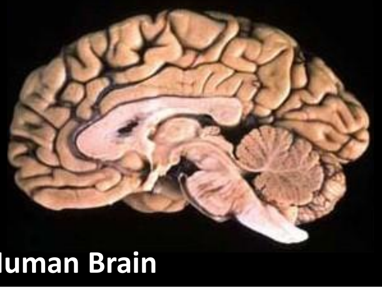
Synapse

CROSSBAR JUNCTION



~10^{10} intersections/cm^2 @ 100 nm pitch

~10^{10} synapses /cm^2



Human Brain

Approved for Public Release, Distribution Unlimited

Getting it Straight

- A neuromorphic computer is another kind of repurposable computing platform like a CPU, GPU, FPGA, etc.
- A neuromorphic computer will be more / less efficient than another computing architecture depending on the algorithm
    - A key question in designing a neuromorphic computer is understanding the structure of the algorithms it will likely run
- Neuromorphic computers may be good choices for implementing some machine learning algorithms, but these should not be confused with brains
- **A neuromorphic computer is not a brain**, although if we were ever to figure out how to simulate a brain on a computer, a neuromorphic computer would likely be an efficient option.

What about building a brain?

Reductionist approach

- Proposition: By understanding the component parts and functions of the brain, we can build brain-like systems from an arrangement of similar components.

- Approach: Study the brain as system of components and subsystems and infer their relevance to overall brain function. Create a brain-like system by mimicking the components and structure of the brain.

- Example: Create dynamical models of biological neurons and synapses and configure them in a network inspired by brain anatomy. Implement these ideas in software or hardware.

Reductionist conundrum

- What is the appropriate level of abstraction needed to build a brain?
  - What components / functions of the brain correspond to its “computational primitives”?
- How do I distinguish relevant from irrelevant features in a real brain?
- How do I deal with the interactions among the components?
  - How does neuroanatomy correspond to the brain’s “architecture”?
- How do I deal with the interactions with a larger environment?
- Is there an algorithm of the brain that the components execute?

- *Reductionism as a strategy to building a brain is equivalent to the basic neuromorphic / cognitive computing proposition*

Limits of reductionism

- Science shows repeatedly that understanding lower levels of organization is insufficient to understand high levels. In general a new description is required at each new level. For example
  - Chemistry cannot be derived from physics
  - Microbiology cannot be derived from chemistry
  - Organisms cannot be derived from microbiology
  - Ecosystems cannot be derived from organisms

- *More is different* - Phil Anderson

Why more is different

- The (typically massive) interaction / feedback that is characteristic of real world systems eliminates the concept of an independent part or piece. When "everything is connected to everything," it becomes difficult to assign an independent function (input-output relationship) to the components.
- Higher levels of organization evolve from their lower level components in response to interaction with their environment. Higher level organization depends strongly on influences external to the system of its components.

Todd’s Top 10 List of Challenges in Building a Brain

10. Neuroscience is too little help (tlh)

- We cannot possibly simulate all the detail of a biological brain
- We don’t understand the function of very simple nervous systems
- There are far, far too few “observables” to guide the development of any model or technology

9. Computational Neural Models are tlh

- Too many assumptions
- Too many parameters
- No general organizing principle
- Models are (usually) incomprehensible
- Unclear connection to applications

8. Other things that are tlh

- Cortical column hypothesis
- Sparse distributed representations
- Spiking neural networks, STDP
- Hierarchies of simple and complex cells
- Insert your favorite ideas here
  - Spatio-temporal, scale invariance
  - Criticality, critical branching
  - Causal entropic forcing

7. Whole System Requirement

- Brains are embodied and bodies are embedded in an environment (Edelman)
- Testing often requires embedding the neuromorphic computer in a complex body /environment.

6. Whole System Interdependence

- Brains / bodies / environments are complex systems whose large scale function (almost certainly) cannot be analytically expressed in terms of its lower level structure / dynamics
- System design methodologies are inadequate because the system cannot be decomposed into independent parts

5. No Easy Path for Technology Evolution

- The benchmark for performance comparison is either
  - A human
  - A well-engineered, domain-specific solution

4. Massive Computing Resources

- Any model that does anything that anyone will care about requires a massive computational resource for development and implementation
- Development is slow and expensive
- Custom hardware in state of art process is needed for any large scale application
- Software and hardware must co-evolve
  - Cannot develop the algorithms first
  - Cannot specify the hardware first

3. Competition for Resources

- It is easy for anyone who doesn’t like your project to claim that
  - It is making no progress
  - It is not competitive with the state of the art
  - You are doing it wrong
  - You are an idiot
- This happened to me regularly at DARPA

