Languages for Molecular Cell Biology

Luca Cardelli

Microsoft Research Cambridge

Redmond TAB, 2003-06-24

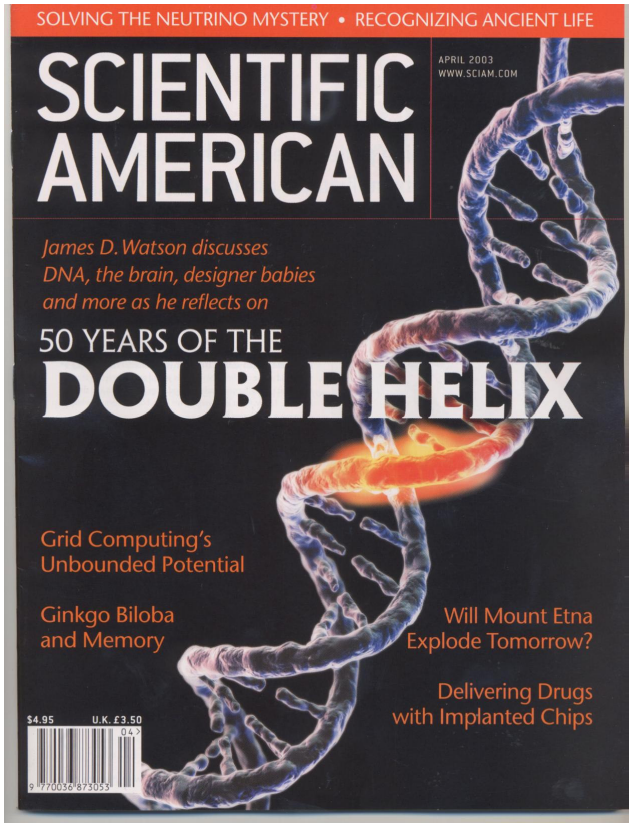
Ekaterina M. Panina - Molecular Biology Institute, UCLA
Aviv Regev - Bauer Center for Genomic Research, Harvard
Ehud Shapiro - CS and Applied Math, Weizmann Institute, Israel
William Silverman - CS and Applied Math, Weizmann Institute, Israel

Introduction

- Involved “accidentally” via prior language work
- Learned many fascinating things
  - Certainly related to future of computing
  - Possibly related to software, in the standard sense of prog. languages and devel. tools
  - “VS for biotech”
- Here to tell you
  - Some of the most surprising things I learned (from a “software engineering” point of view)
  - A little bit of what we did
  - Need to explain, first, what systems we are trying to describe

Molecular Cell Biology

- How cells work:
  - DNA stores information
  - Ribosomes are instructed (by DNA) to assemble proteins
  - Proteins (>10000) do things:
    - Process signals, activate DNA
    - Catalyze reactions to produce substances (including ribosomes)
    - Control energy (ATP) production and consumption
  - Bootstrapping still a mystery
    - Happened a long time ago; not well understood, but who cares.



Like: try to reconstruct DOS sources by looking at NT kernel.

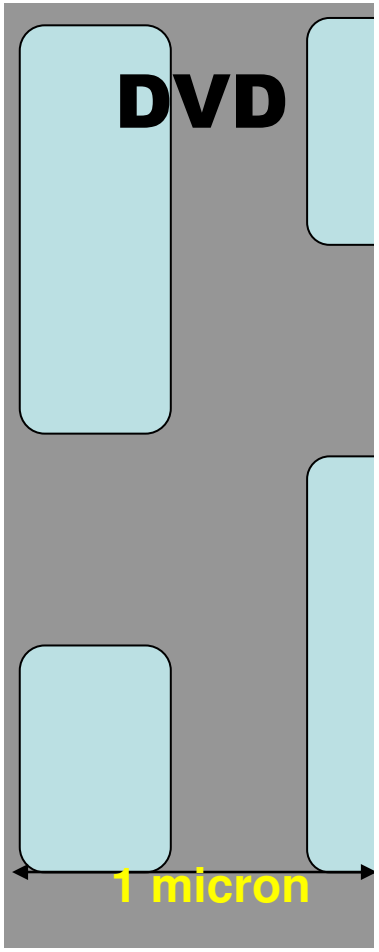
A System. What kind of system?

- A cell is the “smallest self-contained functional unit”, often also an independent organism.
  - Small enough and general enough to be interesting.
- Fun to try and understand in “our” terms.
  - As complex as Windows? 10 times Windows? 10 million times Windows?
  - How is it structured? Why is it structured that way?
  - Could it be done better?
  - What kind of “development tools” would we (or biologists) need to “do it better”?

What a Cell is Not

- Not a quantum-mechanical fuzzball
  - Activity is based on the discrete interactions of highly specific high-level building blocks (macromolecules)
- Not a chemical soup
  - Almost all reactions are “deliberately” triggered, and are localized to very specific membranes or compartments, with active transportation between different sites.
- Not a brain
  - Representation and processing of information is (fairly) understood.
  - So are most fundamental processes, at least in broad outlines.
- It’s more like:
  - Energy management
  - Mechanical assembly
  - A lot of signal processing
  - The result of a huge amount of fine tuning

