Inductive Process Modeling for Learning the Dynamics of Biological Systems

Sašo Džeroski
Department of Knowledge Technologies
Jozef Stefan Institute, Ljubljana, Slovenia

Inductive Process Modeling for Systems Biosciences

Sašo Džeroski
Department of Knowledge Technologies
Jozef Stefan Institute, Ljubljana, Slovenia

Systems Sciences: What do they study?

Systems sciences

study complex systems in nature and society

- Behavior/Dynamics: State of system changes
- Structure: Components & interconnections
- Environment: Inputs & outputs
- The system is more than the sum of its parts

Systems biosciences: Study biological systems

- Systems ecology (macro-scale; populations)
- Systems biology (micro- to meso-scale; molecular to organismal)

Systems Sciences: How do they do it?

A key concept/tool in systems sciences:

Model of the system:
Simplified (formal) representation of the system

The use of models

Understanding the system studied

- Structure (components, connections)
- Behavior

Predicting the behavior of the system

Achieving desired behavior

Systems Sciences: The tasks

The central task in systems sciences:

Modeling: Constructing a model of the studied system, in a given formalism, including structure and parameters

Typically done manually by an expert or a team thereof

Related tasks in systems sciences:

Analysis: Given model; examine possible behaviors

How model parameters/structure influence behavior

Identification: Given observed behavior; generate model

Control: Given model, desired outputs; generate inputs

Design: Given desired behavior, generate model

The diversity of Systems Sciences: System representation and modeling formalisms

The model of a system describes how the **system state changes/evolves** over some **context**

Context space: **time**/space/space-time (discr./cont.)

State space: System variables (discrete vs. **continuous**)

Evolution rules:

- **Deterministic** (one possible evolution in time, approximates avg. behavior of system)
- Stochastic (closer correspondence with nature, e.g. discrete/small number of molecules and stoch. react.))

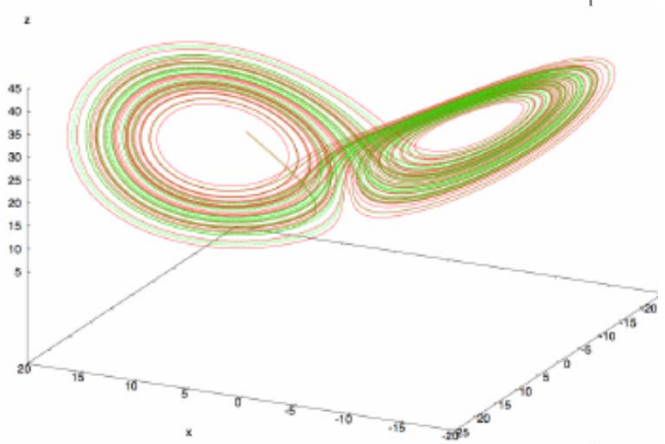
The prototypical case: **systems of ODEs**

Two Models of Dynamic Systems: An abstract and a real-world example

- The Lorenz system: Atmospheric convection

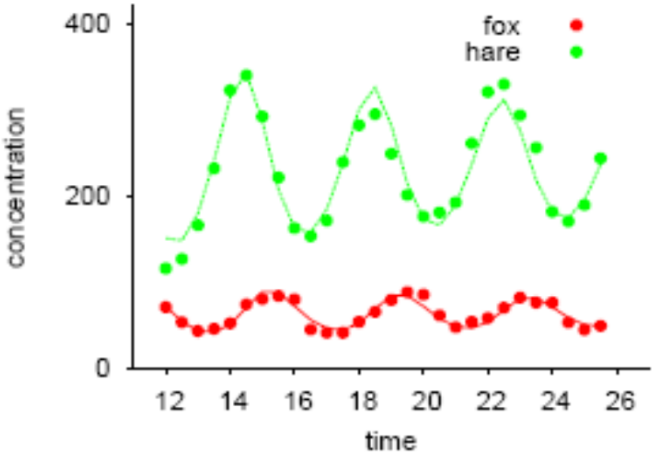
$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$

σ	ρ	β
10	28	8/3



- Lotka-Volterra: Predator-prey interaction in ecology

$$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$$
$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$



Systems Sciences: The challenge

Complexity of systems studied **increases**

The **models** that are used in systems sciences are mostly **constructed manually**

Manual modeling is a knowledge-intensive, time-consuming (and thus **expensive**) task

This represents a **bottleneck** in the further development of systems sciences

The Knowledge-Driven Modeling Paradigm

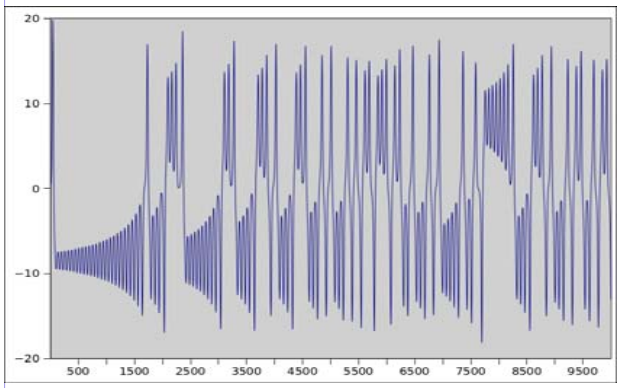
In knowledge-driven (theoretical) modeling:

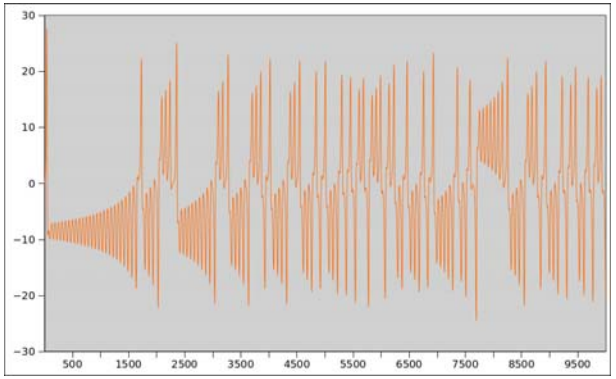
1. Expert derives a proper model (structure)
   - Based on domain-specific knowledge and
   - Knowledge of modeling formalism

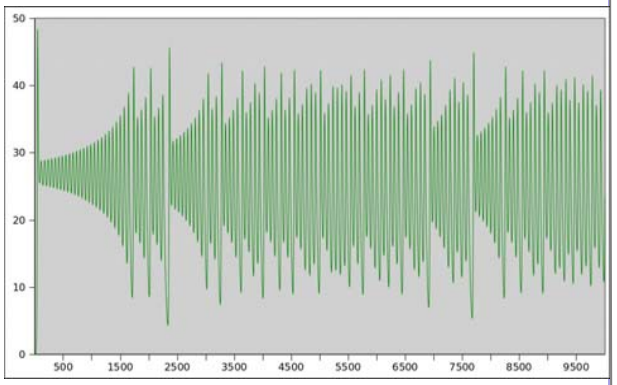
2. Typically, both the structure and parameters of the models are derived by the expert from
   - Knowledge about processes (e.g., bear reproduction)
   - Knowledge about process rates/parameters (e.g., birth rates = average number of cubs/litter)

Data-Driven Modeling: An Example

- Input: Observed behavior of dynamic system







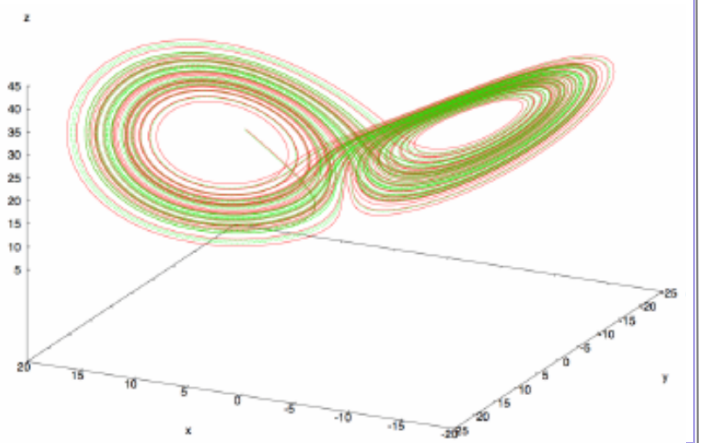
- Output: Set of ordinary differential equations(ODEs)

$$\frac{dx}{dt} = \sigma(y - x)$$

$$\frac{dy}{dt} = x(\rho - z) - y$$

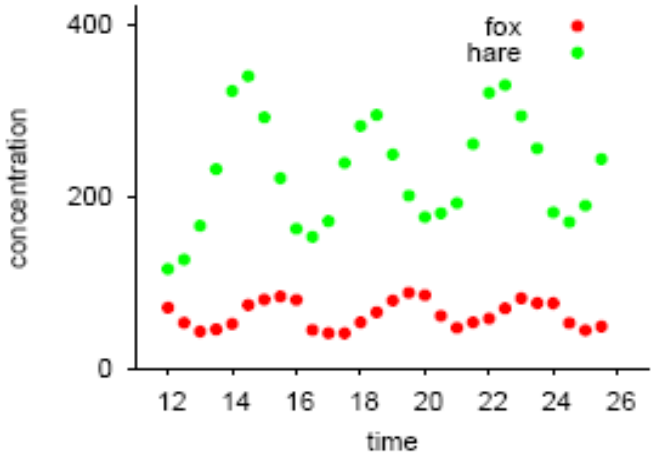
$$\frac{dz}{dt} = xy - \beta z$$

σ	ρ	β
10	28	8/3



Learning Models of Dynamic Systems: A Real-world Example from Ecology

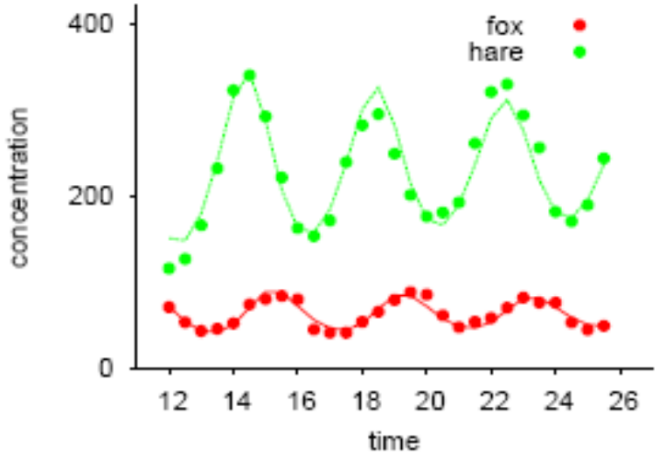
- Input: Observed behavior of dynamic system



- Output: Set of ordinary differential equations(ODEs)

$$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$$

$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$



Learning Models of Dynamic Systems: The task

- Given example behavior(s) of a dynamic system
  - Measurements of system variables
  - Over the course of time

Time	System variables			
	v_1	v_2	$\ldots$	v_n
t_0	$v_{1,0}$	$v_{2,0}$	$\ldots$	$v_{n,0}$
t_1	$v_{1,1}$	$v_{2,1}$	$\ldots$	$v_{n,1}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_m	$v_{1,m}$	$v_{2,m}$	$\ldots$	$v_{n,m}$

- Find a set of ODEs (structure and parameters)
- That describe the dynamics of the system (fit the observed behavior(s))

The Data-Driven Modeling Paradigm

In data-driven (empirical) modeling:

1. (Many) Different model structures (from a given class) are considered in a trial& error fashion

2. A model = structure + parameter values that fits the data best is returned

System identification typically

- Fixes the models structure, finds the parameters (parameter identification)
- Considers linear model structures

Data-Driven or Knowledge-Driven Modeling?