2. Computers can compute anything

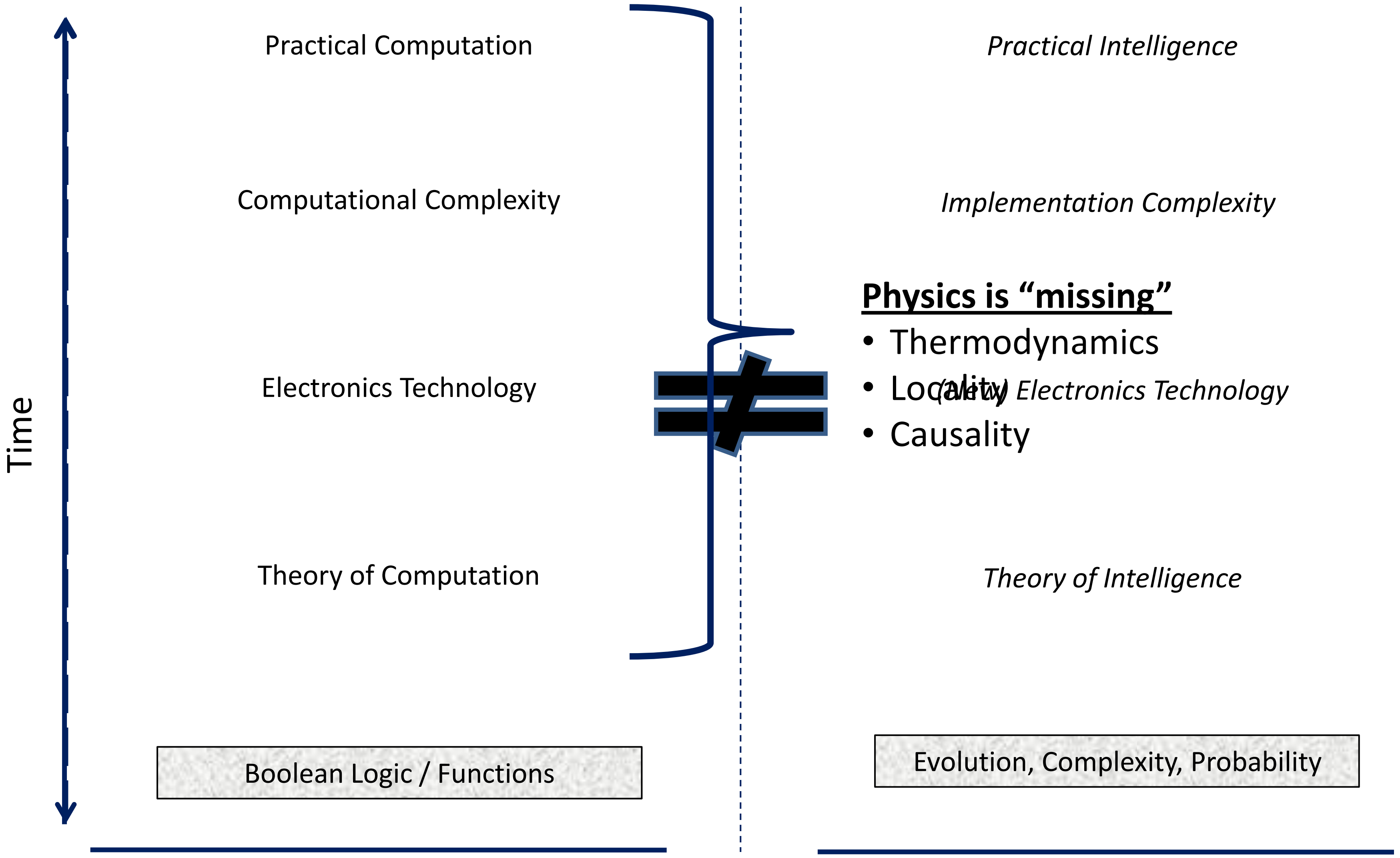
- The computer is a blank slate
- We must generate all the constraints to build a neuromorphic computer
- Changing computing architecture only changes the classes of algorithms that it computes efficiently

1. Brains are not Computers

- Brains are thermodynamical, bio/chemo/ physical systems that evolved from and are embedded in the natural world
  - Computers are symbolic processors executing algorithms designed by humans
- Brains designed computers.
  - Can computers design brains?