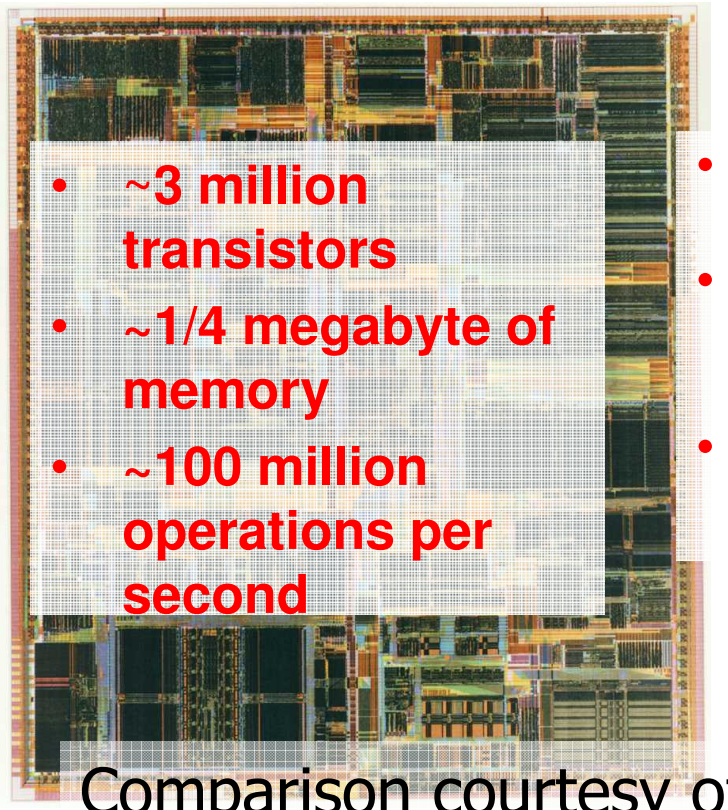
Size



$\lambda = 0.25$ micron
in Pentium II

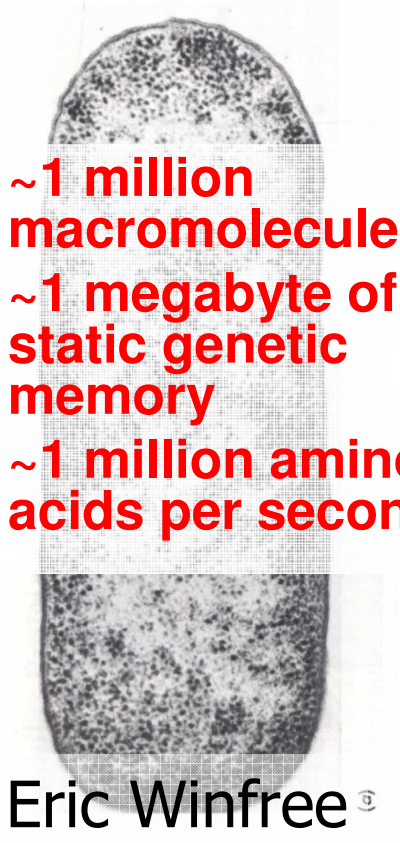
Performance

Pentium II



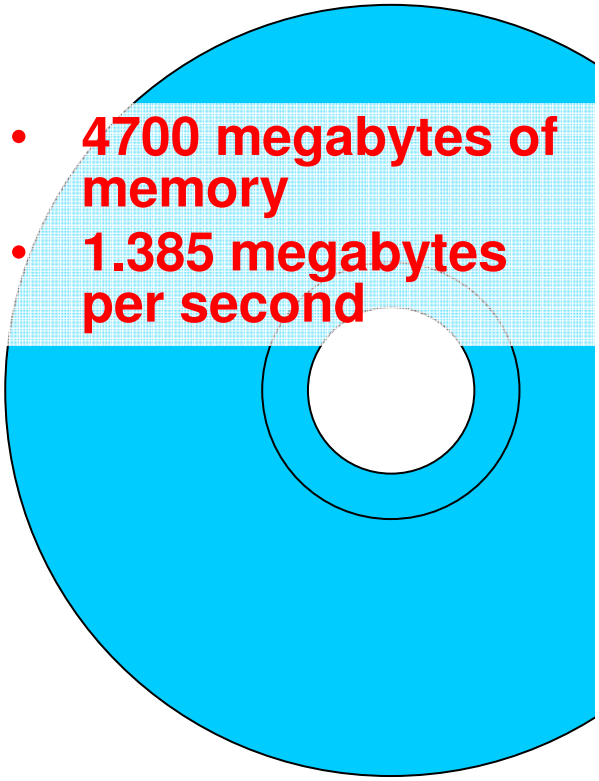
- ~3 million transistors
- ~1/4 megabyte of memory
- ~100 million operations per second

E. Coli



- ~1 million macromolecules
- ~1 megabyte of static genetic memory
- ~1 million amino-acids per second