In knowledge-driven (theoretical) modeling:

- A lot of domain knowledge is needed, but
- Little (if any) data are needed and
- The result is an understandable model that makes sense from the domain point of view

In data-driven (empirical) modeling:

- A lot of data of good quality are needed, but
- Little (if any) domain knowledge is needed
- However, the result is most often a model that makes no sense from the domain point of view (or is even not understandable)

Data-and-Knowledge-Driven Modeling!

- We would like to have the best of both worlds and integrate knowledge- and data-driven modeling
- We would like to flexibly trade-off between data and domain knowledge and handle
  - Lots of knowledge and little data or
  - Lots of data and little knowledge, as the case may be
- We would like to have models that fit the data well
- But are also understandable and make sense from the domain point of view

Data-and-Knowledge-Driven Modeling!

To integrate knowledge-driven and data-driven modeling, we will need

- A formalism for representing models and domain knowledge
  - The formalism has to be powerful and allow precise modeling and simulation of dynamic system
  - It has to be understandable
- Methods for learning models from domain knowledge and data
  - That will take data and domain knowledge as input
  - Will produce accurate models in the formalism, and fast

Knowledge Representation: Dynamic Systems

Dynamic systems consist of

- **Entities** and
- **Processes**

These are very general notions and key concepts in ontological representations of the world

E.g., the Basic Formal Ontology is a framework that consists of a number of ontologies that describe

- **Continuants**, i.e., entities
- **Occurents**, i.e., processes

Our formalism: Process-Based Models

Process-based models have two major components

- **Entities** and
- **Processes**

Entities participate in **processes**

Each process-based model, consisting of completely specified processes, corresponds to a system of ODEs

The Es and Ps specify the qualitative aspect of the model

The ODEs specify the quantitative aspect of the model

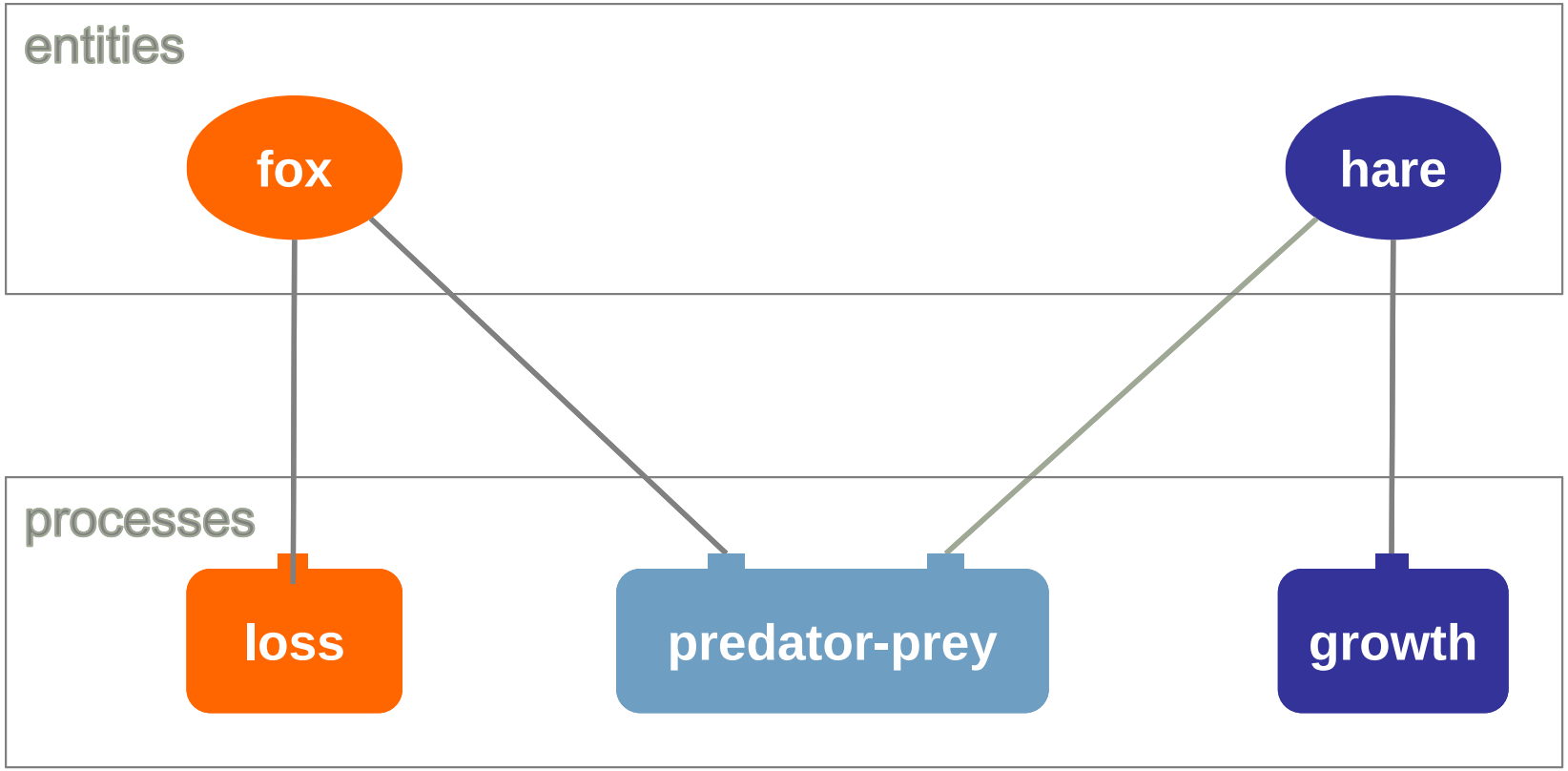
Process-based Models (PBM)

- Integrate two aspects of equation-based models
  - Quantitative aspect: ODEs with given structure and parameter values that allow simulation

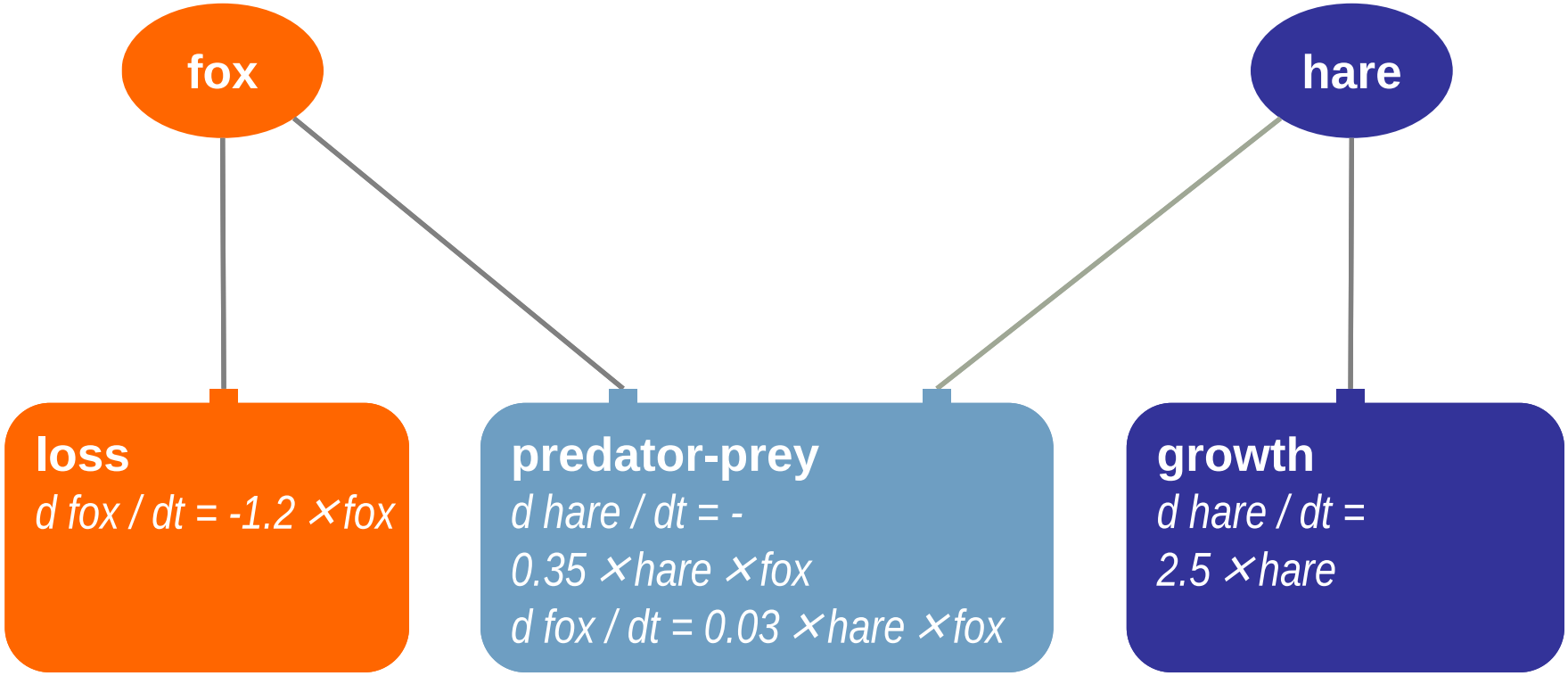
$$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$$
$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$

