Agent-based modeling in Systems Biology a case study of ULS systems

Emanuela Merelli

School of Science and Technology
University of Camerino - Italy

GII Doctoral School 2010 on Ultra Scale Systems
28th September 2010

Outline

Part one	Computational Systems Biology the challenge of 21st century
Part two	Agent-based paradigm a promising technology for modelling ULS systems
Part three	Space, Geometry, Motion and Interactions future aspects
Case studies	Bone remodelling *Cardiac tissue modelling* *Cell Cycle*

Computational Systems Biology

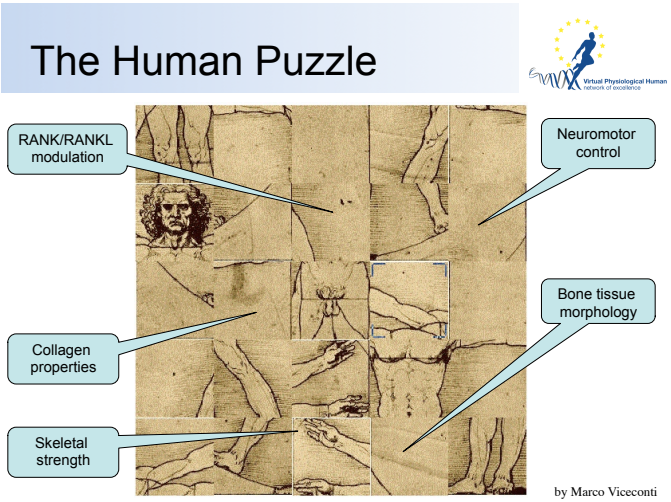
Systems Biology aims to *understand how* biological molecules in living systems **integrate** to complex systems, how these systems **function** and their **evolution**.
Computational Systems Biology is an emerging field in biological simulation that attempts to model or simulate intracellular and intercellular events using data gathered from genomic, transcriptomic, proteomic or metabolomic experiments.

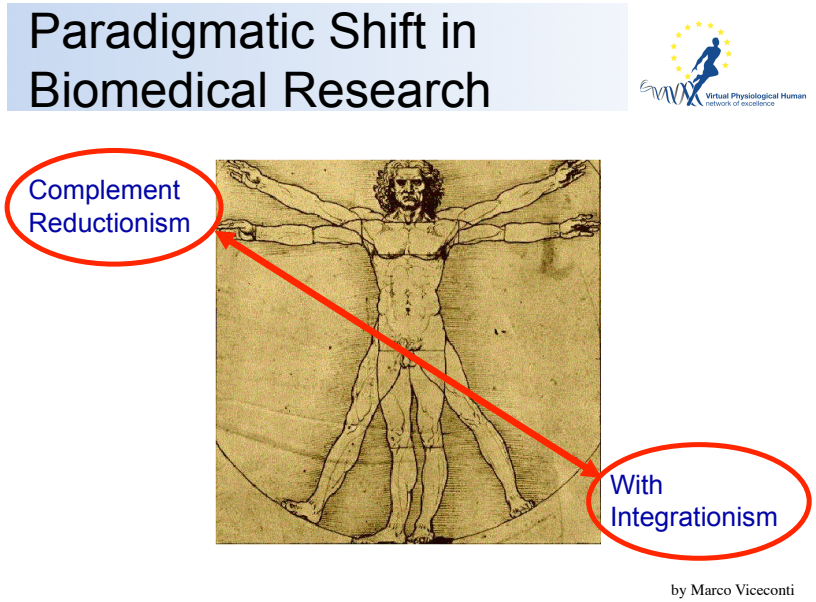
Why so important?

Models, from cells to organisms, will be used in **drugs research** for understanding their effects and side-effects

- avoiding to "torture" animals
- for building **personalized medicine**
- ...

The challenge of 21th century





VPH: Virtual Physiological Human FP7 ICT

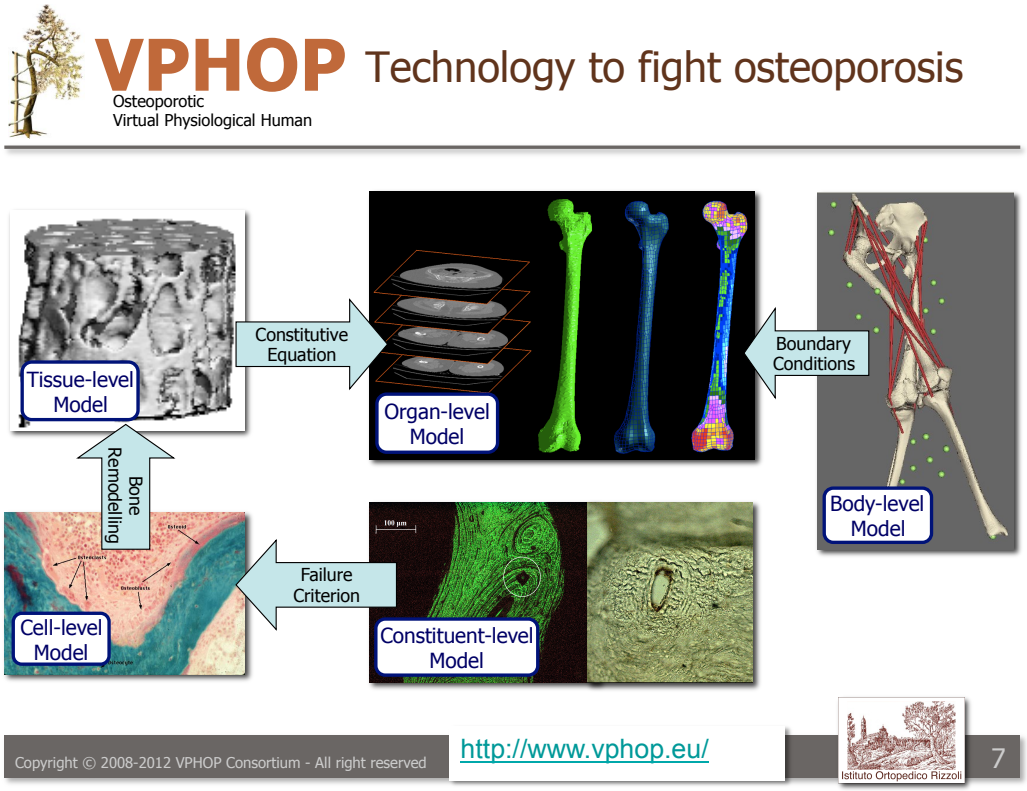
Future ICT for the personalised, predictive and integrative understanding of life and health

- *The virtual physiological human: computer simulation for integrative biomedicine,* M. Viceconti, P. Kohl, Royal Society Publishing, 2010 (free access)

- *Computational challenges of Systems Biology,* A. Finkelstein et al. IEEE Computer 2004

VPH

The vision of integrative biomedical science, shared by a number of distinctly innovative new approaches including systems biology, multiscale modeling, and the physiome, will become practically possible only when an entirely **new framework of methods and technologies** has been developed for investigating organisms as single systems.



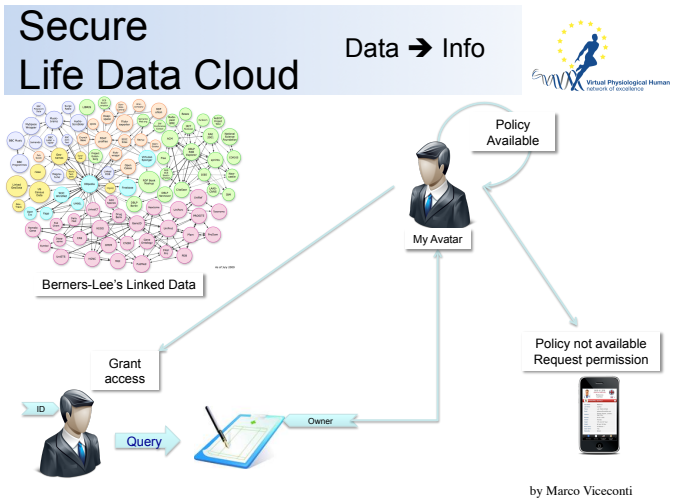
VPH Framework (1/2)

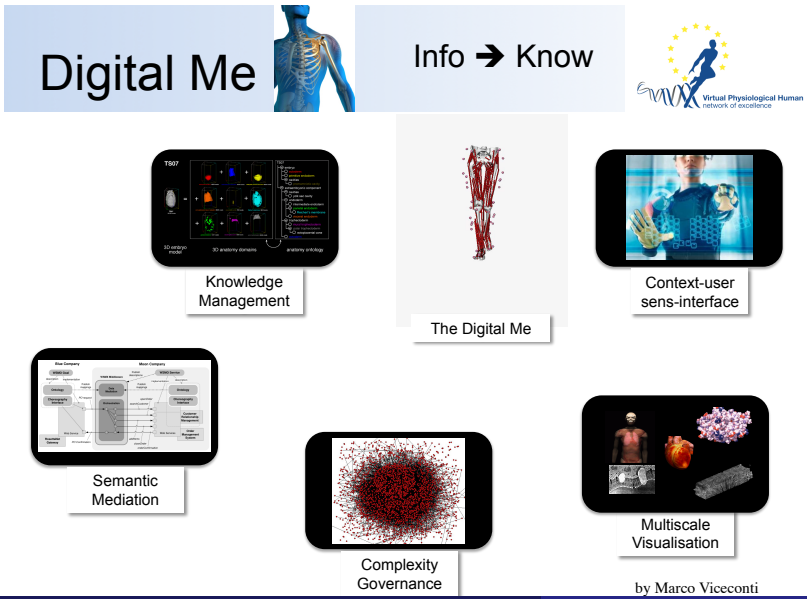
Develop radically innovative ICT solutions for harvesting the huge amount of human physiological and pathological data at all scales from molecules to whole populations for the personalized, predictive and integrative understanding of life and health

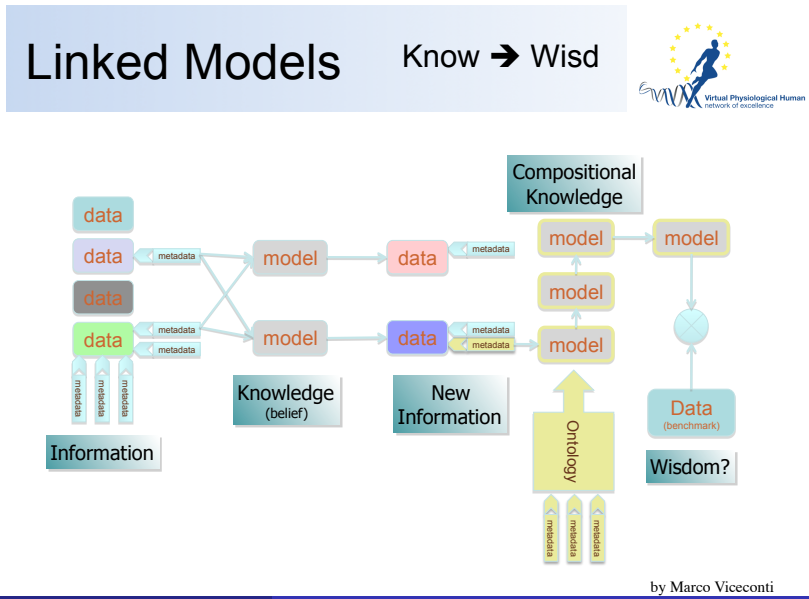
Re-think the human body as a single integrated and dynamic system (USL systems), which operates simultaneously at dramatically diverse space-time scales, from molecules to whole populations, from nanosecond to lifetime

- From data to information $\rightarrow$ Secure Life Data Cloud: all health data (clinical, personal, research, industrial) into a single cloud virtually accessible from anywhere, but under strict access control
- From information to knowledge $\rightarrow$ The Digital Me: process data into subject-specific predictive models to assist prevention, diagnosis, planning, treatment, and rehabilitation (Systems Biology)
- From knowledge to wisdom $\rightarrow$ A world wide web of predictive models: all predictive models into an integrative cloud that represents the infinitely complex system of life

VPH Framework (2/2)







VPH 2020: Double Impact

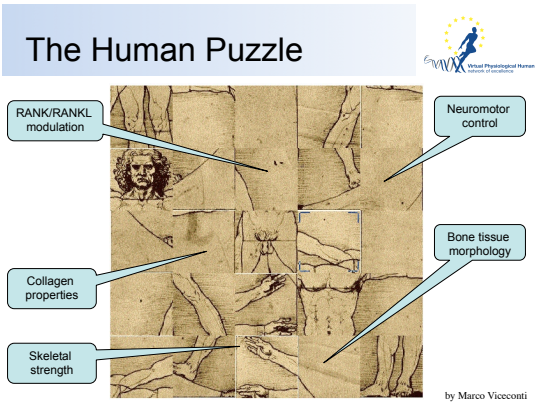
- On biomedicine
  - Providing longer and better life to EU citizens and to humanity at large
  - Improving the quality of life of the aging society
  - Leading EU industry in growth and demand for ICT innovation
- On ICT
  - VPH vision forces to think out of the box, due to the size, the complexity and the nature of the ICT problem we are facing
  - VPH can lead to radical innovations in ICT at Ultra Large Scale level
    - Secure as a banking ICT
    - Big as physics ICT
    - Complex as social sciences ICT
    - Easy to use as consumer ICT

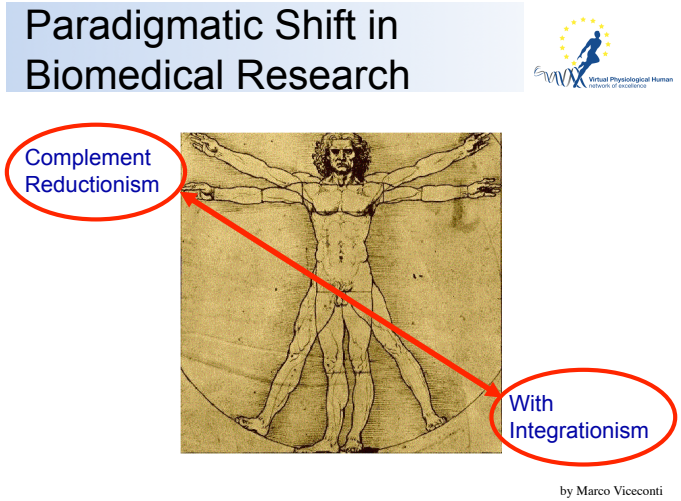
Ultra-large-scale (ULS) Systems

by SEI, Carnegie Mellon

ULS systems will be interdependent *webs* of software-intensive systems, people, policies, cultures, and economics.

- extremely large-scale and highly complex software system.
- Currently impossible to build due to limitations in the fields of software design and systems engineering.





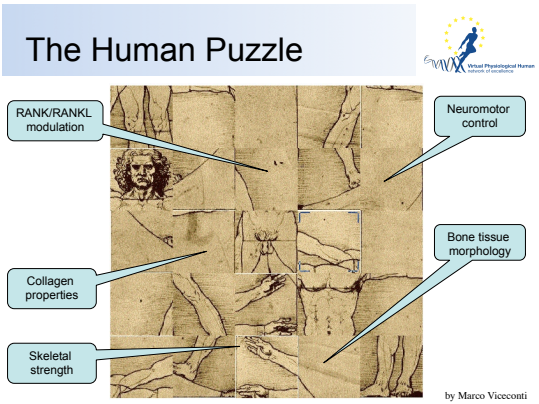
We consider VPH-systems as ULS Systems

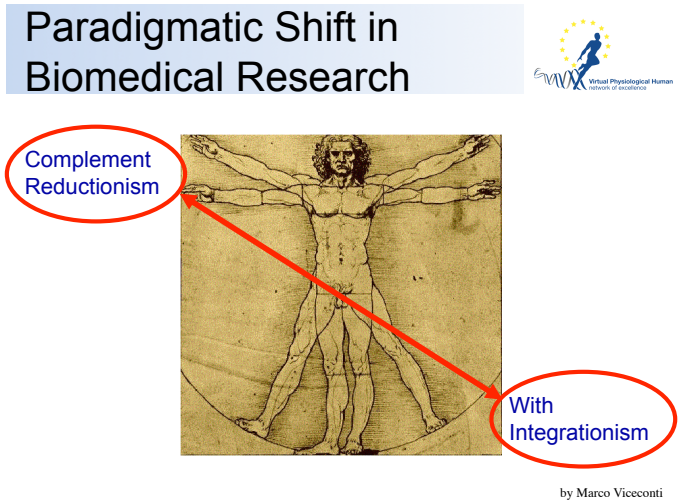
Ultra-large-scale (ULS) Systems

by SEI, Carnegie Mellon

ULS systems will be interdependent *webs* of software-intensive systems, people, policies, cultures, and economics.

- extremely large-scale and highly complex software system.
- Currently impossible to build due to limitations in the fields of software design and systems engineering.

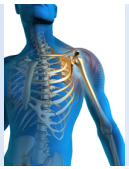




We consider VPH-systems as ULS Systems

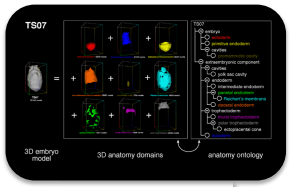
From information to knowledge → The Digital Me

Digital Me

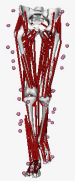


Info ➔ Know

Virtual Physiological Human
network of excellence

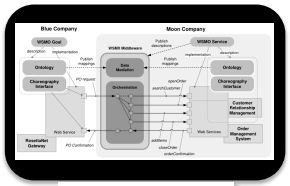


Knowledge Management

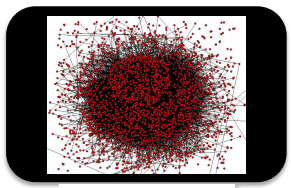


The Digital Me

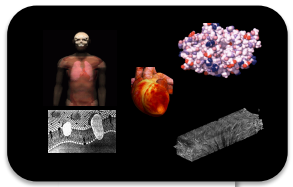
Context-user sens-interface



Semantic Mediation



Complexity Governance



Multiscale Visualisation

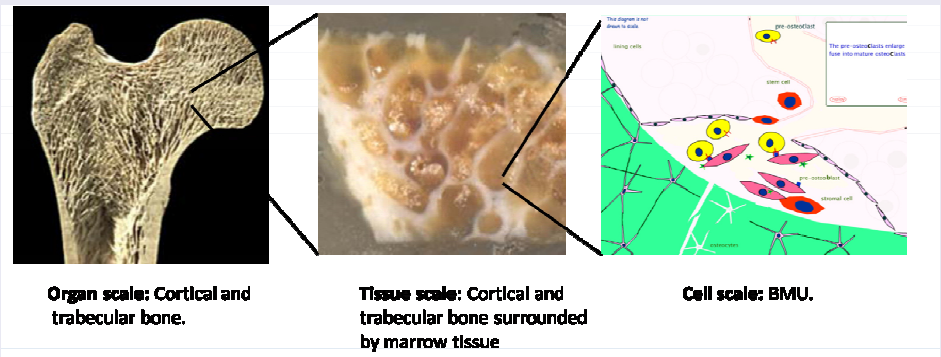
by Marco Viceconti

We need new ICT for creating predictive models

Bone Remodelling

Movies by Susan Ott - University of Washington

- normal turn over
- new BMU

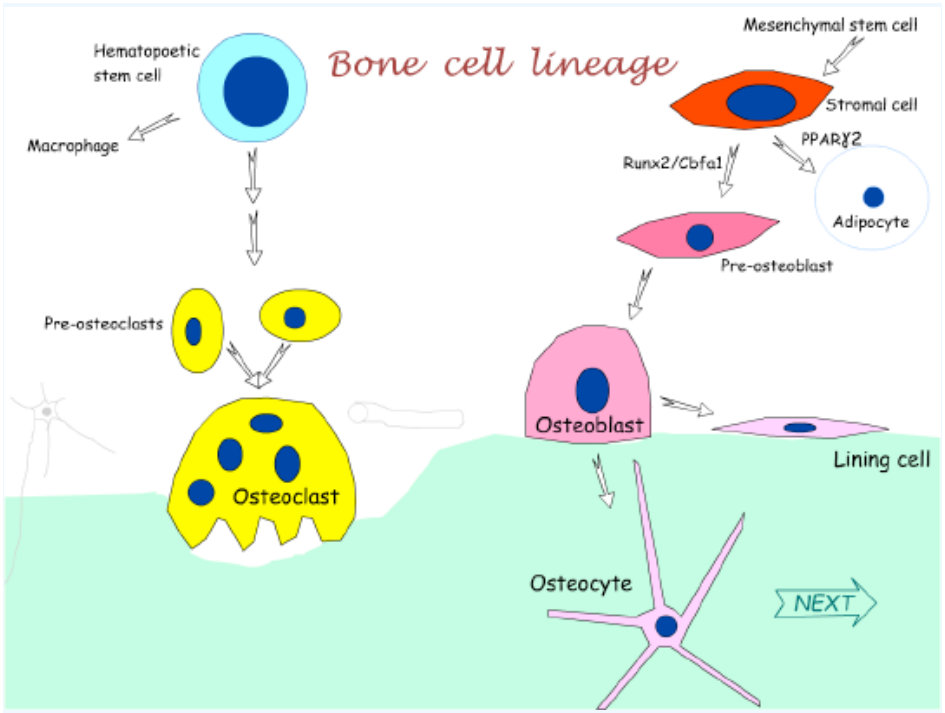


http://courses.washington.edu/bonephys/physremod.html

Note: Extrimely large and complex (ULS) system

BMU

The Basic Multicellular Unit (BMU) is a wandering team of cells that dissolves an area of the bone surface and then fills it with new bone.



Osteoporosis

Osteoporosis is a bone disease in which the amount of bone is decreased and the structural integrity of trabecular bone is impaired. Cortical bone becomes more porous and thinner. This

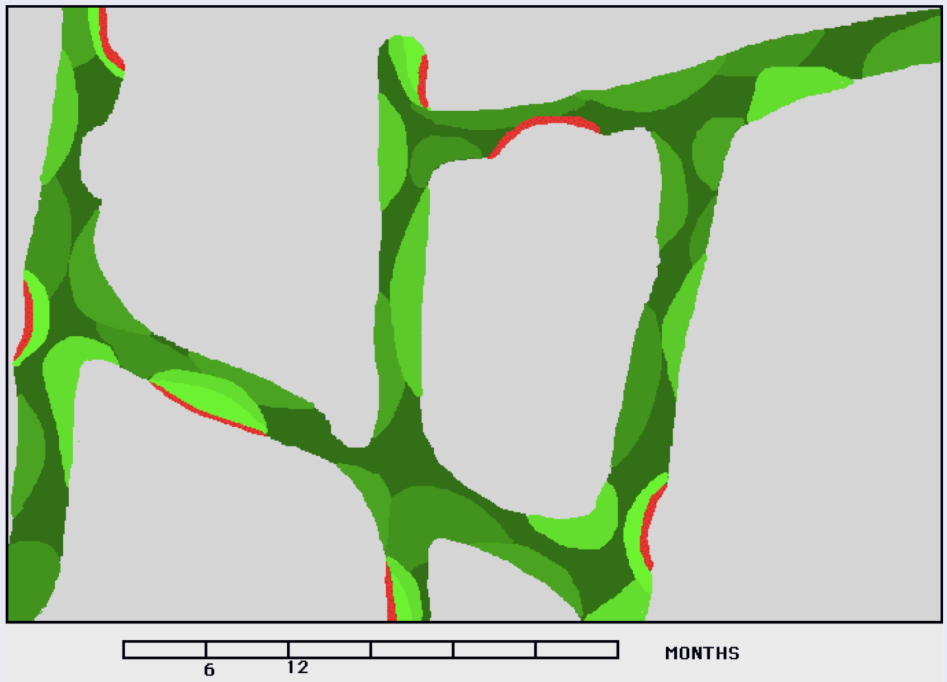
The BMU remodeling sequence

1. Origination	After microdamage to the bone, following mechanical stress, a BMU will originate. The osteocytes secrete messages to the surface cells.
2. Osteoclast recruitment	Stromal cells that have been activated by messages from osteocytes will produce M-CSF (macrophage factor) which stimulates differentiation of c
3. Resorption	The mature osteoclasts resorb bone by forming a space on the matrix surface
4. Osteoblast recruitment	Osteoblasts are derived from marrow stromal cells
5. Osteoid formation	The osteoblasts make layers of osteoid and slowly refill the cavity.
6. Mineralization	When the osteoid is about 6 microns thick, it begins to mineralize.
7. Mineral maturation	For months after the cavity has been filled with bone, the crystals of mineral are packed more closely and the density of the new bone increases.
8. Quiescence	The final osteoblasts turn into lining cells which participate in the minute-to-minute release of calcium from the bones

Bone Remodelling

Movies by Susan Ott

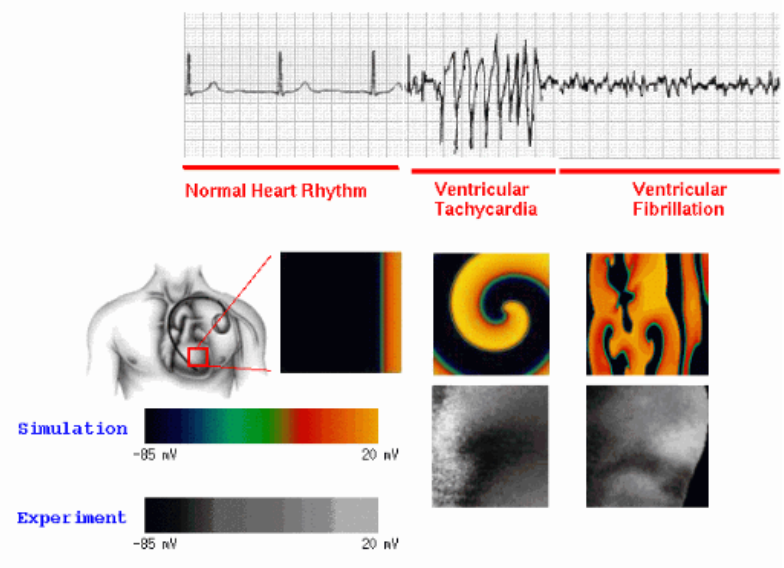
- BMU remodel
- Osteoporosis



http://courses.washington.edu/bonephys/physremod.html

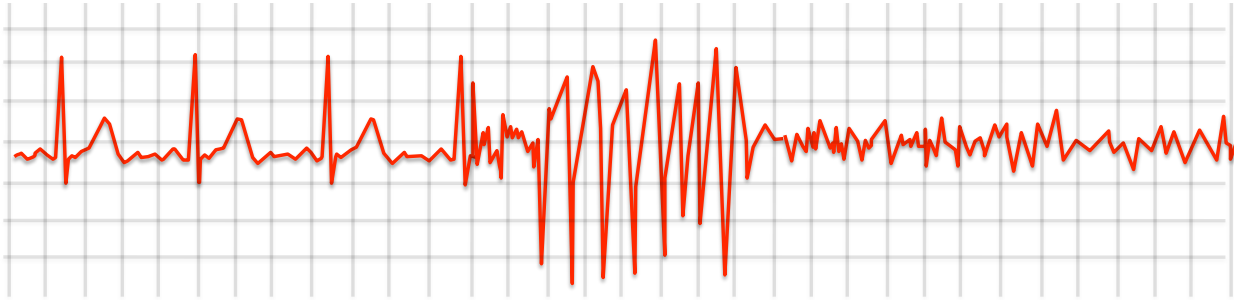
Cardiac cell

Cardiac Cells (size 10x100 microns)

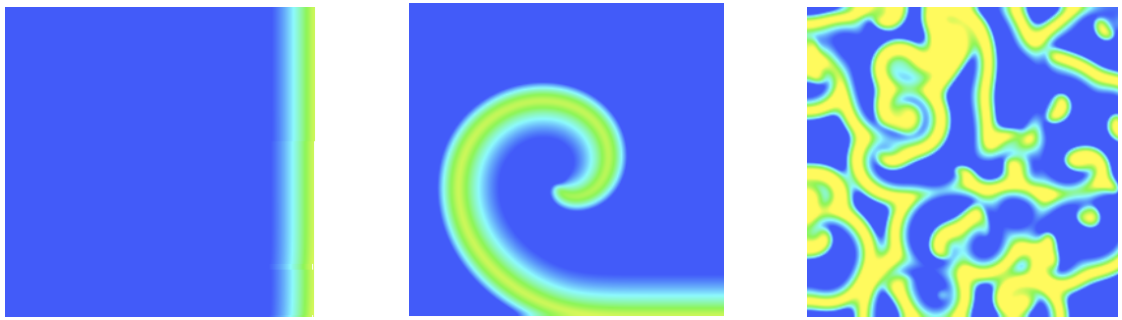


Spiral wave breakup due to APD (Action Potential Discordant) oscillations produced arithmia.
http://thevirtualheart.org/

Spiral



Normal Heart Rhythm | Ventricular Tachycardia | Ventricular Fibrillation



a) Normal Wave | b) Spiral Wave | c) Spiral Wave breakup

Cell Membrane

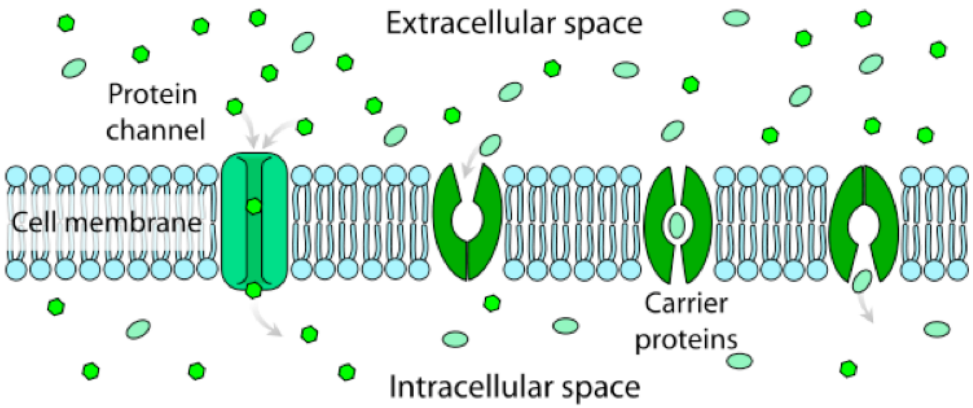


Figure 2.2: Cell membrane and passive transport using ionic channels.