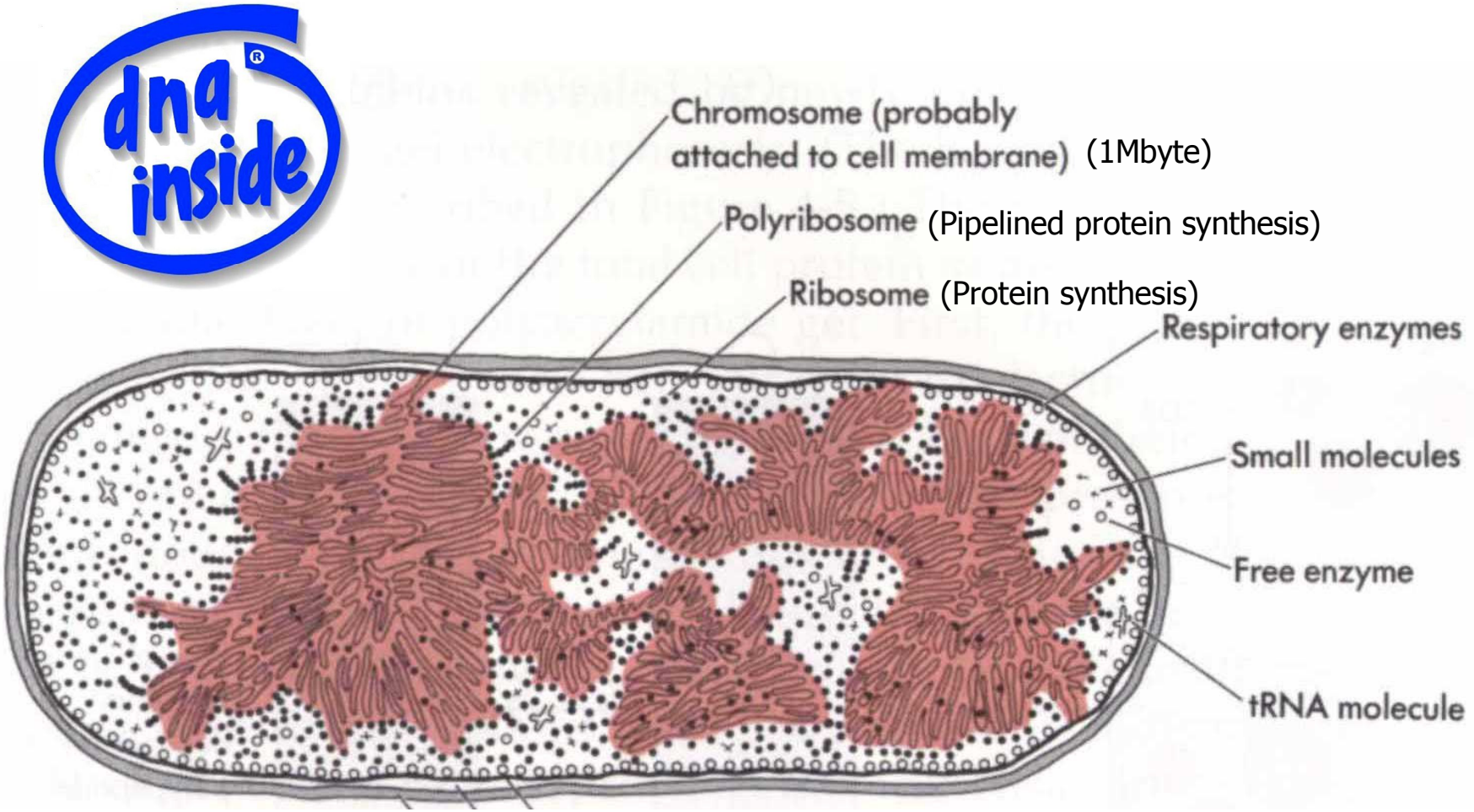
DVD



- 4700 megabytes of memory
- 1.385 megabytes per second

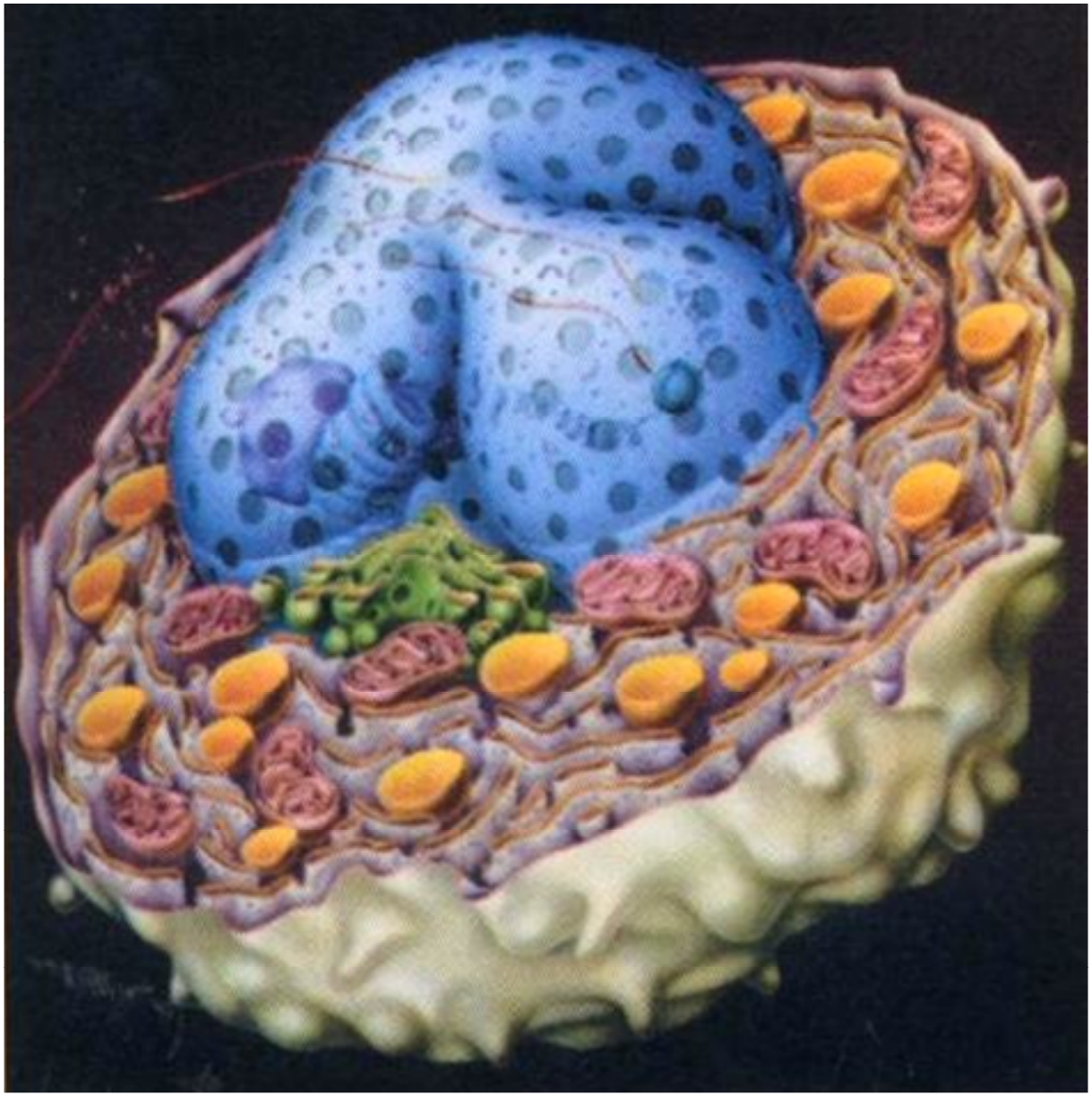