  - Qualitative aspect: explanation of the structure of the modeled systems in terms of entities and processes
- In equations: Variables correspond to **entities**, terms correspond to **processes**
  - Exponential growth of hare population
  - Exponential loss of fox population
  - Predator-prey interaction between the two species

PBM: Qualitative Aspect



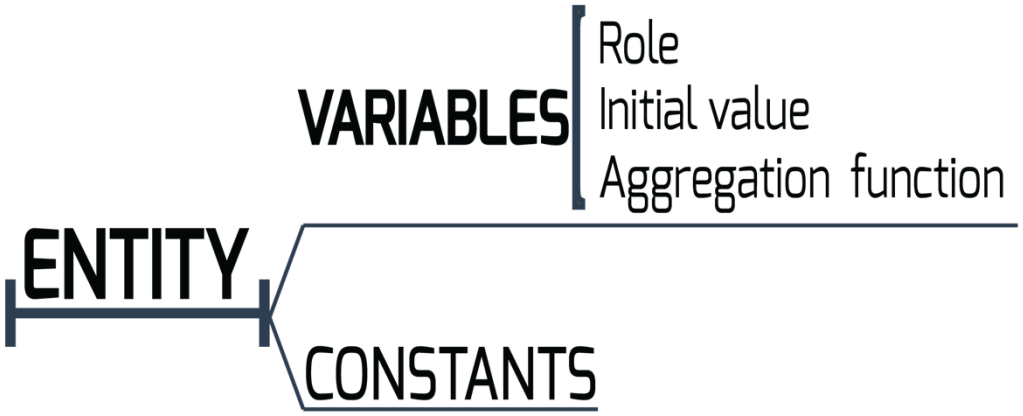
PBM: Quantitative Aspect



Entities in Process-Based Models

Entities (e.g., person) have as properties **constants** (e.g., gender) and **variables** (e.g., weight or height)

The latter have an *initial value* that changes over time



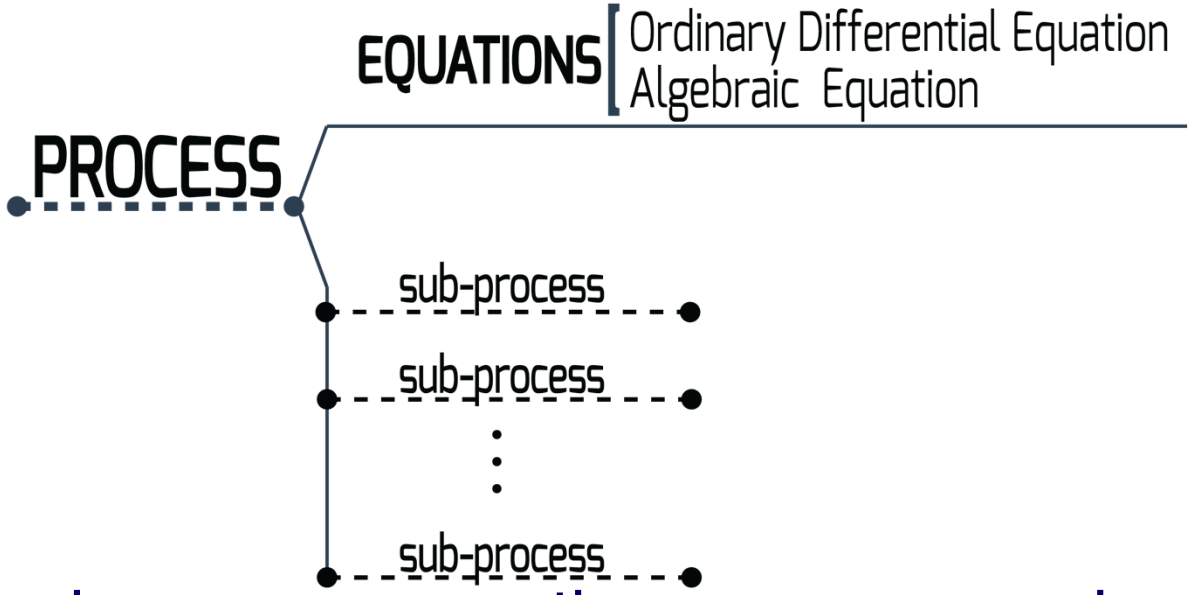
The variables appear in (OD) equations: Their *role* determines whether they are endogenous (system) vars. or exogenous (input) variables

Entities participate in processes and are influenced by them. *Agg. Function* = How influences are combined

Processes in Process-Based Models

Processes have entities as participants, which they influence

Besides participants, **processes** have equations, which specify their kinetic rates, and **sub-processes**



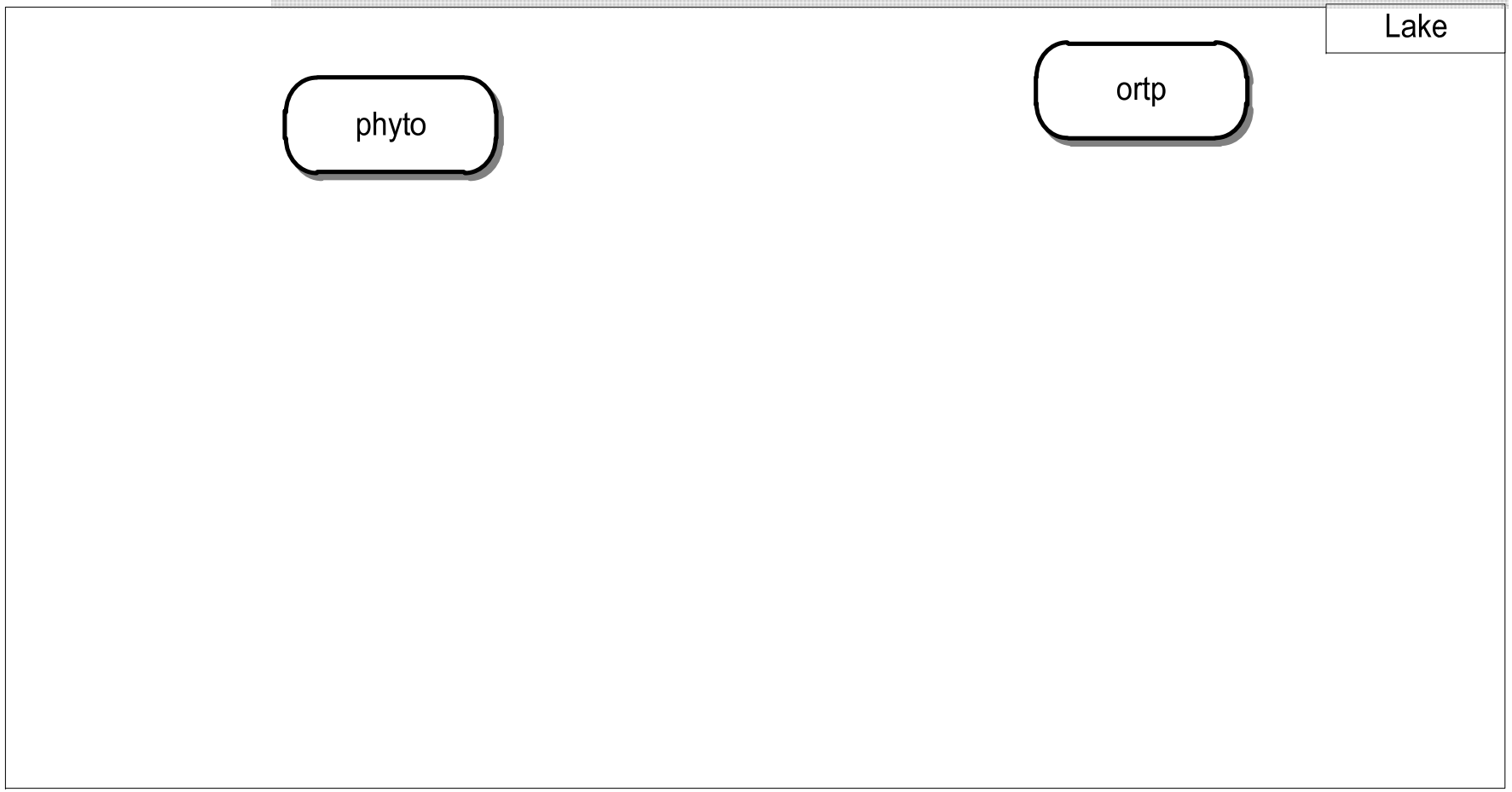
Each process contributes a term to the (OD) equations for the variables of each participating entity