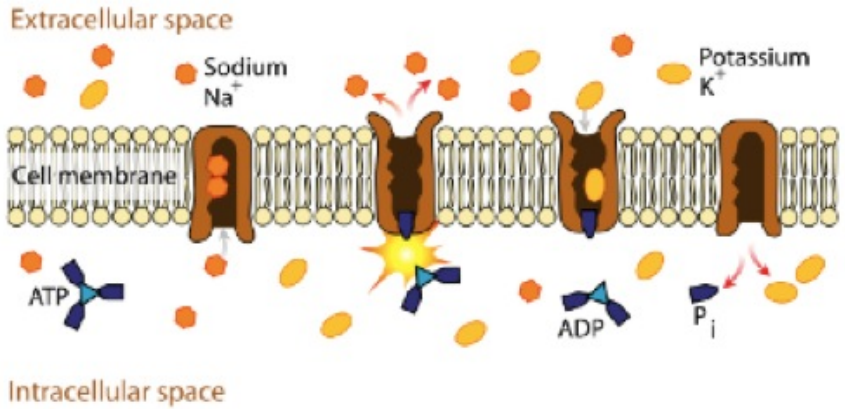
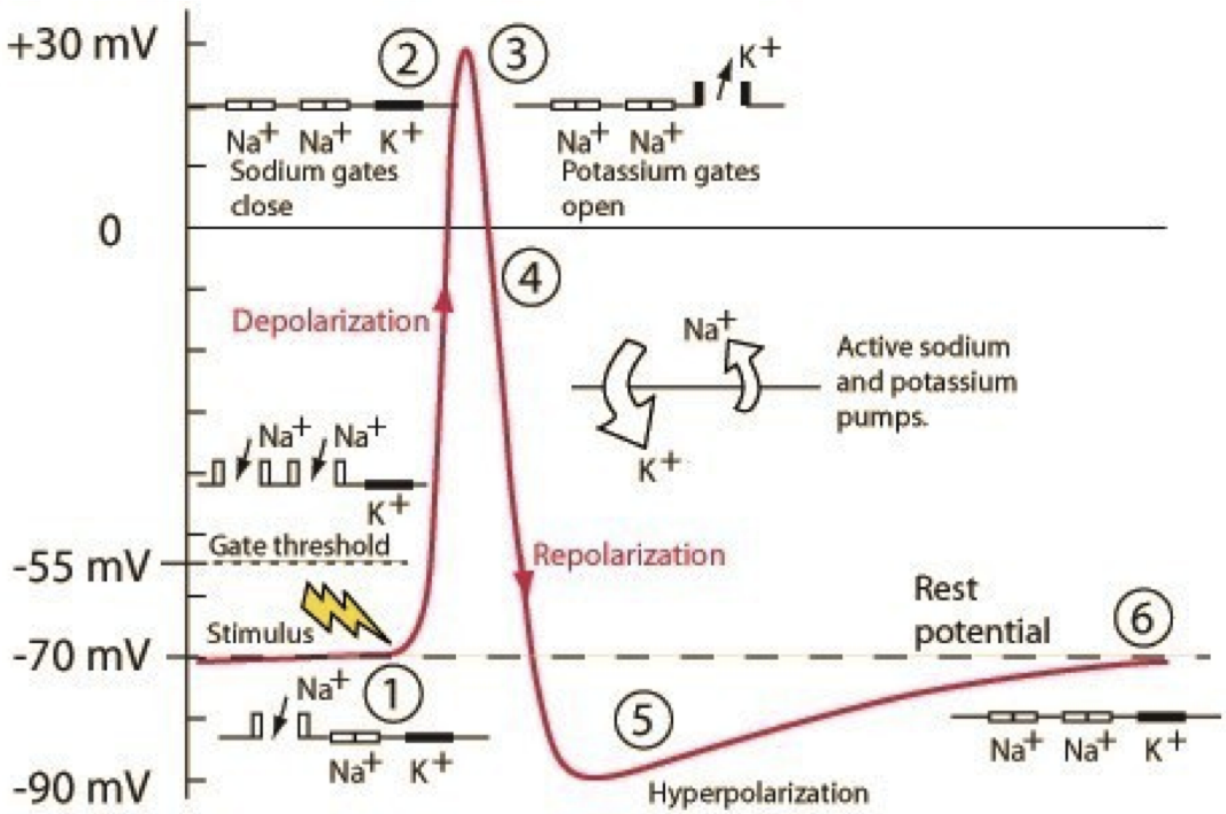
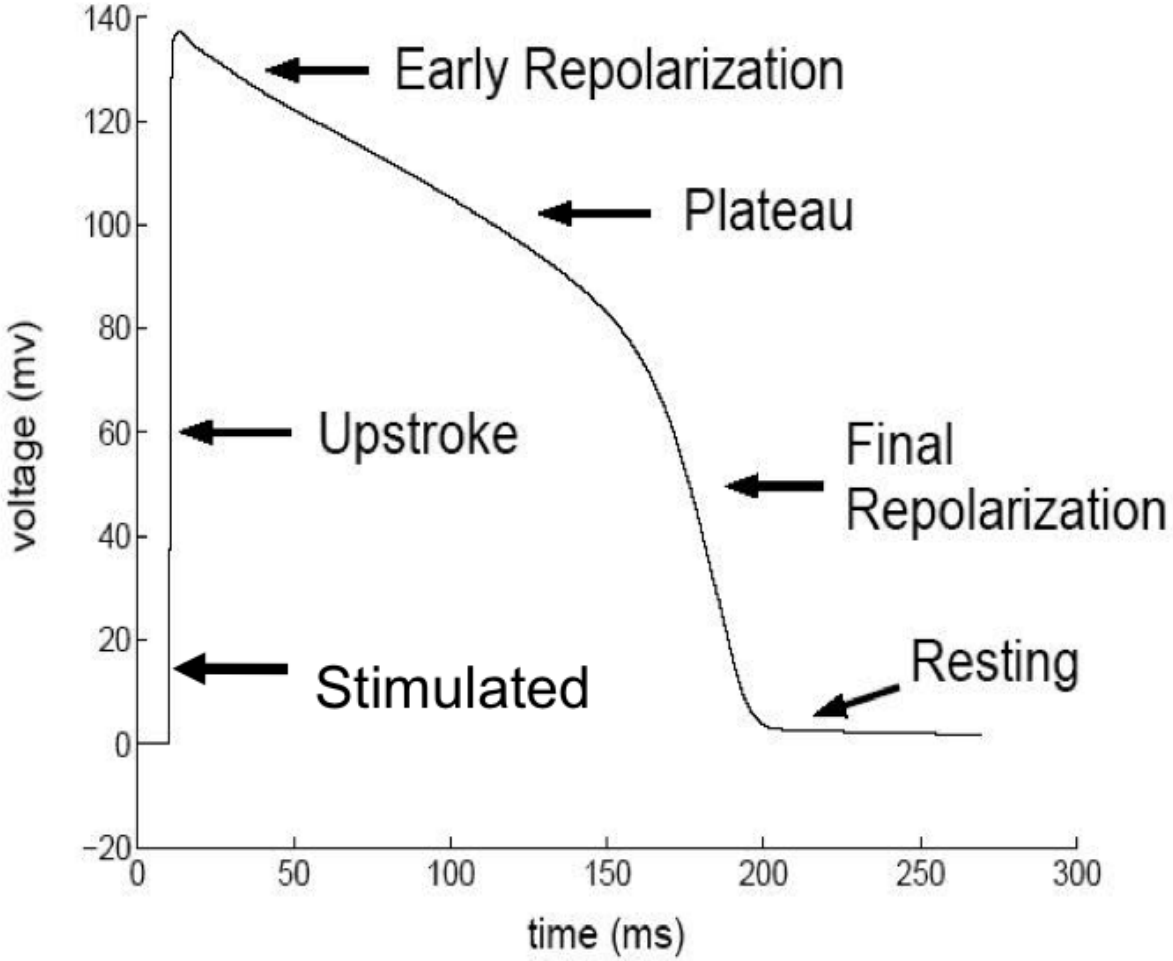


Figure 2.1: Cell membrane and active transport using ion pumps.

Action Potential

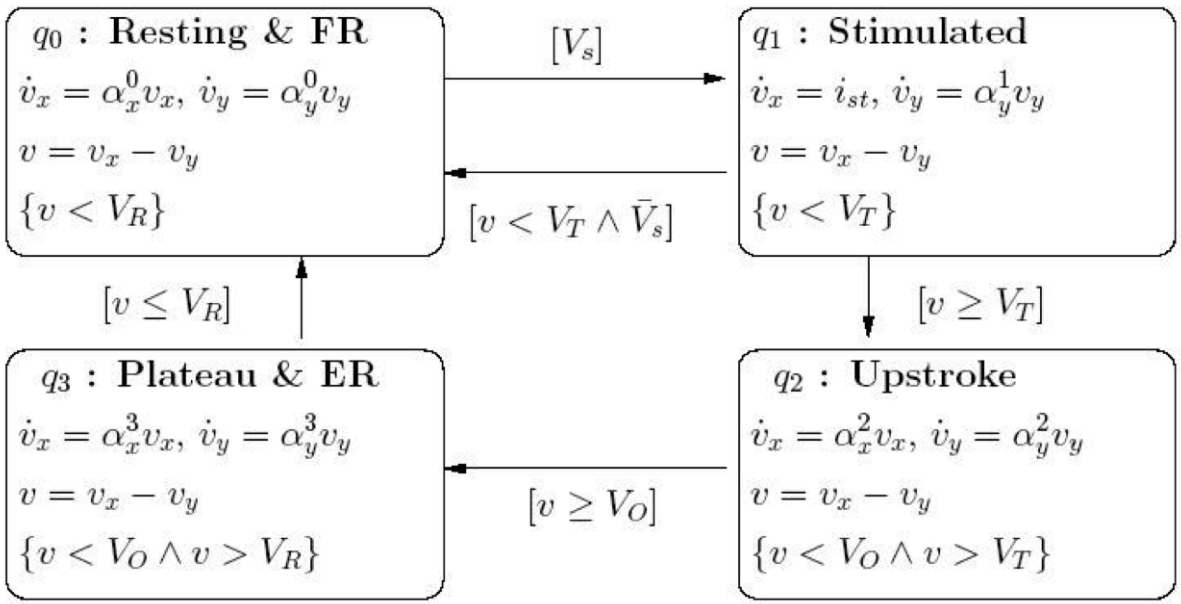


Action Potential



Hybrid Automata

- C: Capacity of the cell
- V: Transmembrane Voltage
- g_{na}, g_k, g_L: Maximum conductance of the channel
- m, n, h: Gate variable of ionic channel
- I_{st}: stimulus



Diffusion Model

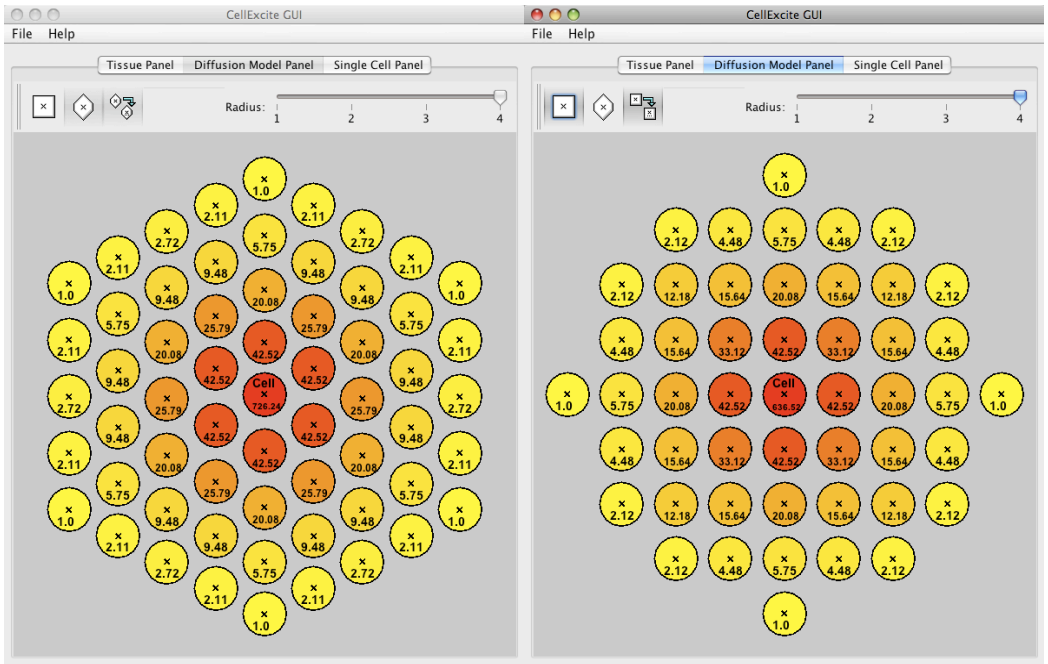


Figure 5.4: Selecting a diffusion model: on the left an exagonal diffusion schema, on the right a squarish diffusion schema.

Tissue Model

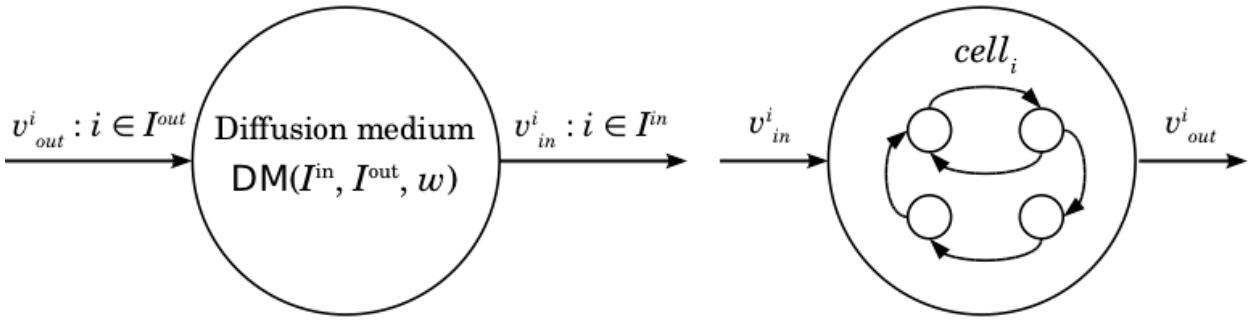


Figure 4.1: Main components of a tissue: diffusion medium and cells.

Cell Excite Architecture

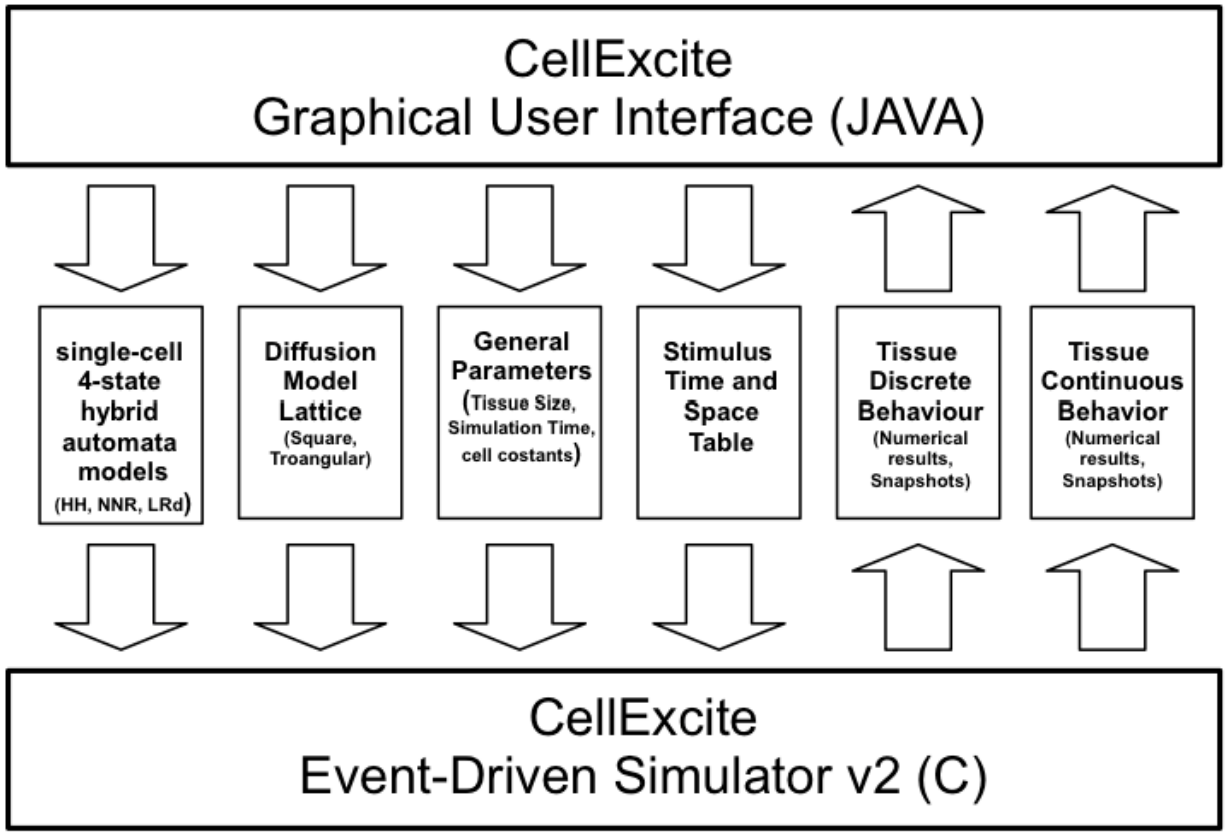


Figure 5.1: Architecture of CellExcite

Cell Panel

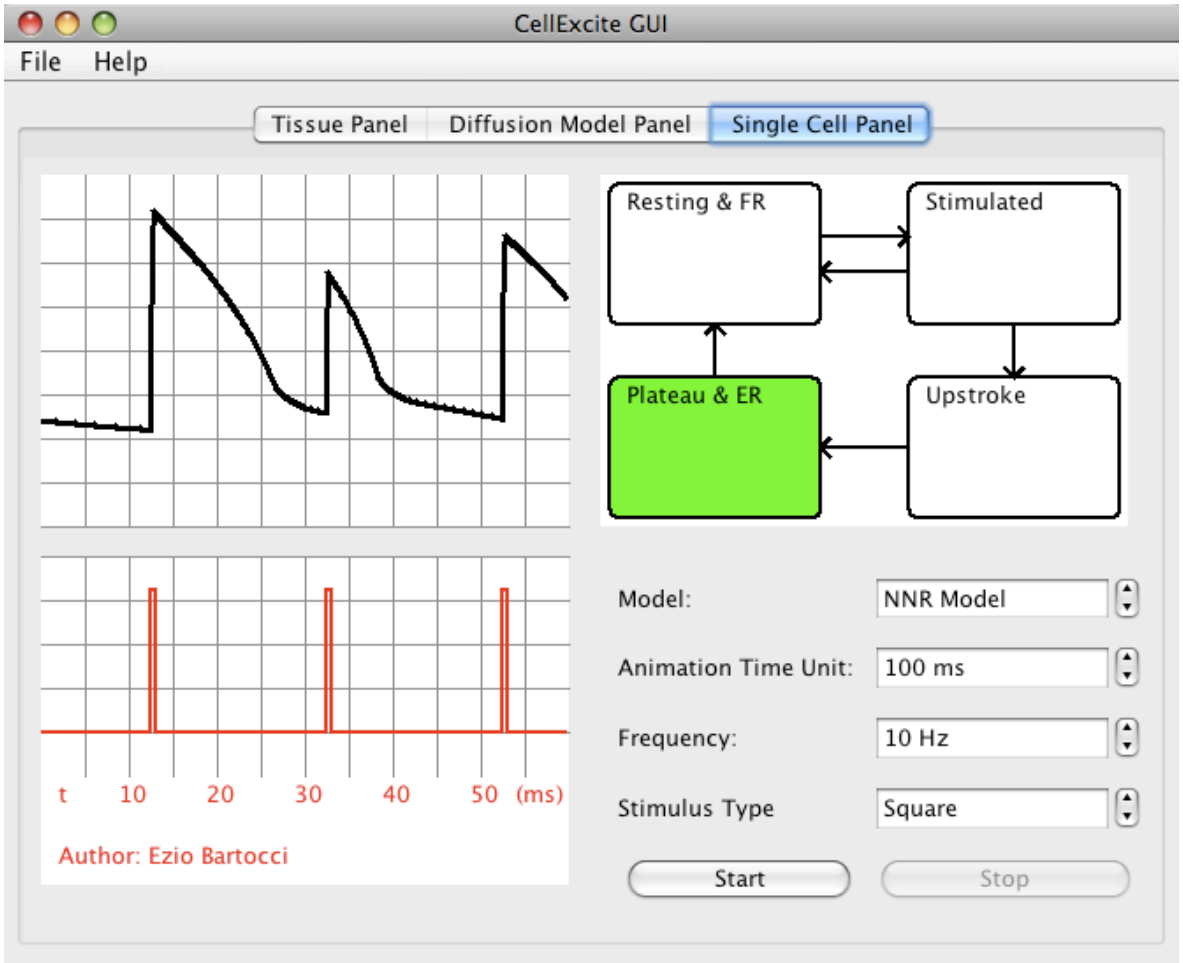


Figure 5.2: Single Cell Panel.

Tissue Panel

- Choose NNR (Neo-Natal Rat) as the 4-state HA model
- Set the distance between two neighboring cells to 0.01 cm
- Set the membrane capacitance to 1 μ F/cm
- Set the simulation time step to 0.001 msec

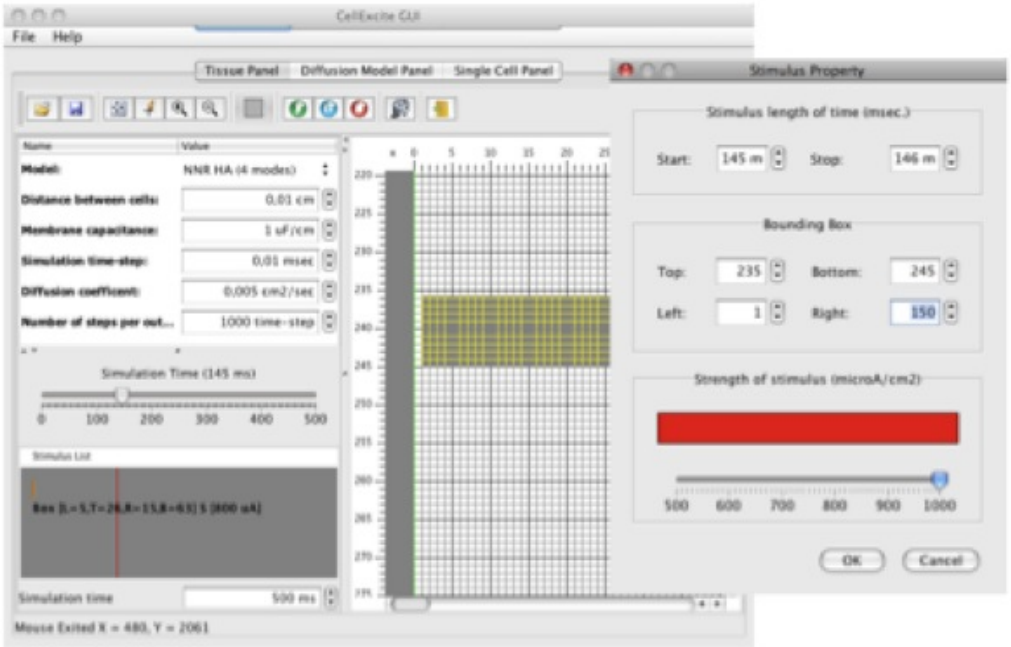
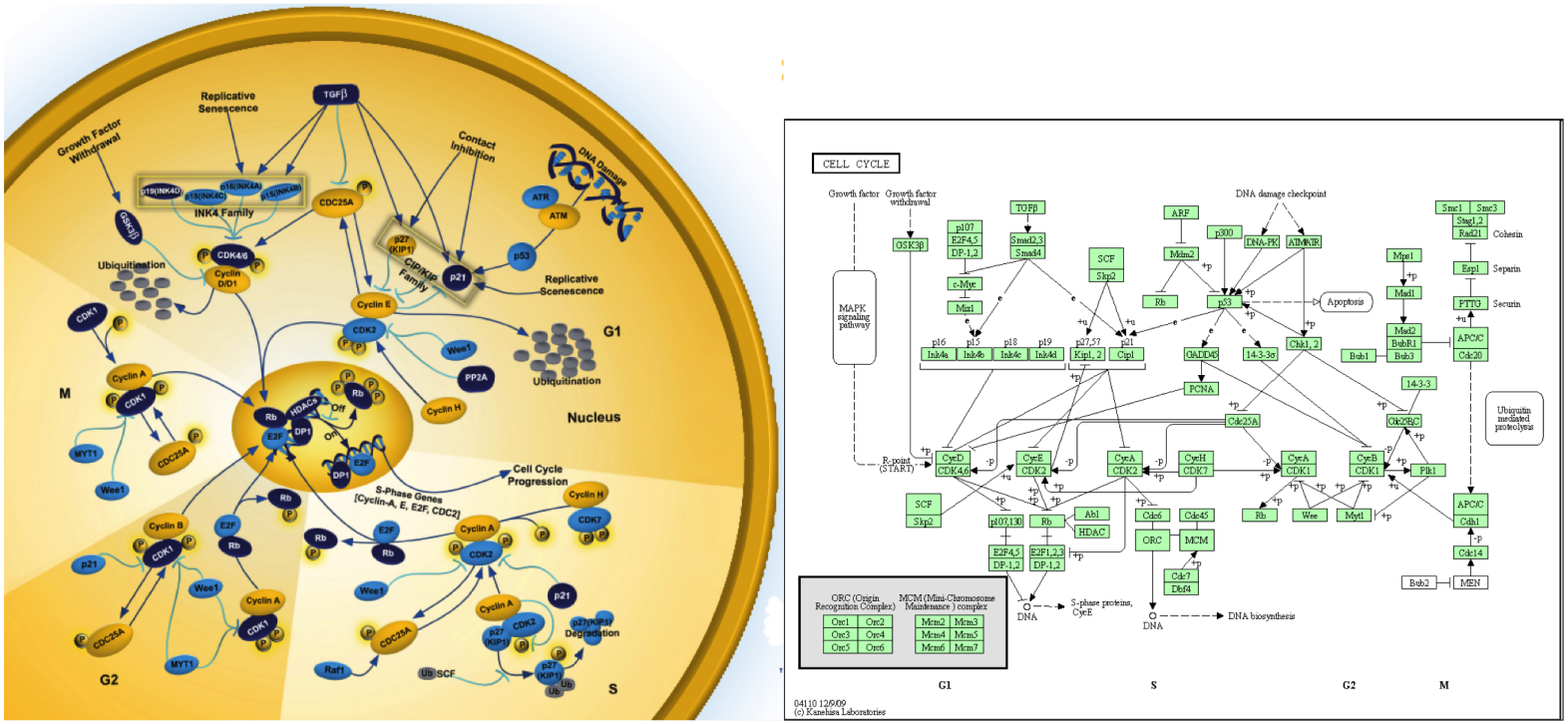


Figure 5.3: Planning a stimulus using the Tissue Panel.

Inside the Cell: Cell Cycle

The cell cycle is an ordered set of events, culminating in cell growth and division into two daughter cells



The cell cycle is a frequently investigated process to verify the impact differently regulated genes can have in normal and cancer cells

Cell Cycle regulation

Cell Cycle Regulation

- Cancer is a disease where the regulation of the cell cycle is altered leading to an uncontrolled cell division
- While normal cells will stop dividing if there is a mutation in the DNA, cancer cells will continue to divide with mutation

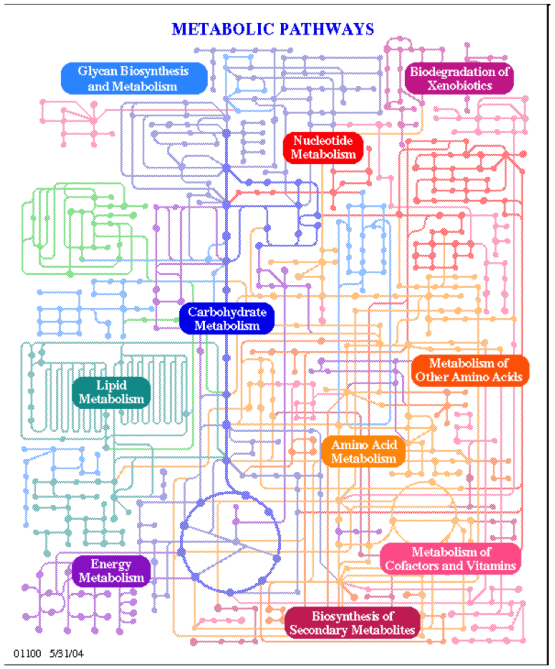
Normal Division Cell

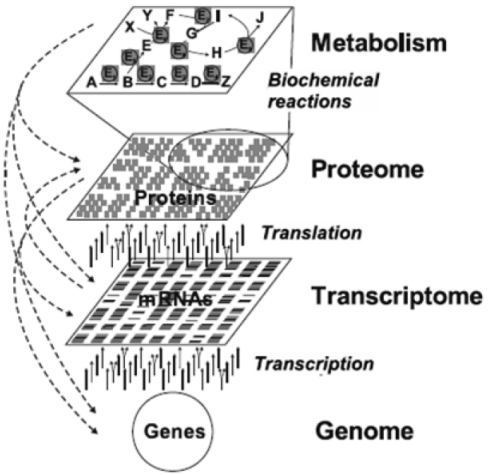
1. DNA is replicated properly
2. Chemical signals start and stop the cell cycle
3. Cells communicate with each other so they dont become overcrowded

Cancer Cells

1. Mutations occur in the DNA when it is replicated
2. Chemical signals that start and stop the cell cycle are ignored
3. Cells do not communicate with each other and tumors form

Cell Cycle Complexity





The complexity of cellular systems appears relies on:

- high number of components and interactions
- different types of control mechanisms
- sophisticated spatial organization
- multiple level organization and multiple time-scales

The organization of different cell types into tissues and organs and finally of multiple tissues and organs into higher organisms is a crucial level of complexity

Data Integration

- Data integration is an essential task to accomplish in order to achieve a view of the biological knowledge as much complete as possible in emerging fields such as bioinformatics and systems biology

- The integration of biological knowledge in the systems biology field is related to different levels, such as genomics, transcriptomics, proteomics and network interactions.