Alternatives Ways of Thinking About Building a Brain

Perspective – What we need in order to build a brain



Time

Practical Computation

Computational Complexity

Electronics Technology

Theory of Computation

Boolean Logic / Functions

Intelligence & Computation

Computation

Practical Intelligence

Implementation Complexity

<u>Physics is “missing”</u>

- Thermodynamics
- Locality
- Causality

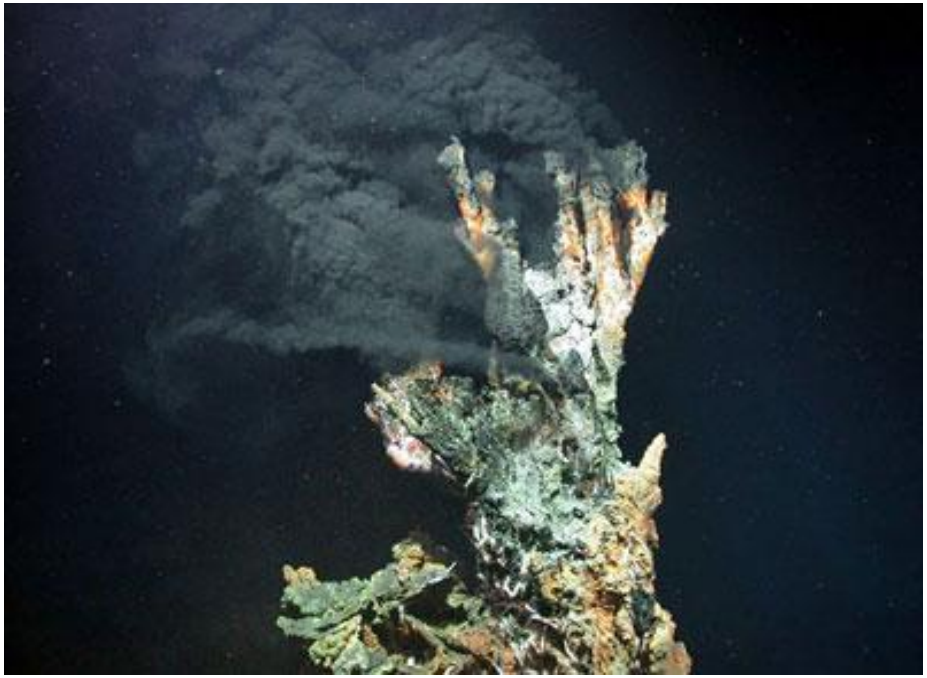
(New) Electronics Technology

Theory of Intelligence

Evolution, Complexity, Probability

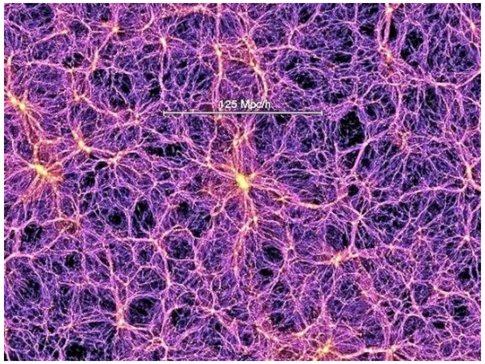
Intelligence

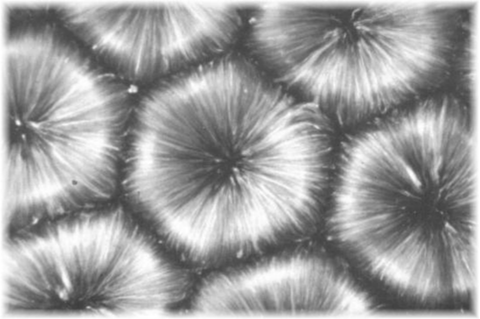
Life is Autotrophic

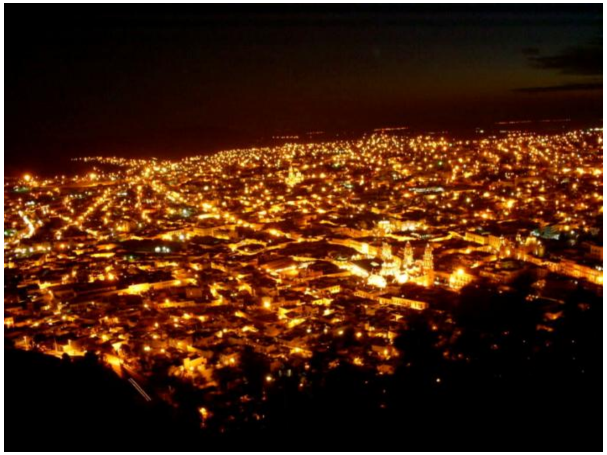


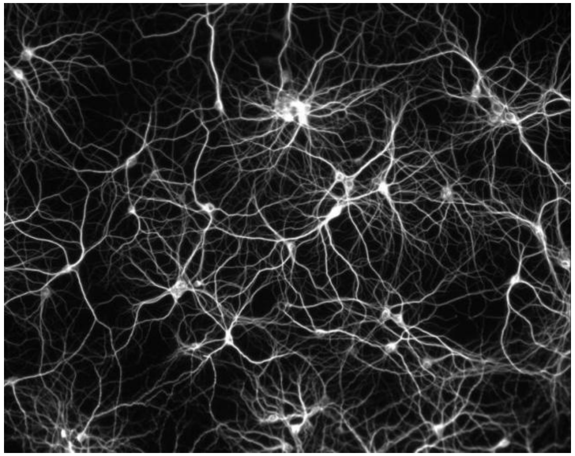
On hydrothermal vents, life is sustained by *chemoautrophic* bacteria, which derive energy and materials from purely *inorganic* sources. These bacteria provide an efficient means to consume energy through a chemical cascade that would otherwise not be possible. At the ecosystem level, all life is *autotrophic* in that it is derived from inorganic sources (and sunlight). In general, life provides a means to relieve chemical potential “gradients” that could not otherwise be accessed (because of energetic activation barriers).

Thermodynamically Evolved Structures



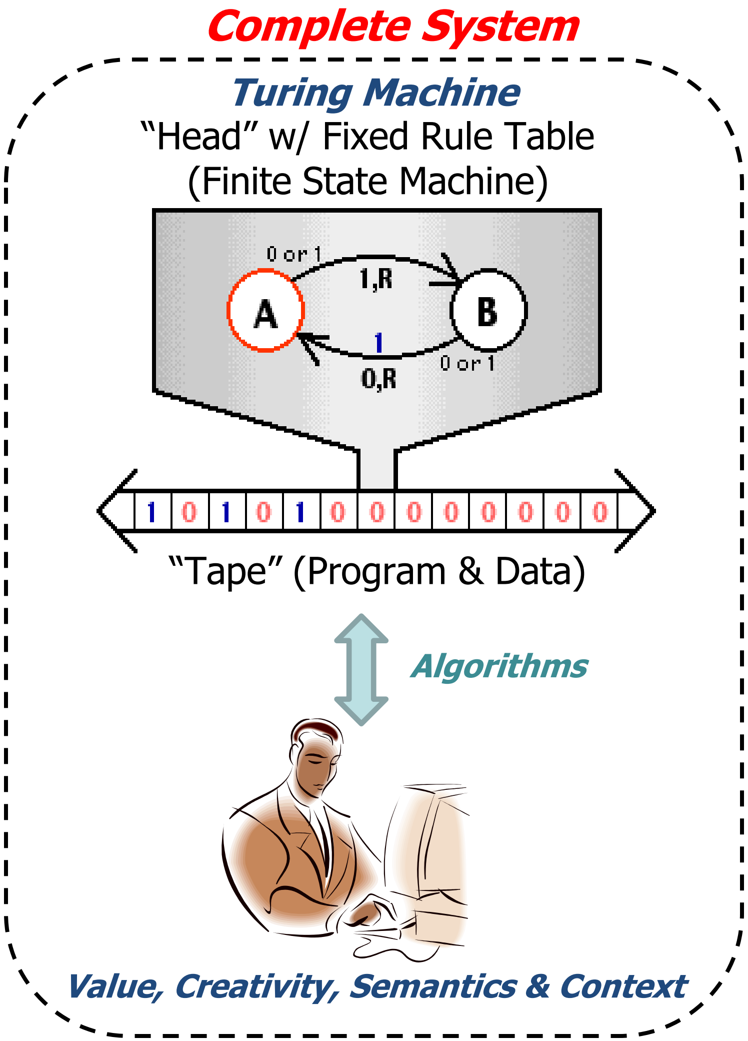






Conceptual Issues – Foundations of Computing

- Observation - The Turing machine (and its many equivalents) is the foundational idea in computation and has enabled decades of success in computing, *but*
  - The **machine** is an abstraction for symbol manipulation that is *disconnected* from the physical world.
  - **Humans** provide contact with the physical world via the *creation and evaluation of algorithms*.
- Question – With such foundation, is it reasonable to suppose that the **machine** can understand, adapt and function in a complex, non-deterministic, evolving problem or environment?