Process-based models

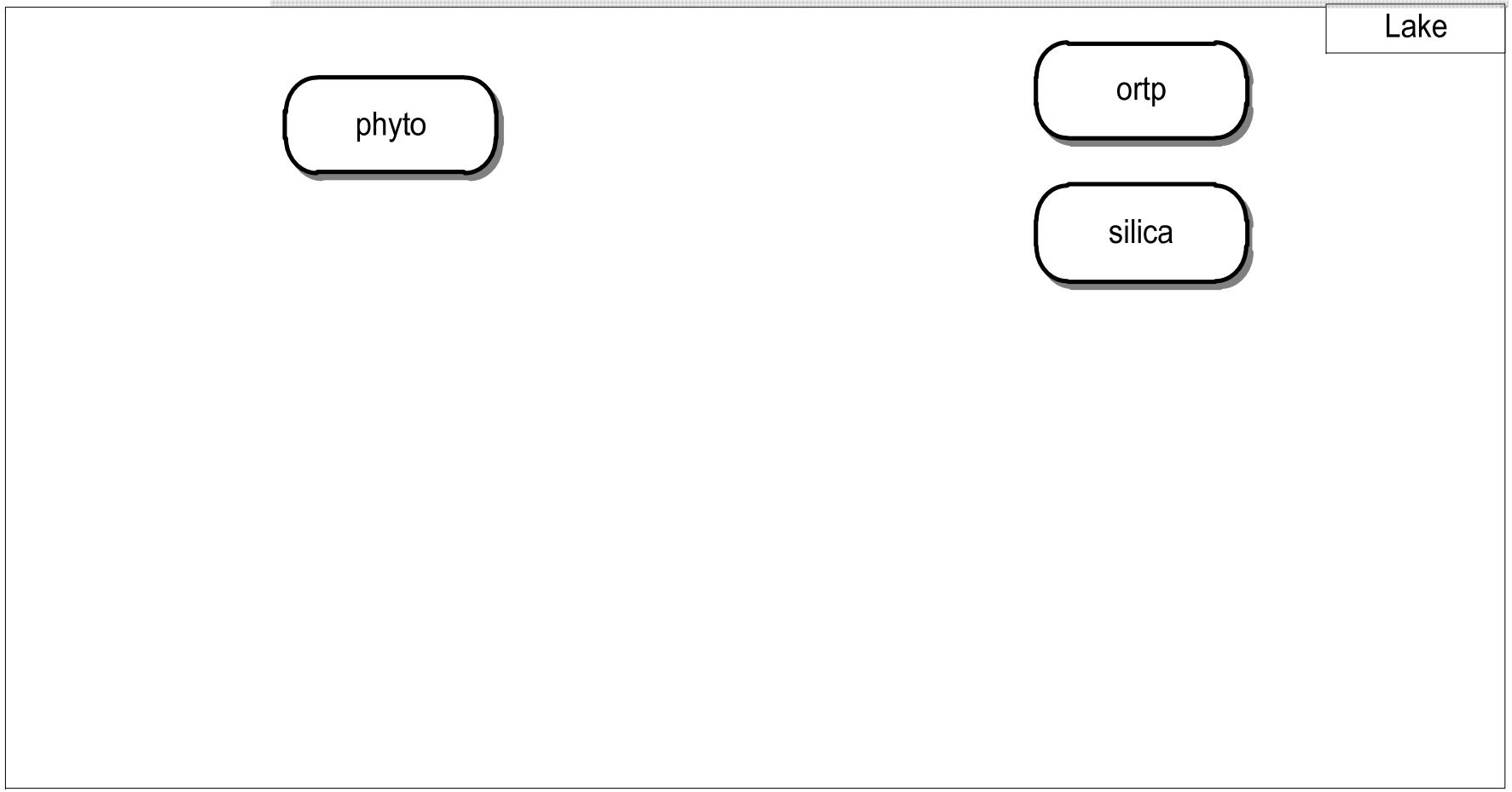
Lake

Process-based models

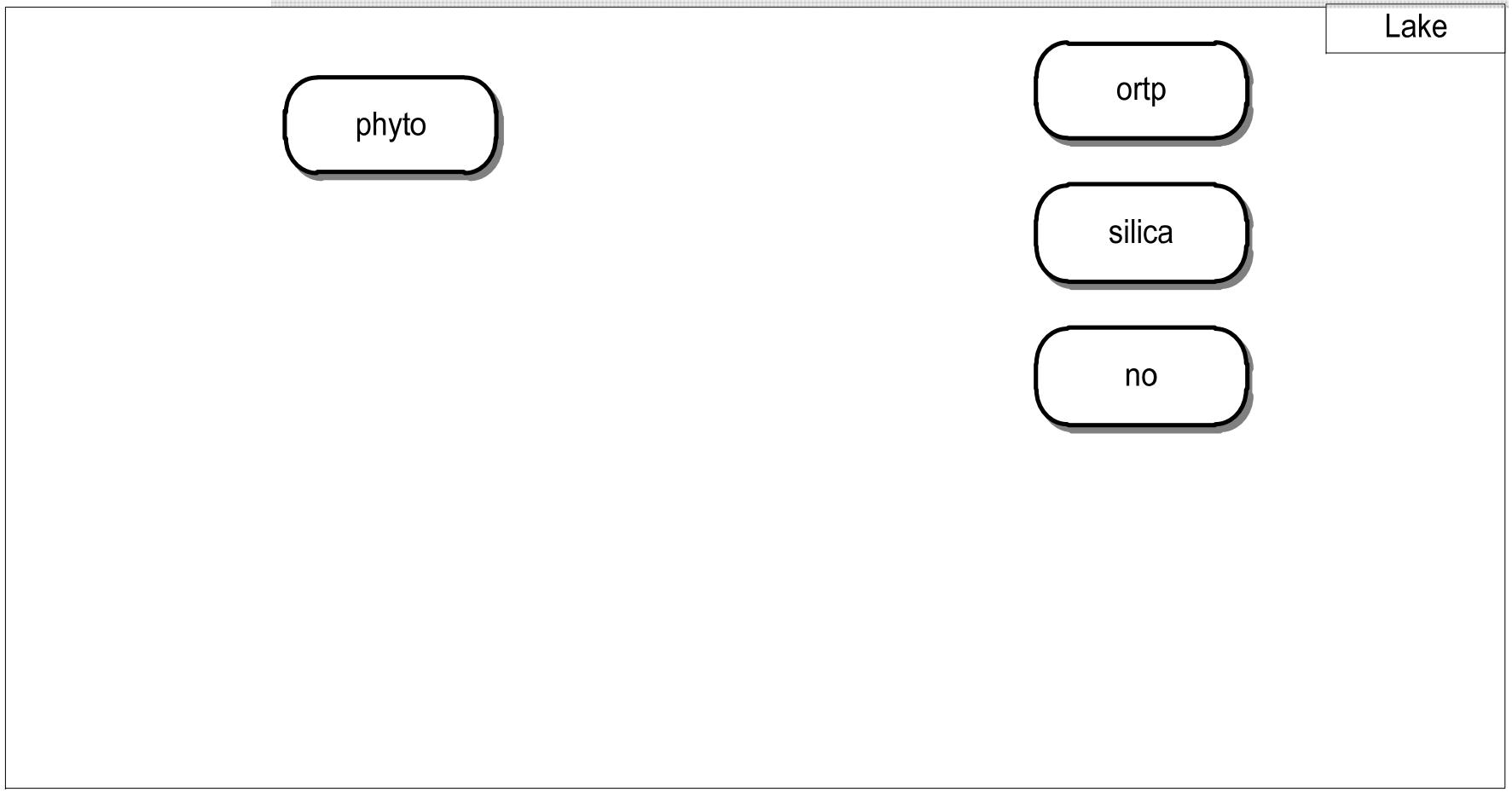
Process-based models



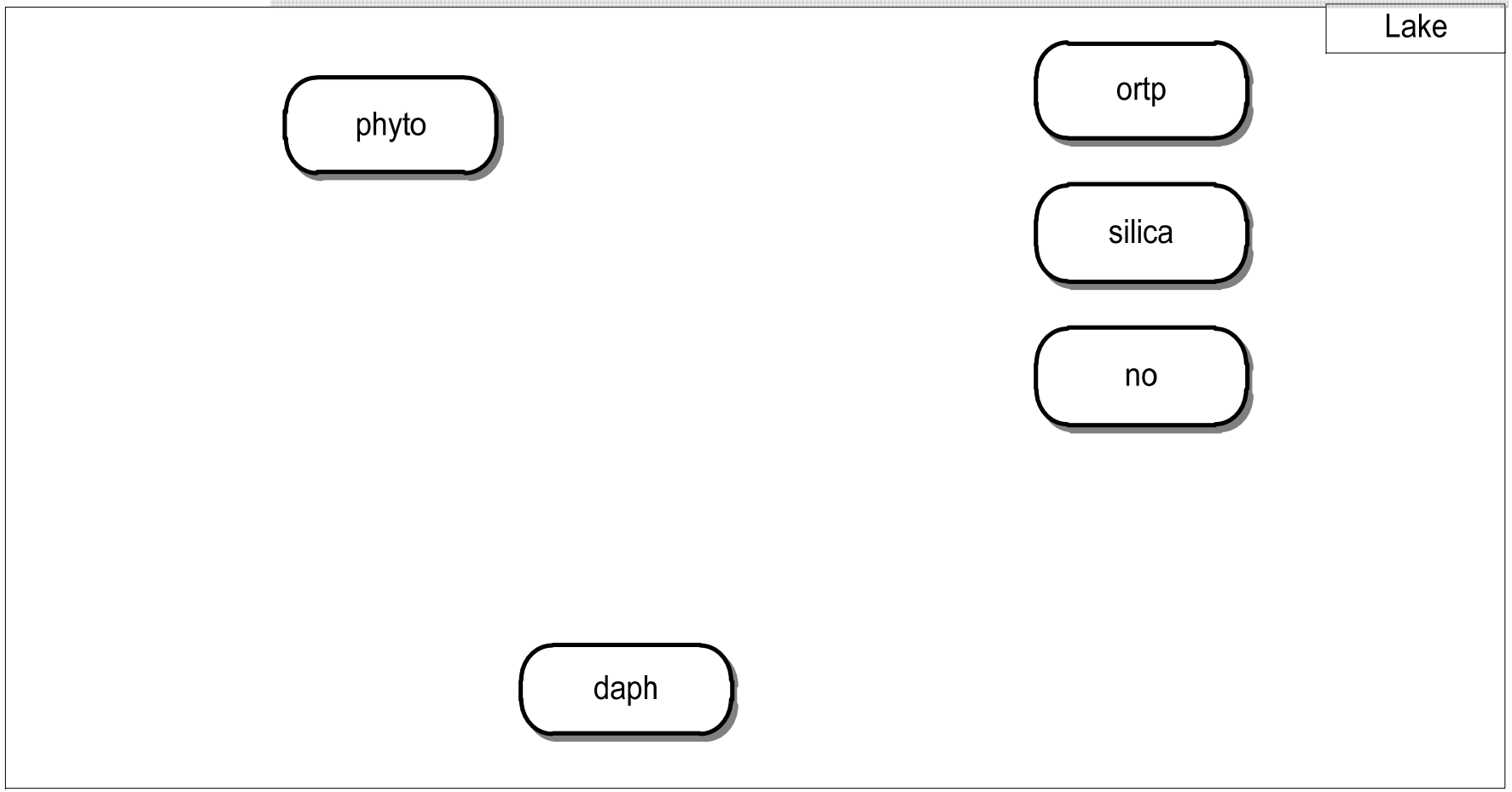
Process-based models



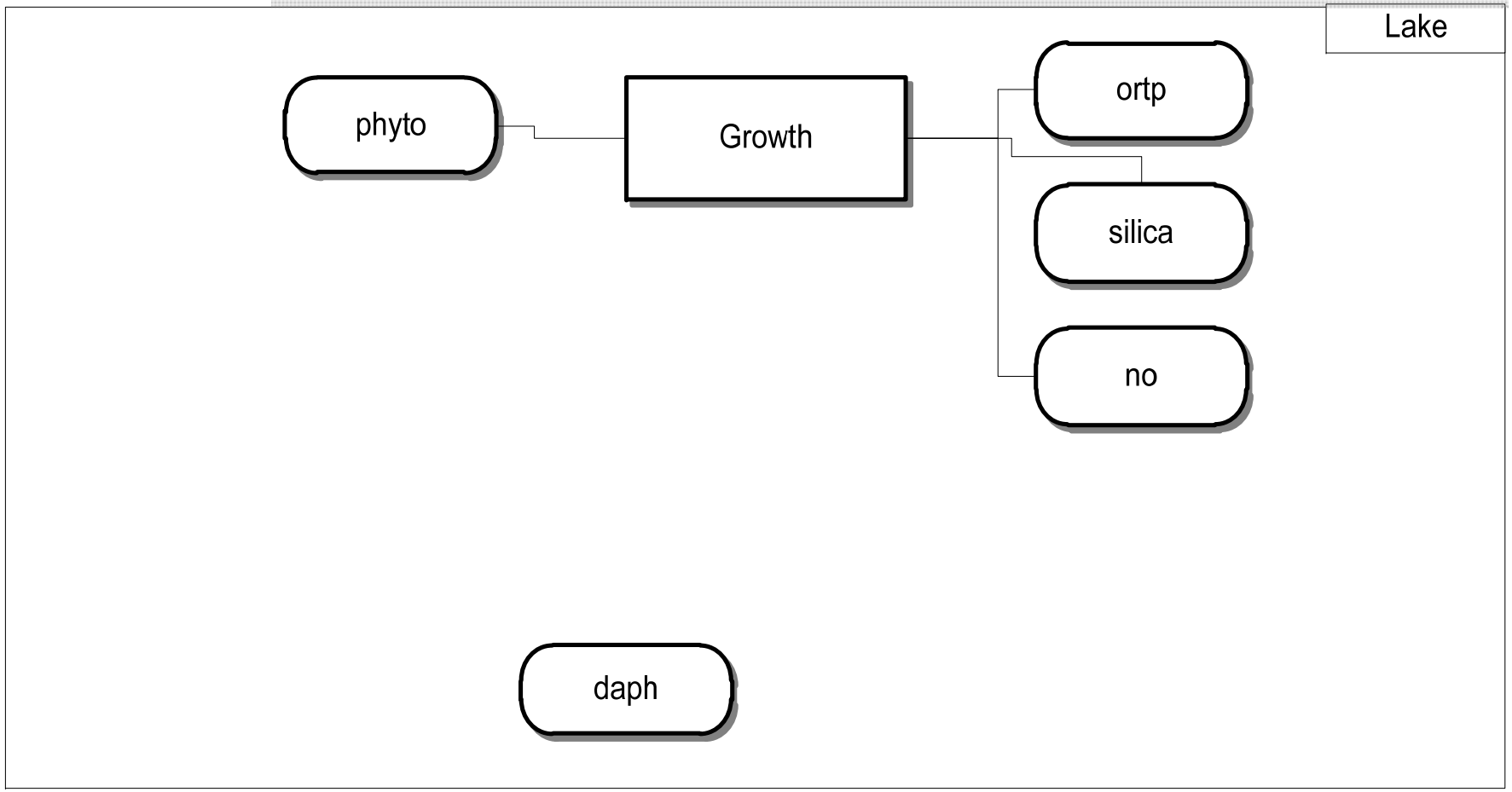
Process-based models



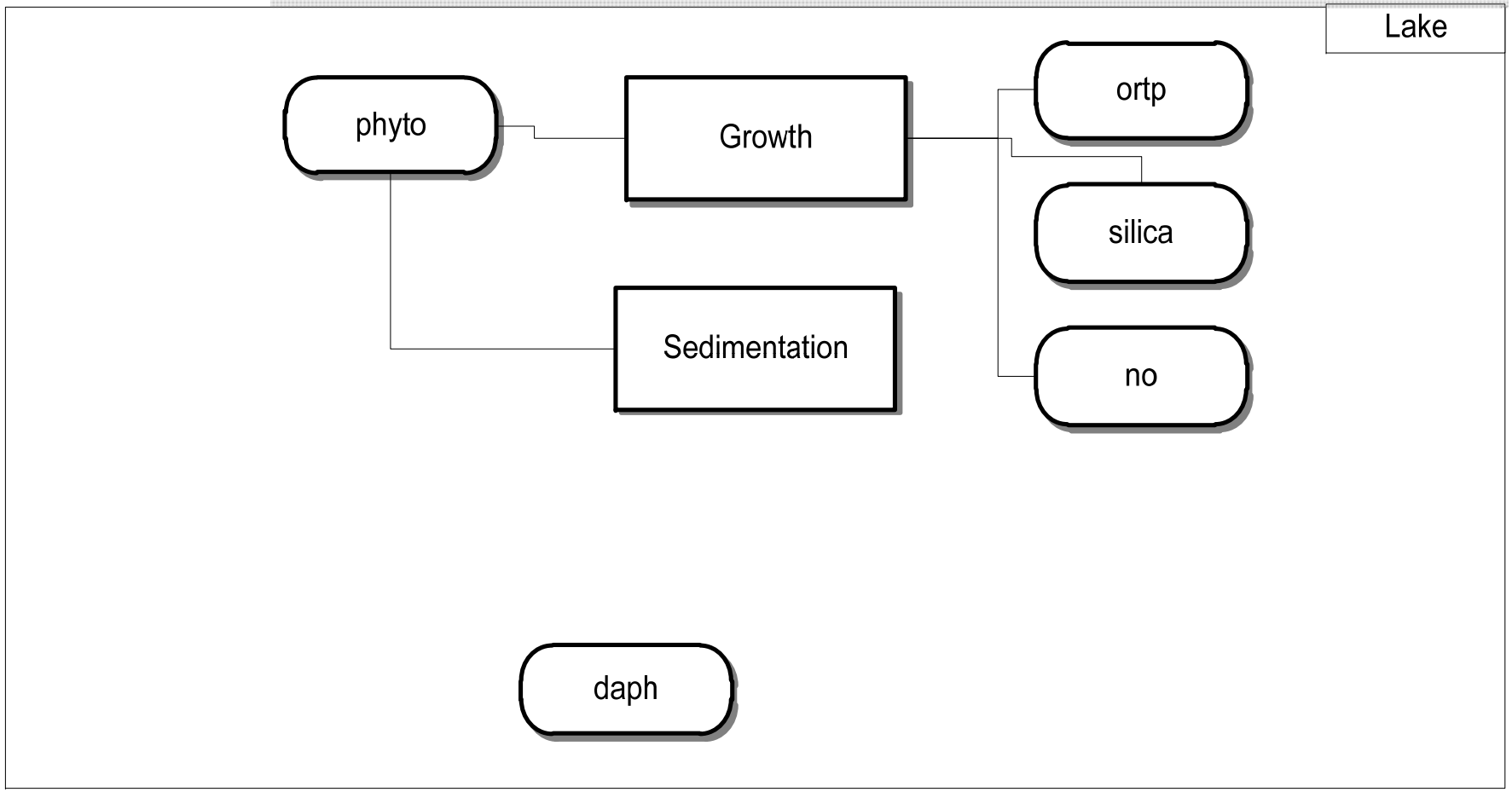
Process-based models