- Data integration is crucial:
  - to maintain constantly up-to-date the huge amount of data arising from experiments;
  - efficiently support the mathematical modeling

Computational Systems Biology typically requires tools to

integrate the huge amount of experimental data

understand the behaviour of biological systems

- by creating the model
- by simulating the model

discover new relationships between the various parts of a biological system, such as organelles, cells, organs, organisms

- by perturbing the model of a biological system (biologically, genetically, or chemically)

access to a huge amount of computational-power

Classical Approach

The process of modelling

- The act of describing something in a schematic representation, usually on a smaller scale (general definition)
- Design and analysis of a mathematical representation of a biological system to outline unknown properties of that system, the emergent properties (systems biology definition)

Mathematical representation of a biological process

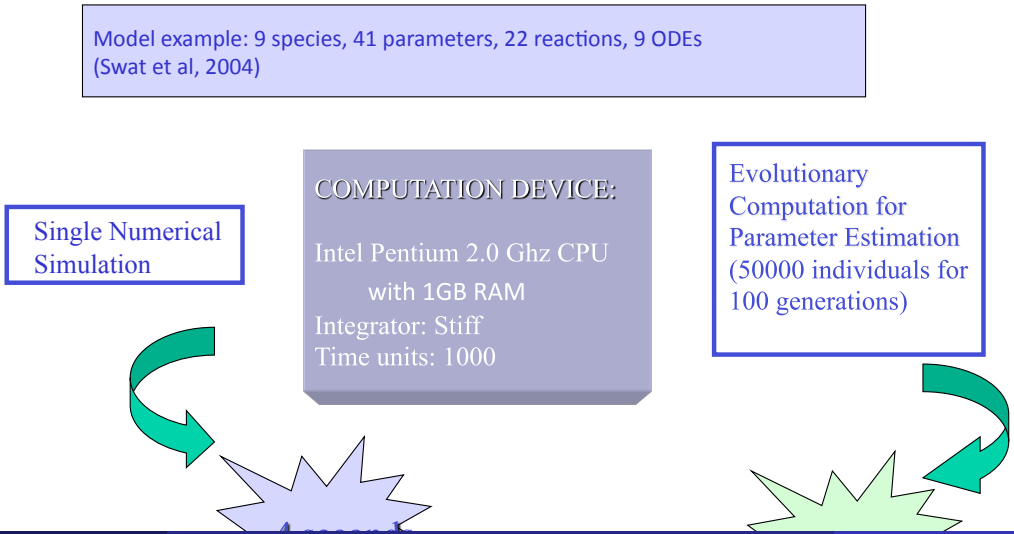
- Set of kinetic equations to define biochemical reactions
- System of Ordinary Differential Equations to describe the dynamic behaviour of the model components
- Initial parameters for kinetic equations
- Initial concentration of the model species

HPC: High Performance Computing

ODE simualtion

Simulation of an ODE system is possible on a single workstation: the numerical integration of an ODE system is not very time consuming

Parameters estimation Parameters estimation, the evaluation of the best set of parameters which define the model relating to a specific experimental dataset, requires High Performance Computing techniques, since the computational load needed in finding the best model is very great



Summary of Part One

Systems Biology - ingredients

- huge amount of data
- huge amount of metabolic pathways (the contexts)
- huge amount of different actors (agents) with different roles (behaviors)
- huge amount of spatial interactions (perception + communication)
- ...
- huge amount of unknown information (probabilities)
- ...
- highly dynamic environment (non determinism)
- many levels of abstraction (multiscale)

Systems Biology - methods

- agents interaction $\longrightarrow$ synchronous and asynchronous communication
- agent cooperation and orchestration $\longrightarrow$ distributed and centralized coordination
- non deterministic, stochastic, compositional behaviours $\longrightarrow$ process calculi
- emerging and collective behaviour $\longrightarrow$ multiscale models
- multi-agent system (MAS) verification and validation $\longrightarrow$ model checking (probabilistic, simulation-time)

Summary Part One - Systems Biology and other disciplines

- Spacial motion $\longrightarrow$ Physics
- Interaction and perception $\longrightarrow$ Chemistry
- Qualitative Analysis $\longrightarrow$ Mathematics
- Quantitative Analysis $\longrightarrow$ Computer Science
-
- Models Theory, Game Theory $\longrightarrow$ Mathematics
- Computational Models Theory and Languages Theory$\longrightarrow$ Computer Science
- Control Theory $\longrightarrow$ Engineering
-
- Multiscale model theory $\longrightarrow$?
- Proximal model theory $\longrightarrow$?

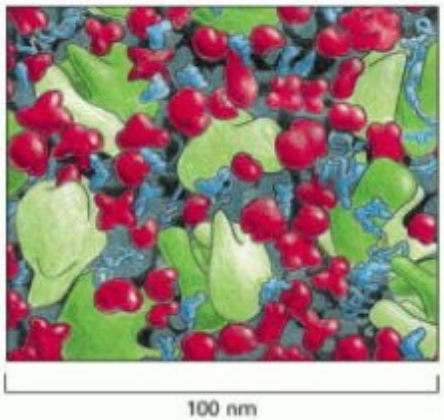
Part Two

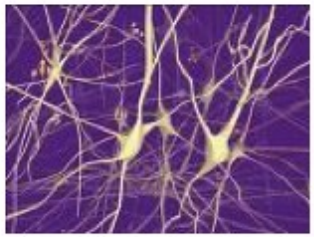
Part two

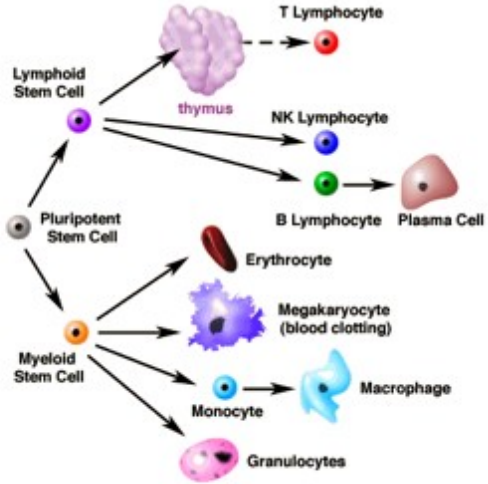
Agent-based paradigm, a promising technology for engineering ULS

- Agents paradigm
  - Reactive, Adaptive and Proactive agent
- Multiagent Systems
  - Coordination models and languages

Challenge of 21th century







Biological artifact

Biological artifacts exhibit the characteristics of **concurrent reactive systems** (Harel & Pnueli,86) on many levels: from the molecular, via the cellular, and all the way up to organs and full organisms even entire populations.

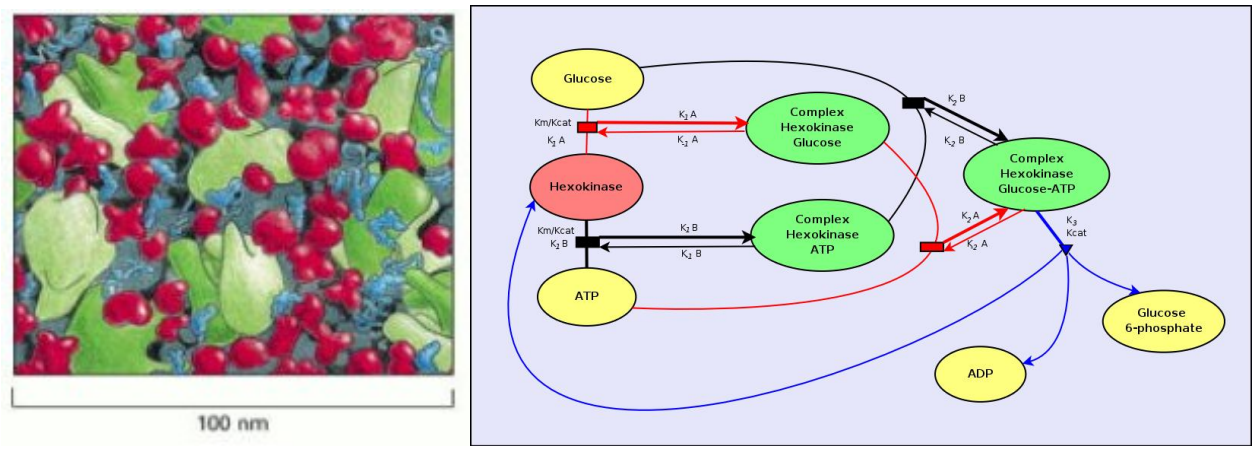
Orchestration

Orchestrated and **coordinated** concurrency of systems from Nature can be remarkably beautiful and inspiring

The **collective behaviour** of the system emerges from **local interactions** among components

- Denis Noble, *The Music of Life*, Cambridge press

A glance into the dimension of the problem



Modelling and simulation of biological systems

The simulation of biological systems requires sharing of large amount of dynamic data and gaining access big computing power.

- To model a portion (10^{15} liters) of a cytoplasm, we must simulate the movement and the interactions of about 26.883.929 autonomous entities (agents).
- Each interaction has associated 2 constants; in many cases those values are still undiscovered; but, each day one of them could be discovered by one of the hundreds of research groups distributed all over the world

Multiagent system a promising technology

Robin Milner - Turing Award 1991

In 1989 Milner wrote ... each of the several part of the system has its own identity, which persist through time; we shall call these parts agents

Recently, Luca Cardelli . . . about collective "stochastic" behaviour, said that 'a large set of interacting finite state automata

- *is not quite* **language automata** *(large set)*
- *is not quite* **cellular automata** *(interacting, but not in a grid)*
- *is not quite* **process algebra** *(collective behaviour)*
- *consider to be a multiagent system*

Multi-Agent System (MAS)

A multiagent system consists of a number of agents interacting one-another in a dynamic environment.

- agents will be acting with their different **goals and motivations**;
- agents will require **the ability to communicate, cooperate, and perceive** other agents.

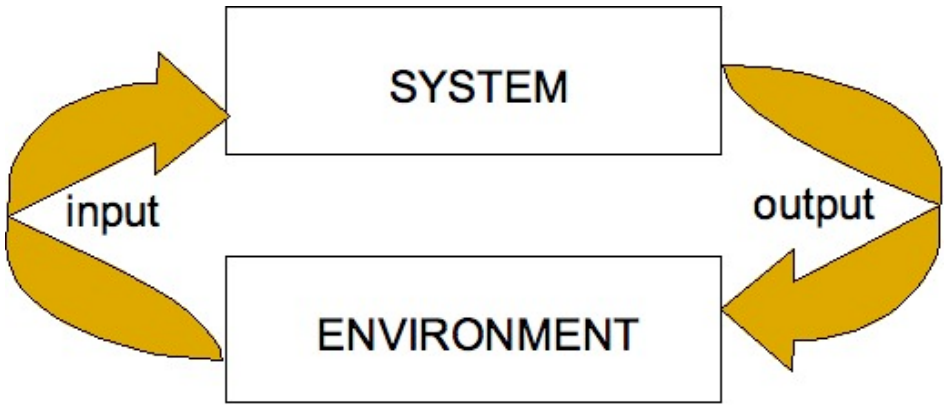
[*An Introduction to MultiAgent Systems* by Michael Wooldridge, John Wiley & Sons, 2002]

Agent definition

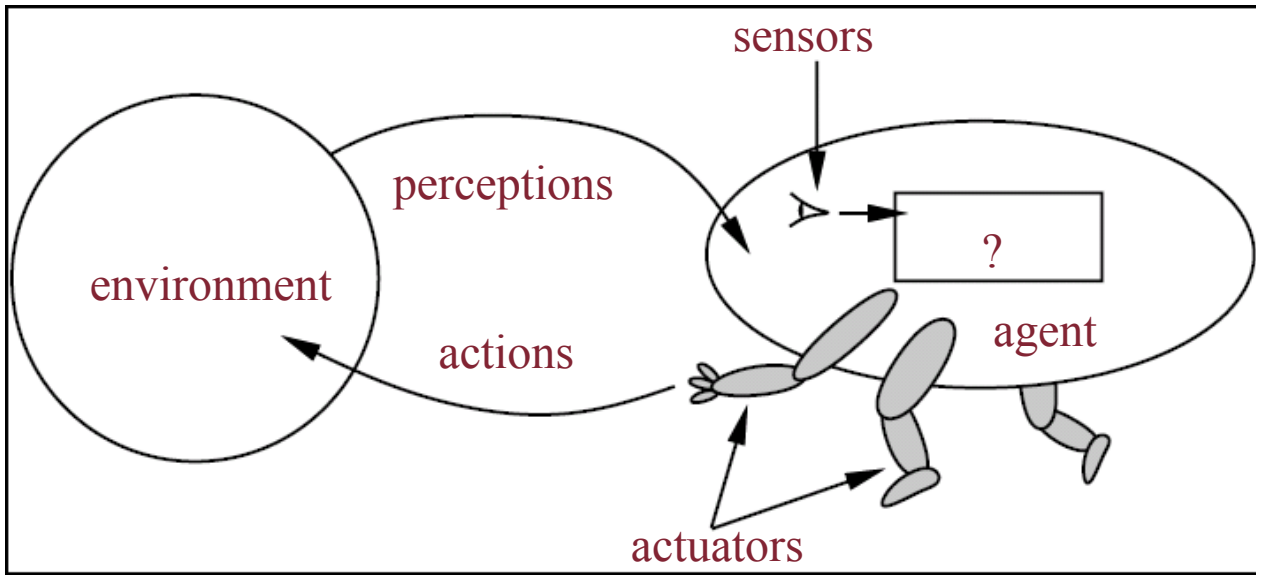
Agent

An **agent** is a computer system **situated** in some **environment** and capable of **autonomous**, **flexible** action in that environment in order to meet its design objectives

[M. Wooldridge, *Agent-based software engineering.* In IEE Proceedings of Software Engineering, 1997]



Agent as a robot

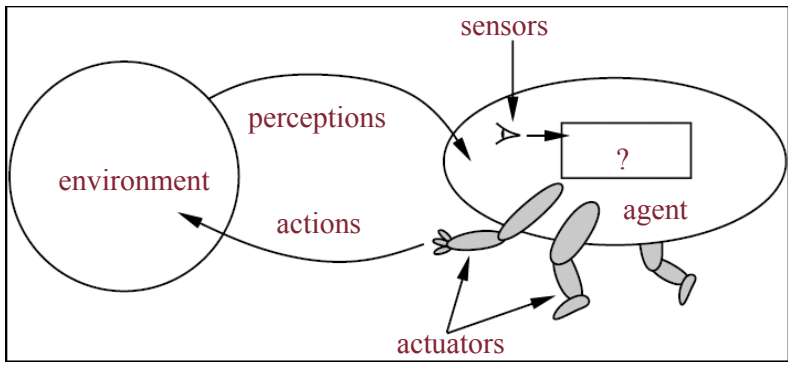


An agent is an encapsulated computer system that is situated in some open, unpredictable and typically multi-agent environment, and that is capable of flexible, autonomous (problem-solving) action in that environment in order to meet its design objectives.

Autonomous agent

An autonomous agent is a computer system capable of **autonomous** (problem-solving), flexible action, situated in dynamic, likely open, unpredictable and typically multi-agent domains.

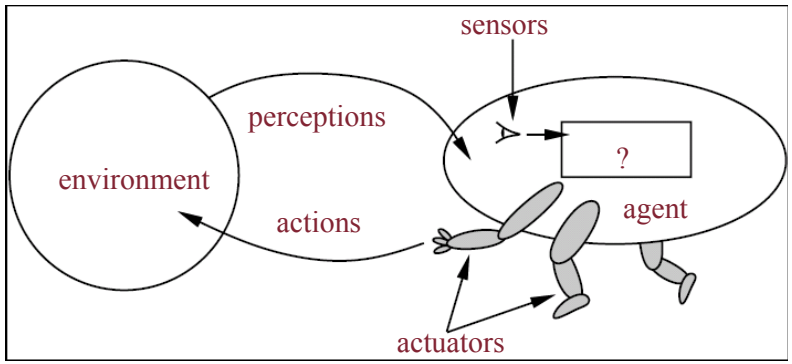
The agent has the control over its internal state and over its own behaviour



Situated agent

A situated agent is a computer system capable of flexible, autonomous (problem-solving) action, **situated** in dynamic, likely open, unpredictable and typically multiagent environment.

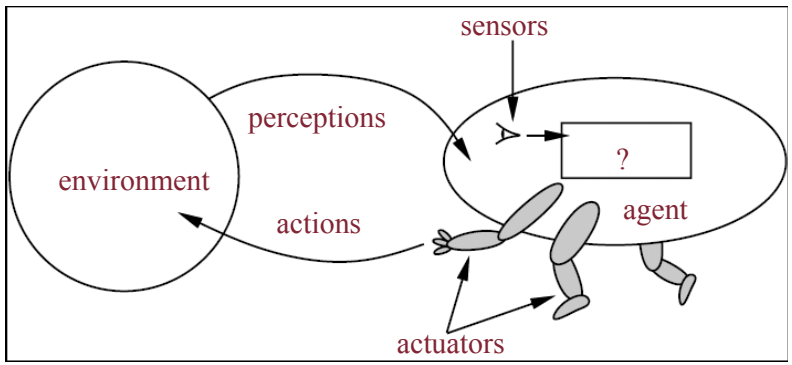
The agent perceives the environment through sensors and acts through sensors



Mobile agent

A mobile agent is a computer system capable of flexible, autonomous (problem-solving) action and **movement** from one to another distributed environment, likely open, unpredictable and typically multiagent.

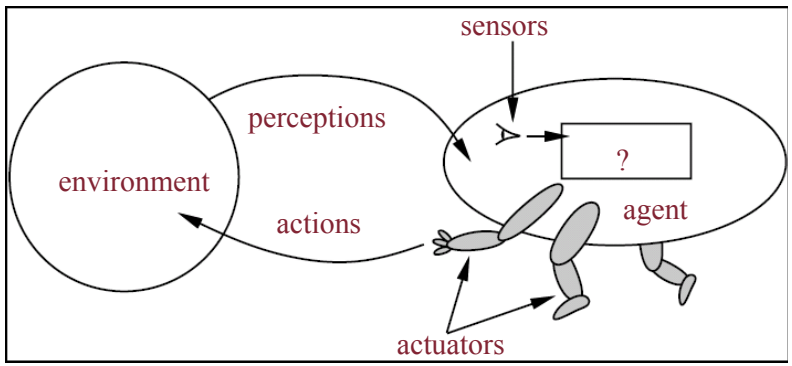
An agent moves from one environment to another through its actuators



Flexible agent

A flexible agent is computer system capable of **flexible**, autonomous (problem-solving) action, situated in dynamic, eventually open, unpredictable and typically multi-agent domains.

- reactive: the agent responds in timely fashion to environmental change
- adaptive: the agent responds to the environmental change according to its internal state
- proactive: the agent acts in anticipation of future goals



Some Formalism

Legenda

Let

- E be a finite set of discrete, instantaneous states of the environment whose elements are ranged over by $e, e', e'', \ldots$
- Ac be a finite set of actions an agent can perform, whose elements are ranged over by $\alpha, \alpha', \alpha'', \ldots$
- $\mathcal{R}$ be the set of all such possible finite sequences (runs) over E and Ac, whose elements are ranged over by $r, r', r'', \ldots$
- $\mathcal{R}^{Ac}$ be the subset of runs that end with an action $\alpha \in Ac$
- $\mathcal{R}^{E}$ be the subset of runs that end with an environment state $e \in E$
- $\mathcal{AG}$ be the set of all agents whose elements are ranged over by $Ag, Ag', Ag'' \ldots$

A Run

Run

A run, $r \in \mathcal{R}$, of an agent in an environment is a sequence of interleaved environment states and actions

$$r : e_0 \xrightarrow{\alpha_0} e_1 \xrightarrow{\alpha_1} e_2 \xrightarrow{\alpha_2} e_3 \xrightarrow{\alpha_3} \cdots \xrightarrow{\alpha_{u-1}} e_u$$

Where $\mathcal{R}$ be the set of all possible finite sequences (runs) over E and Ac

Environment

State Transformer Function

A state transformer function τ maps the set $\mathcal{R}^{Ac}$ of runs that end with an action of an agent to a set of possible environment states - those that could result from performing the action.

$$\tau : \mathcal{R}^{Ac} \rightarrow 2^{E}$$

If $\tau(r) = \emptyset$ then there are no possible successor state to r. The system has ended its run.

Environment

An environment is a 3-tuple

$$Env =< E, e_0, \tau >$$

where E is a set of states of the environment, $e_0 \in E$ is the initial state and τ is the state transformation function

Environment features

The environment is assumed to be

- history dependent
  the next state depends on the action and the current state, but the current state depends on the early action, thus it implies
- non-deterministic
  i.e. there is "uncertainty" about the result of performing an action in some states.

Agent

Agent

Agent is a function which maps runs to actions

$$Ag : \mathcal{R}^E \rightarrow Ac$$

Where $\mathcal{R}^E$ is the set of runs that end with an environment state.

Note:

1. An agent makes a decision about what action to perform based on the history of the system that it has witnessed to date.
2. While the environments are implicitly non-deterministic, agents are assumed to be deterministic.

Agent system

System

An System is a pair, (Ag, Env), of an agent function Ag and the environment Env.

We denote by $\mathcal{R}(Ag, Env)$ the set of runs of an agent Ag in environment Env.

Note: We assume that $\mathcal{R}(Ag, Env)$ contains only runs that terminate.

Agent Run

Formally, a sequence $r \in \mathcal{R}(Ag, Env)$ represents a run of an agent Ag in environment $Env = (E, e_0, \tau)$ if

1. e_0 is the initial state of Env
2. $\alpha_0 = Ag(e_0)$ and
3. for $u > 0$

$$e_u \in \tau((e_0, \alpha_0, e_1 \ldots \alpha_{u-1}))$$
$$\alpha_u = Ag((e_0, \alpha_0, e_1 \ldots e_u))$$

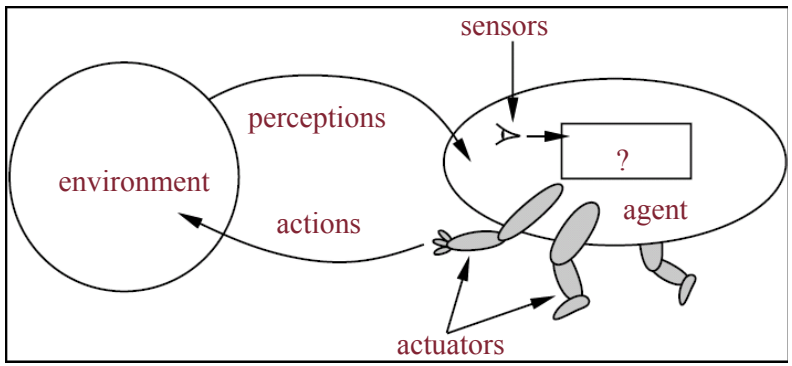
Where
$\tau : \mathcal{R}^{Ac} \rightarrow 2^E$
and
$Ag : \mathcal{R}^E \rightarrow Ac$

Autonomus, situated, flexible and mobile agent

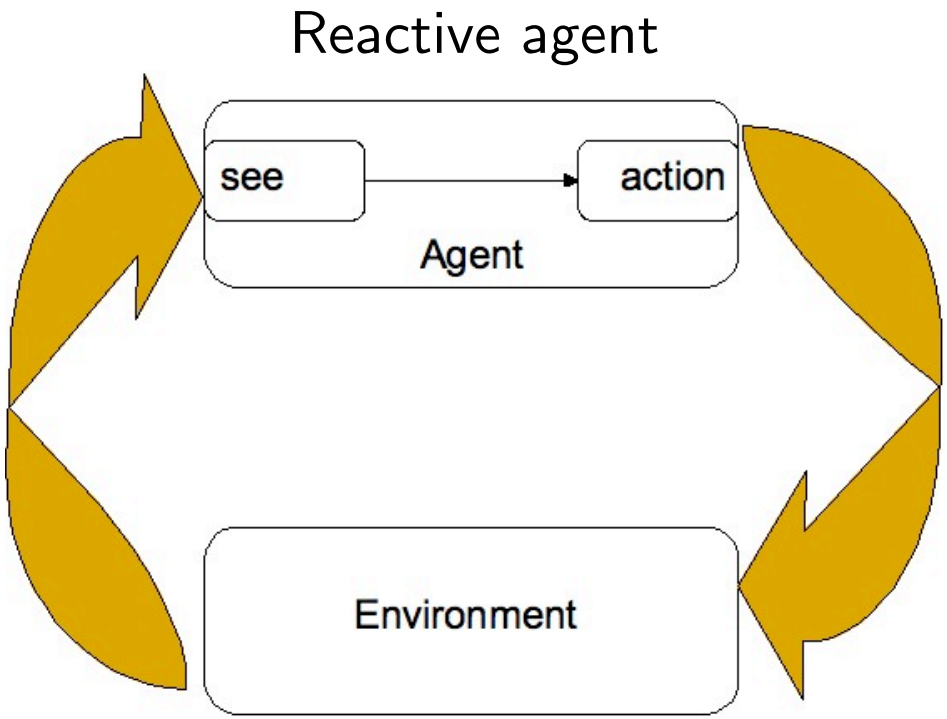
reactive: the agent responds in timely fashion to environmental change without reference to their history

adaptive: the agent responds to the environmental change according to its internal state

proactive: the agent acts in anticipation of future goals



Reactive Agent - thermostat



A thermostat is a purely reactive agent

$$Action(e) = \{ \begin{array}{l} \text{off if e=temperature OK} \\ \text{on otherwise} \end{array}$$

Perception

The *see* function is the agent ability to observe its environment, whereas the *action* function represents the agents decision making process and *do* function acts on the environment.
Output of the *see* function is a percept:

$$see : E \rightarrow Per$$

which maps environment states to "percepts". Per is a (non-empty) set of perception.
action is a function

$$action : Per \rightarrow Ac$$

which maps sequences of percepts to actions.
do is a function which

$$do : Ac \times Per \rightarrow E$$

Reactive Agent

Reactive agents can be defined by a 6-tuple

$$< E, Per, Ac, see, do, action >$$

where

- E is the set of all states for the environment
- Per is a partition of E (representing the perception of the environment from the agent's point of view)
- Ac is a set of actions
- $see : E \rightarrow Per$
- $action : Per \rightarrow Ac$
- $do : Ac \times E \rightarrow E$

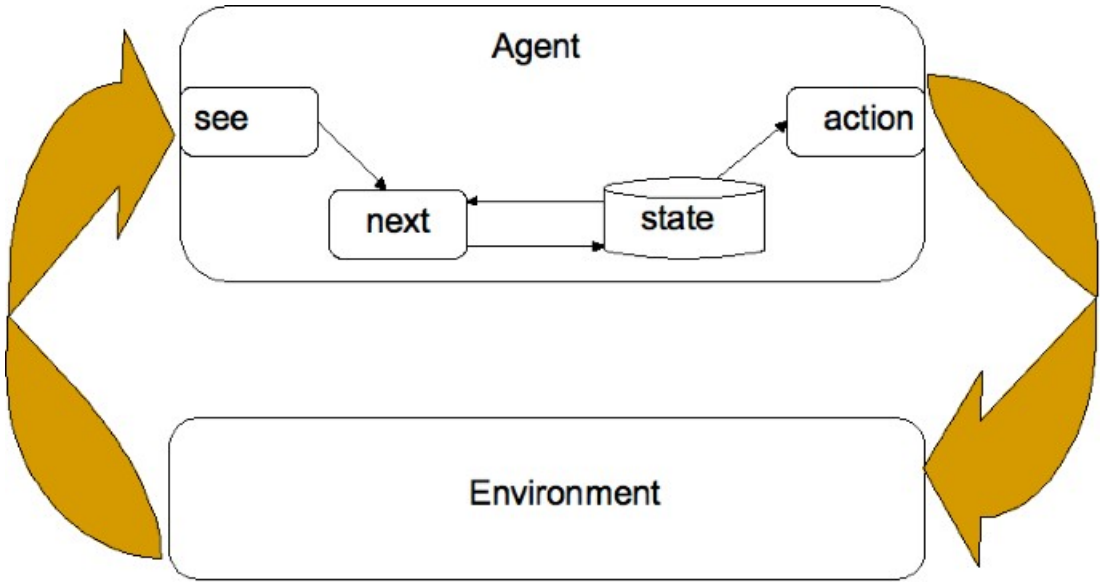
These agents observe the environment (see), find the appropriate action ($action$), and act (do) on the environment.$action$ function is a specialization of Ag function.

Questions

1. can a reactive agent be considered as an ordinary Markov process?
2. can agent be described by a Finite State Automaton (FSA)?

Adaptive Agent

Adaptive/proactive agent



Thus, a **simple adaptive agent** uses its memory to store observations of the state of the system and then it uses these observations to change its behavior.

Agent with a state

Adaptive

agents have some internal data structure, which is typically used to record information about the environment state and history.
Let I be the set of all internal states of the agent.
The perception function *see* for a state-based agent is unchanged:

$$see : E \rightarrow Per$$

The *action*-selection function action is now defined as a mapping

$$action : I \times Per \rightarrow Ac$$

from internal states to actions. An additional function *next* is introduced, which maps an internal state and percept to an internal state

$$next : I \times Per \rightarrow I$$

Adaptive Agent

Adaptive agents can be defined by a 9-tuple

$$< I, E, P, Ac, i_0, see, next, do, action >$$

where

- I is a set of internal states
- E, Per, Ac, see and do are the same as for reactive agents
- i_0 is the initial internal state
- $action : I \times Per \rightarrow Ac$
- $next : I \times Per \rightarrow I$

These agents observe the environment (see), choose their next action according to their internal state ($action$), act (do), and update their internal state (next).

Questions

1. can an **adaptive agent** be described as a push down automaton (PDA)?
2. can he be described as an Hidden Markov Model?

Knowledge

The knowledge of the agent is store in KB as set of predicate calculus (knowledge base)

$$see : E \rightarrow Per$$

The *action*-selection function action is now defined as a mapping

$$action : KB \times Per \rightarrow Ac$$

An additional function *update* is introduced, which updates the knowledge base of the agent

$$update : KB \times Per \rightarrow KB$$

Proactive Agent

Proactive agents can be defined by a 9-tuple

$$< KB, E, Per, Ac, d_0, see, update, do, action >$$

where

- KB is a set of predicate calculus (knowledge base)
- E, Per, Ac, see and do are the same as for reactive agents
- d_0 is the initial predicate
- $action : KB \times Per \rightarrow Ac$
- $update : KB \times Per \rightarrow KB$

These agents predicate observe the environmental (*see*), choose their next action according to some kind of **reasoning** on their internal knowledge base (*action*), act (*do*), and update their knowledge base (*update*).

Agent control loop

1. Agent starts in some initial internal state i_0
2. Observes its environment state e, and generates a percept $see(e)$
3. Internal state of the agent is then updated via *next* function, becoming $next(i_0, see(e))$
4. The action selected by the agent is $action(next(i_0, see(e)))$. This action is then performed.
5. Goto (2)

Task

- We build agents in order to carry out tasks.
- The task must be explicitly specified.
- But we want to tell agents what to do without telling them how to do it.

Utility Function over States

- One possibility is to associate utilities with individual states the task of the agent is then to bring about states that maximise utility.
- A task specification is a function

$$u : E \longrightarrow \Re$$

which associated a real number with every environment state.

But, what is the value of a run ...

- minimum utility of state on run?
- maximum utility of state on run?
- sum of utilities of states on run?
- average?

Difficult to specify a long term view when assigning utilities to individual states.

Utility over Run

- Another possibility: assigns a utility not to individual states, but to runs themselves

$$u : \mathcal{R} \longrightarrow \Re$$

- Such an approach takes an inherently long term view.