Comparison courtesy of Eric Winfree

Architecture (Bacterial Cell)



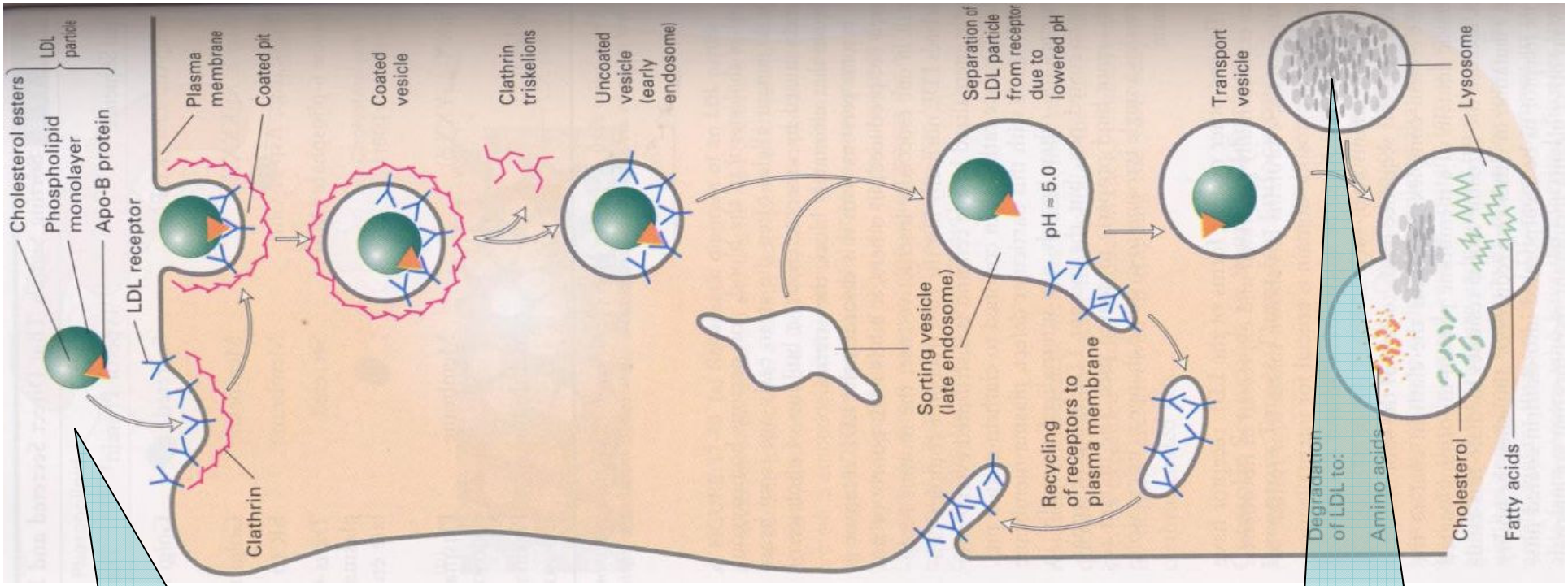
Architecture (Eukaryotic Cell)