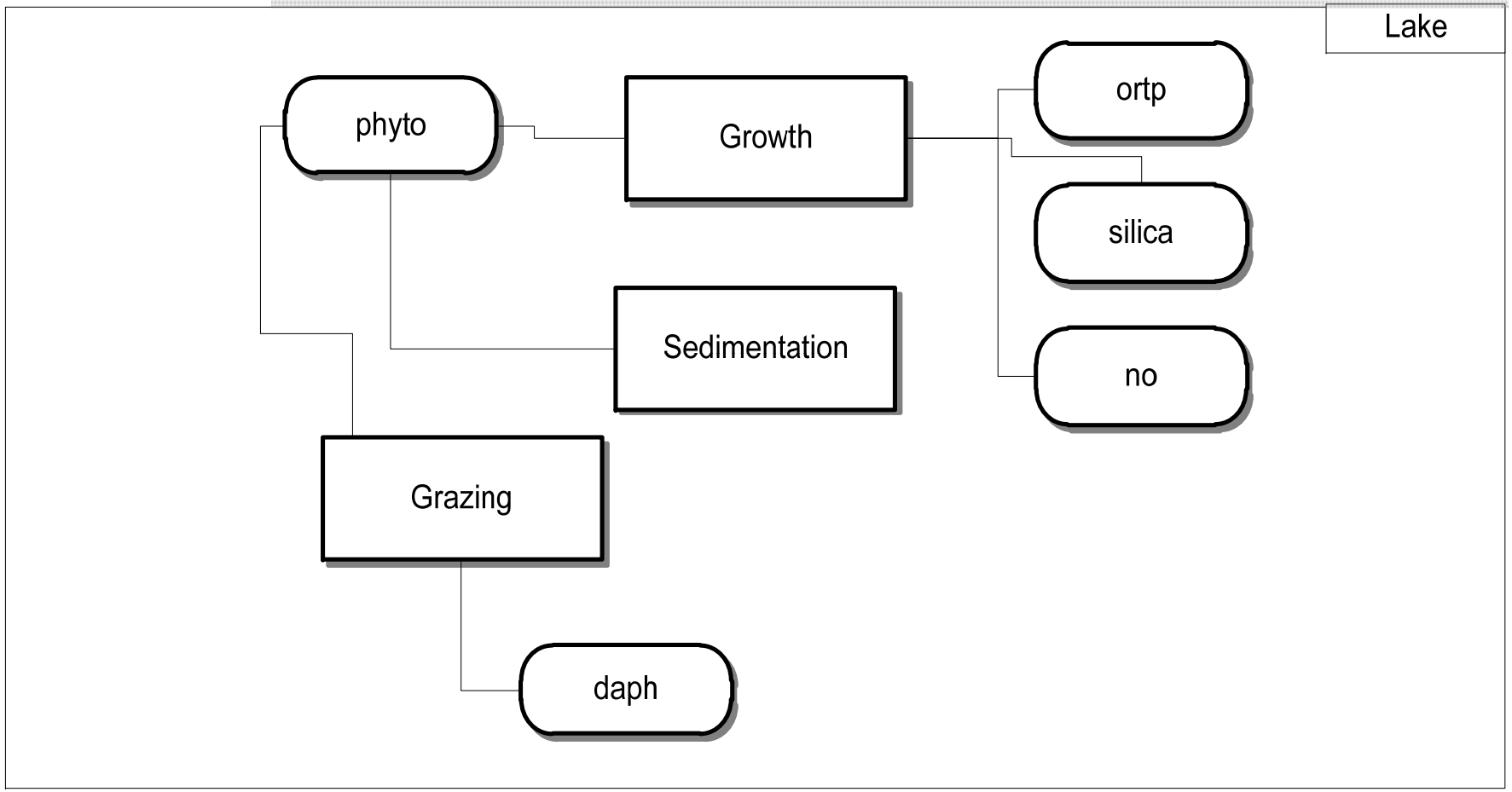
Process-based models



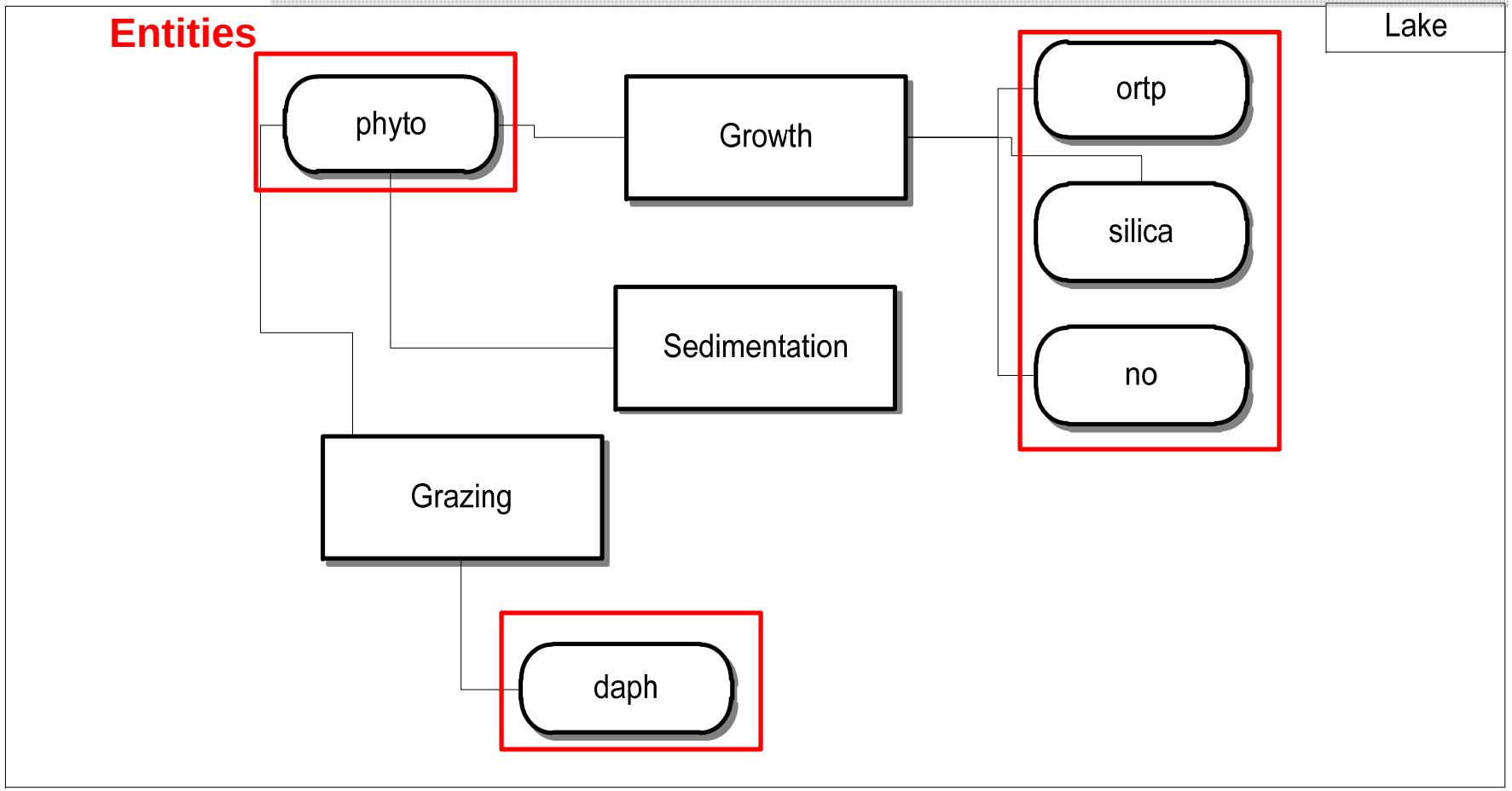
Process-based models



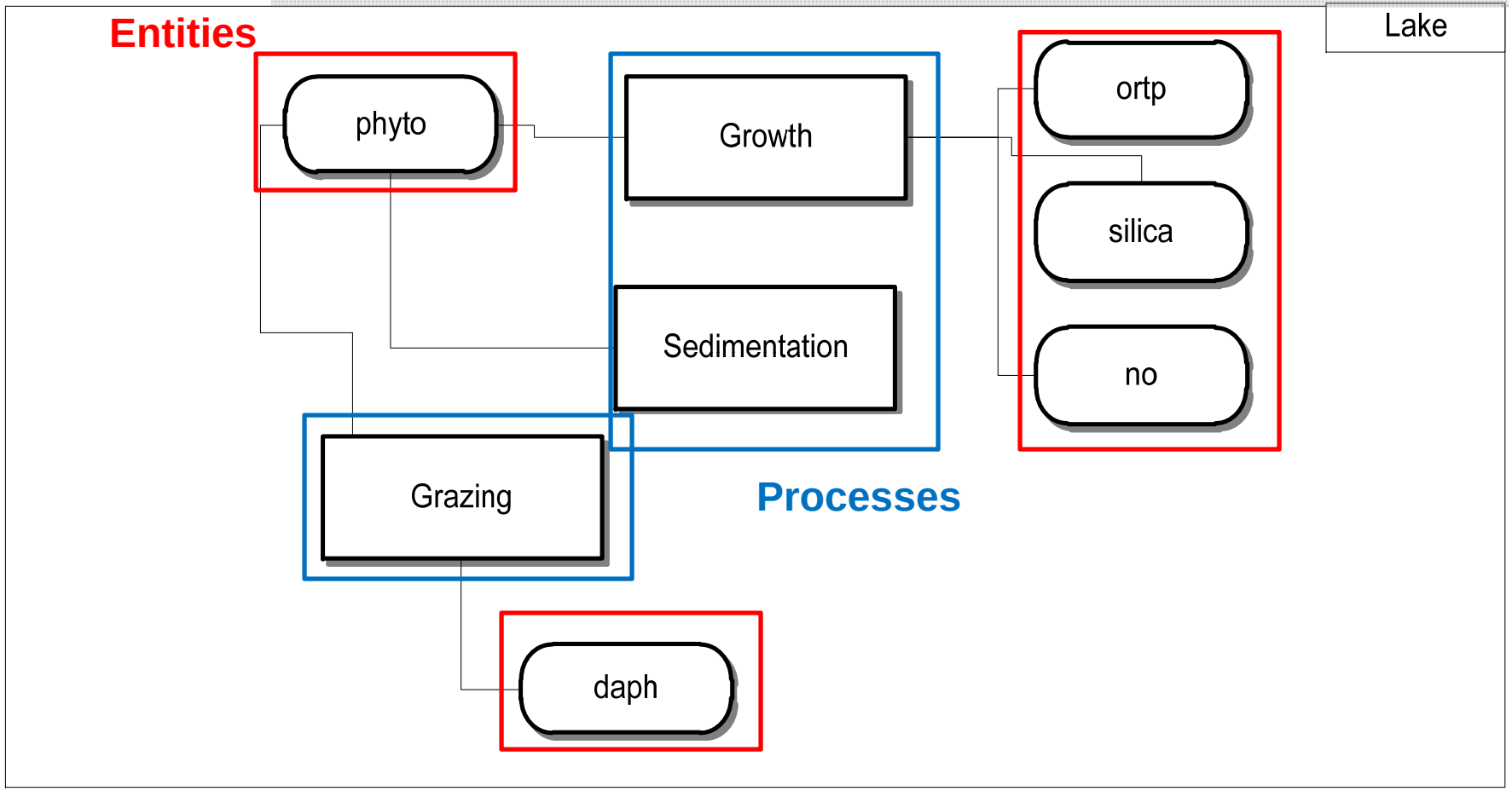
Process-based models



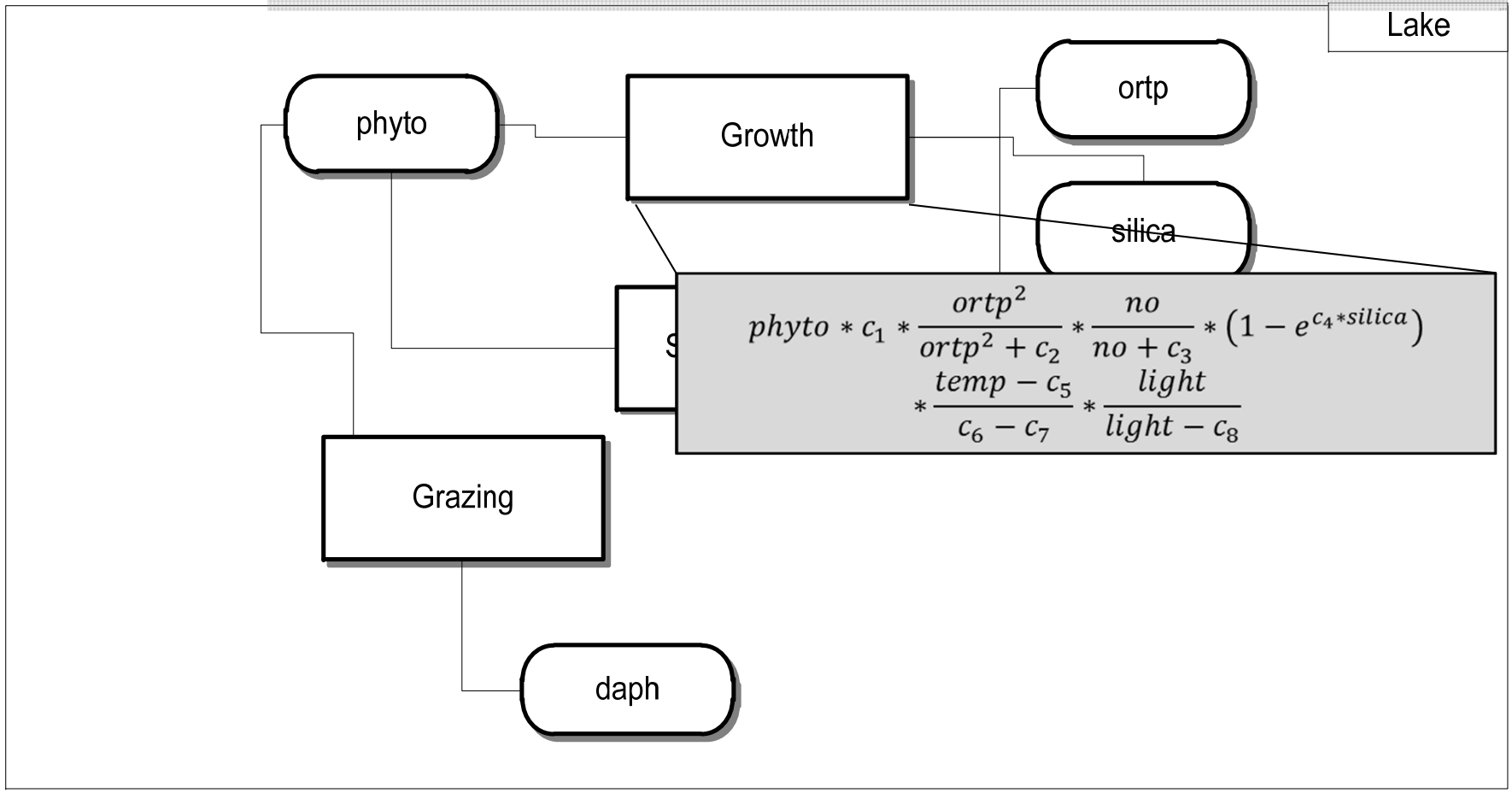
Process-based models



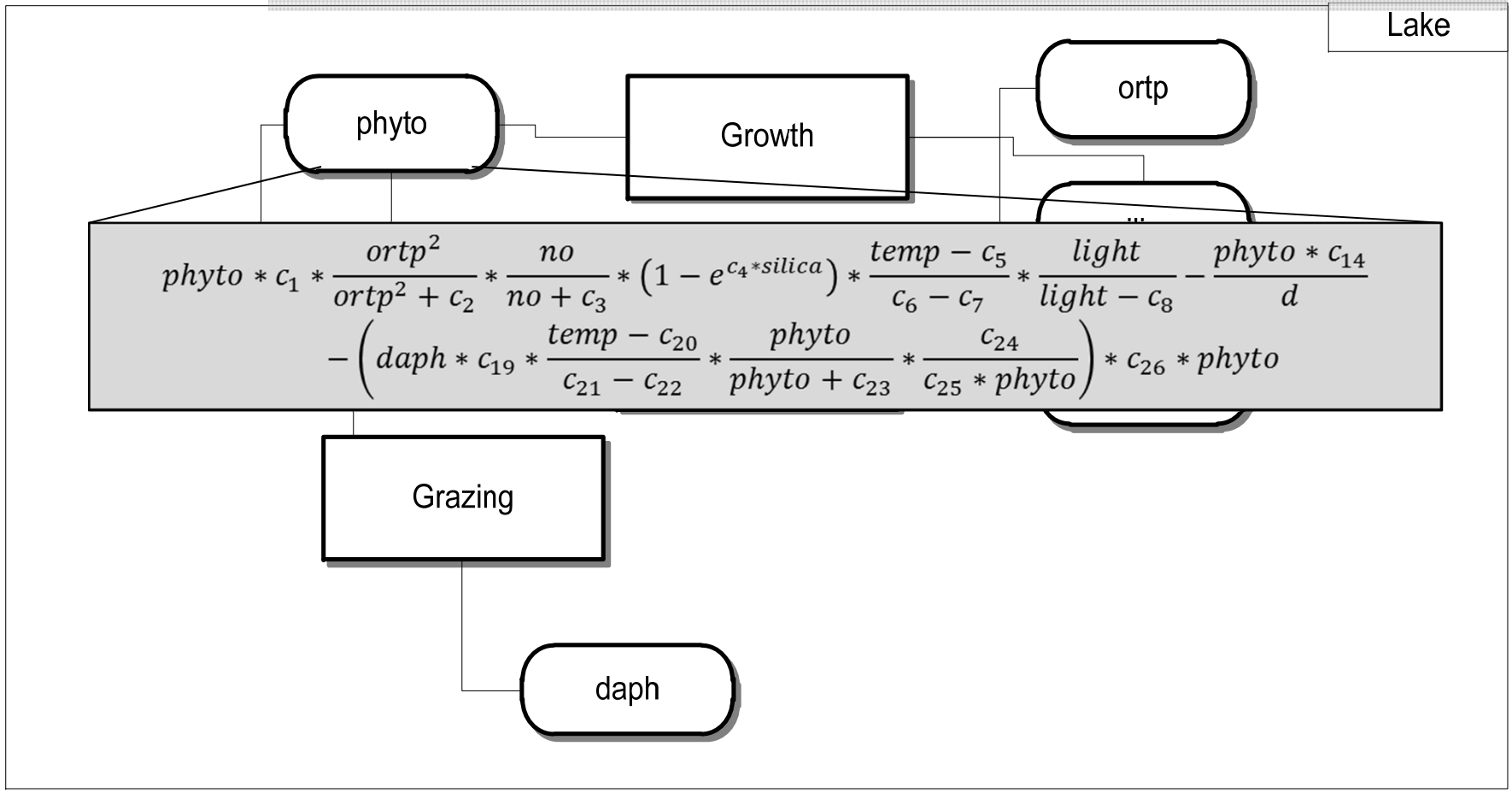
Process-based models



Process-based models



Process-based models