Complete System

Turing Machine

"Head" w/ Fixed Rule Table

(Finite State Machine)

"Tape" (Program & Data)

Algorithms

Value, Creativity, Semantics & Context

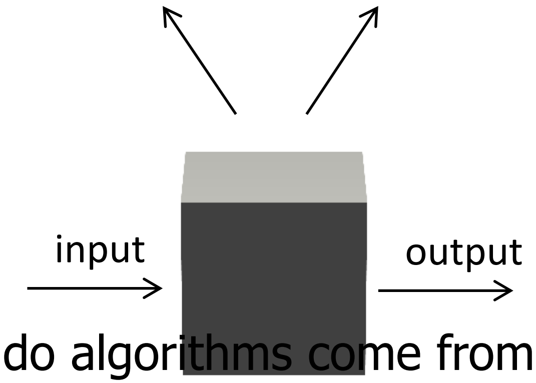
Hypothesis #1: Intelligence concerns the ability to create (useful) algorithms.

Evolution of Intelligence

— *physical* energy — *biological* evolution — *psychological* stimulus-response — *gnoseological* cognition & mind — *sociological* cooperation & competition →

Current Paradigm: *Cognitive Computing*

- Brains are universal computers
  - Algorithms determine behavior
  - Memory = storage of data and algorithms
  - Thinking = application of algorithms to data
- Intelligence is algorithmic
- Intelligence ⊆ computation



Where do algorithms come from?

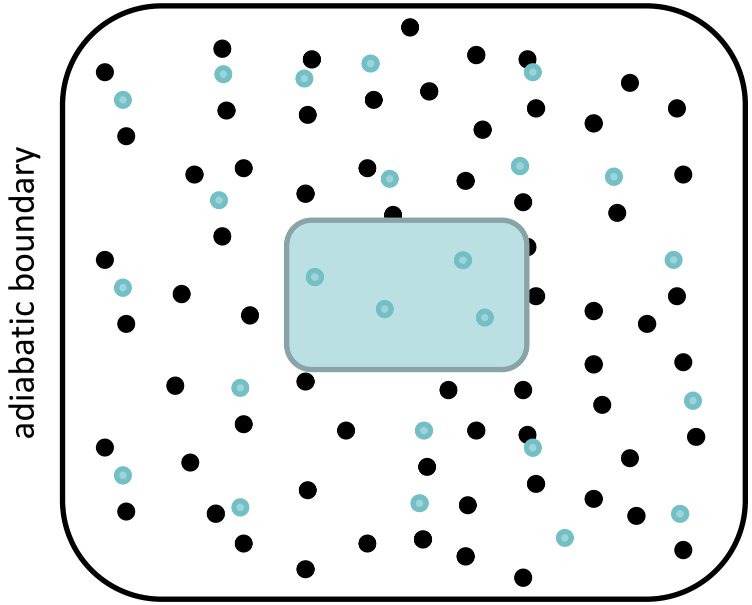
unphysical, static, unscalable, black-box efforts targeting the highest levels of intelligence

Today's Approach

Hypothesis #2: Intelligence is part of a pervasive evolutionary paradigm that applies to the physical and biological world. The computational metaphor for intelligence is inadequate. *Intelligence is physical*.