- Lots of compartments
- Active transport



Dynamic Compartments

- Breakdown of cholesterol

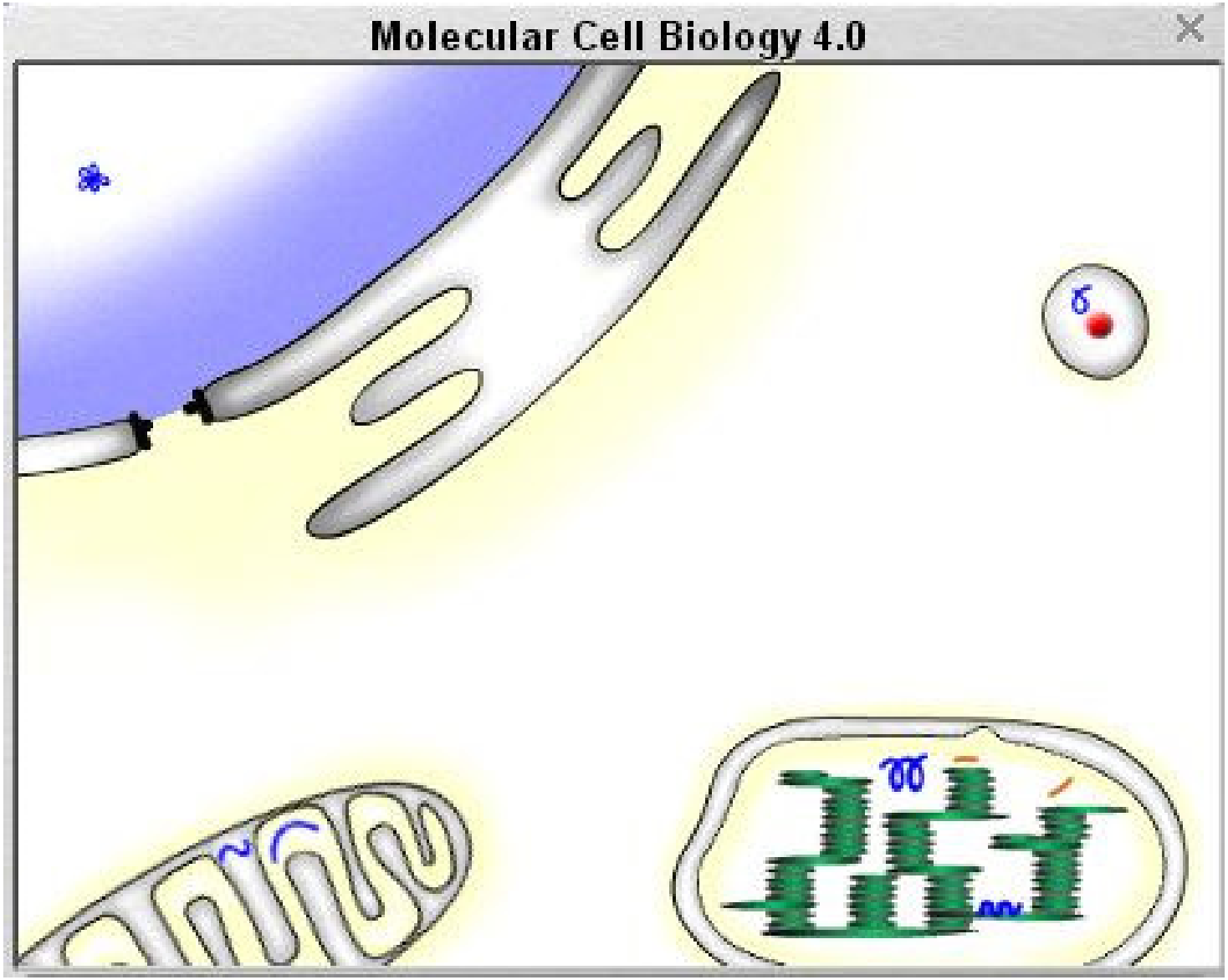


An “algorithmic” collection of discrete step.

How to get this bit…

… way over here.

Dynamic Compartments



A System. How to describe it?

- Biochemistry needs a formal language
  - To precisely describe the architecture and dynamics of a complex biological system based on discrete, nondeterministic interactions.
  - As opposed to nice diagrams or their XML encodings.
  - As opposed to popular description tools based on differential equation and stochastic simulation.
  - Biochemistry has UML and CMOS simulators, but no C++.
- We now need to understand the software of biology
  - How do all those subunits end up behaving like a cell?
  - Think an x86 (easy) running SNOBOL (uh?).

Languages for Biochemistry

- A useful language for biochemistry must be:
  - High level (protein/cell interactions, not atom interactions)
  - Highly concurrent (the “soup” aspect)
  - Structured (membranes, compartments, transport)
  - Stochastic (quantities/frequencies/equilibrium)
- Could be used for:
  - Documentation (precise description of known reactions)
  - Analysis/simulation (exploring paths, extracting traces)
  - Formulating and testing hypotheses “in silico” (drug devel)
  - Compiling biological systems (maybe for biocomputing)
- Formal proposals:
  - Variants of Petri Nets (non-compositional)
  - Variants of statecharts (David Harel et al.)
  - Variants of process calculi (Ehud Shapiro et al.)

BioSpi [Shapiro et al.]

- The Biochemical Stochastic π-calculus.
  - A process language for describing biochemical reactions.
  - "Interaction" here is: small molecules exchanging electrons (chemistry), or large molecules exchanging smaller molecules (biochemistry).
  - "Communication ports" are the multiple active domains of proteins.
  - Provides excellent opportunities for "program analysis".
- A simulation tool
  - Interactions have reaction rates (Gillespie algorithm).
  - Good for quantitative modeling/simulation at various abstraction levels.

"Control Flow Analysis"

- Cascades of biochemical reactions
  - Are usually drawn in biology publications as *biochemical pathways*, i.e. "concurrent traces" such as the one here, summarizing known facts.
  - This one, however, was automatically generated from a program written in BioSpi. [Curti, Priami, Degano, Baldari]
  - One can play with the program to investigate various hypotheses about the pathways.