The Syntax of Process-Based Models: Entities

- Three entities in an aquatic ecosystem

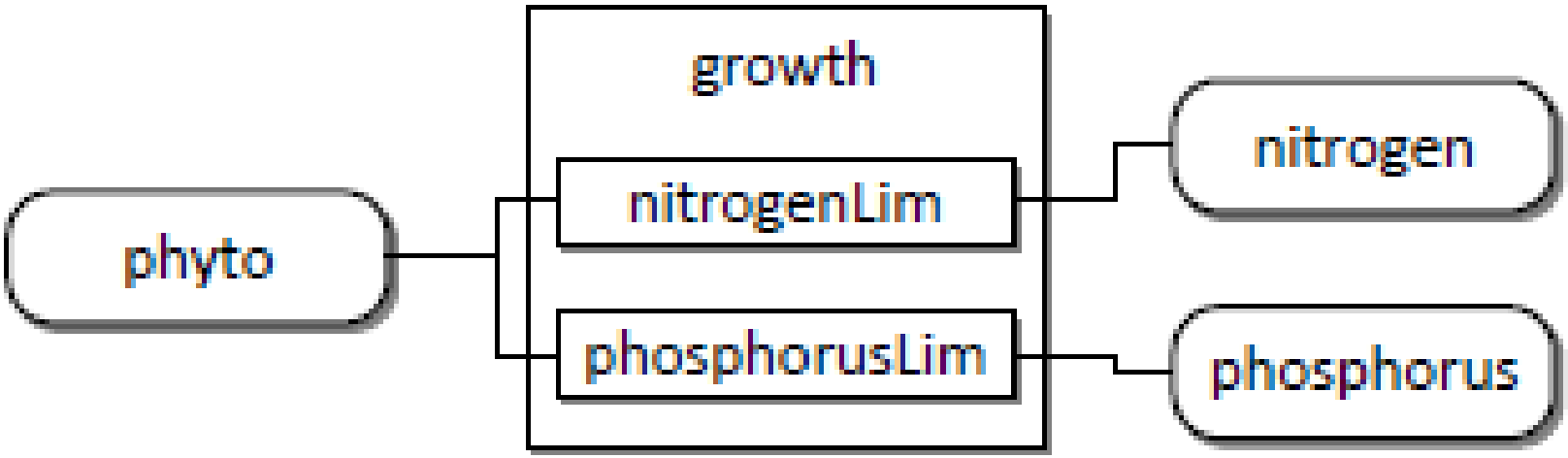
```
entity phyto {
    vars:
        conc{role: endogenous; aggregation: sum; initial: 10},
        nutrientLim{aggregation: product};
    consts:
        maxGrowthRate = 0.5,
}
entity phosphorus {
    vars:
        conc{role: endogenous; initial: 3};