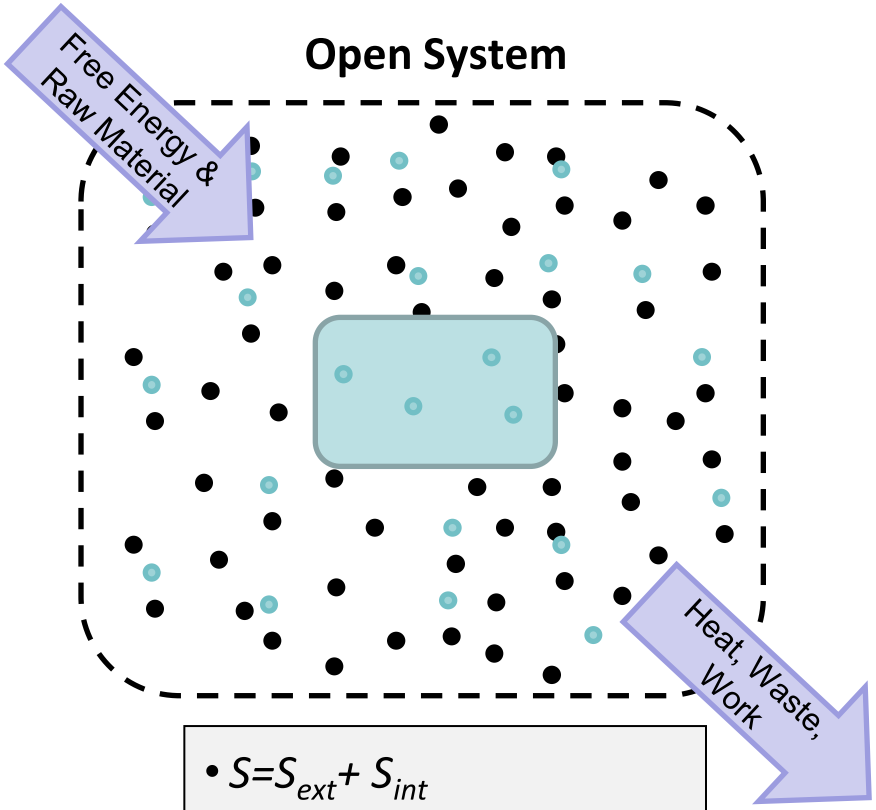
Thermodynamics of Open Systems

Isolated System



- S = *Entropy*
- $S(t) \rightarrow S_{max}$
- $dS/dt > 0$

Open System



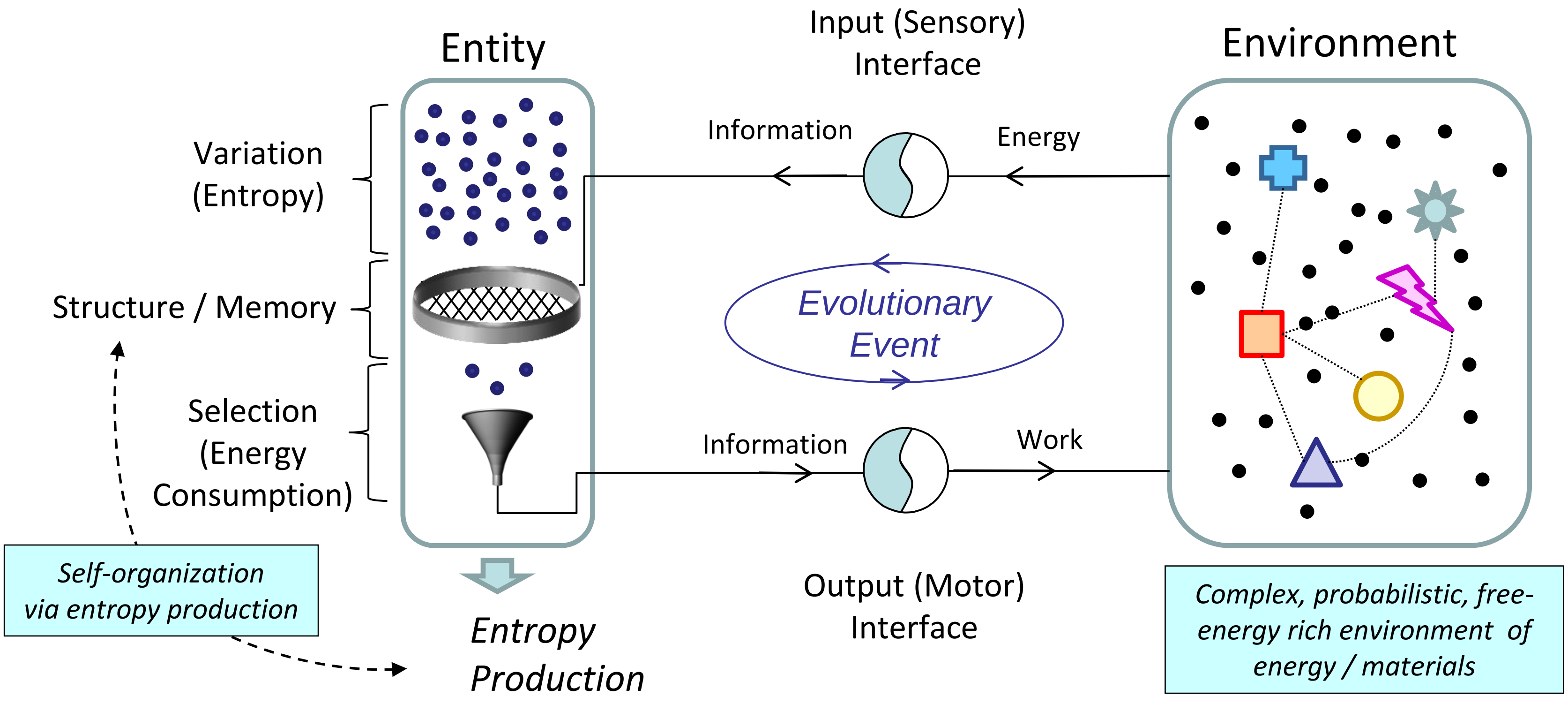
- $S = S_{ext} + S_{int}$
- $dS/dt > 0$

$\Rightarrow dS_{int}/dt < 0$

- $dS/dt \rightarrow (dS/dt)_{max}$?

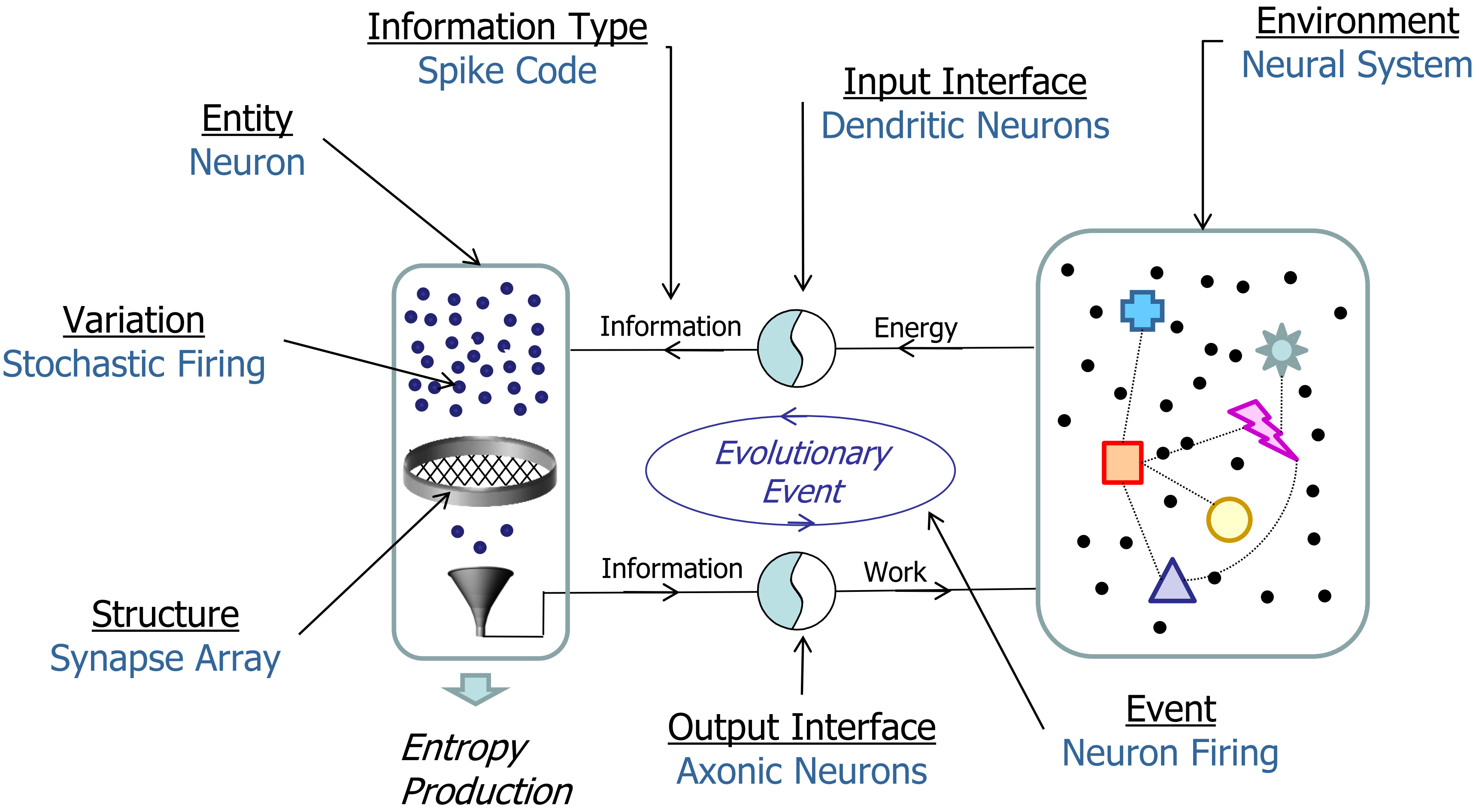
Open thermodynamic systems spontaneously evolve structure via entropy production in the external environment.