- Write the behavior of each molecule as a BioSpi process (a small stochastic FSA).
- Compose all the molecules in parallel.
- Run it as a simulation, or:
- Compute traces of all possible interactions between molecules.
- Extract the *causality relation*: what interactions causally depend on what.
- Represent it as a Hasse diagram.
- Quantitative information can be extracted as well, from the known frequencies of individual reactions.

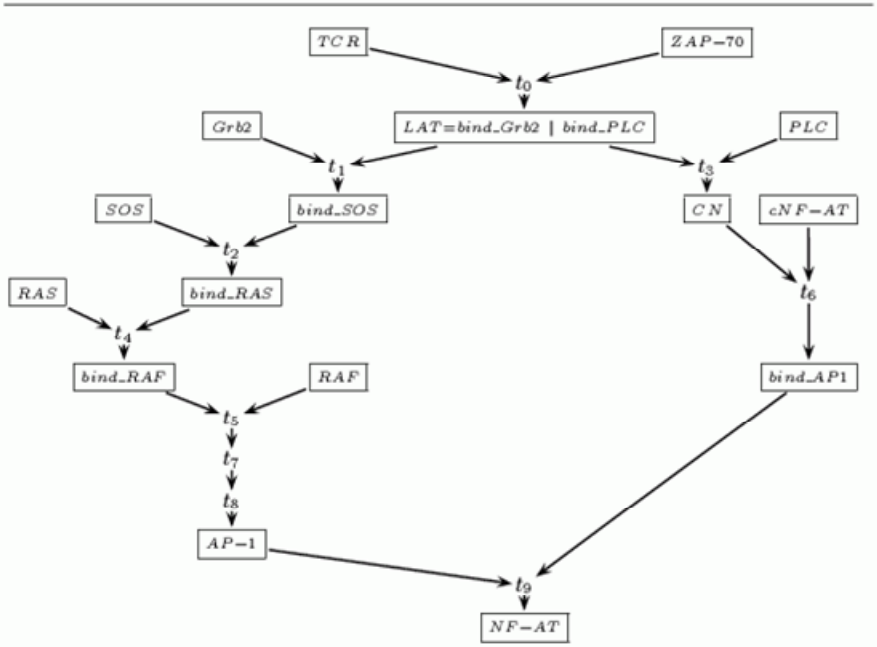


Fig. 2. A computation of *Sys*. For readability, the processes, enclosed in boxes, have no address. Causality (both on transitions and processes) is represented by the (Hasse diagram resulting from the) arrows; their absence makes it explicit concurrent activities.

BioAmbients

- *BioAmbients* is an extension of BioSpi with:
  - Membranes: important to describe many biological processes.
  - Dynamic membranes: operations for merging, splitting, interacting through membrane channels.
  - Good abstraction: partitions subsystems
    - do not want to simulate every molecule forming a membrane
    - do not want to compute the interaction of every molecule in a membrane with every other molecule in a different membrane
  - Implemented by Aviv Regev et al. at Weizmann Institute as an extension of BioSpi.
- An adaptation of Ambient Calculus
  - A process language for dynamic containers (mobile agents, distributed locations, etc.) – this is where I came in.

Pathway Diagrams

- Pathway diagrams are informal representations of biochemical processes, used by biologists
- This example was chosen because it involves several levels of biological organization: molecular, cellular, and anatomical.