    consts:
        halfSaturation = 0.02,
        alpha = 0.1;
}
entity nitrogen {
    vars:
        conc{role: exogenous};
    consts:
        halfSaturation = 0.2,
        alpha = 0.7;
}
```

The Syntax of Process-Based Models: Processes

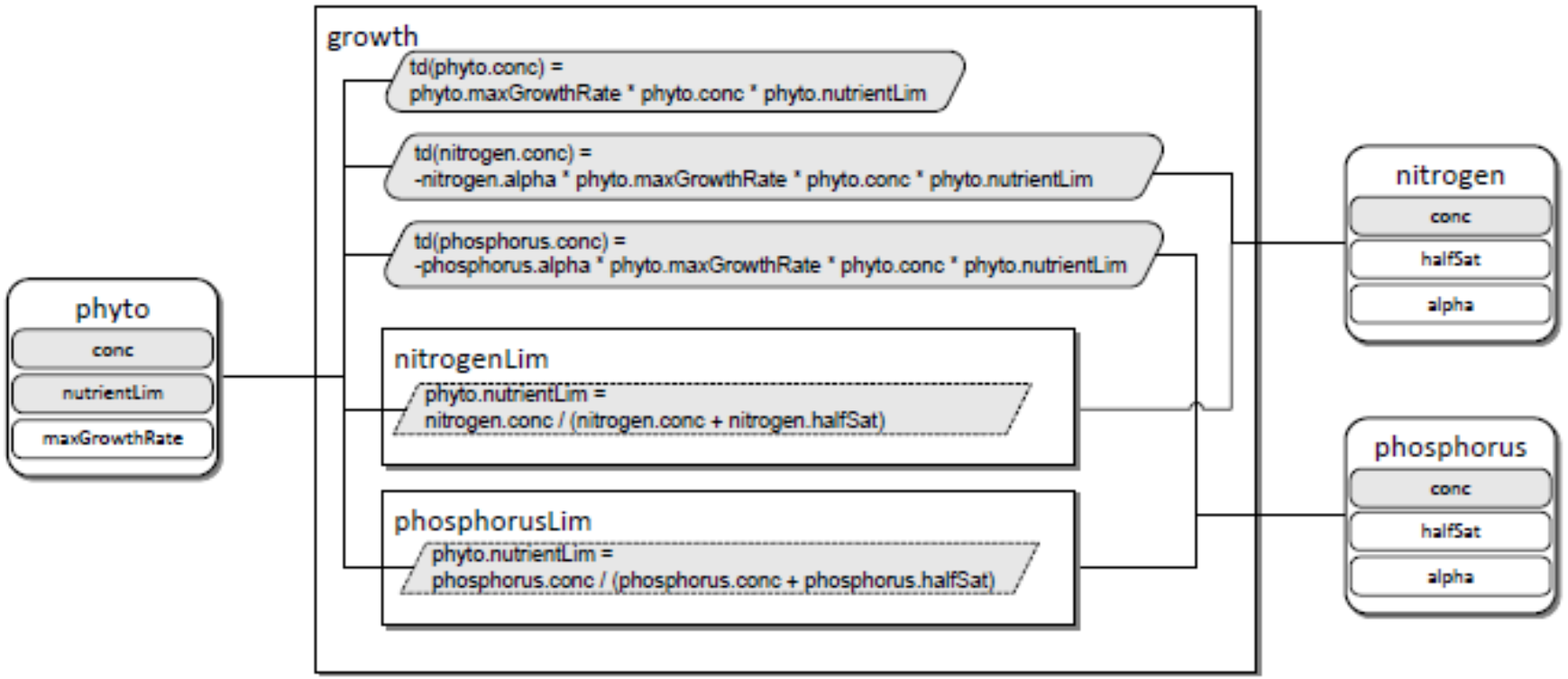
- A process with two sub-processes connecting the three entities in an aquatic ecosystem