Expected Utility

- Write $P(r|Ag, Env)$ to denote probability that run r occurs when agent Ag is placed in environment Env.
- Note

$$\sum_{r \in \mathcal{R}(Ag,Env)} P(r \mid Ag, Env) = 1$$

- The expected utility of agent Ag in environment Env (given P, u), is then:

$$EU(Ag, Env) = \sum_{r \in \mathcal{R}(Ag,Env)} u(r) P(r \mid Ag, Env)$$

Example

Consider the environment $Env_1 = \langle E, e_0, \tau \rangle$ defined as follows:

$$E = (e_0, e_1, e_2, e_3, e_4, e_5)$$
$$\tau(e_0 \xrightarrow{\alpha_0}) = (e_1, e_2)$$
$$\tau(e_0 \xrightarrow{\alpha_1}) = (e_3, e_4, e_5)$$

There are two agents possible with respect to this environment:

$$Ag_1(e_0) = \alpha_0$$
$$Ag_2(e_0) = \alpha_1$$

The probabilities of the various runs are as follows:

$$P(e_0 \xrightarrow{\alpha_0} e_1 \mid Ag_1, Env_1) = 0.4$$
$$P(e_0 \xrightarrow{\alpha_0} e_2 \mid Ag_1, Env_1) = 0.6$$
$$P(e_0 \xrightarrow{\alpha_1} e_3 \mid Ag_2, Env_1) = 0.1$$
$$P(e_0 \xrightarrow{\alpha_1} e_4 \mid Ag_2, Env_1) = 0.2$$
$$P(e_0 \xrightarrow{\alpha_1} e_5 \mid Ag_2, Env_1) = 0.7$$

Assume the utility function u_1 is defined as follows:

$$u_1(e_0 \xrightarrow{\alpha_0} e_1\ ) = 8$$
$$u_1(e_0 \xrightarrow{\alpha_0} e_2\ ) = 11$$
$$u_1(e_0 \xrightarrow{\alpha_1} e_3\ ) = 70$$
$$u_1(e_0 \xrightarrow{\alpha_1} e_4\ ) = 9$$
$$u_1(e_0 \xrightarrow{\alpha_1} e_5\ ) = 10$$

What are the expected utilities of the agents for this utility function?

Optimal Agent

- The optimal agent Ag_{opt} in an environment Env is the one that maximizes expected utility:

$$Ag_{opt} = \arg \max_{Ag \in \mathcal{AG}} EU(Ag, Env)$$

- The fact that an agent is optimal does not mean that it will be best; only that on average, we can expect it to do best.

Predicate Task Specifications

- A special case of assigning utilities to histories is to assign 0 (false) or 1 (true) to a run.
- If a run is assigned 1, then the agent succeeds on that run, otherwise it fails
- Call these predicate task specifications.
- Denote predicate task specification by Ψ

$$\Psi : \mathcal{R} \rightarrow \{0, 1\}$$

Task Enviornment

- A task environment is a pair $\langle Env, \Psi \rangle$ where Env is an environment, and

$$\Psi : \mathcal{R} \rightarrow \{0, 1\}$$

is a predicate over runs. Let $\mathcal{TE}$ be the set of all task environments

- Write $\mathcal{R}_\Psi(Ag, Env)$ to denote the set of all runs of the agent Ag in the environment Env that satisfy

$$\mathcal{R}_\Psi(Ag, Env) = \{r \mid r \in \mathcal{R}(Ag, Env) \text{and} \Psi(r) = 1$$

- We then say that an agent Ag succeeds in task environment $\langle Env, Ag \rangle$ if

$$\mathcal{R}_\Psi(Ag, Env) = \mathcal{R}(Ag, Env)$$

Probability of Success

- Let $P(r|\ Ag, Env)$ denote probability that run r occurs if agent Ag is placed in environment Env
- Then the probability $P(\Psi|\ Ag, Env)$ that Ψ is satisfied by Ag in Env would then simply be:

$$P(\Psi|\ Ag, Env) = \sum_{r \in \mathcal{R}_\Psi(Ag, Env)} P(r|\ Ag, Env)$$

Agent Synthehsis

- Agent synthesis is automatic programming: goal is to have a program that will take a task environment, and from this task environment automatically generate an agent that succeeds in this environment:

$$syn : \mathcal{TE} \rightarrow (\mathcal{AG} \cup \{\bot\})$$

(Think of $\bot$ as being like null in JAVA).

- Synthesis algorithm:
  - is sound if, whenever it returns an agent, then this agent succeeds in the task environment that is passed as input; and
  - is complete if it is guaranteed to return an agent whenever there exists an agent that will succeed in the task environment given as input.

Synthesis algorithm syn is sound if it satisfies the following condition:

$$syn(\langle Env, \Psi \rangle) = Ag \text{ implies } \mathcal{R}(Ag, Env) = \mathcal{R}_{\Psi}(Ag, Env)$$

and complete if:

$$\exists Ag \in \mathcal{AG} \text{s.t.} \mathcal{R}(Ag, Env) = \mathcal{R}_{\Psi}(Ag, Env) \text{ implies } syn(\langle Env, \Psi \rangle) \neq \perp$$

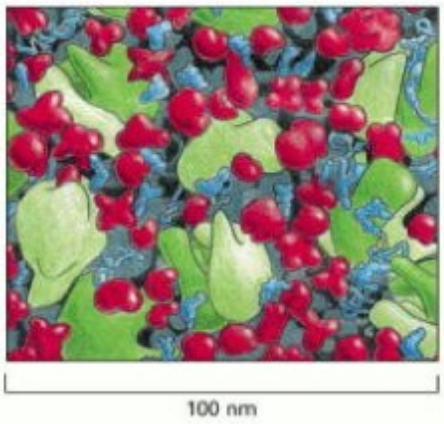
Post-Declarative Systems

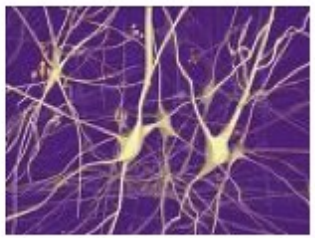
This view of agents leads to a kind of post-declarative programming:

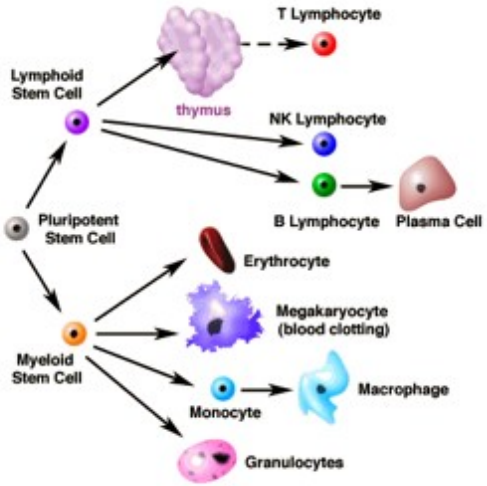
In procedural programming, we say exactly what a system should do

In declarative programming, we state something that we want to achieve, give the system general info about the relationships between objects, and let a built-in control mechanism (e.g., goal-directed theorem proving) figure out what to do

With agents, we give a very abstract specification of the system, and let the control mechanism figure out what to do, knowing that it will act in accordance with some built-in theory of agency.

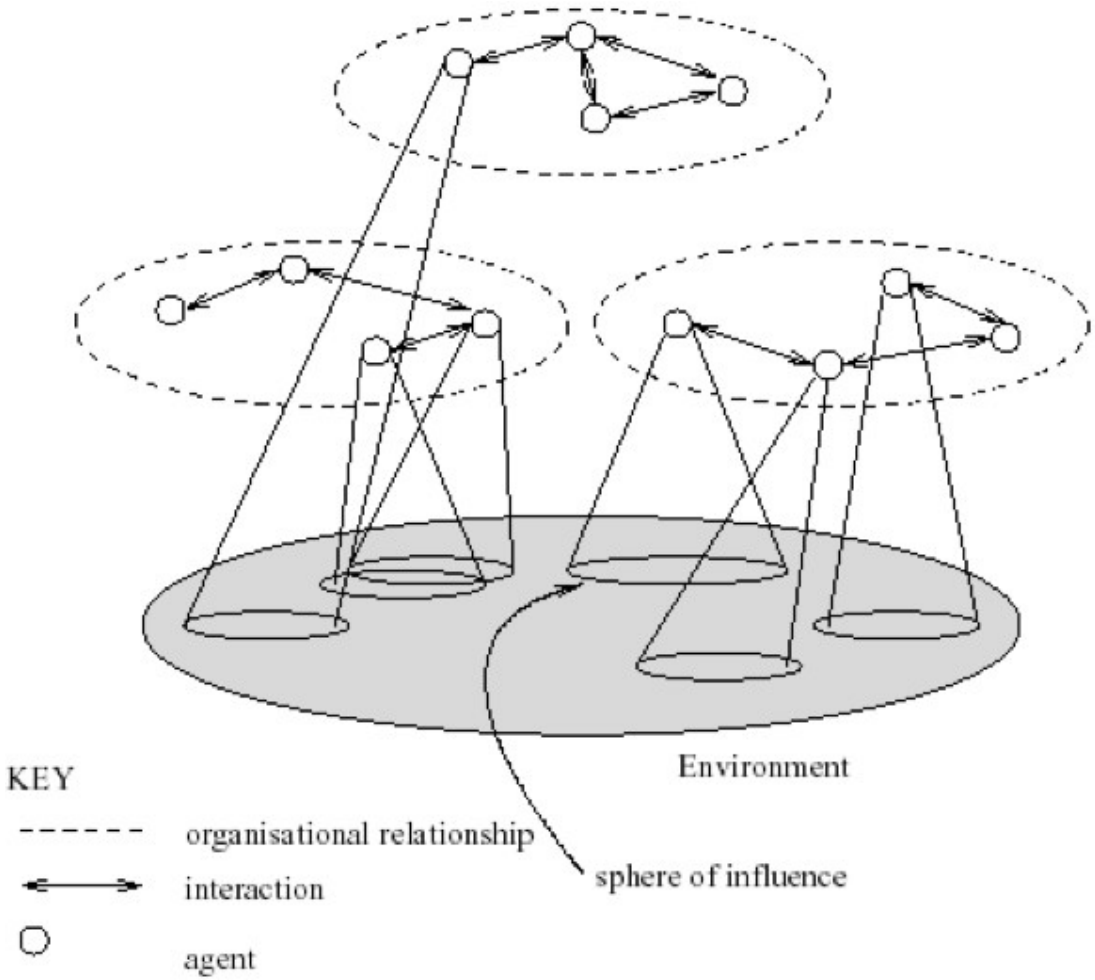




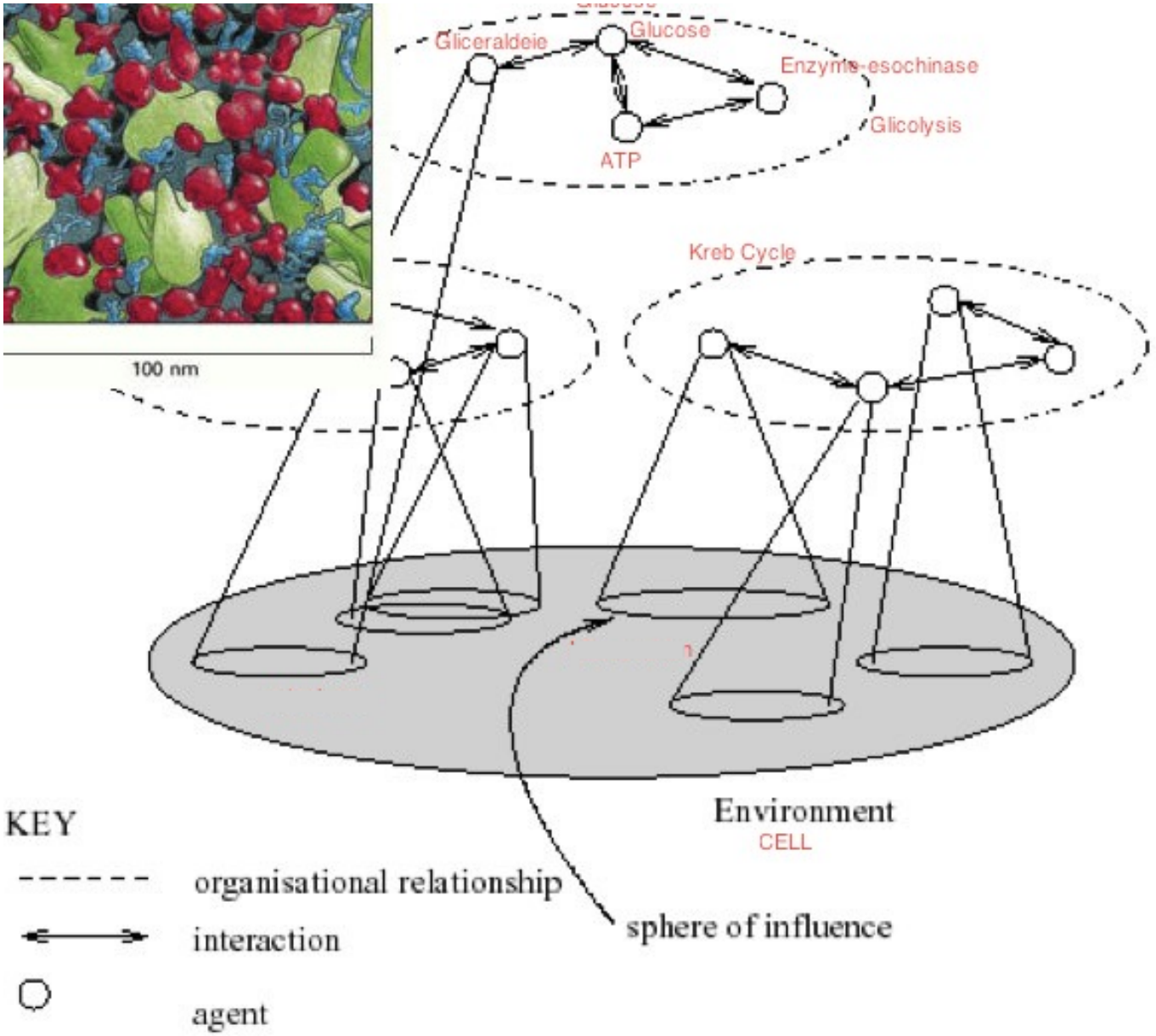


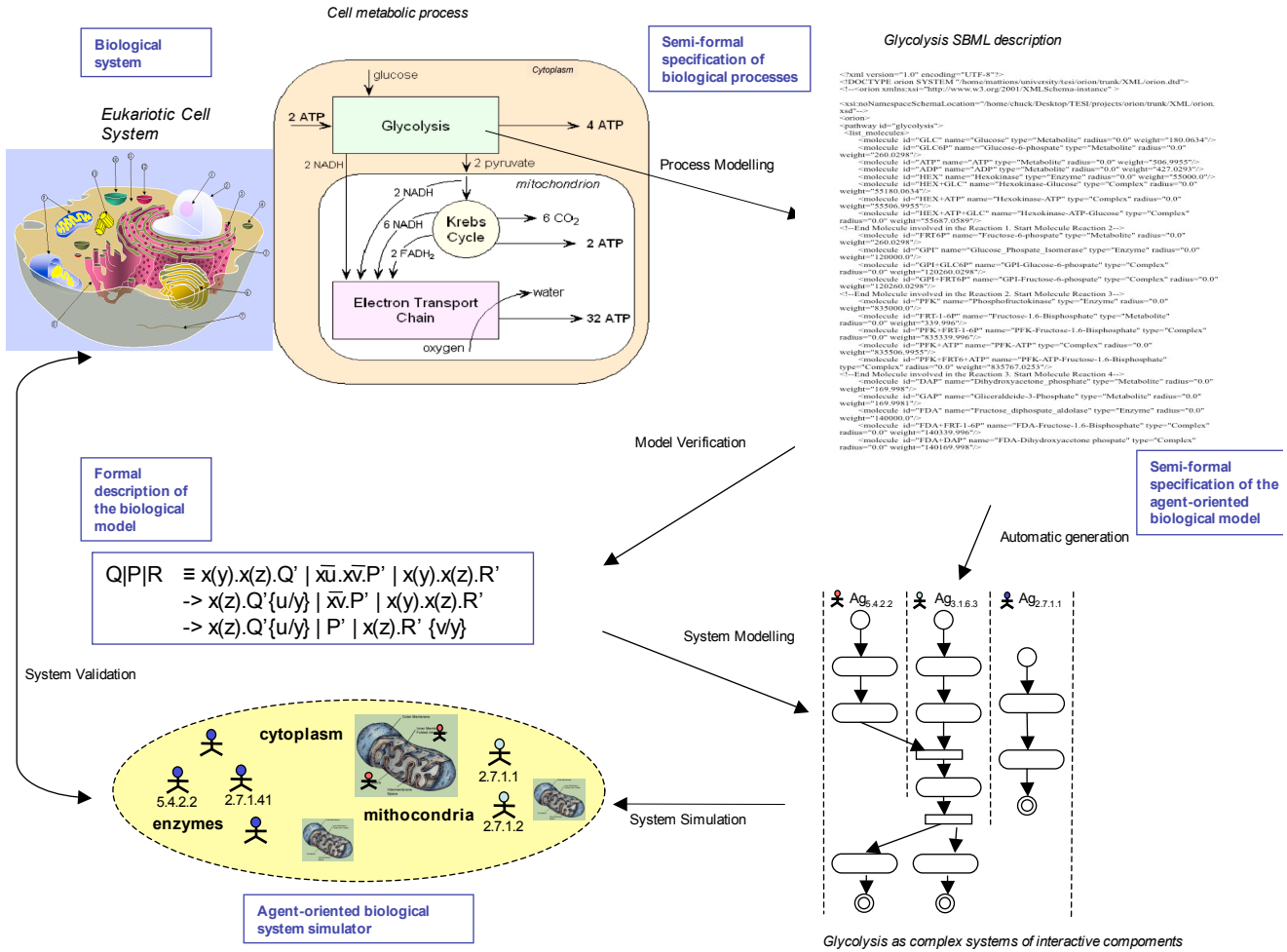
A Multiagent system contains a number of agents which:

- interact trough communication
- are able to act in an environment
- have different spheres of influence
- will be linked by many relationships (organizational)



[*An Introduction to MultiAgent Systems* by Michael Wooldridge, John Wiley & Sons, 2009]


Gliceraldeie
Glucose
Enzyme-esochinase
Glicolysis
ATP
Kreb Cycle
100 nm
KEY
organisational relationship
interaction
agent
Environment
CELL
sphere of influence

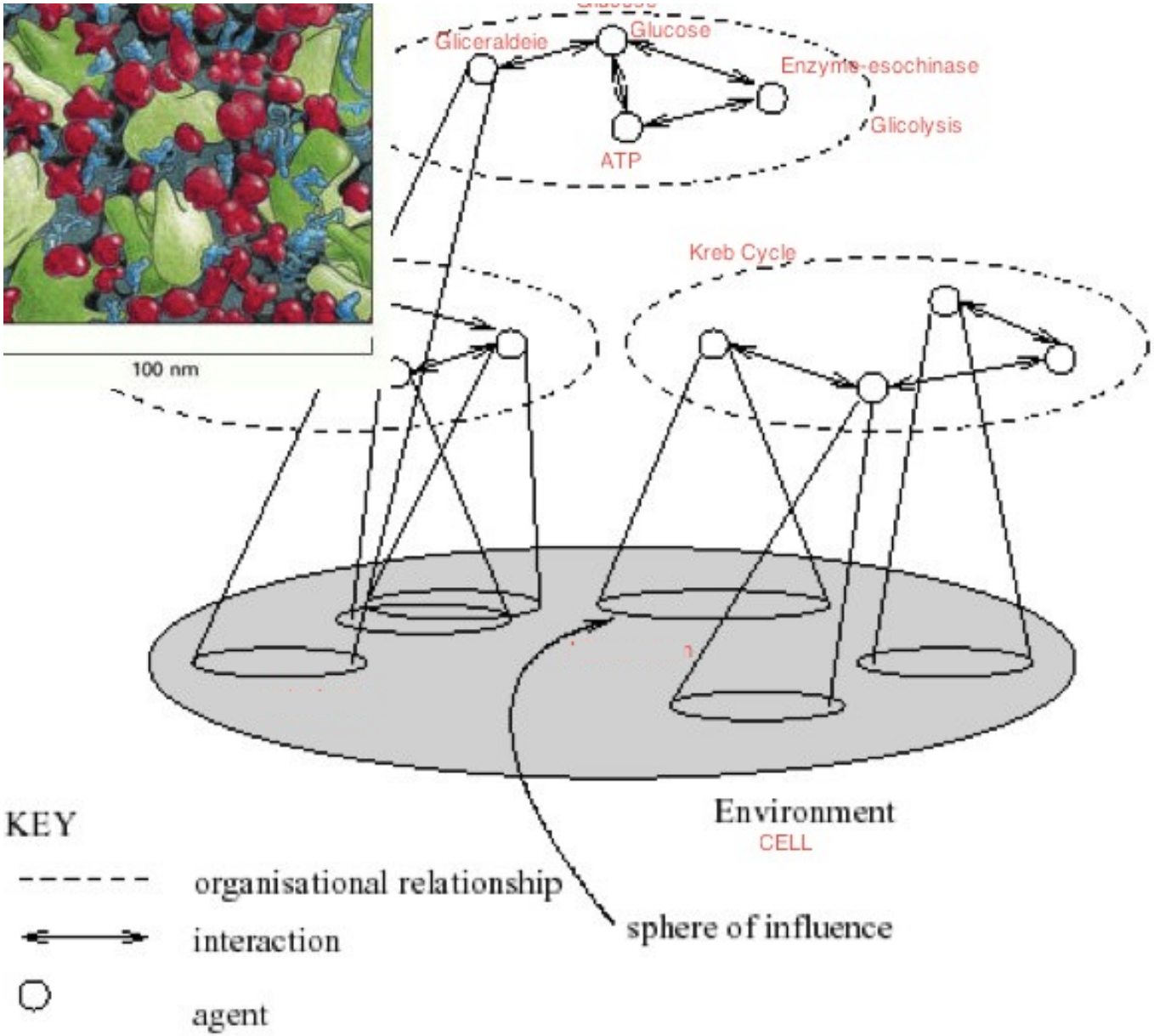

Cell metabolic process
Biological system
Eukariotic Cell System
glucose
Cytoplasm
2 ATP
Glycolysis
4 ATP
2 NADH
2 pyruvate
mitochondrion
2 NADH
6 NADH
Krebs Cycle
6 CO_2
2 ATP
2 $FADH_2$
Electron Transport Chain
water
32 ATP
oxygen
Semi-formal specification of biological processes
Process Modelling
Glycolysis SBML description
Model Verification
Formal description of the biological model
Semi-formal specification of the agent-oriented biological model
Automatic generation
Q|P|R ≡ x(y).x(z).Q' | x̄u.x̄v.P' | x(y).x(z).R'
-> x(z).Q'{u/y} | x̄v.P' | x(y).x(z).R'
-> x(z).Q'{u/y} | P' | x(z).R' {v/y}
System Modelling
System Validation
cytoplasm
5.4.2.2
2.7.1.41
enzymes
mithocondria
2.7.1.1
2.7.1.2
System Simulation
Agent-oriented biological system simulator
Glycolysis as complex systems of interactive compoments

Multiagent System - MAS

- A multiagent system consists of
  - a collection of agents, $\mathcal{AG}$ and
  - a set of coordination rules

- An agent has associated a set of behaviours.
- An agent adopts different behaviours at different times, depending on the environmental conditions $Per \subset E$

Mutiagent Systems



Mutiagent System

A well-designed MAS is one that achieves a global task through the tasks of single agent and through their interactions.

Main design steps consist of

- decomposing a global task into distributable subcomponents, yielding tractable tasks for each agent;
- establishing communication channels that provide sufficient information to each agent to achieve its task; and
- coordinating the agents in a way they cooperate on the global task, or at the very least, does not allow them to pursue conflicting strategies in trying to achieve their tasks.

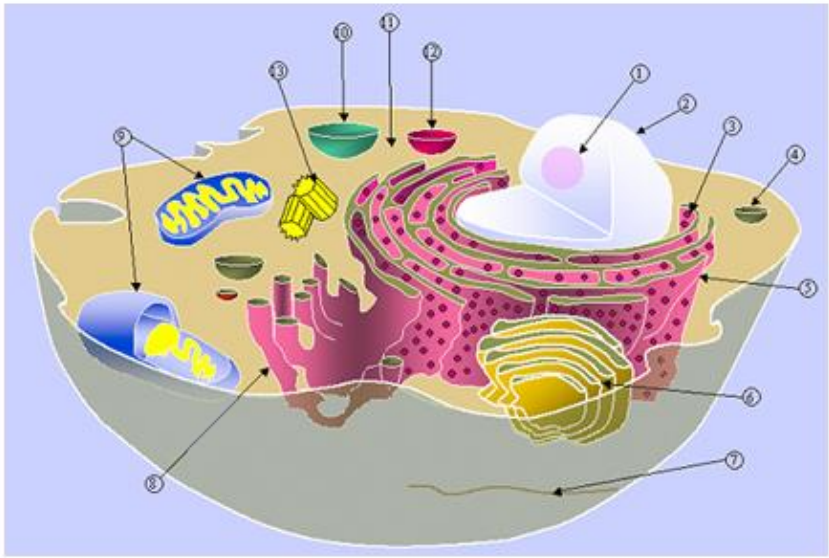
The fundamental problem, in analyzing/designing such MAS, is in determining how the combined actions of a large number of agents, leads to "coordinated" behavior on the global task.

The Cell

The Carbohydrate Oxidation is the energy production process performed by two active components of the Cell: Cytoplasm and Mitochondrion

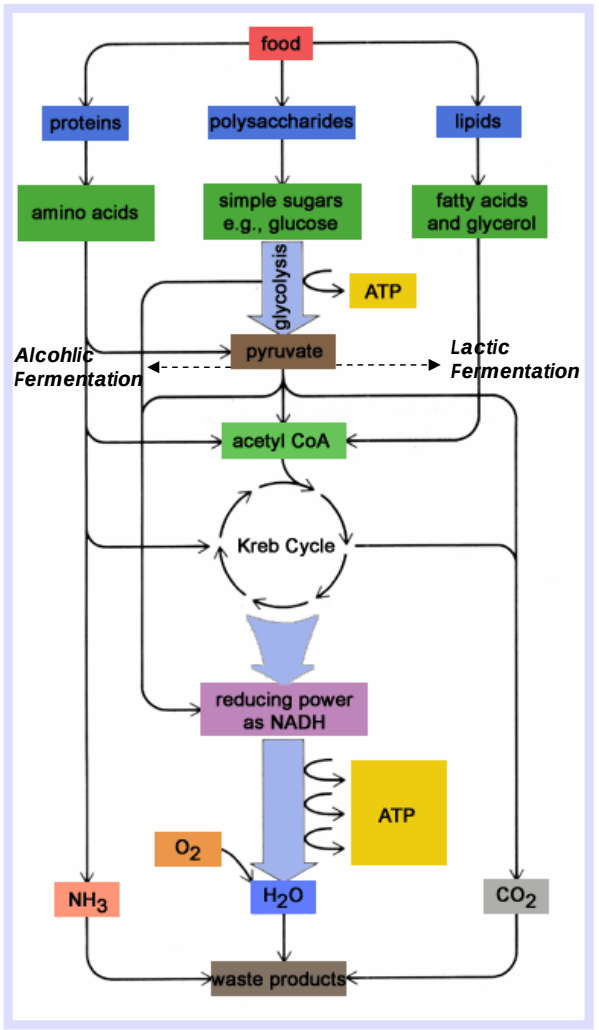
Cell Components

- Cytoplasm
- Mithocondrion
  - Mithocondrial Matrix
  - Mithocondrial Inner Membrane



- Cell metabolism is the process by which individual cells process nutrient molecules.
- The series of chemical reactions within a cell that realise cell metabolism are called metabolic pathways.
- Carbohydrate oxidation is a set of metabolic pathways by which a cell produces energy through chemical transformation of carbohydrates like fructose, glucose, mannose, maltose, lactose, saccharose, glycogen and starch.