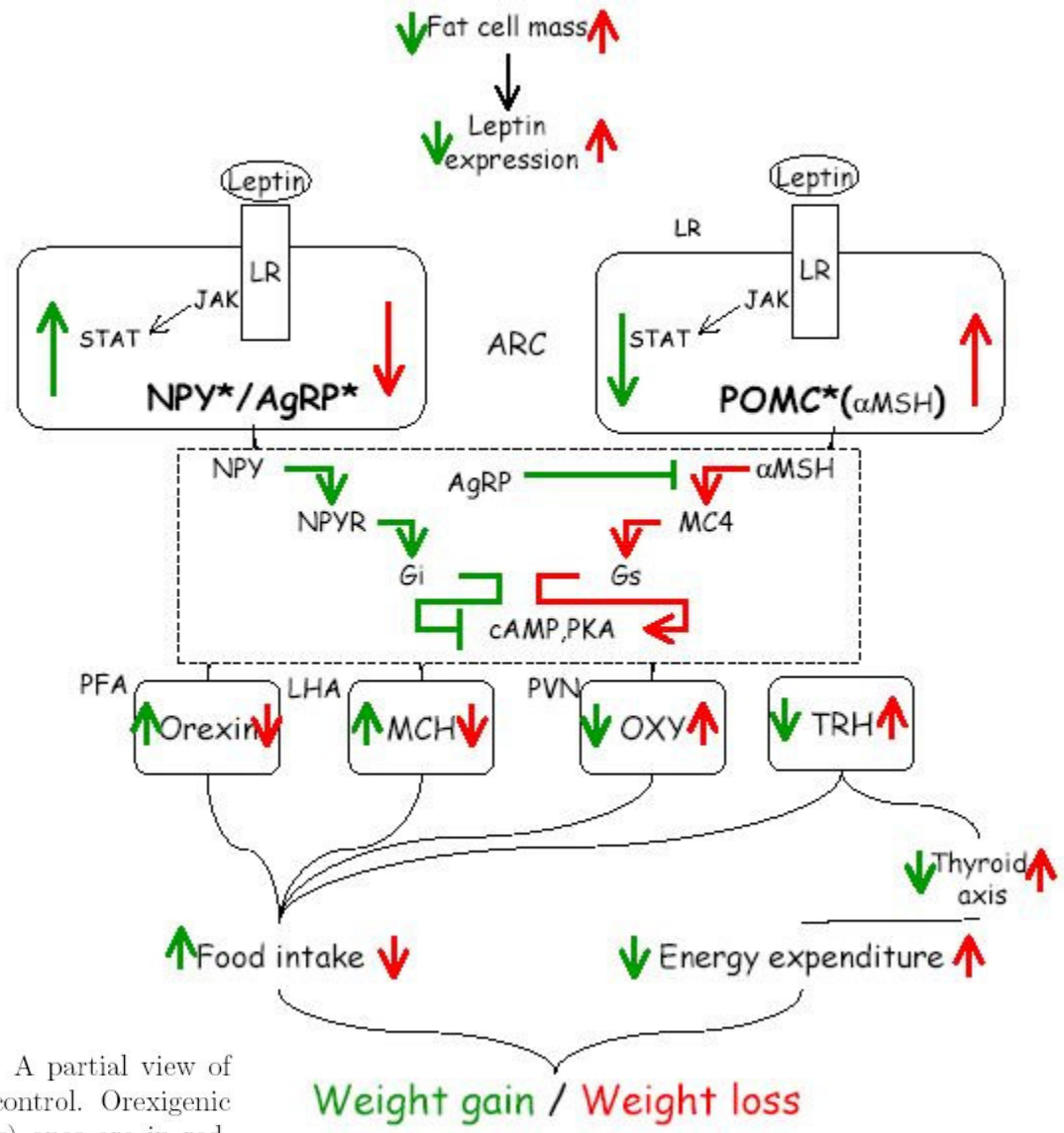


Fig. 16. **Hypothalamic pathways for weight regulation.** A partial view of molecular pathways, neurons and nuclei involved in weight control. Orexigenic (weight gaining) signals are in green, anorexigenic (weight loss) ones are in red. For further details see the main text. Adapted from [17].

BioAmbients

- Formal representation of the same system.
- (Schematic representation of a BioAmbients script, hand-drawn)