Thermodynamic Evolution Paradigm


Entity
Input (Sensory) Interface
Environment
Variation (Entropy)
Information
Energy
Structure / Memory
Evolutionary Event
Selection (Energy Consumption)
Information
Work
Self-organization via entropy production
Entropy Production
Output (Motor) Interface
Complex, probabilistic, free-energy rich environment of energy / materials

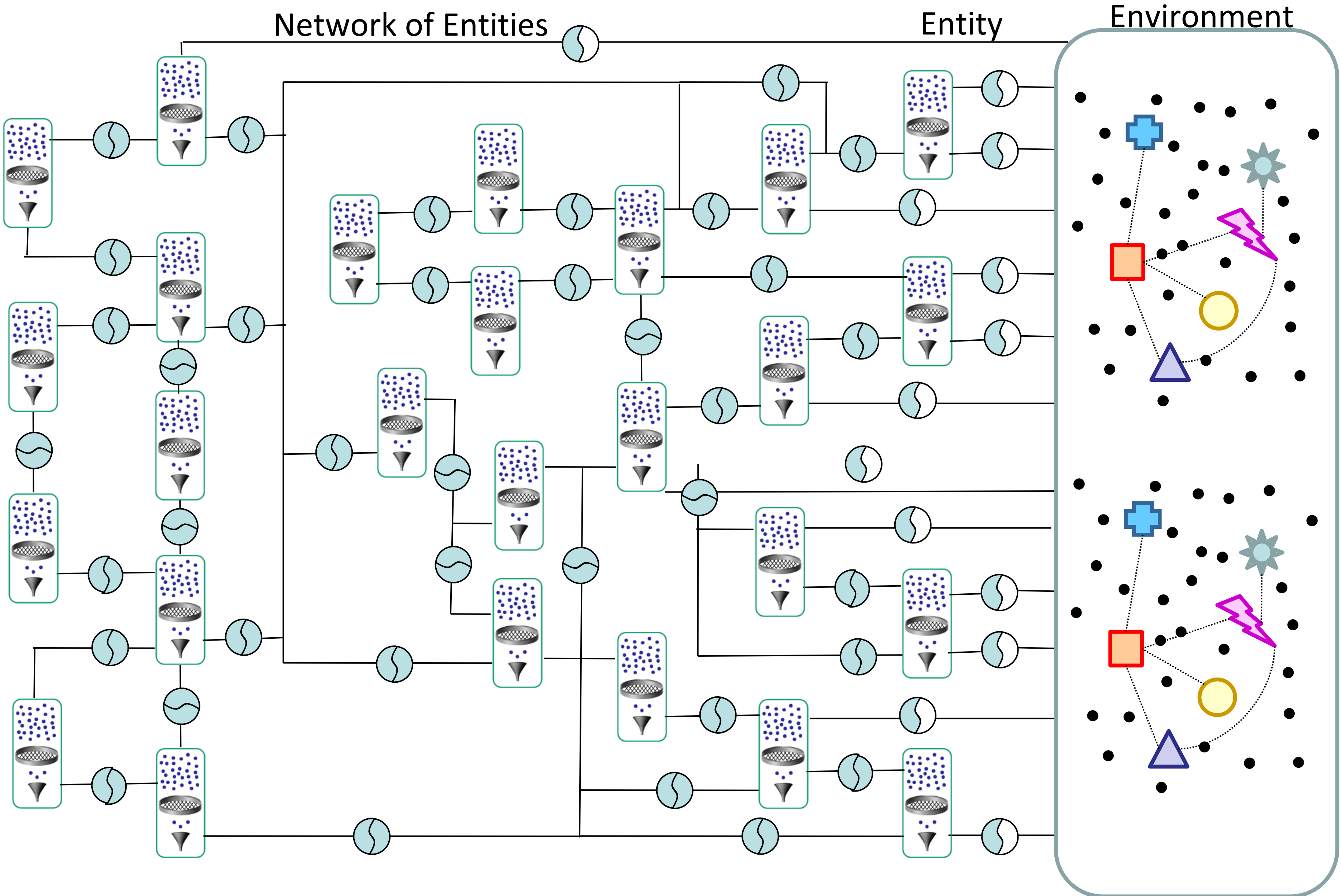
- Entities extract resources from their environment through evolutionary variation & selection.
- Entropy production rate selects for (Algorithmic) Structure / Memory among entropic variations.
- (Algorithmic) Structures / Memories constrain the variation repertoire in future selections.
- Entities are distinguished from their environment by their level of integration.