Carbohydrate Oxidation Cellular Process



Step 1:
Transformatrion of glucose into pyruvate

Step 2:
Transformation of acetyl CoA into NADH

Step 3:
Transportation of NADH into ATP

Legend

Metabolic pathway

Compounds reaction

Molecules

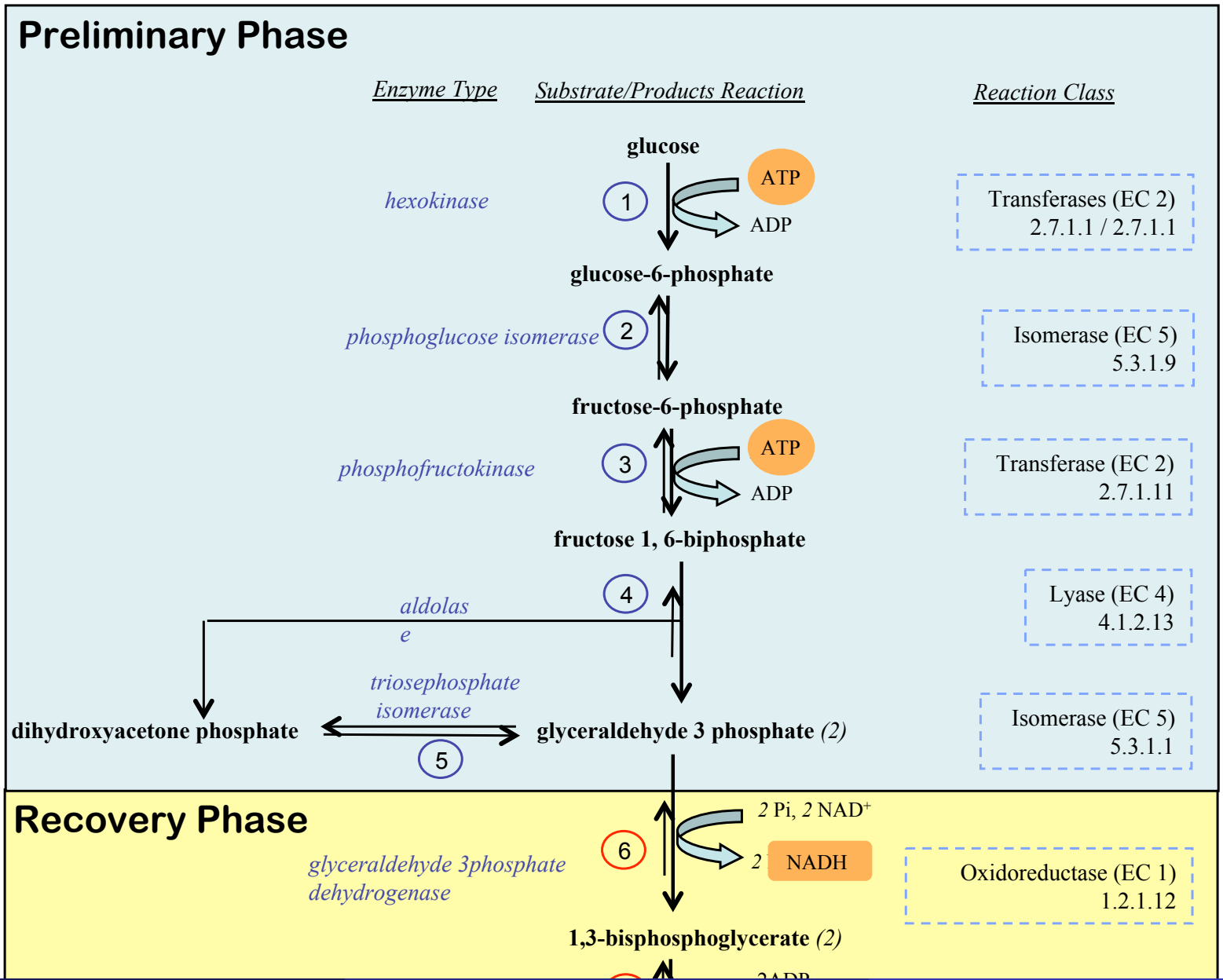
Compartment 1:
Cytoplasm

Compartment 2:
Mitochondrion

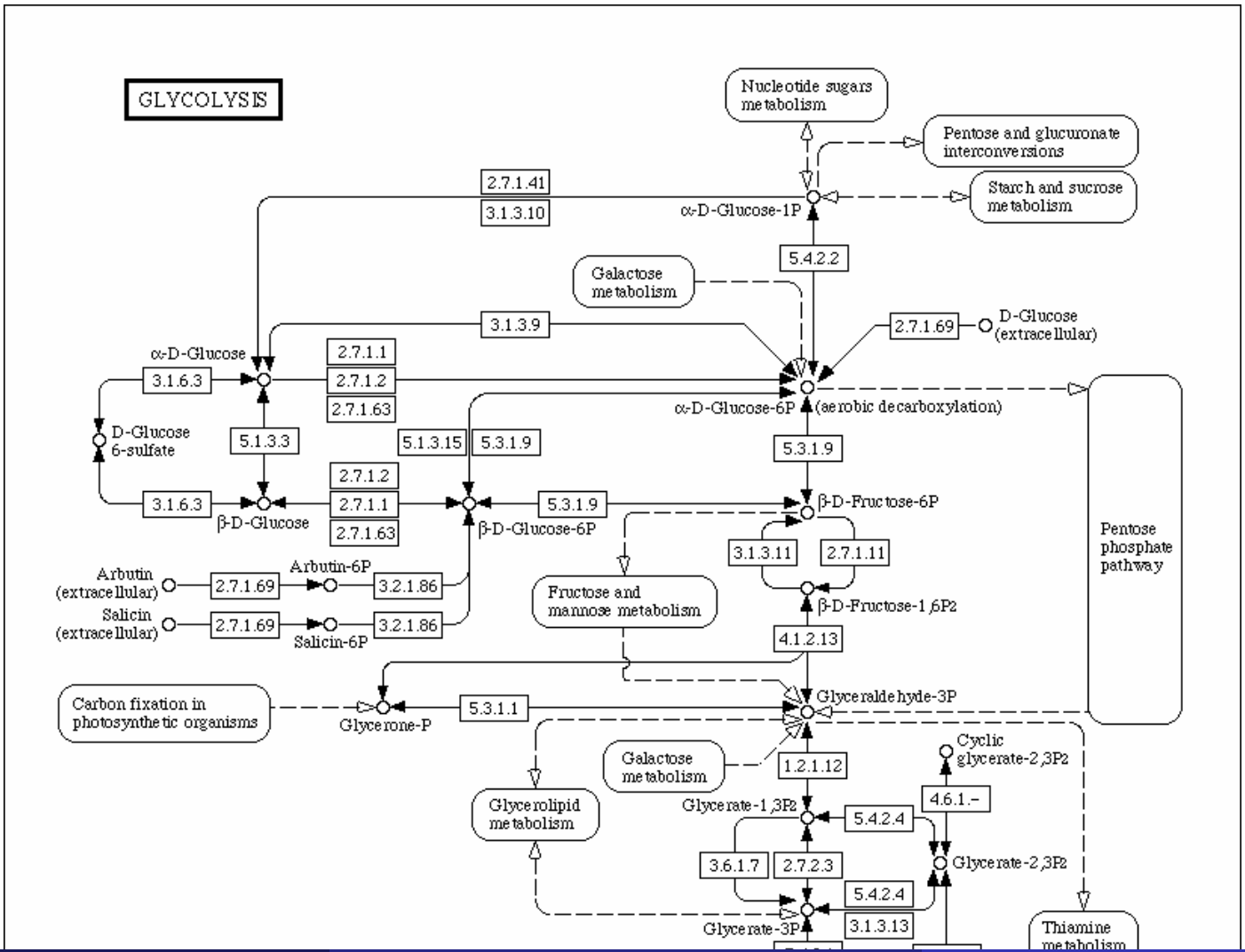
Compartment 2.1:
Mitochondrial Matrix

Compartment 2.2:
Inner Mitochondrial Membrane

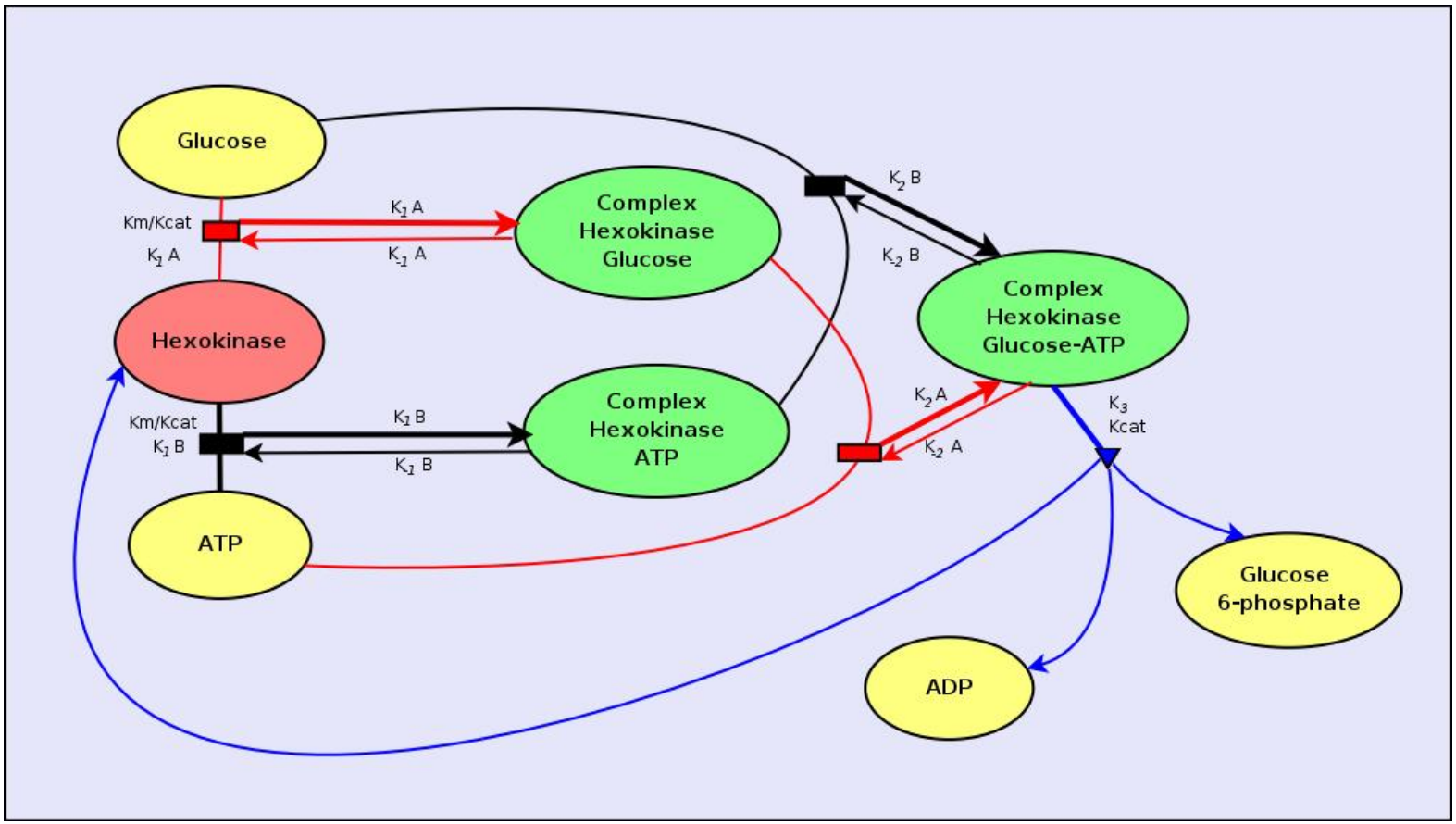
Glycolysis


Preliminary Phase
Enzyme Type
Substrate/Products Reaction
Reaction Class
glucose
hexokinase
1
ATP
ADP
glucose-6-phosphate
Transferases (EC 2)
2.7.1.1 / 2.7.1.1
phosphoglucose isomerase
2
Isomerase (EC 5)
5.3.1.9
fructose-6-phosphate
phosphofructokinase
3
ATP
ADP
fructose 1, 6-biphosphate
Transferase (EC 2)
2.7.1.11
4
aldolase
Lyase (EC 4)
4.1.2.13
triosephosphate isomerase
dihydroxyacetone phosphate
5
glyceraldehyde 3 phosphate (2)
Isomerase (EC 5)
5.3.1.1
Recovery Phase
glyceraldehyde 3phosphate dehydrogenase
6
2 Pi, 2 NAD+
2
NADH
Oxidoreductase (EC 1)
1.2.1.12
1,3-bisphosphoglycerate (2)

KEGG Interaction Diagram



Glycolysis - Step 1



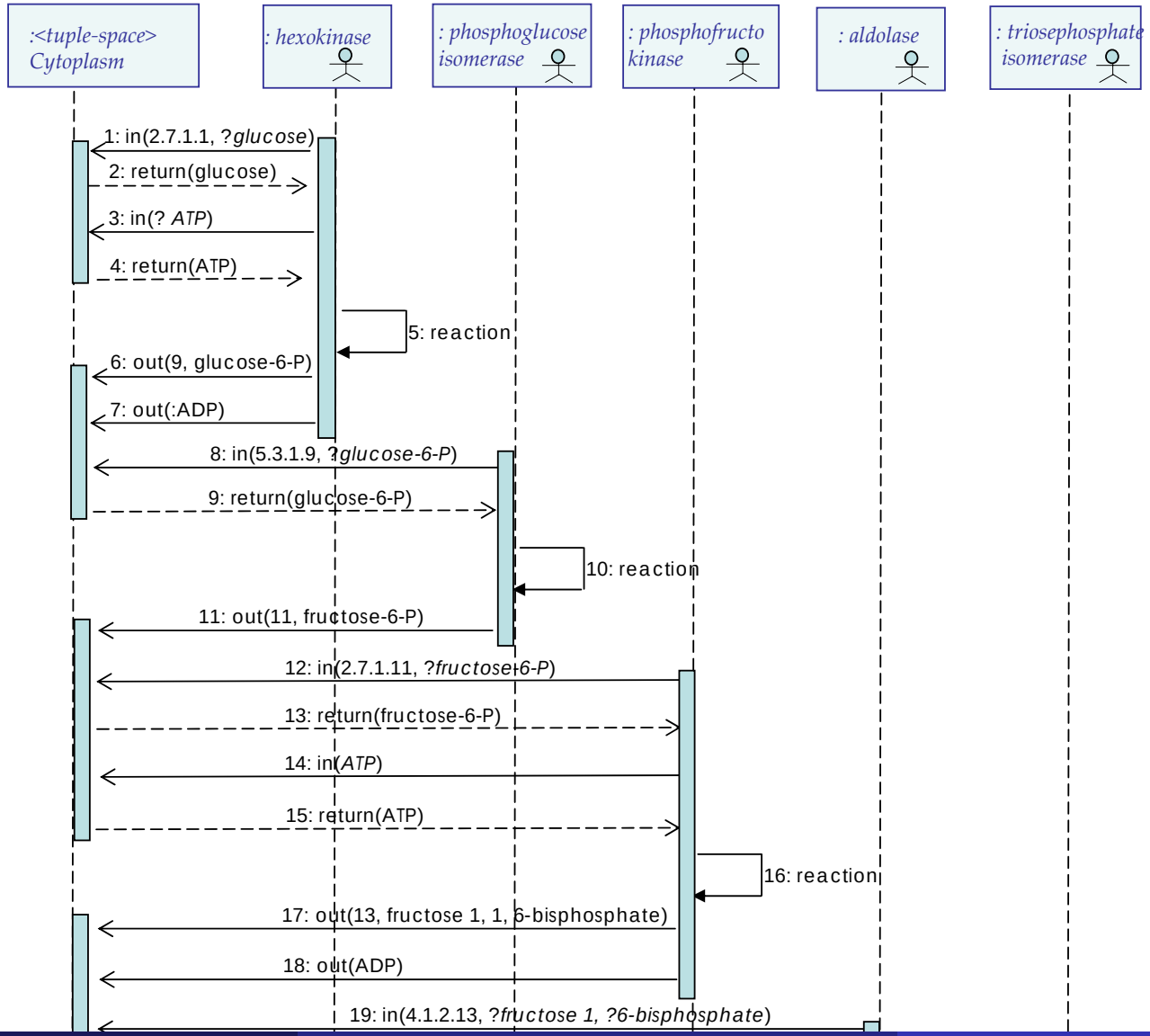
```
  <?xml version="1.0" encoding="UTF-8" ?>
- <sbml xmlns="http://www.sbml.org/sbml/level2"
xmlns:jd="http://www.sys-bio.org/sbml"
xmlns:sl2="http://projects.eml.org/bcb/sbml/level2" metaid="metaid_0000001"
level="2" version="1"> - <model metaid="metaid_0000002" id="Glycolysis_Nielsen"
name="Nielsen1998_Glycolysis"> - <notes> - <body
xmlns="http://www.w3.org/1999/xhtml">
  <p>Reference: Nielsen et al; Biophys. Chem. (1998) 72:49-62</p>
- <p>
  The reaction looks like this:
  <br />
  reaction_1: GLC + ATP -> F6P + ADP;
  <br />
  reaction_2: F6P + ATP -> FBP + ADP;
  <br />
  reaction_3: FBP => 2 * GAP;
  <br />
  reaction_4: GAP + NAD -> DPG + NADH;
  <br />
  reaction_5: DPG + ADP => PEP + ATP;
  <br />
  reaction_6: PEP + ADP -> PYR + ATP;
  <br />
  reaction_7: PYR -> ACA;
  <br />
  reaction_8: ACA + NADH => EtOH + NAD;
  <br />
  reaction_9: AMP + ATP => 2 * ADP;
  <br />
  reaction_10: F6P -> P;
  <br />
  flow reactor:
  <br />
  flow = 1.1e-2 reactorvolume/s
  <br />
  inflow concentrations:
  <br />
```

Glycolysis

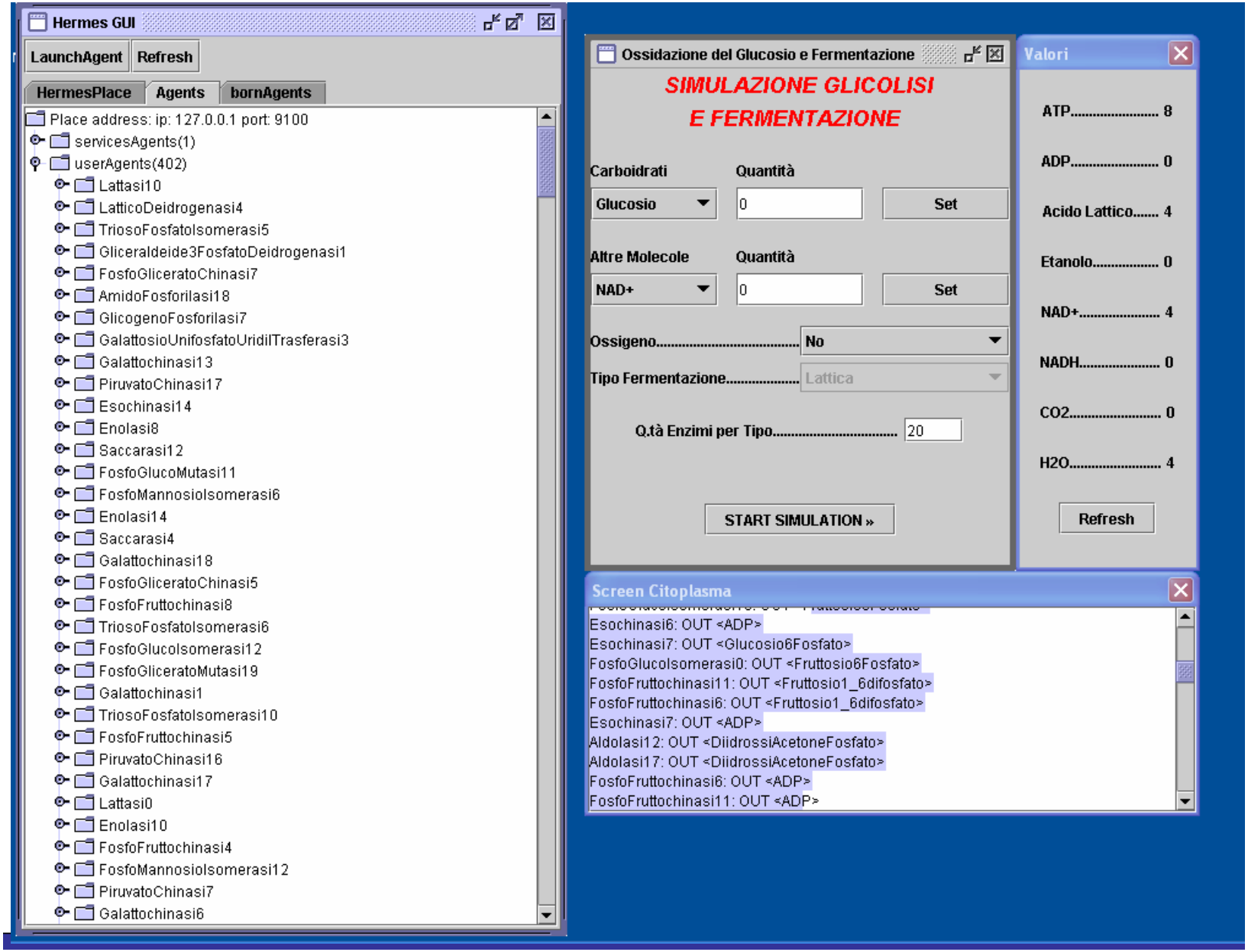
$$glucose + 2ATP + 4ADP + 2Pi + 2NADox \rightarrow 2pyruvate + 2ADP + 4ATP + 2NADrid + 2H_2O$$

ULM specification

UML Sequence diagram of the preliminary phase of glycolysis



Second Simulator: Glycolysis pathway

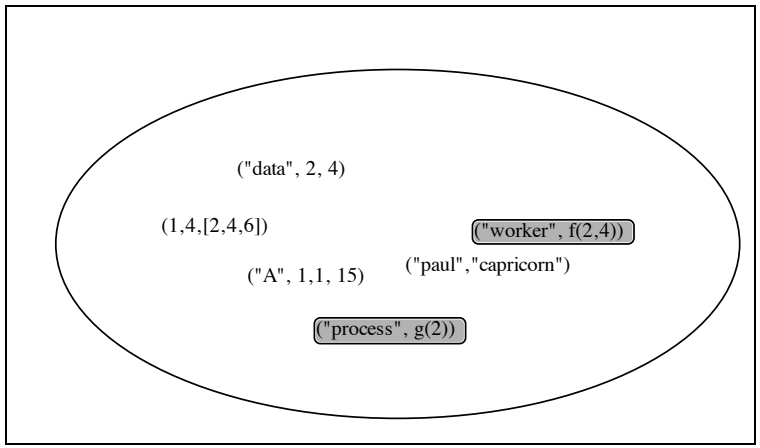


MAS and Coordination

- Centralize Coordination $\longrightarrow$ Linda-like models
- Distributed Coordination $\longrightarrow$ Message-passing communication

A Coordination model -Tuple space

The Tuple Space is a global computing environment conceptually including both data, in form of passive tuples, and agents, in form of active tuples



- All tuples are created by agents; they are atomic data (they can only be created, read or deleted)
- Agents cannot communicate directly:
  an agent can only read (rd) or consume (in) a tuple, write (out) a new tuple or create (eval) a new agent that when terminates becomes a data tuple

LINDA model

Linda model

Based on generative communication: processes (agents) communicate by inserting or retrieving data objects (called Tuples) from shared Data Space (called Tuple space)

Operations

- out(t) insert a tuple t into the Tuple space (non-blocking)
- in(t) find and remove a *matching* tuple from the tuple space; block until a matching tuple is found
- rd(t) like in(t) except that the tuple is not removed
- eval(t) add the active tuple t to the tuple space

The model is structurally recursive: a tuple space can be one field of a tuple.

Linda TS operations

Tuples are created and manipulated by agents using the following operations:

out(t) puts a new passive tuple in the Tuple Space, after evaluating all fields; the caller agent continues immediately

eval(t) puts a new agent in the Tuple Space (each field containing a function to be computed starts a process); the caller agent continues immediately; when all active fields terminate the tuple becomes passive

in(t) looks for a passive tuple in the Tuple Space; if not found the agent suspends; when found, reads and deletes it

rd(t) looks for a passive tuple in the Tuple Space; if not found the agent suspends; when found, reads it

inp(t) looks for a passive tuple in the Tuple Space; if found, deletes it and returns TRUE; if not found, returns FALSE

rdp(t) looks for a passive tuple in the Tuple Space; if found, copies it and returns TRUE; if not found, returns FALSE

Coordination

- *Coordination* is a key concept for studying the *activities* of complex *dynamic systems*
- *Coordination* is managing *dependencies* between activities

Coordination by Malone and Crowston '94

Coordination includes *agents* performing *activities* that are *interdependent*

Coordination

Coordination by Carriero and Gelernter '92

Coordination is the process of building programs by gluing together active pieces

Active pieces are processes, objects with threads, agents or whole applications

- *Sequential* programming = computation
- *Distributed* programming = communication + computation
- *Coordinated* programming = coordination + computation

Coordination model

Coordination model

A coordination model provides a minimun framework in which the interaction of individual agents can be expressed

- *dynamic agent creation and termination*,
- *control of communication flows among agents*,
- *control of spatial distribution* and
- mobility of agents and
- *as well as synchronization and distribution of actions over time*

Coordination language

- A Coordination language is the linguistic embodiment of a coordination model, and consists of some coordination mechanisms that are added to an host (sequential) language
- In practice, a coordination language includes clearly defined mechanisms for communication, synchronization, distribution, and concurrency control.

Coordination language - features

- Distribution
  - computing over different (explicit) localities
- Mobility
  - moving agents and computations over localities
- Concurrency
  - considering parallel and non-deterministic computations
- Access Rights
  - maintaining privacy and integrity of data

Klaim: Kernel Language for Agent Interaction and Mobility

Process Calculus Flavored

- Small set of basic combinator
- Clean operational semantics

Linda based communication model

- Asynchronous communication;
- Shared tuple spaces;
- Pattern Matching

Explicit use of localities

- Multiple distributed tuple spaces;
- Code and Process mobility.

From Linda and Process Algebras to Klaim

Explicit Localities to model distribution

- Physical Locality (sites)
- Logical Locality (names for sites)
- A distinct name self (or here) indicates the site a process is on

Allocation environment to associate sites to logical localities

- This avoids the programmers to know the exact physical structure

Process Algebras Operators to compose programs

- Sequentialization
- Parallel composition
- Creation of new names

Klaim Syntax

$$
\begin{array}{lcll}
P & ::= & \mathbf{nil} & \text{(null process)} \\
 & | & a.P & \text{(action prefixing)} \\
 & | & P_1 \mid P_2 & \text{(parallel composition)} \\
 & | & P_1 + P_2 & \text{(choice)} \\
 & | & X & \text{(process variable)} \\
 & | & A(\widetilde{P}, \widetilde{l}, \widetilde{e}) & \text{(process invocation)}
\end{array}
$$

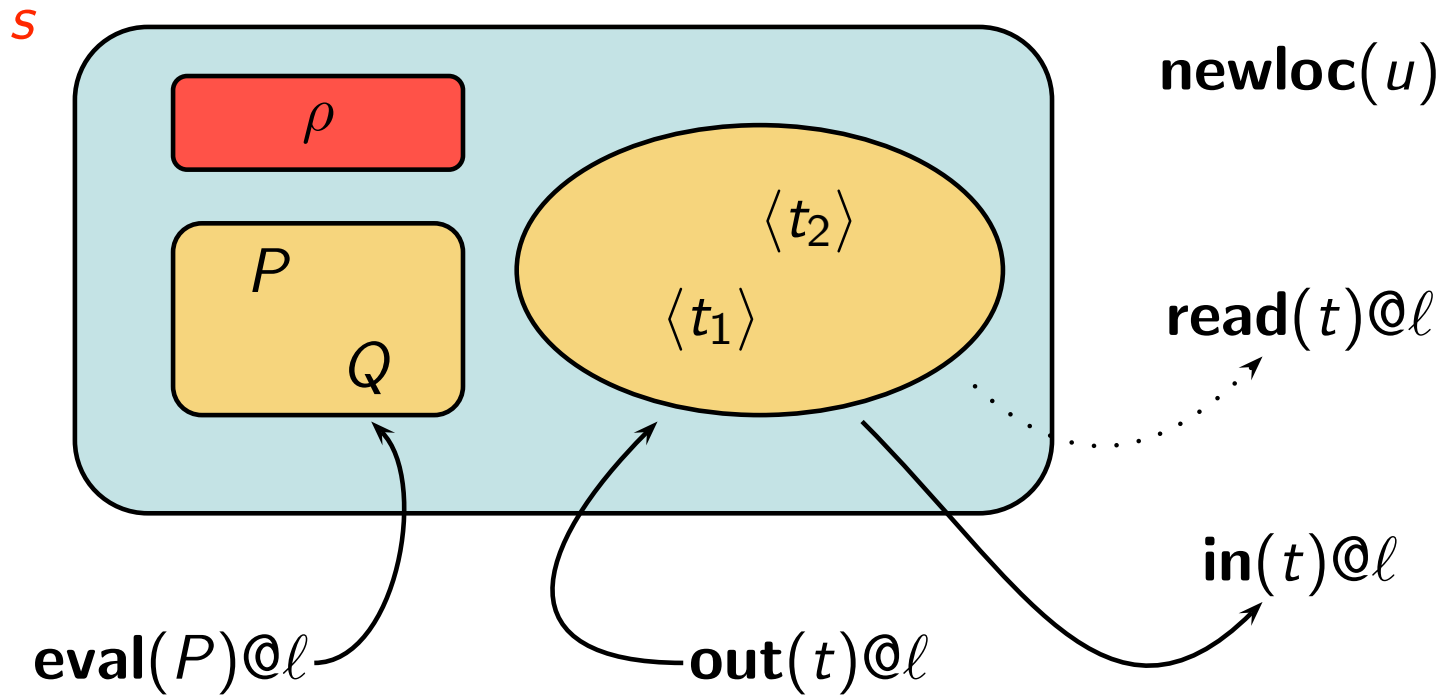
$$
a ::= \mathbf{out}(t)@l \mid \mathbf{in}(t)@l \mid \mathbf{read}(t)@l \mid \mathbf{eval}(P)@l \mid \mathbf{newloc}(u)
$$

$$
t ::= e \mid P \mid l \mid !x \mid !X \mid !u \mid t_1, t_2
$$

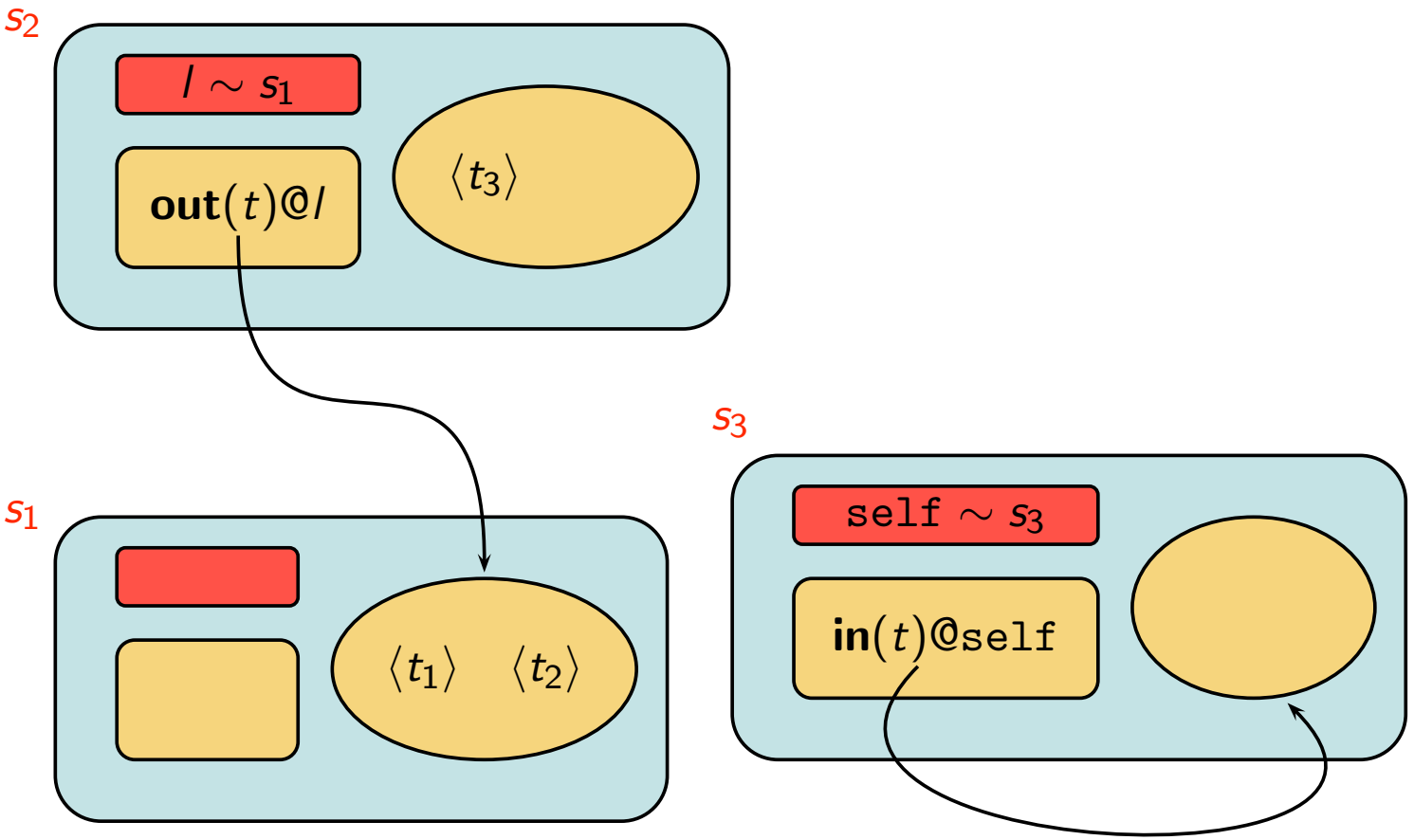
$$
\begin{array}{lcll}
N & ::= & s ::_\rho P & \text{(node)} \\
 & | & N_1 \| N_2 & \text{(net composition)}
\end{array}
$$