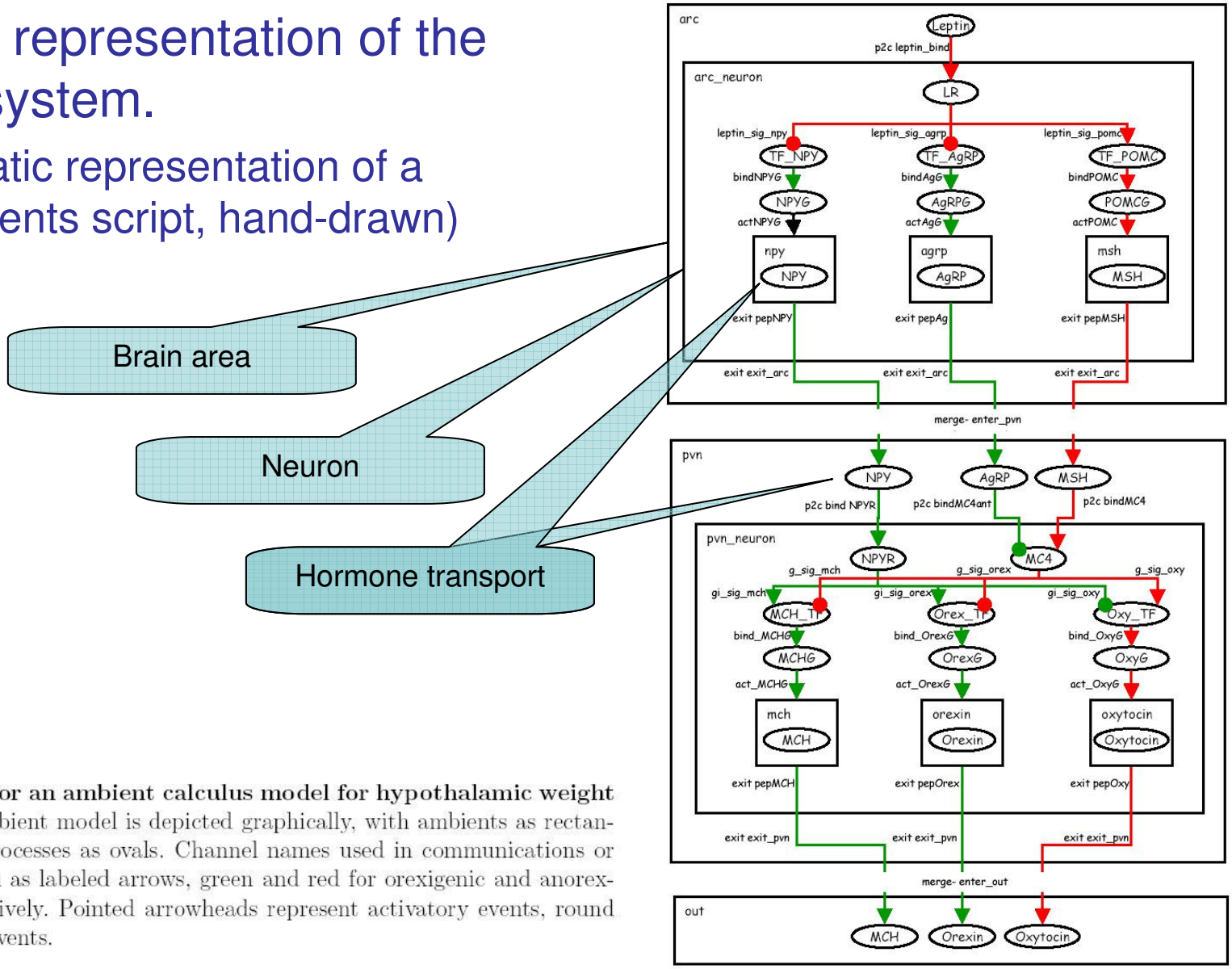
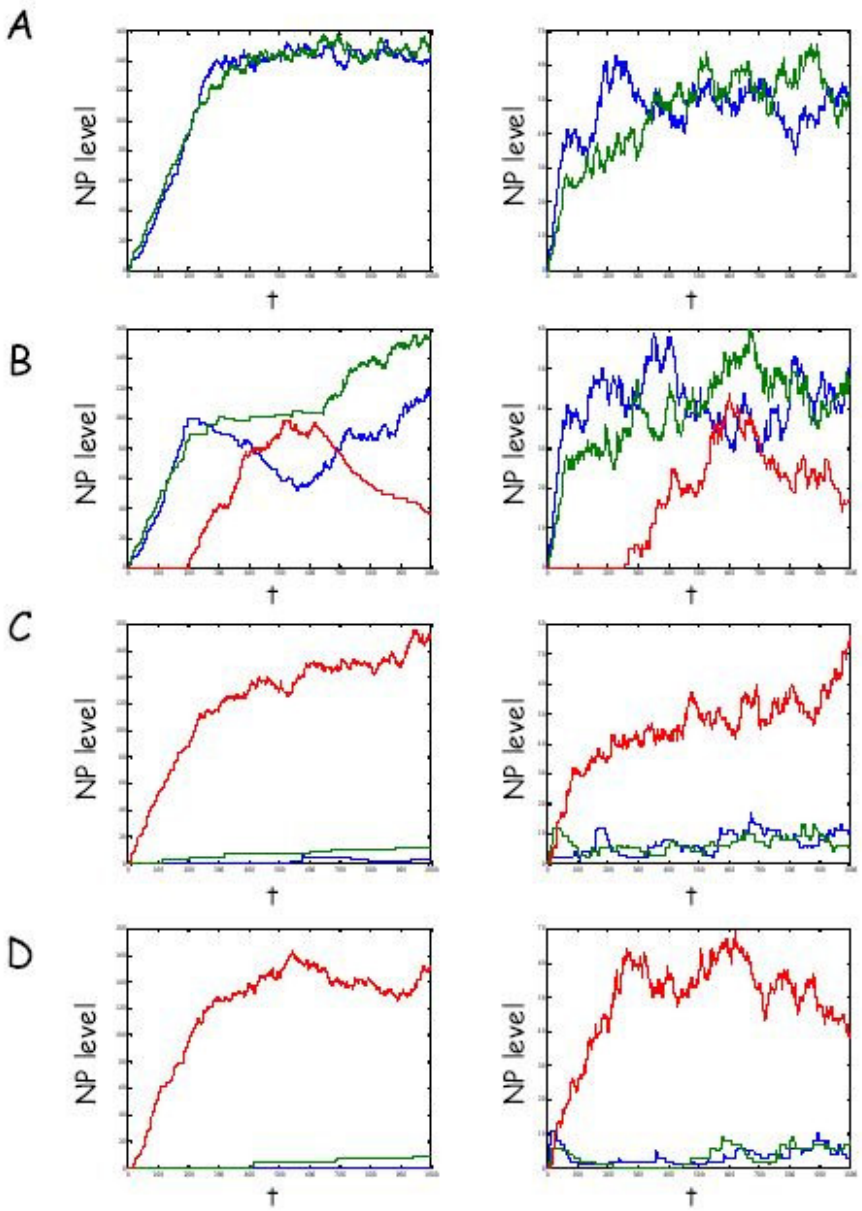


Fig. 17. **A scheme for an ambient calculus model for hypothalamic weight regulation.** The ambient model is depicted graphically, with ambients as rectangles and molecular processes as ovals. Channel names used in communications or capabilities are shown as labeled arrows, green and red for orexigenic and anorexigenic signals, respectively. Pointed arrowheads represent activatory events, round heads for inhibitory events.

BioAmbients Simulation



- Stochastic simulation (1 neuron per functional area, ~100 receptors per neuron)
- N.B.: discrete processes: thousands of components are enough, not billions.

Fig. 18. **Neuropeptide profiles under different levels of Leptin.** Simulation results of neuropeptide levels under various Leptin creation rates (A) 0.0001 (B) 0.01 (C) 1 (D) 100. In each panel first order hormones, AgRP (blue), NPY (green), and MSH (red) are shown on the right, and second order hormones, MCH (blue), Orexin (green) and Oxytocin (red) are shown on the left. The anorexigenic hormones (red) are high when leptin levels are high, while the orexigenic ones (green,blue) are high when Leptin is low, as expected.

Conclusions

- Fascinating “software” systems:
  - Behaviorally: so much of the interesting activity is above the “raw hardware” level.
  - Structurally: DNA is pure code, and very subtle.
- Familiar architectural and analytical notions.
  - Components, hierarchy, discrete (although stochastic) behavior, algorithms.
  - Control flows to discover and analyze.
    (Drugs need to modify control flows, not just molecules.)
- Amazing complexity
  - Requiring complex tools…