Example Evolving System

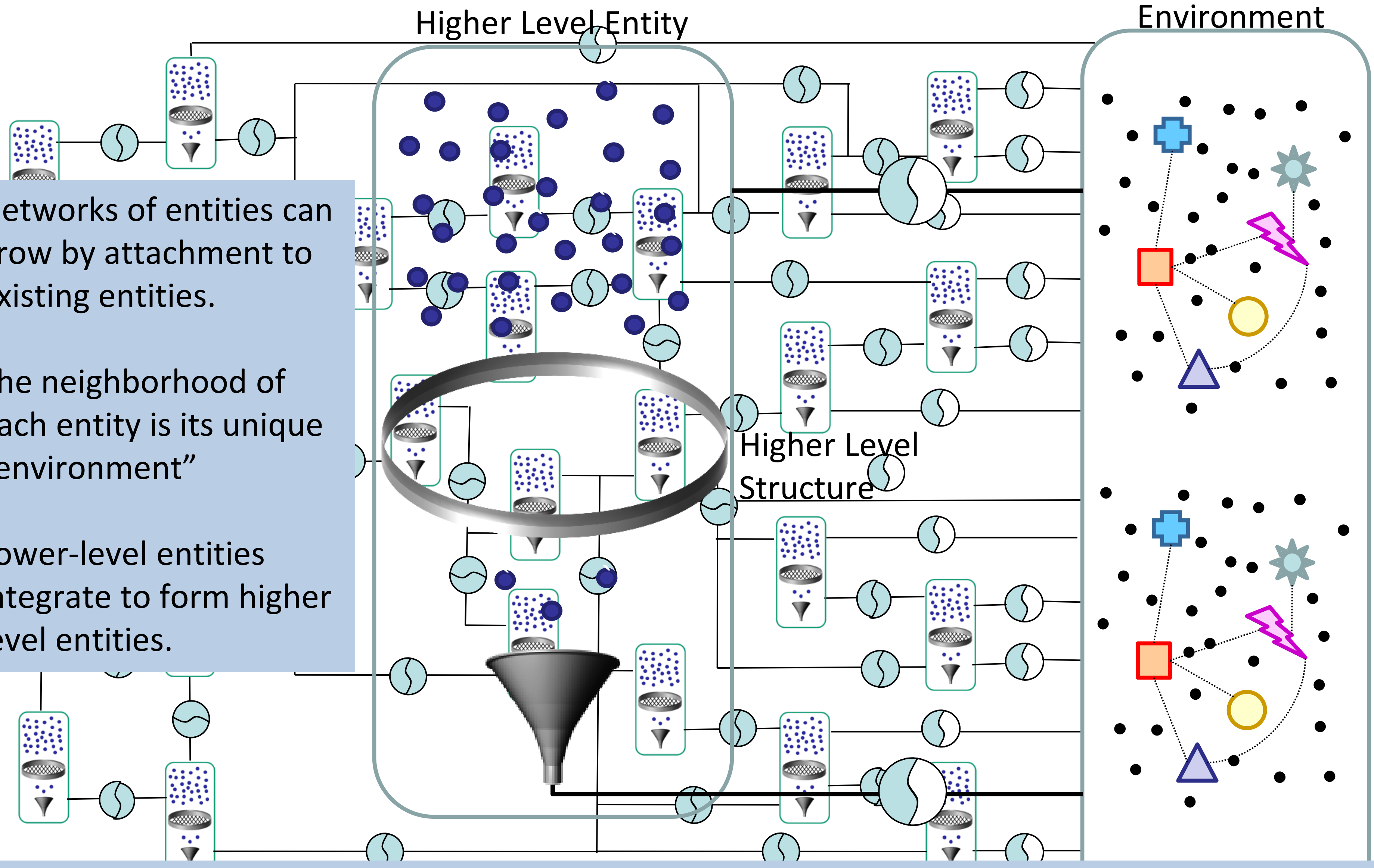


Neural systems qualitatively fit the thermodynamic evolution paradigm.

Structure Growth, Integration & Scaling



Structure Growth, Integration & Scaling



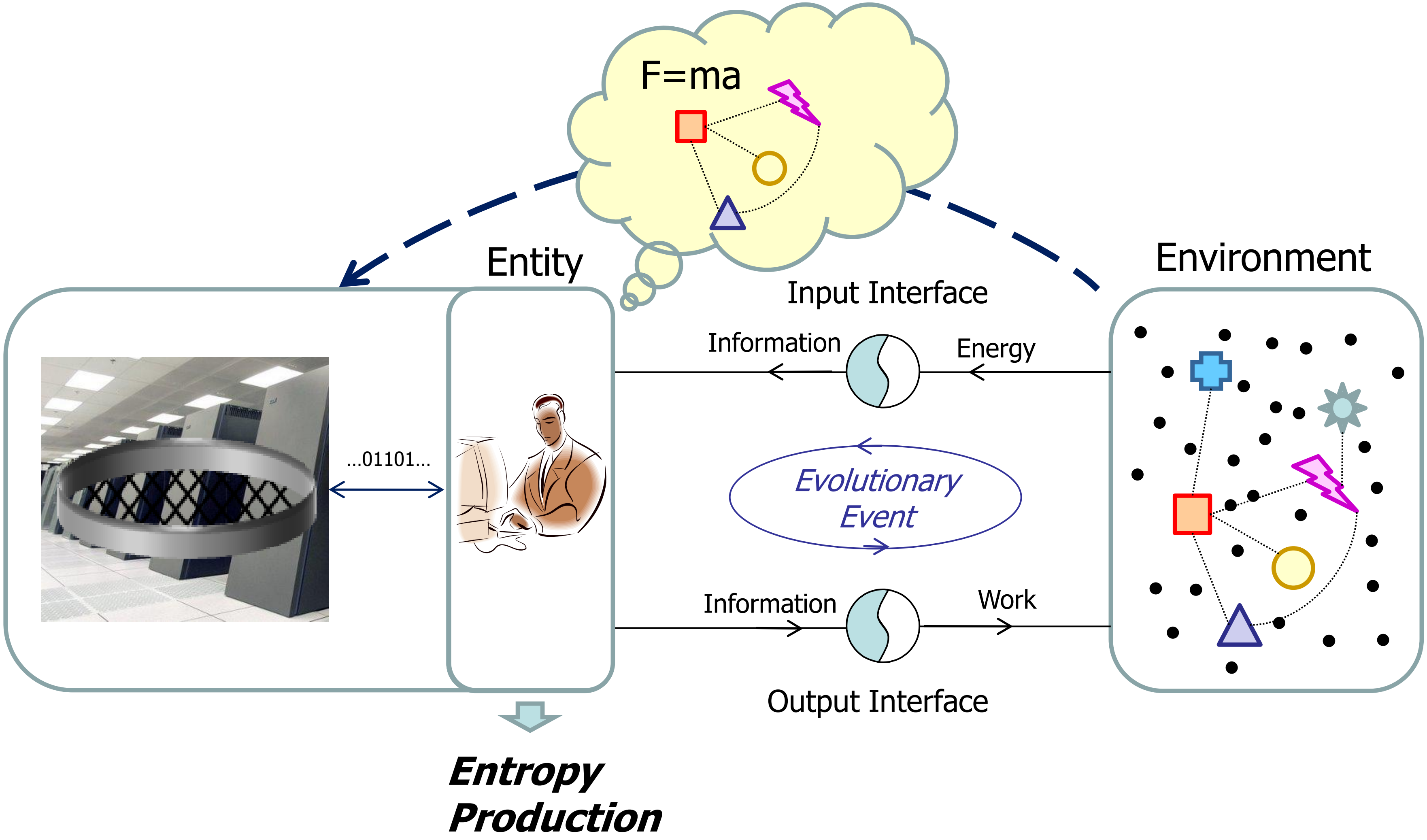
Networks of entities can grow by attachment to existing entities.