Klaim Node

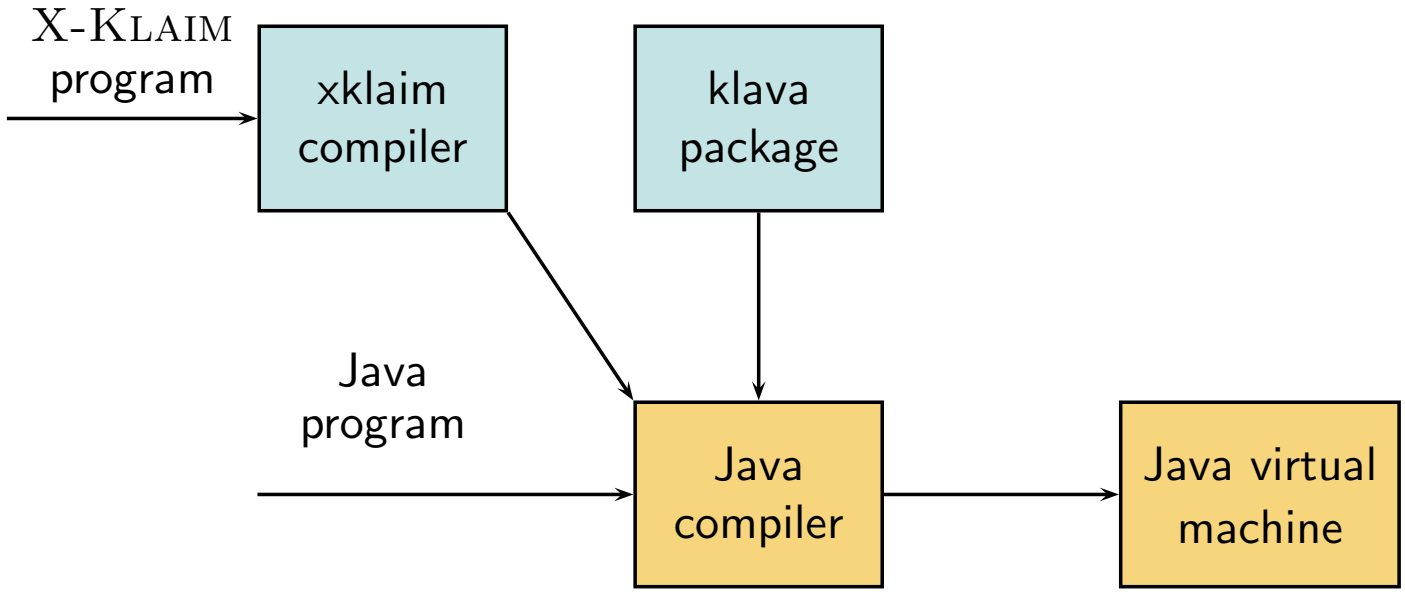
- Locality
- Processes
- Tuple Space
- Allocation Environment



Klaim Nets


s_2
$l \sim s_1$
$\langle t_3 \rangle$
out(t)@l
s_1
$\langle t_1 \rangle$ $\langle t_2 \rangle$
s_3
self $\sim s_3$
in(t)@self

Programming framework for Klaim

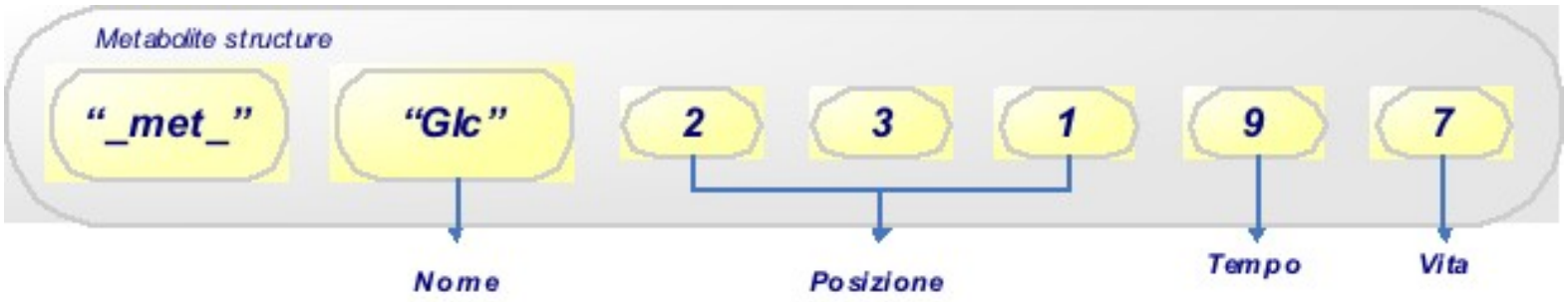


Kalliope Simulator

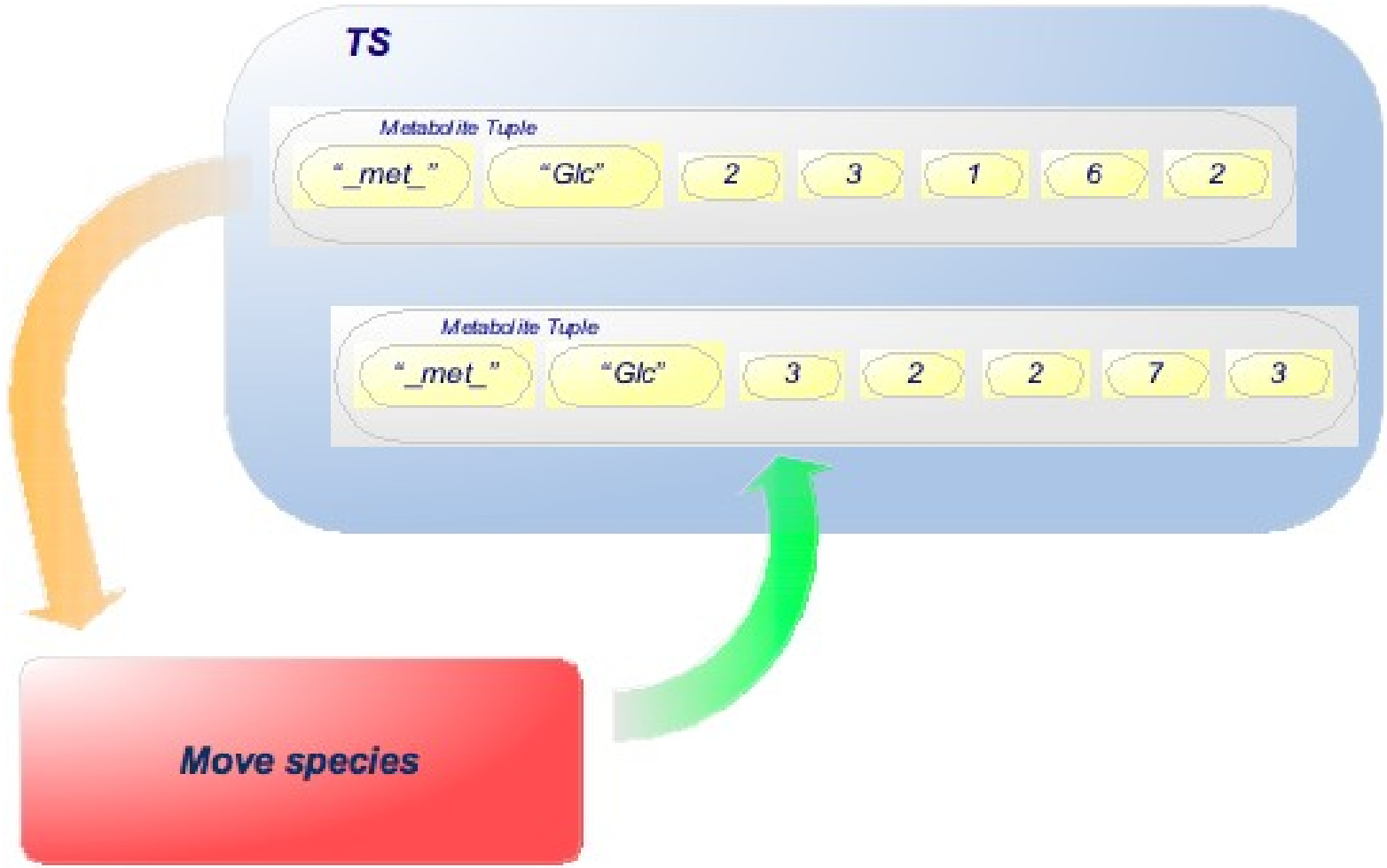
Kalliope has been developed in X-Klaim and run over Klava framework

- a Klaim Tuple Space (TS) represents the Cytoplasm
- the Cytoplasm molecules are tuples in the TS
- a set of Klaim agents manipulated the TS to simulate the Cytoplasm reaction during the Glycolysis

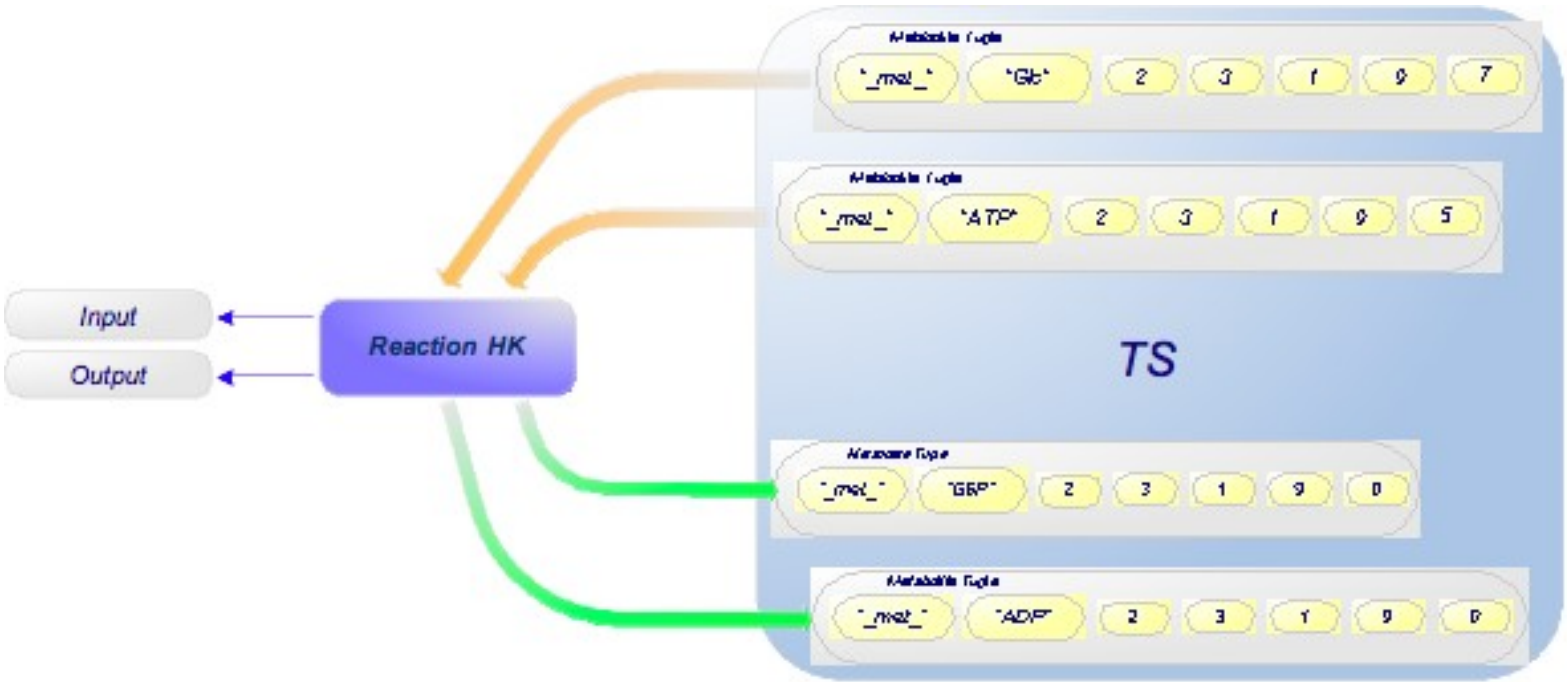
Kalliope-Klaim tuples



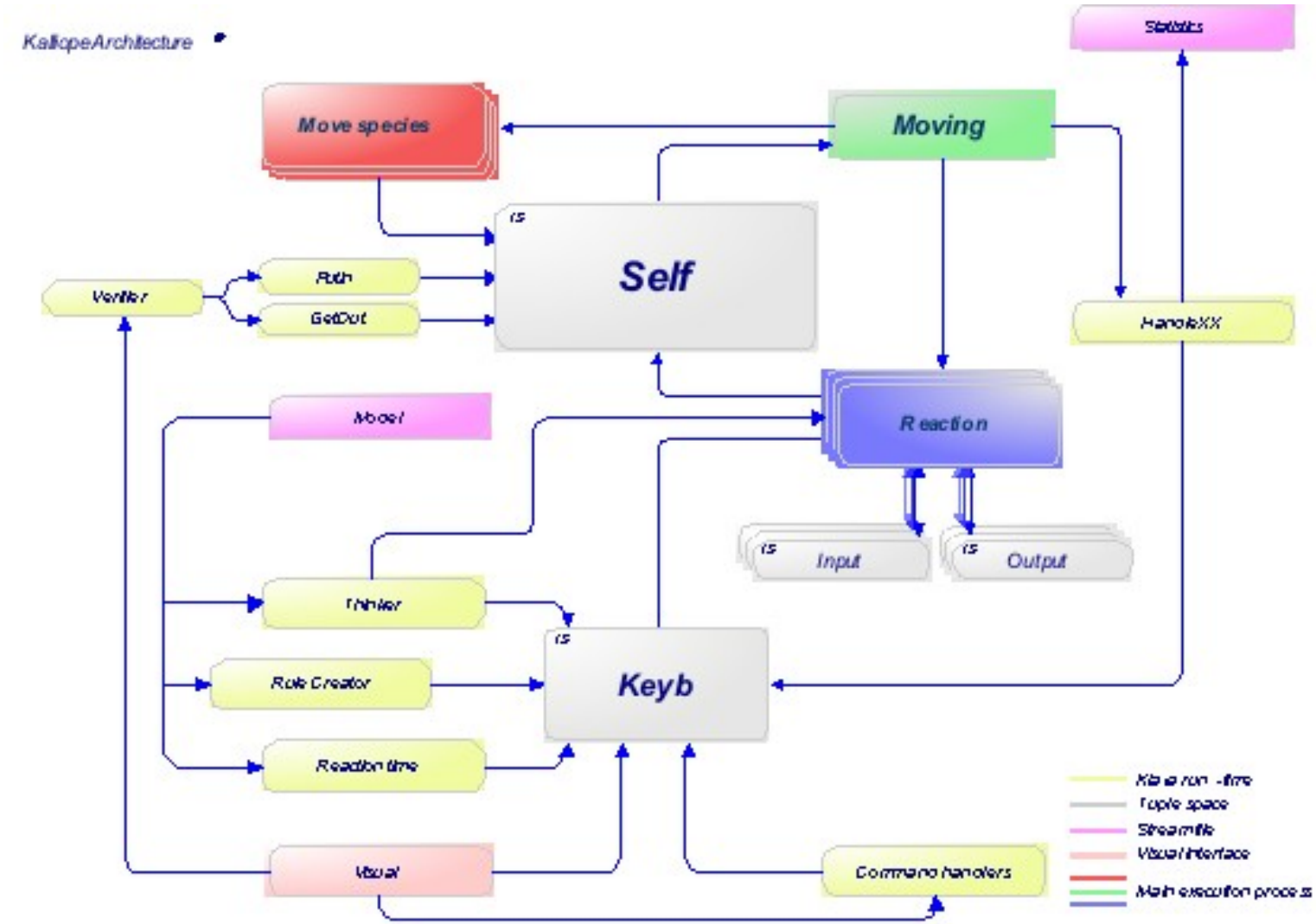
Kalliope movement



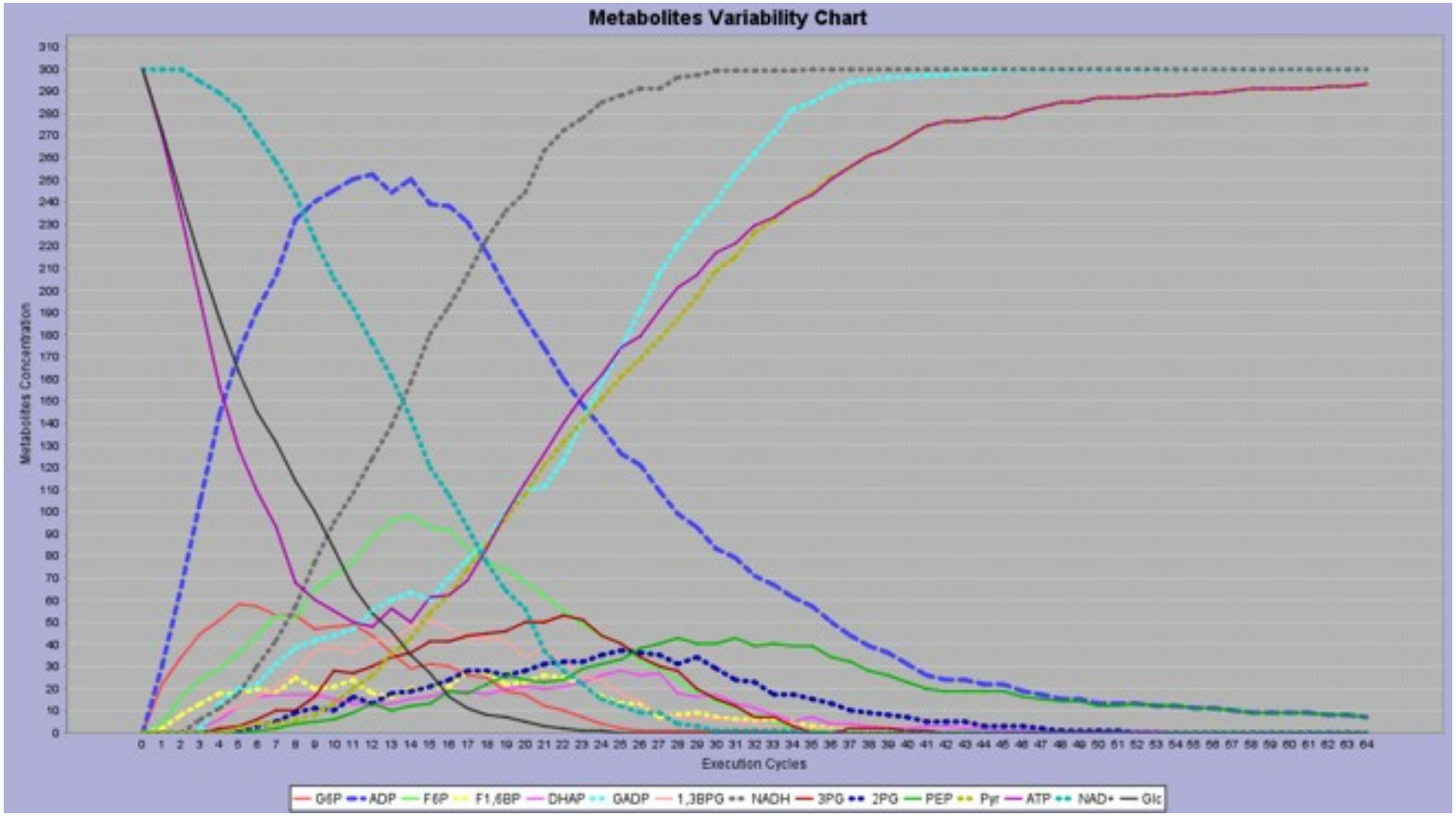
Kalliope reactions



Kalliope simualtor architecture



Kalliope simulation results



Part Three

Part Three: Space, Geometry, Motion and Interactions

- Space in models and languages
- Shape Calculus
- BIOSHAPE Simulation Environment
- Multiscale case study: Bone Remodelling

Spatial models and languages

- Traditional computer science models do not consider space in an explicit way
- Main reason: it is not a primary issue when modelling software systems
- In some languages a **qualitative** notion of space is considered, e.g. "location"
- Quantitative, or better, geometric space cannot be expressed, unless the model contains "values" with which it can be emulated

The same story of time

- Traditionally, only qualitative time could be expressed, both in modelling languages and in property specification languages
- Since about 1990, **timed** models and languages have been introduced
- Discrete or real time became first citizen in many extensions of classical formalisms
- It's now the turn of space ...

Motivation

- Biological systems are very complex systems to modeled
- In Computational (Algorithmic) Systems Biology we want the computer science approach is used to model such systems
- Space, in particular 3D space, plays a fundamental role in the real systems (the ones to model)
- The geometrical shape of an object often determines its functionality

Motivation

Within Computational Systems Biology

- Designing "in-silico" experiments (sets of simulations)
- Exploring known and unknown biological interactions
- Modeling different **scales**: molecules, cells, tissue, organ

Virtual Physiological Human

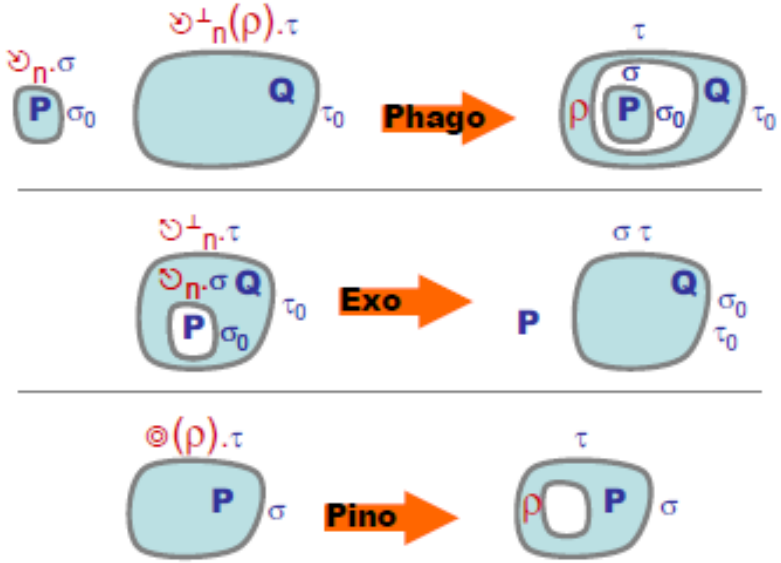
- Having a digital self, with personalized parameters
- Pre-test of drugs or therapies on the virtual self

Nanotechnology

- Molecular self-assembly
- Molecular recognition

Topological approaches

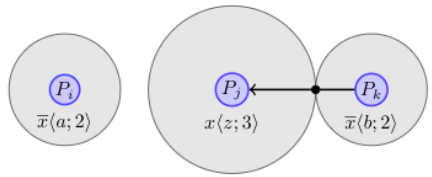
Describe space as a set of hierarchical and communicating well-stirred compartments: BioAmbients (Regev et al. 2004), Brane Calculi (Cardelli 2005), Membrane Computing (Paun 2000), etc ...



http://lucacardelli.name/

Sphere based approach

Entities are modeled as spheres in space: Spatial CLS (Barbuti et al. 2009), SpacePI (John et al. 2008), etc ...



SpacePI: Spatial extension of π calculus

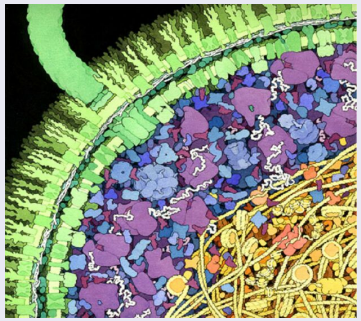
Shape is not considered

The shape of a biological entity plays a very important role in his interaction

Our Approach: Space, Motion and Shape

Space/Motion

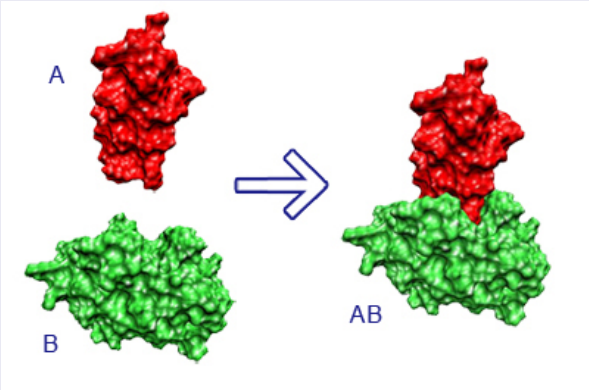
- The exact positions and velocities of all biological entities are known



http://www.anl.gov

Shape and communication

- Contacts (collisions) and shapes transformation determines biomolecular interactions



Our Approach: twofold

Shape Calculus

- Formal calculus for describing processes that are the shaped biological entities
- Language of specification: CCS-like
- Semantics of the evolution of a network of processes given by SOS-rules

BioShape simulation environment

- Distributed environment for simulation of a given biological scenario
- Language of specification: (Java/C++) coding
- Stub of all entities and simulation features of shapes, motion and collision detection embedded

Synergy between Simulation and Verification

Using the same system model (possibly at different levels of abstraction)

By simulation

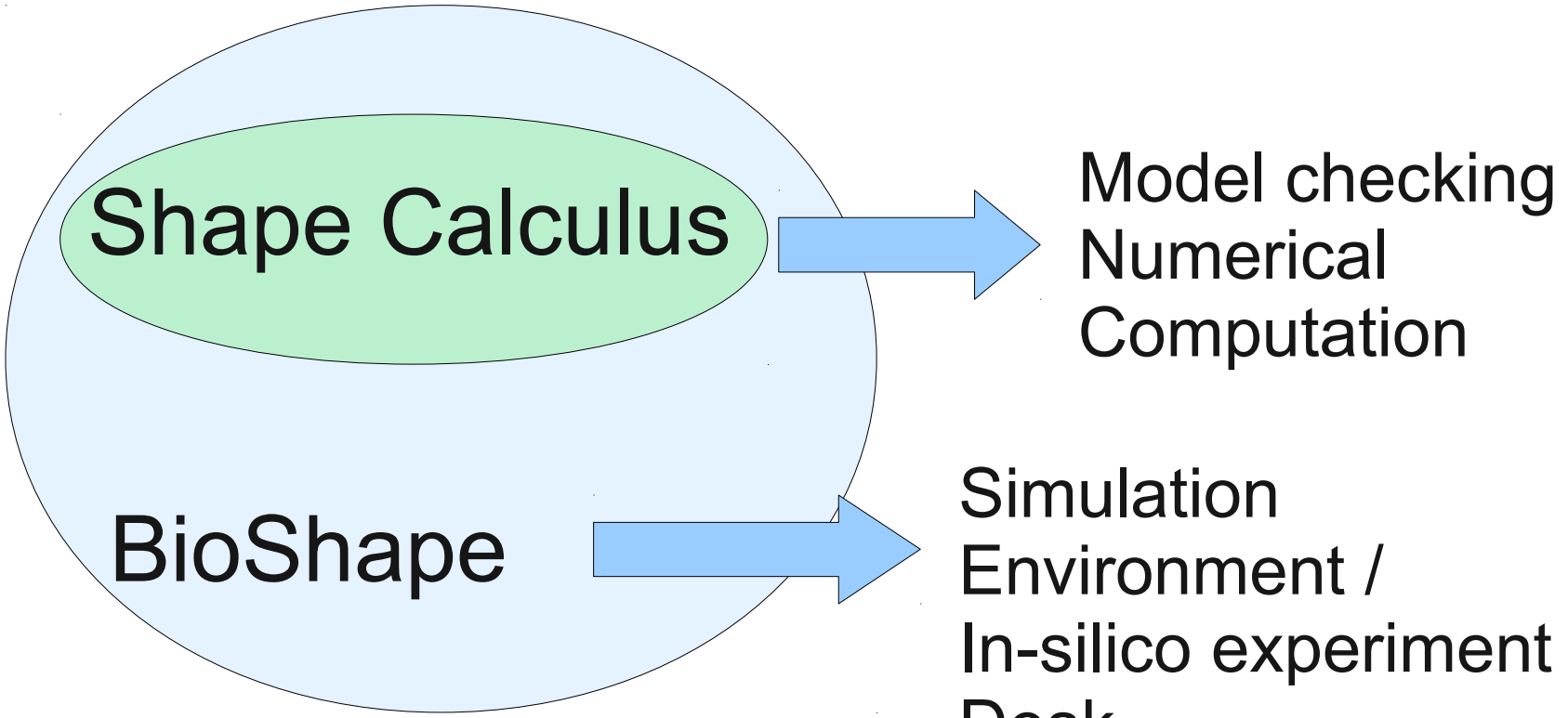
- Model validation and refinement by aggregation of simulation results
- Hypothesis testing rejecting unlikely scenarios
- Determination of one or more parameters by fitting known aggregated data, e.g. concentrations, trends, rates, etc ...
- Gain new biological information by perturbing a validated model, e.g. useful in drug design. In general could permit to avoid in vivo experiments

Synergy between Simulation and Verification

By Verification

- Sanity checks of the model using known biological properties, by model-checking or equivalence checking
- Computation of quantitative information if the model is quantitative, e.g. by diagnostic information of model-checking
- Gain information derivable form the model, e.g. by numerically computing the probability of a scenario if the model is probabilistic

Shape Calculus vs BioShape



Shape calculus

Main features:

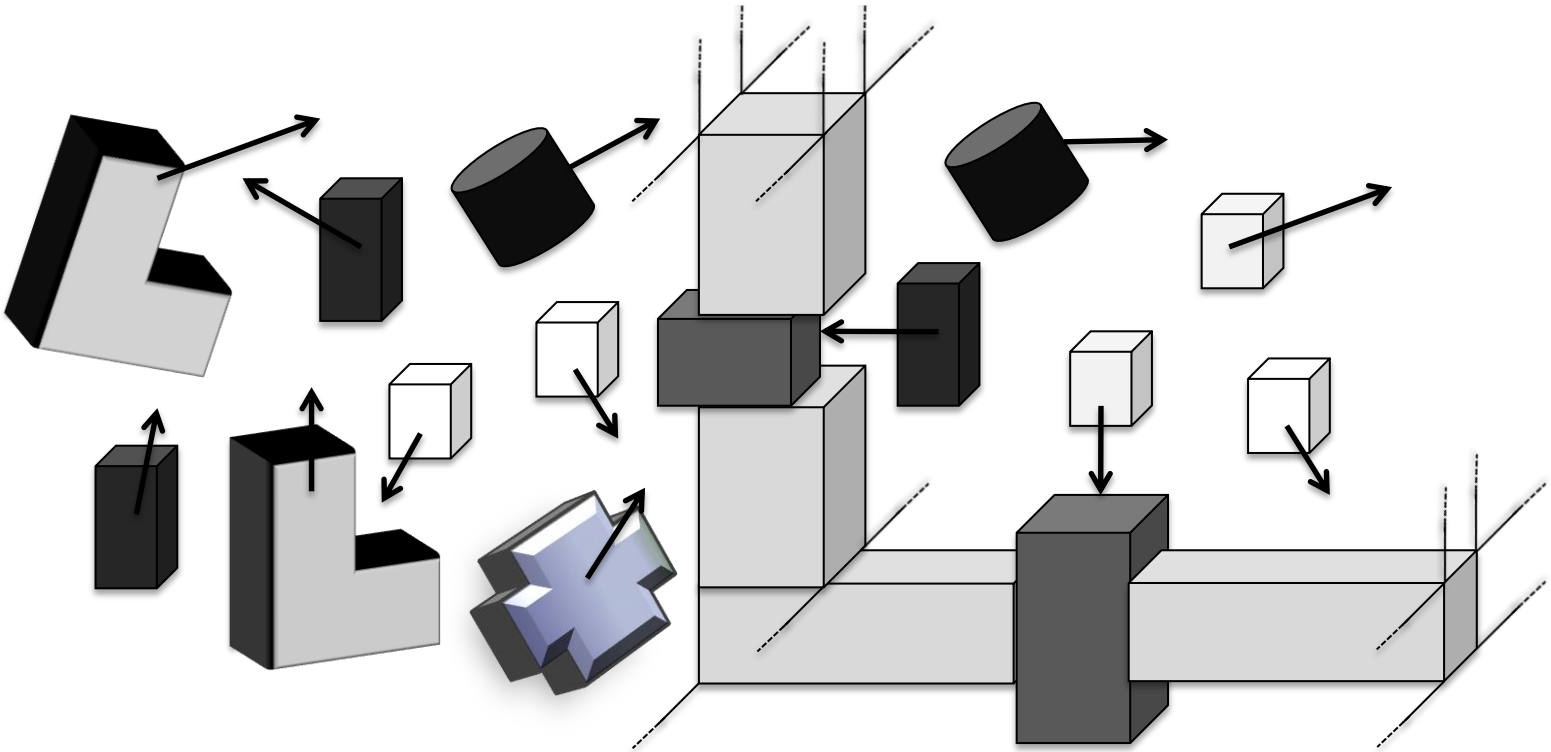
- 3D processes: a 3D shape with a behavior
- A 3D process is situated in space and move accordingly to its specific motion law
- Processes can collide and possibly bind
- The binding depends on channels $\langle a, X \rangle$, X is a portion of the surface of the process's shape in which the channel is "active"

Shape calculus

Main features:

- Compound processes can split weakly, by non-deterministically releasing a previously established bond, or "react", by splitting strongly in as many pieces as the products of the reaction are
- If communication (i.e. binding) does not occur, the collision of two 3D processes is considered elastic, i.e. the shapes bounce and proceed independently

A network of processes at a glance

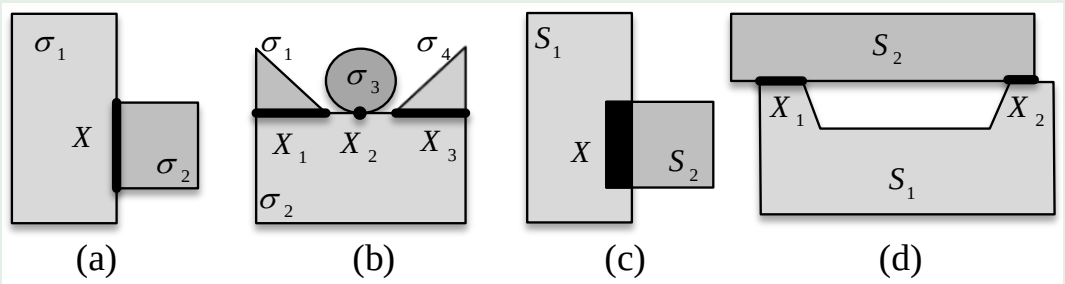


Shapes in Shape Calculus

Shape Syntax

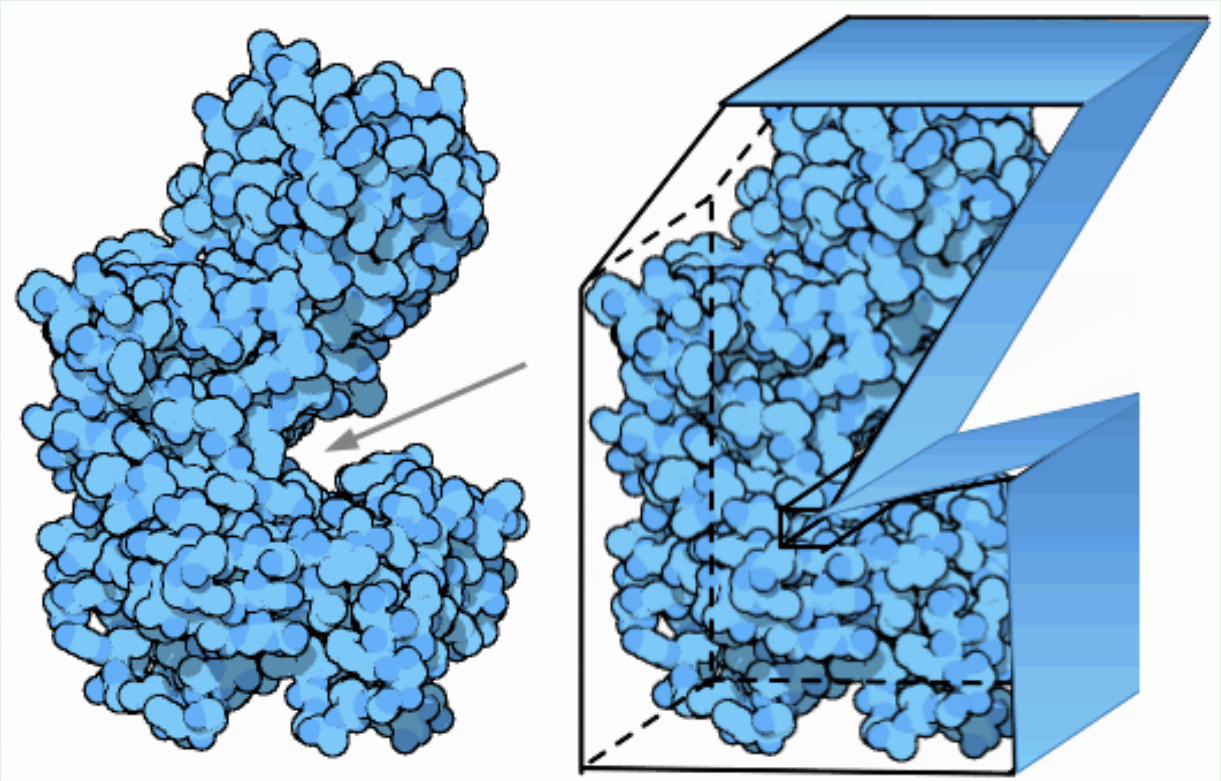
$S ::= \sigma \mid S \langle X \rangle S$ where $\sigma \in$ Basic and $X \subseteq \mathbb{R}^3$ is a *non-empty* set of points. The set X is intended to be the common surface on which the two shapes are attached.

Examples of compound shapes in 2D



Representation of enzymatic reaction in Shape Calculus

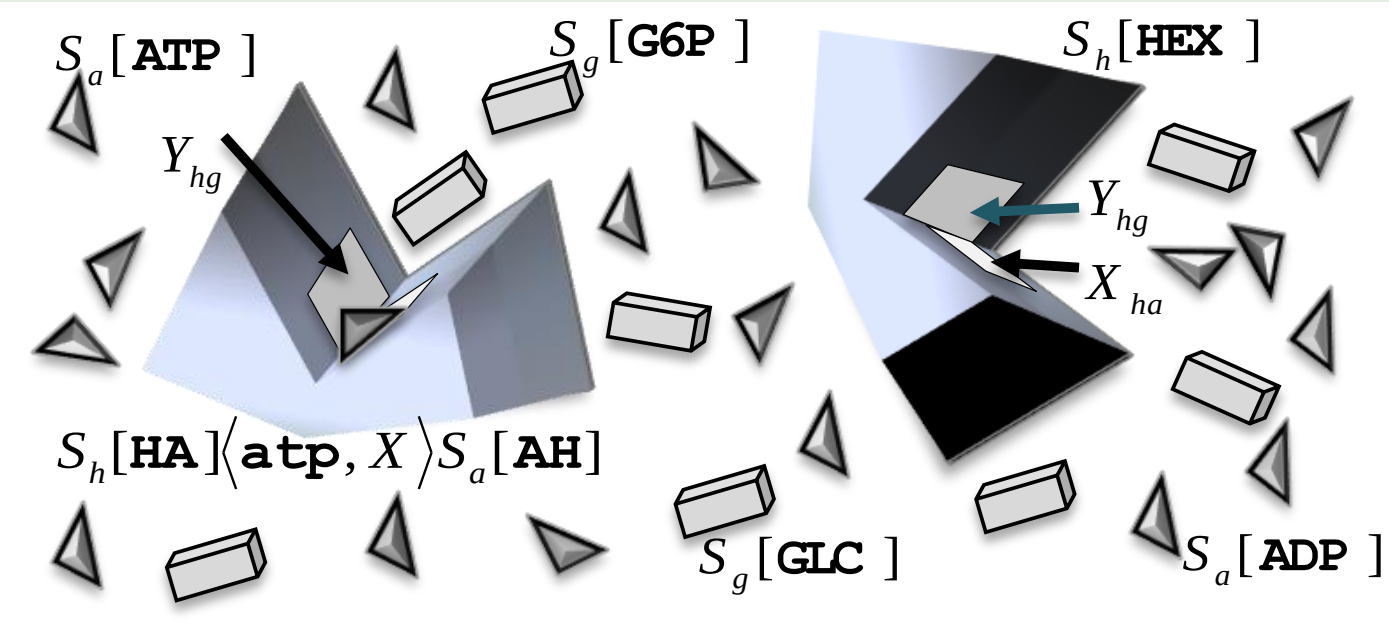
Shape approximation



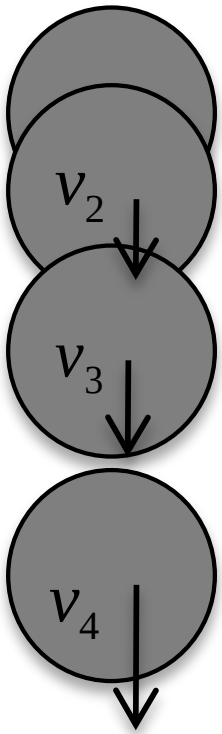
Examples

Representation of enzymatic reaction in Shape Calculus

Network and binding sites



Trajectories of Shapes



$$steer\ t_i S = [0, -\frac{1}{2} g(i+1)\Delta, 0] m/s$$

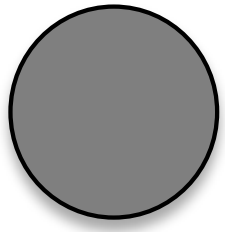
$$t_i = t_{i-1} + \Delta$$

$$\Delta = 0.05s$$

$$v_1 = steer\ t_0 S = [0, -0.245, 0] m/s$$

$$v_2 = steer\ t_1 S = [0, -0.49, 0] m/s$$

$$S =$$



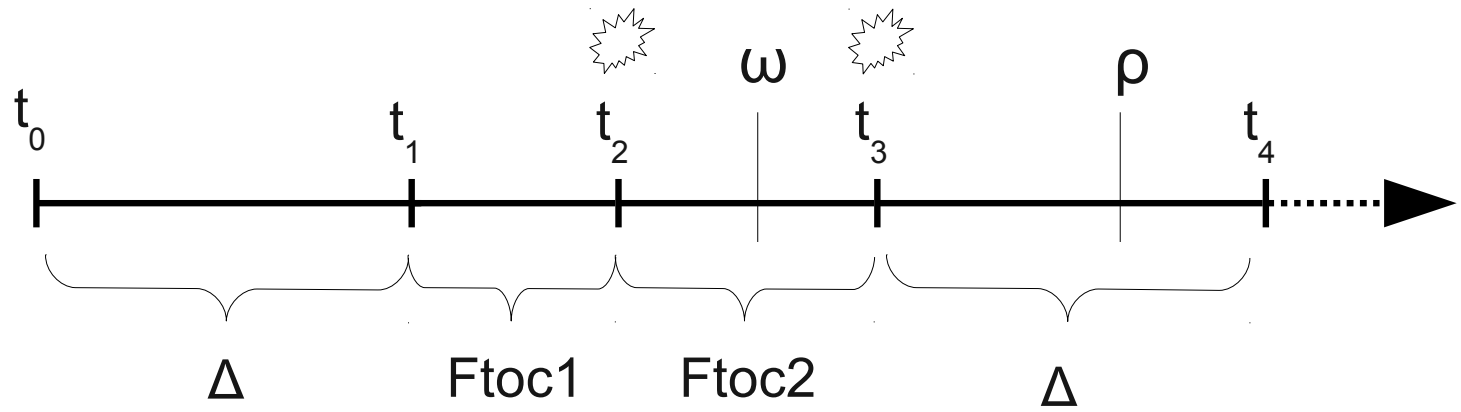
$$v_3 = steer\ t_2 S = [0, -0.735, 0] m/s$$

$$v_4 = steer\ t_3 S = [0, -0.98, 0] m/s$$

$$v = [v_x, v_y, v_z]$$

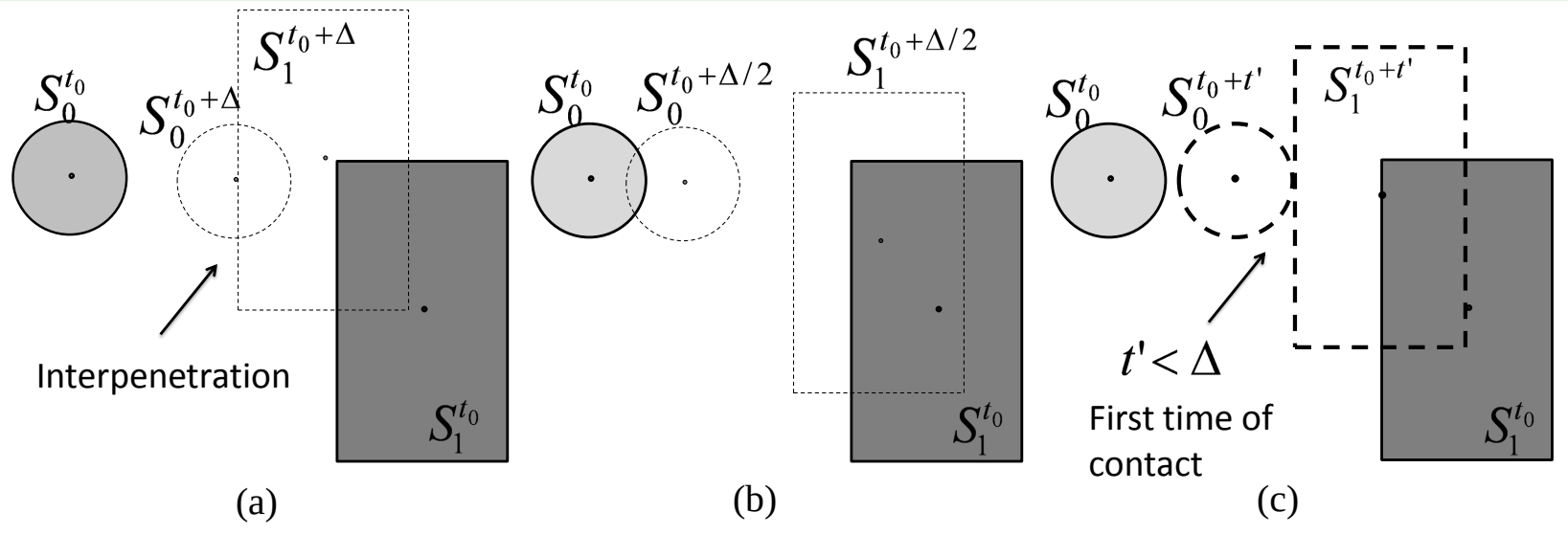
Time evolution and velocity update

- Time domain $\mathbb{T} = \mathbb{R}_0^+$ is then divided into an infinite sequence of movement time steps t_i such that $t_0 = 0$ and $t_i = t_{i-1} + \min(\Delta, \text{Ftoc})$.
- The updating of the velocities is represented by a function steer: $\mathbb{T} \rightarrow \text{Shapes} \hookrightarrow \mathbb{V}$ gives the velocity vector steer t S to assign to shape S at time t



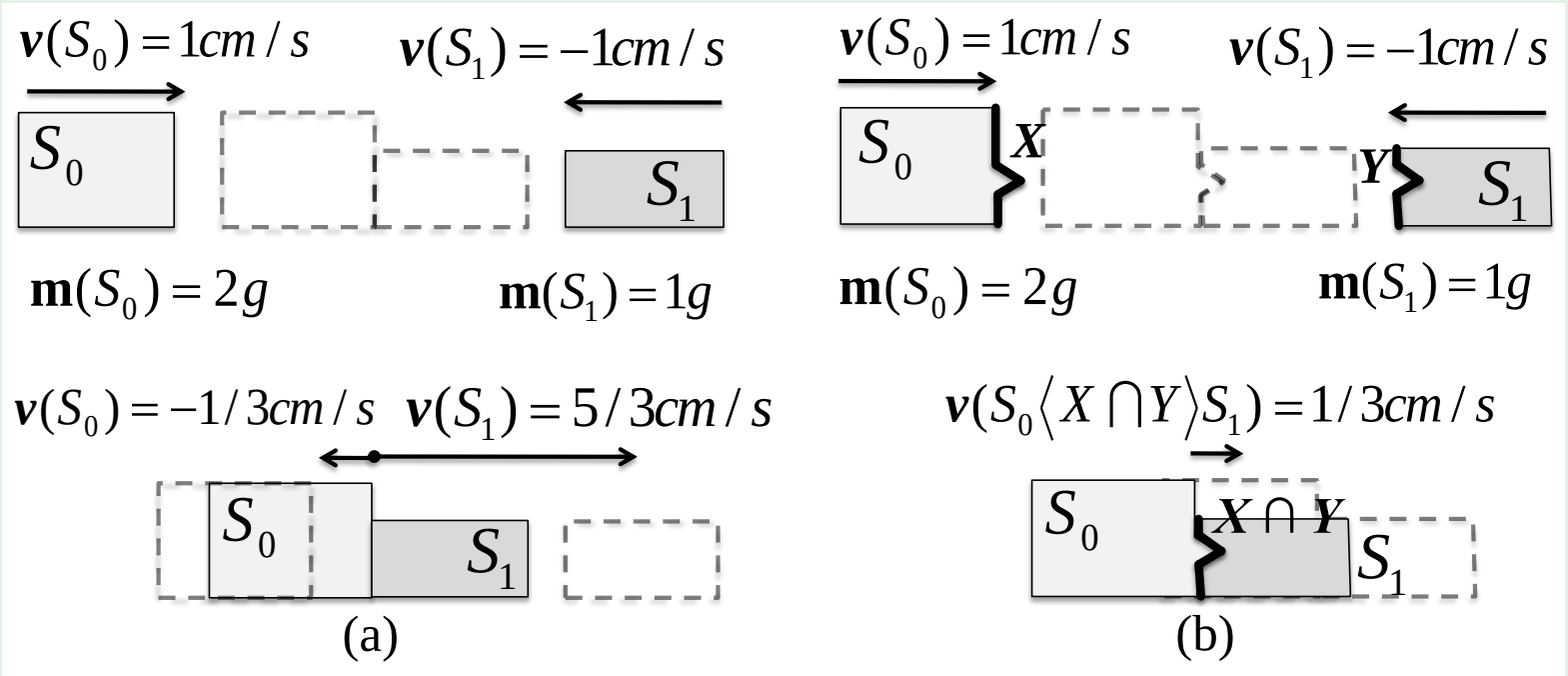
Collision Detection

First time of contact



Collision Response

Elastic and inelastic collision (one dimensional case)



Modeling Hexokinase behavior

The set $\mathbb{B}$ of *shapes' behaviors* is generated by the grammar

$$B ::= \text{nil} \mid \langle \alpha, X \rangle.B \mid \omega(\alpha, X).B \mid \rho(L).B \mid \epsilon(t).B \mid B + B \mid K$$

where $\langle \alpha, X \rangle \in \mathcal{C}$, L is a non-empty subset of $\mathcal{C}$ whose channels are pairwise incompatible, $t \in \mathbb{T}$ and K is a process name in $\mathcal{K}$.

The set 3DP of *3D processes* is generated by the following grammar:

$P ::= S[B] \mid P \langle a, X \rangle P$, where $S \in$ Shapes, $B \in \mathbb{B}$, $a \in \Lambda$ and $X \subseteq \mathbb{R}^3$ intersection between the surface active sites that are bound

Modeling Hexokinase in Shape Calculus

$S_h[\text{HEX}]$ where $\text{HEX} = \langle \text{atp}, X_{ha} \rangle.\text{HA} + \langle \text{glc}, Y_{hg} \rangle.\text{HG}$

Functional behavior of $\mathbb{B}$-terms

$$\textsc{Pref}_a \frac{\mu \in \mathcal{C} \cup \omega(\mathcal{C})}{\mu.B \xrightarrow{\mu} B} \qquad \textsc{Del}_a \frac{B \xrightarrow{\mu} B'}{\epsilon(0).B \xrightarrow{\mu} B'} \qquad \textsc{Sum}_a \frac{B_1 \xrightarrow{\mu} B'}{B_1 + B_2 \xrightarrow{\mu} B'}$$

$$\textsc{Def}_a \frac{B \xrightarrow{\mu} B'}{K \xrightarrow{\mu} B'} \quad \text{if } K \stackrel{\text{def}}{=} B \qquad \textsc{Spt}_1 \frac{L = \{\langle \alpha, X \rangle\}}{\rho(L).B \xrightarrow{\rho(\alpha,X)} B}$$

$$\textsc{Str}_2 \frac{L = \{\langle \alpha, X \rangle\} \cup L' \;\; L' \neq \emptyset}{\rho(L).B \xrightarrow{\rho(\alpha,X)} \rho(L').B} \qquad \textsc{Str}_3 \frac{B \xrightarrow{\rho(\alpha,X)} B'}{\rho(L).B \xrightarrow{\rho(\alpha,X)} \rho(L).B'}$$

Modeling HEX, ATP and Glucose behaviors

$$\mathrm{HEX} = \langle \mathrm{atp}, X_{ha} \rangle.\mathrm{HA} + \langle \mathrm{glc}, X_{hg} \rangle.\mathrm{HG},$$

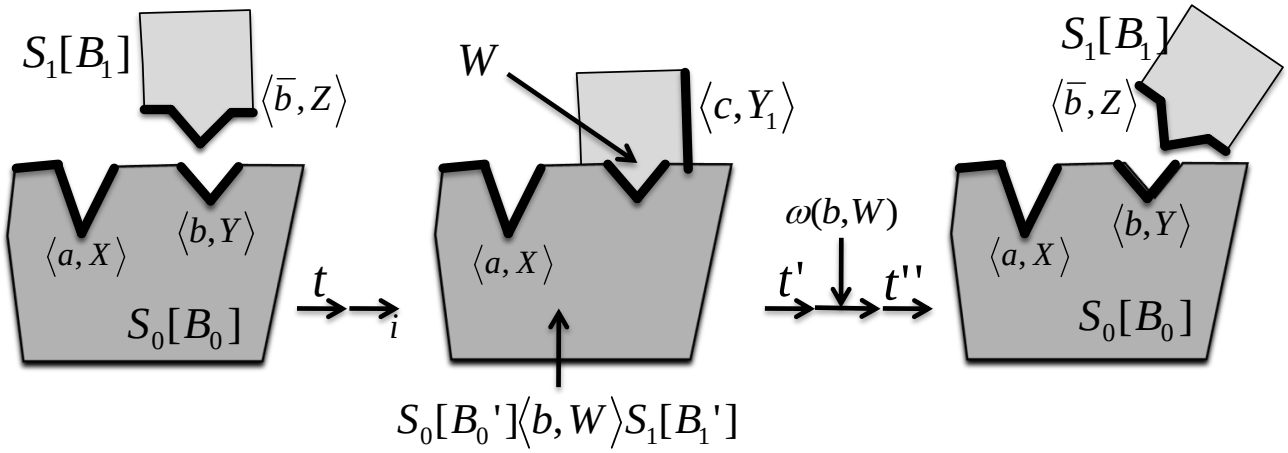
$$\mathrm{HA} = \omega(\mathrm{atp}, X_{ha}).\mathrm{HEX} + \epsilon(t_h).\langle \mathrm{glc}, X_{hg} \rangle.\rho(\{\langle \mathrm{atp}, X_{ha} \rangle, \langle \mathrm{glc}, Y_{hg} \rangle\}).\mathrm{HEX},$$

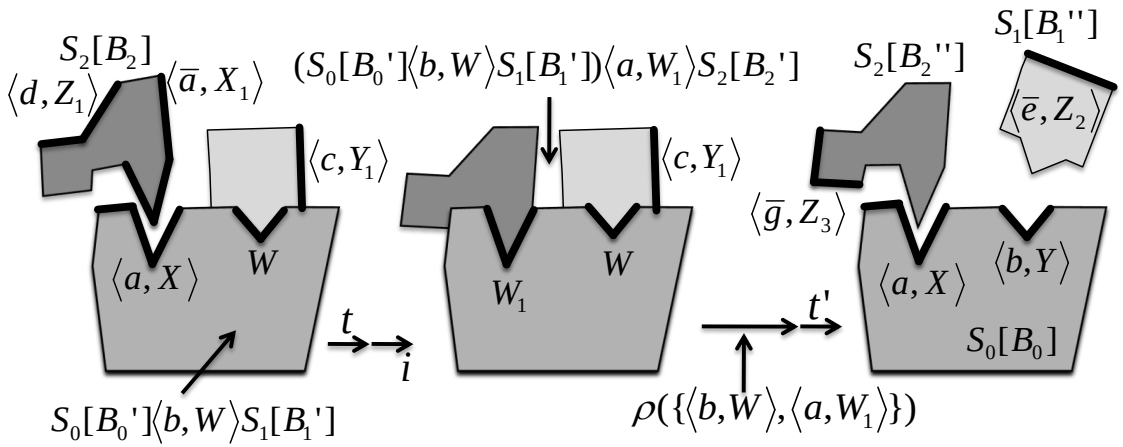
$$\mathrm{HG} = \omega(\mathrm{glc}, X_{hg}).\mathrm{HEX} + \epsilon(t_h).\langle \mathrm{atp}, X_{ha} \rangle.\rho(\{\langle \mathrm{atp}, X_{ha} \rangle, \langle \mathrm{glc}, Y_{hg} \rangle\}).\mathrm{HEX},$$

where X_{ha}, Y_{hg} are the surfaces of contact.

$$\mathrm{ATP} = \langle \overline{\mathrm{atp}}, X_{ah} \rangle.(\epsilon(t_a).\rho(\{\langle \overline{\mathrm{atp}}, X_{ah} \rangle\}).\mathrm{ADP} + \omega(\overline{\mathrm{atp}}, X_{ah}).\mathrm{ATP})$$

Binding and Splitting





Network of 3D processes

The set of networks of 3D processes is generated by the grammar:

$$N ::= \mathsf{Nil} \mid P \mid N \parallel N$$

Functional and Temporal semantics of the networks

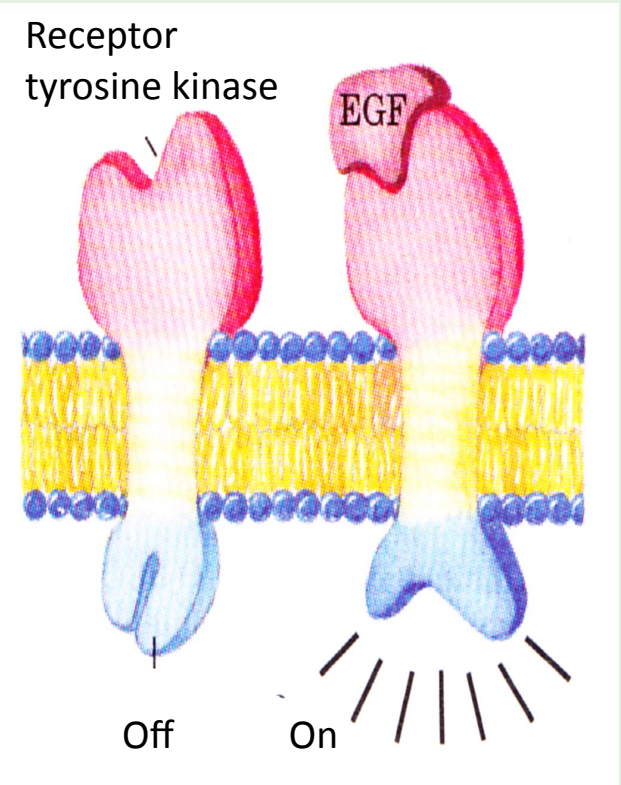
$$\textsc{Empty}_t \frac{}{\mathsf{Nil} \xrightarrow{t} \mathsf{Nil}} \qquad \textsc{Par}_t \frac{N \xrightarrow{t} N' \quad M \xrightarrow{t} M'}{N \parallel M \xrightarrow{t} N' \parallel M'} \qquad \textsc{Par}_{a1} \frac{N \xrightarrow{\mu} N'}{N \parallel N \xrightarrow{\mu} N' \parallel M}$$

Ongoing work on the Shape Calculus

Work in progress

- Manage the Shape transfomation
- Tayloring the Shape Calculus towards **verification** tools

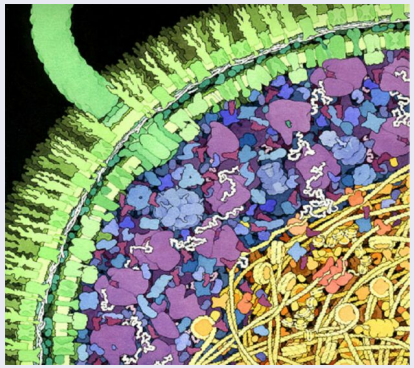
Biosignalling by shape transformation



BIOSHAPE: Geometrical 3D Space, Time and Shape

Space/Time

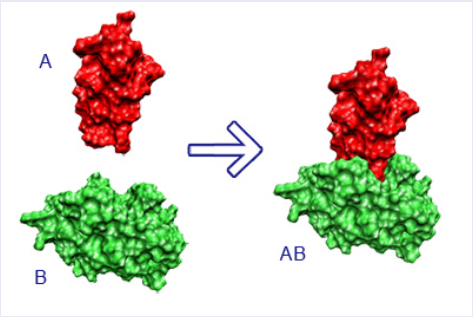
- 3D **geometrical** space, not only logical compartments
- Exact positions and velocities of all entities are tracked



http://www.anl.gov

Shape and communication

- Interactions can be collision-driven **or perception-driven**



BioShape key concepts

- 3D space
- Geometry
- Particle-based models with shapes
- (Personalized) Movement
- Interactions, both collision-driven and perception-driven
- Multiscale feature: a level for each considered scale
- Multiscale feature: levels are related by simple spatial/shape mapping functions