The neighborhood of each entity is its unique "environment"

Lower-level entities integrate to form higher level entities.

Networks of entities evolve and integrate Algorithmic Structure into higher level entities / systems.

Computing in the Physical Intelligence Framework



Computers enhance our ability to extract energy and resources from the environment by allowing us to evolve and use larger, faster and more complex algorithms.

Closing Thoughts

What has changed in 7 years

- End of semiconductor scaling clearly in sight
- Numerous large scale efforts in neuromorphic computing now exist
  - Community has substantially grown
  - Several example systems now exist
- Deep learning algorithms have matured and are being deployed
- BRAIN Initiative and Human Brain Project have been announced/started

Think “Algorithms” and not “Brains” when building a NC

Dynamical Algorithms

- Represent systems of coupled dynamical equations
  - not just feedforward networks
- Interact in real-time in the real world (e.g. robotics)
- Tough to conceive, tough to “debug”

Typical Questions

- What are the plasticity/adaption rules? / What are the dynamical equations?
- What network should I build?
- What is the effect / interaction of the components with the system?
- What / how should I test it?
- How can I figure out what is wrong?
- How do I make it do something (that I want it to do)?

What We Can Do

- Build new kinds of computers that are capable of efficiently executing new classes of algorithms
- Build better machine learning algorithms

Recommendation

- Separate/classify effort into 2 domains
  - Aspirational efforts focused on building a brain (the basic NC proposition)
  - Practical efforts focused on building new, useful computers
- Avoid the temptation to straddle both domains

Backup

Digital or Analog?

- Communications
  - Digital – no controversy
- Neurons
  - Digital – computed dynamics, scales, reproducible, multiplexes, parameterizes
  - Analog – intrinsic dynamics, low power
- Synapses
  - Digital – computed dynamics, scales, reproducible, multiplexes, parameterizes
  - Analog – intrinsic dynamics, low power
- State of the art CMOS technology and design practice generally favors digital implementations
- Groups of highly multiplexed, digital neurons and synapses resemble small processor cores with dedicated memory (like SpiNNaker)
- Mixed analog-digital solutions are also possible

User Focused NC Proposition

- We will build a computer to enable (for example)
  - computational neuroscientists to efficiently model large neural systems.
  - analysts to more easily understand video

- Comment: This kind of proposition is a long way from an engineering specification.

Algorithm Focused NC Proposition

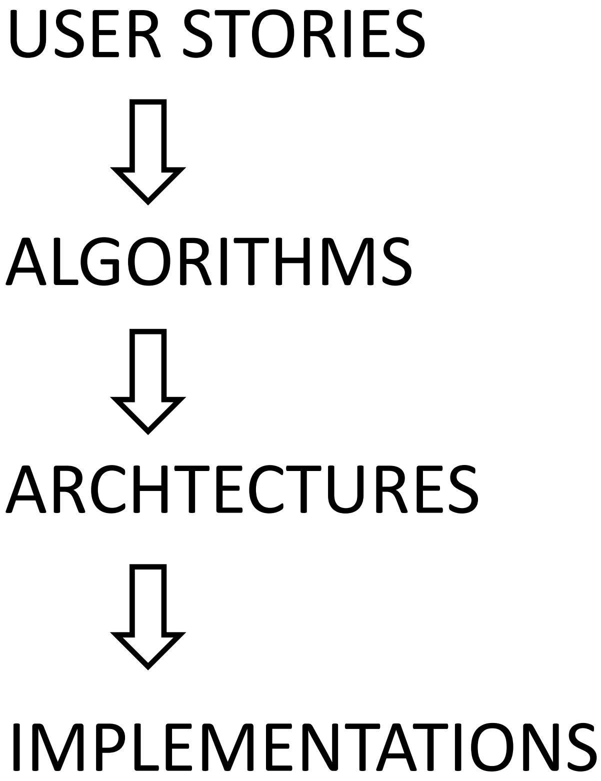
- We will build a computer that efficiently computes certain (classes of) machine learning algorithms

- Comment: This kind of proposition can lead to narrowly focused systems (ASICs).

Architecture Focused NC Proposition

- We will build a computer featuring the following architectural concepts (for example)
  - SDR
  - event-based execution, asynchronous communication
  - highly distributed simple cores within a dense memory
  - neural/synaptic/columnar computational primitives,
  - criticality/homeostasis….
- Comments:
  - Before any specification can be created, a description like this is required
  - It isn’t obvious from such propositions what the computer will be good at / used for.

The Evolution of NC Has Begun



USER STORIES

ALGORITHMS

ARCHTECTURES

IMPLEMENTATIONS