Motion of entities

- At the level of the **Shape Calculus** there is an all-knowing function that specifies movements for each shape at each time (too abstract concept)
- At the BioShape level, every entity has a strategic personalised class that implement its own motion law with its own needed parameters
- Thus, at the moment the behaviour is expressed in UML and then **coded** into the mentioned class
- Future Work: intermediate language (with spatial/values capabilities)

Behaviours of entities

- A high level behaviour (only interactions) can be expressed with the **Shape Calculus**
- However, it is not sufficient for specifying all complex aspects of the entities in a real in-silico experiment
- Again, at the moment the behaviour is expressed in UML and then **coded** into the class of the entity
- Future Work: intermediate language (with probabilistic/quantitative aspects)

Non-shaped actors of the simulator

- Concentration Grouped Entity: represents a concentration of something (of an enzyme, or an hormone, etc)
- Matrix Grouped Entity: represents a quantity of something (a molecule, a certain kind of cell) that is available in a certain position at a certain moment
- Gradient Grouped Entity: represents a field of something (electric charge, chemical attractors, ecc) at a certain moment
- Tuple Grouped Entity: represents a tuple space in which information defined in a policy is stored and read by the involved Actors
- Conservation Controller: controls that a global equilibrium is satisfied (with a given tolerance) by receiving by every involved Actor messages of changes

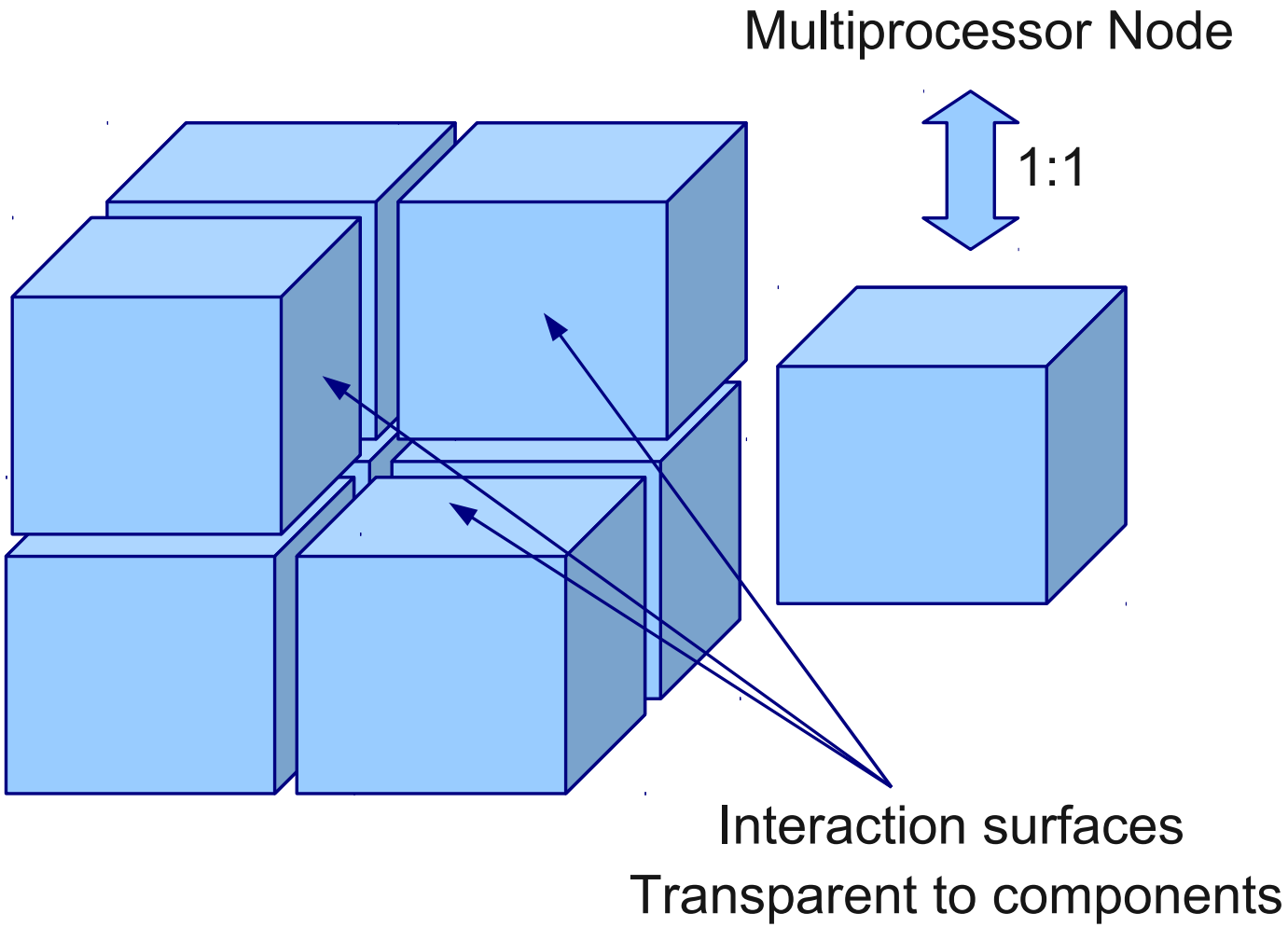
BioShape: current status of implementation

- Prototype implementation in Java (sufficient for testing non-crowded situations), version 0.9
- Prototype distribution of the computation using Agent-based approach - Hermes Platform developed at University of Camerino, untested
- Currently started a porting on C++/MPI over a multiprocessor platform - Multiple Instruction Multiple Data (MIMD), version 0.3

Splitting and Distribution

- The simulated volume is split in several parts, e.g. subcubes of the cube
- Each part has a coordinator that manages the entities at its slice virtual space
- The decomposition is completely transparent to components
- A global coordinator guarantees spatio-temporal consistency
- Each part runs on a different processor over a MIMD architecture

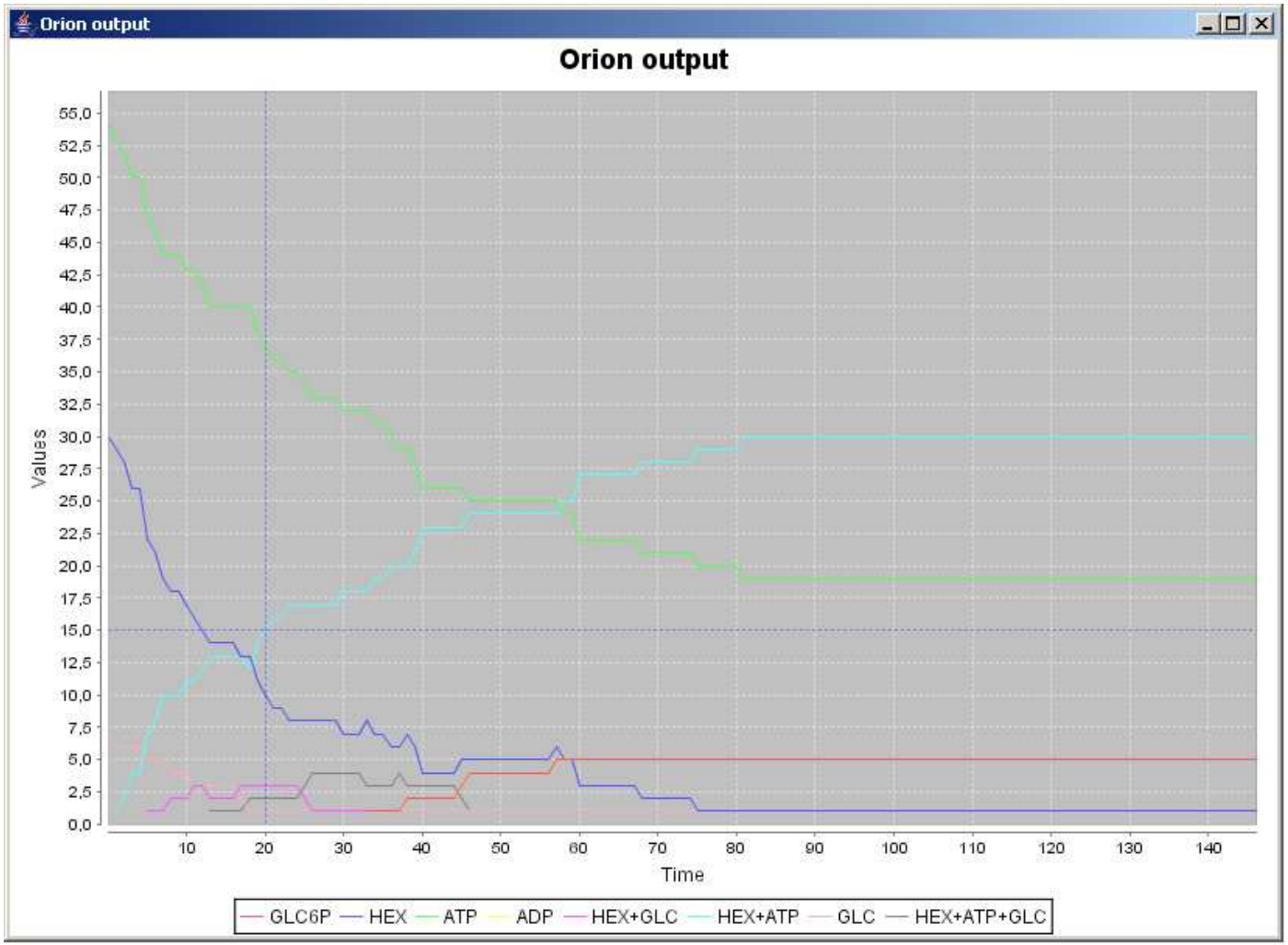
Splitting


Multiprocessor Node
1:1
Interaction surfaces
Transparent to components

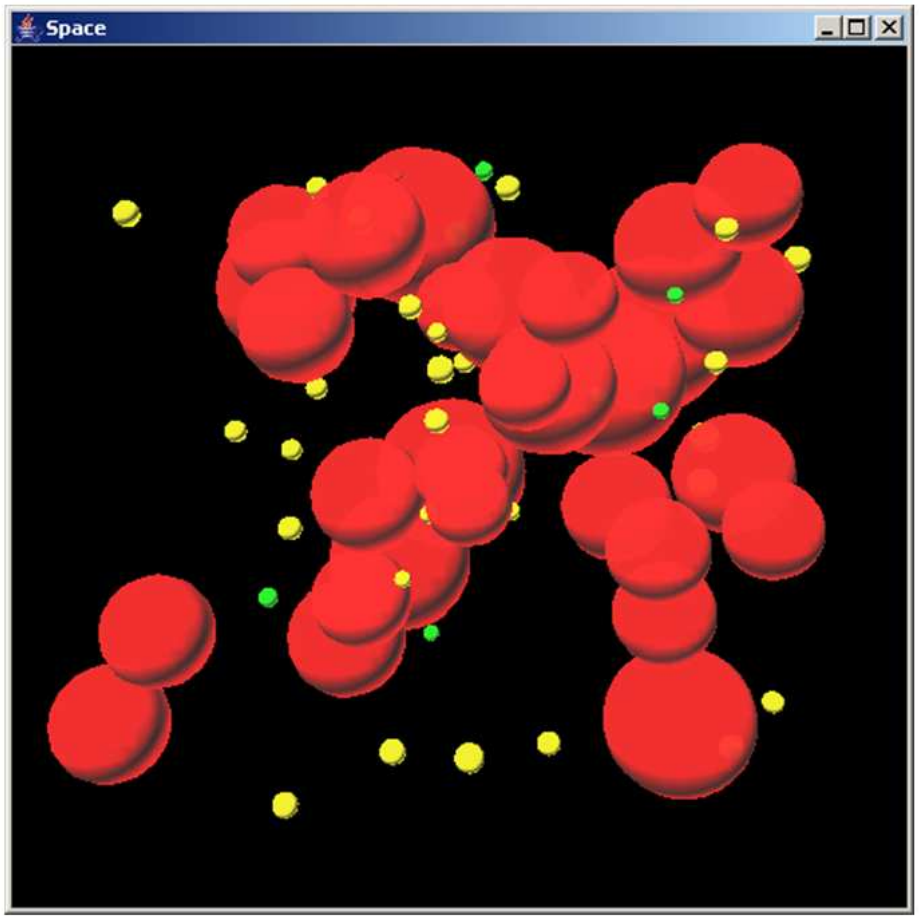
Prototype example run

- Activate only the first reaction of the glycolysys (i.e. the one mediated by Hexokinase)
- Small volume of cytoplasm ($10^{-18}\ell$)
- Initial 0.0112817 $mMol/\ell$ concentration of glucose (GLC), corresponding to 6 molecules
- 54 molecules of ATP and 30 of Hexokinase (HEX)
- After 33ms of simulation time the first molecule of glucose-6-phosphate (GLC6P) appears
- After nearly 60ms we can find 5 molecules of GLC6P in our little portion of cytoplasm. After about 80ms the system stabilizes.

Run results


Orion output
Orion output
Values
Time
55,0
52,5
50,0
47,5
45,0
42,5
40,0
37,5
35,0
32,5
30,0
27,5
25,0
22,5
20,0
17,5
15,0
12,5
10,0
7,5
5,0
2,5
0,0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
GLC6P
HEX
ATP
ADP
HEX+GLC
HEX+ATP
GLC
HEX+ATP+GLC

Visualization - Post-processed



Bone Remodelling

Biological facts:

- Old bone is continuously replaced by new tissue
- Mechanical integrity of the bone is maintained
- In healthy conditions: no global changes in morphology/mass
- A **multiscale** phenomenon: macroscopic behaviour and microstructure strongly influence each other

Case study

Bone remodelling is a suitable case study to test the multiscale capabilities of models and simulators.

Bone Remodelling

Pathological conditions can alter bone remodelling balance. In particular:

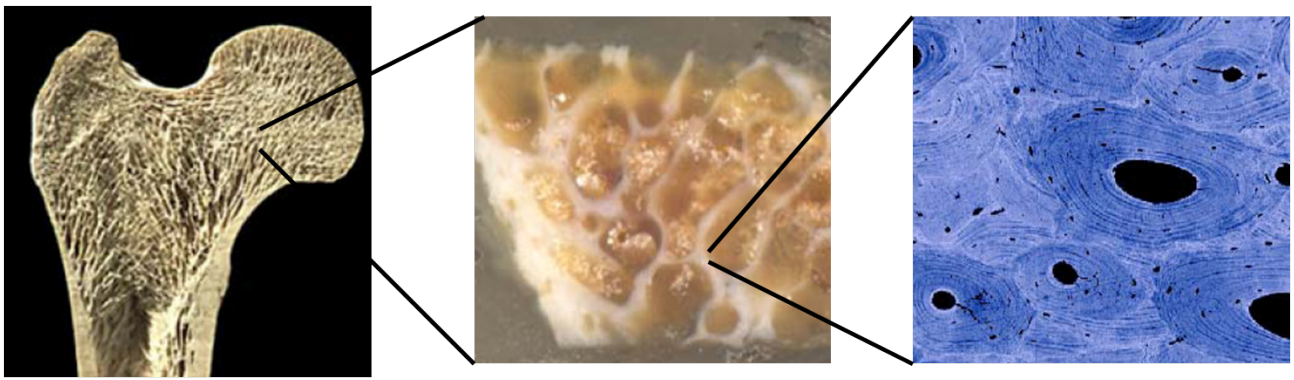
- Osteoporosis: the balance is negative, thus there is higher risk of spontaneous bone fractures
- Bone metastases: secondary tumours forming in the skeleton from malignant tumour cells in blood
- Renal osteodystrophy: defective mineralization resulting from renal disease

Social and Clinical Relevance

A better understanding of the bone remodelling process and of how it is influenced in these and other diseases has social and clinical relevance.

Bone Remodelling scales

- The phenomenon has been studied at different scales
- Typically **from** organ-level, where fractures take place
- **Towards** cells/molecules, where mechanisms are present that influence the whole bone structure and resistance



a. Femur cortical and trabecular bone.

b. Cut surface of bone: cortical and trabecular bone surrounded by marrow tissue

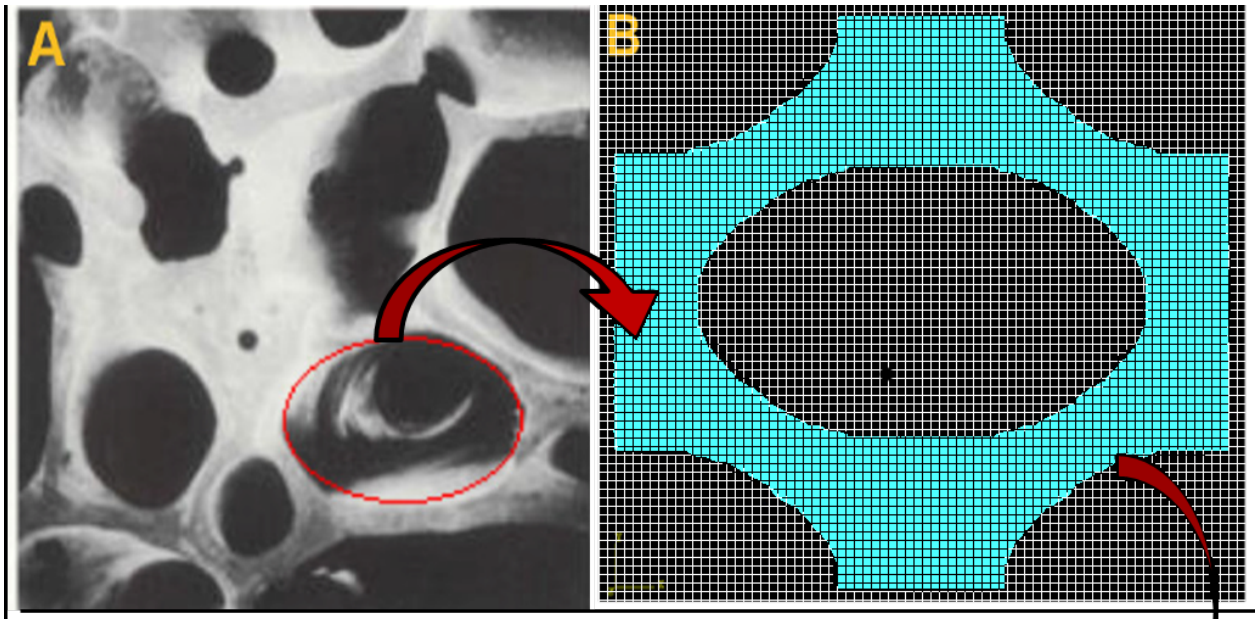
c. Haversian systems in cortical bone. The central canal supplies blood, while the small black dots are spaces containing osteocytes.

In this case study:

- We focus on two levels: tissue and cellular
- Organ-level has been intensively studied in the past
- Existing models of different scales can be integrated, but their coupling is a difficult and tricky task
- We show how to model phenomena of different scales into a unique integrated/homogenized framework
- The simulator BioShape is intended to support this uniform approach

Tissue Scale

- A fixed 3D lattice can be defined
- Each vertex of each cube has a **mineralization** value
- Depending on a given threshold, some cubes are at the **surface**, where remodelling happens



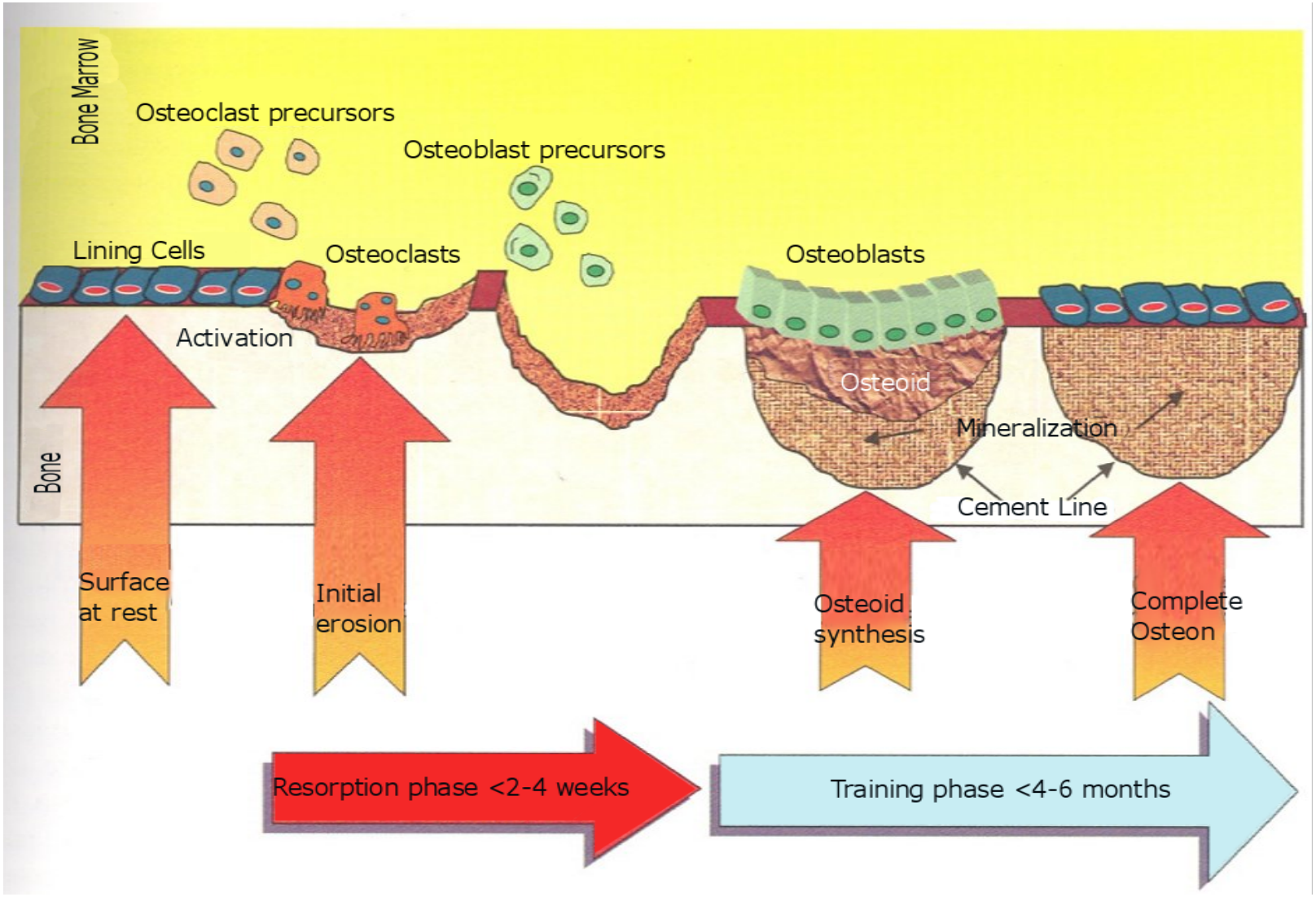
Cellular Scale: Bone Multicellular Unit Animation

A qualitative visual description of the bone remodelling at the cellular level can be found at

`http://courses.washington.edu/bonephys/physremod.html`

© Susan Ott, Associate Professor, Department of Medicine, University of Washington

BMU time scale



Multiscale modelling: coupling two existing models

- Typically, an **integration** approach is attempted
- Two different models of two different scales are taken into account
- A communication between them is somehow constructed
- Adjustment/Abstractions are performed, with possible information loss
- Let us see an example in our bone remodelling scenario

Tissue scale: Meshless Cell Method

- The Meshless Cells Method (MCM) is a mathematical model based on the 3D lattice defined on the tissue
- INPUT: an external field of forces applied to the bone (considered invariant in an in-silico experiment)
- INPUT: degree of mineralisation of each point of the 3D lattice
- OUTPUT: deformation of the bone and possible damages that might occur: both micro-fractures and fractures

Cellular scale: ODE model

- For each cell of the 3D lattice, the average density of mineralization is calculated
- Considering the mechanical force applied to the cell (derived from tissue model)
- Considering other internal systemic factors: osteoblastic and osteoclastic activity, grown factor values, local regulators and hormones
- It is determined the new average density of the cell

Coupling the two models

1. Initial values of cell densities is loaded
2. MCM is applied for deformation and damage detection
3. For a percentage of surface cells, for each damaged section, the ODE method is performed to determine new density values
4. Goto 2

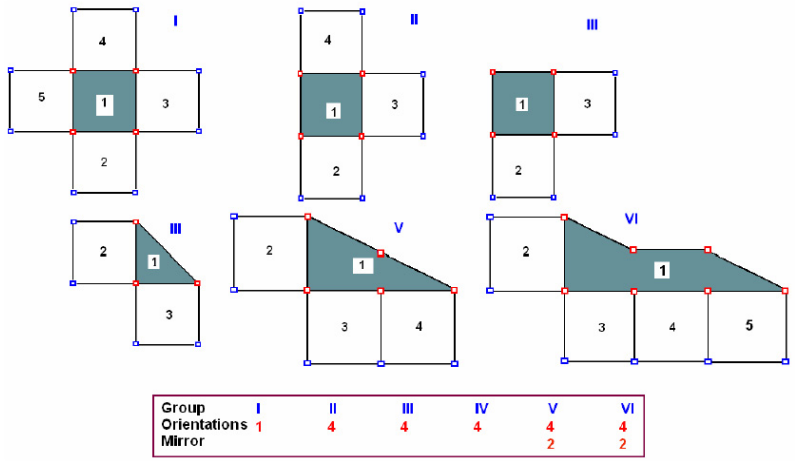
Currently our group is developing a scalable approach to this coupling using nVidia Graphics Processing Units (GPUs) and CUDA (Computer Unified Device Architecture) to resolve all the ODE computations in parallel

Coupling the two models

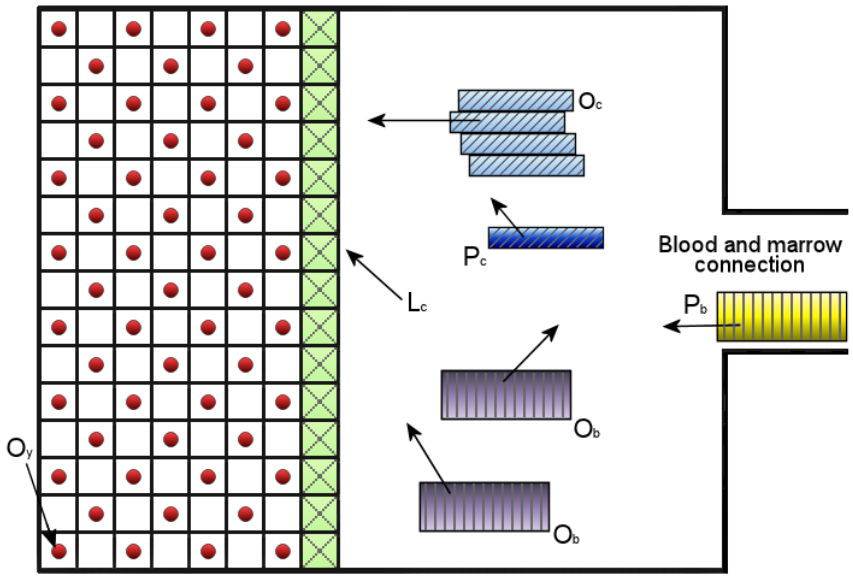
- Problems in determining the correct time scale
- Artificial random choice of the cells with null-balance bone remodelling
- Transfer of data between two different machines hugely impacts efficiency
- Data transformation and exchange protocol must be provided

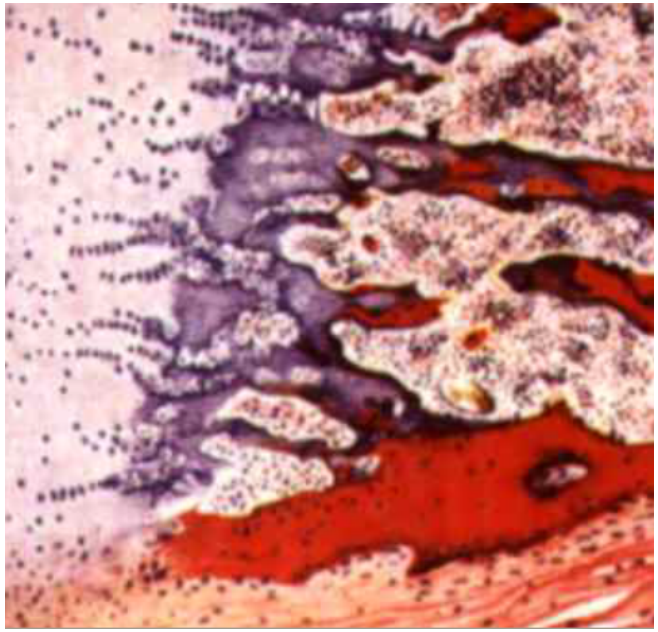
Tissue modelling in BioShape

- We can use the 3D lattice of the MCM method to define cube shapes that do not move
- Each cube is an entity holding the information about the mineralization
- The surface cubes can be decomposed in a finite number of pieces to represent any linear cut of the surface:



BMU modelling in BioShape





BMU modelling in BioShape

Current set of entities:

- Cell (mineralized or not, possibly containing an Osteocyte)
- PreOsteoclast
- AggregatedPreOsteoclast
- Osteoclast
- OsteoclastPathCollector
- RANK-L
- OPG
- PreOsteoblast
- Osteoblast

Coupling the two representation within BioShape

Cycle:

- for each shape at the tissue scale there is a mineralization density value
- each activated shape for remodelling (randomly chosen or determined by the model of forces) is simulated **at the cellular level** with the systemic factors personalised for that shape
- the new value for the density is passed to the tissue model that updates the border surfaces of the remodelled shape (by shape updating)
- re-apply the tissue model to the new data to determine how the changes of density modified the effect of mechanical stimuli

Coupling the two representation within BioShape

The time scale is different at the two levels:

- cell events occurs at a pace of days
- tissue timing is in the order of months
- the simulator basic time step is in the order of a (simulated) day
- when a (simulated) month elapsed, one step of the tissue level is performed

ToDo

- Make the design of a simulation simpler, with a limited number of "handles" to manipulate
- Refine the details of the models
- First validations with available qualitative and quantitative biological information
- Interact with biologists and bioengineers for possibly acquiring suitable qualitative and quantitative information from in-vitro experiments, for validation
- Start using the simulator for hypothesis testing, perturbation analysis, etc . . .

Acknowledgements

LitBio: Laboratory of Interdisciplinary Technologies in Bioinformatics
http://www.litbio.org/

CoSy Research group: http://cosy.cs.unicam.it

Credits

Shape Calculus Design

Flavio Corradini, Emanuela Merelli, Maria Rita Di Berardini, Luca Tesei, Ezio Bartocci

BioShape Design and Development

Flavio Corradini, Emanuela Merelli, Luca Tesei, Diletta Cacciagrano, Federico Buti, Carmine Oliva

Credits

BMU Design and Development in BioShape

Luca Tesei, Diletta Cacciagrano, Federico Buti, Vincenzo Ciro Addeo, Gaston Alanis, Luigi Ambruoso, Andrea Piermarteri, Matteo Rucco

CUDA group

Luca Tesei, Diletta Cacciagrano, Federico Buti, Leonardo Vito, Martino Pani, Nicola Paoletti, Matteo Rucco, Daniele Senigagliesi

Thank you for your attention