process *growth*(*phyto*, [*phosphorus*, *nitrogen*]) {
  processes:
    phosphorusLim, *nitrogenLim*;
  equations :
    td(*phyto.conc*) = *phyto.maxGrowthRate* ∗ *phyto.conc* ∗ *phyto.nutrientLim*,
    td(*phosphorus.conc*) = *phosphorus.alpha* ∗ *phyto.maxGrowthRate* ∗ *phyto.conc* ∗
      phyto.nutrientLim;
}
process *phosphorusLim*(*phyto*, *phosphorus*) {
  equations:
    phyto.nutrientLim = *phosphorus.conc* / (*phosphorus.conc* + *phosphorus.*
      halfSaturation);
}
process *nitrogenLim*(*phyto*, *nitrogen*) {
  equations:
    phyto.nutrientLim = *nitrogen.conc* / (*nitrogen.conc* + *nitrogen.halfSaturation*);
}

The Qualitative Structure of the PMB



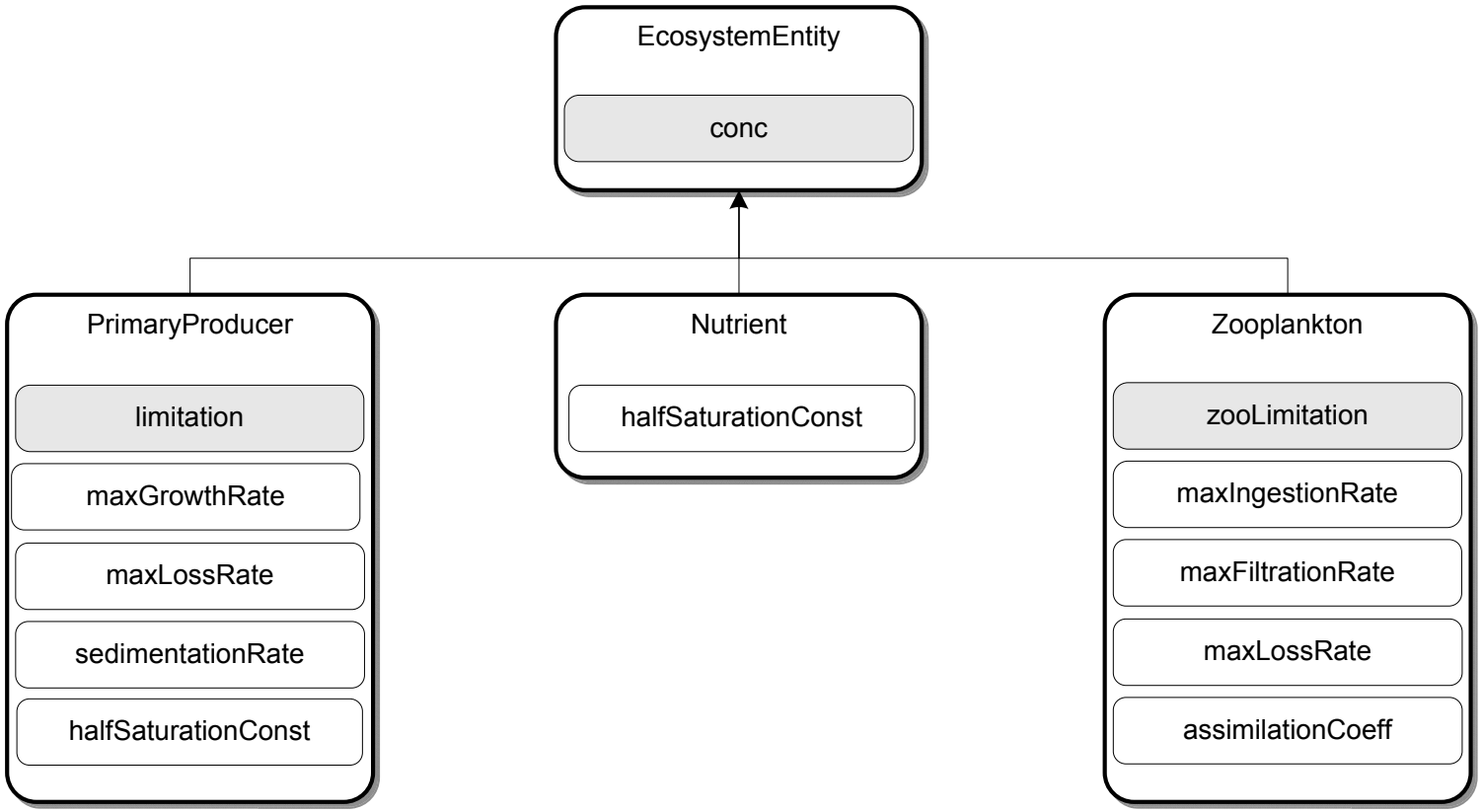
The Quantitative Structure of the PMB



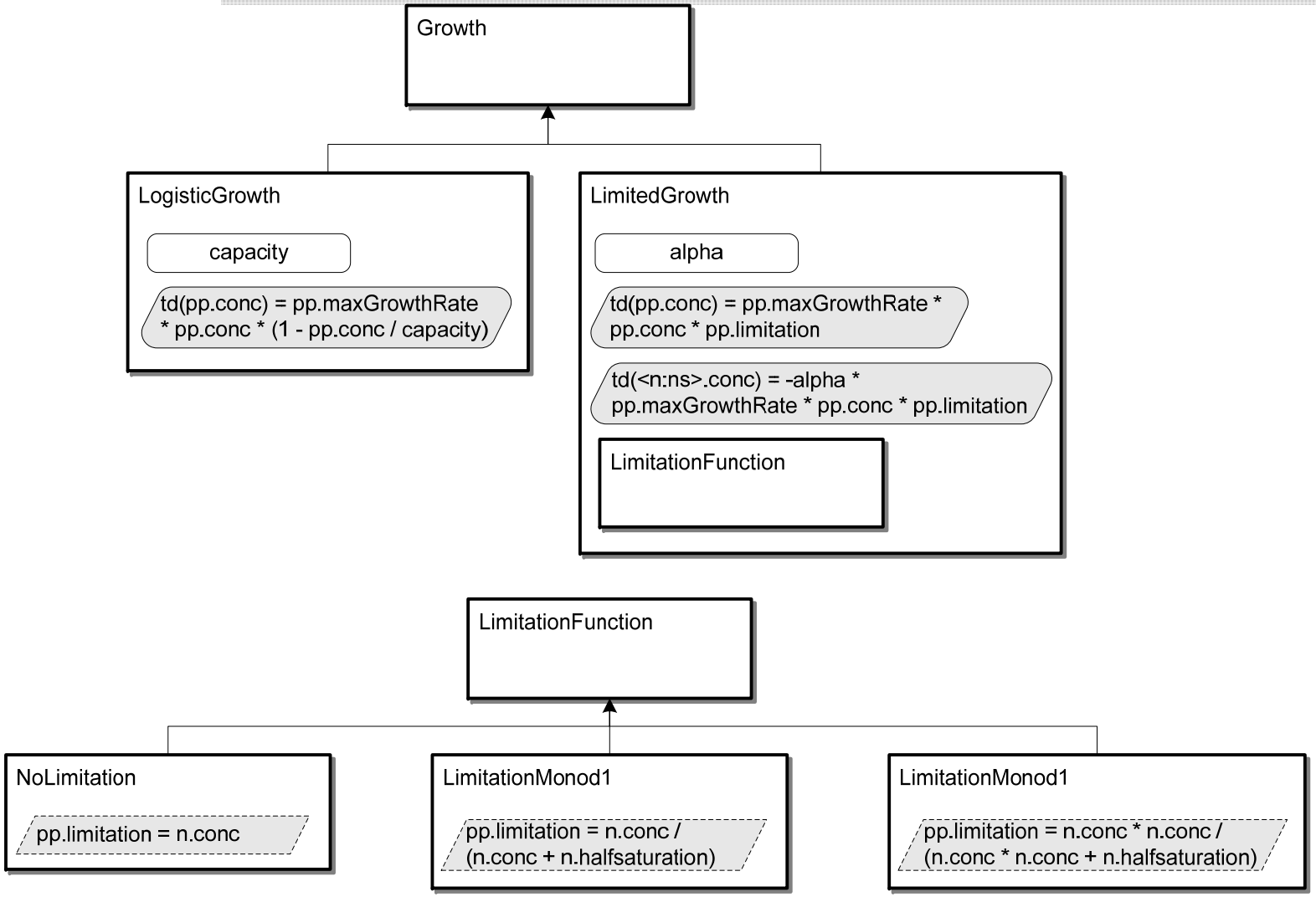
Domain Knowledge for Process-Based Modeling

- Some entities (and processes) are very similar
  - have the same properties
  - have properties that have the same pattern
- Extract these common properties into higher level concepts, called **templates**
- Templates
  - are partial/incomplete specifications that capture common information
  - Entity Templates
  - Process Templates
- Templates are organized into hierarchies

Entity Templates Hierarchy



Process Templates Hierarchy

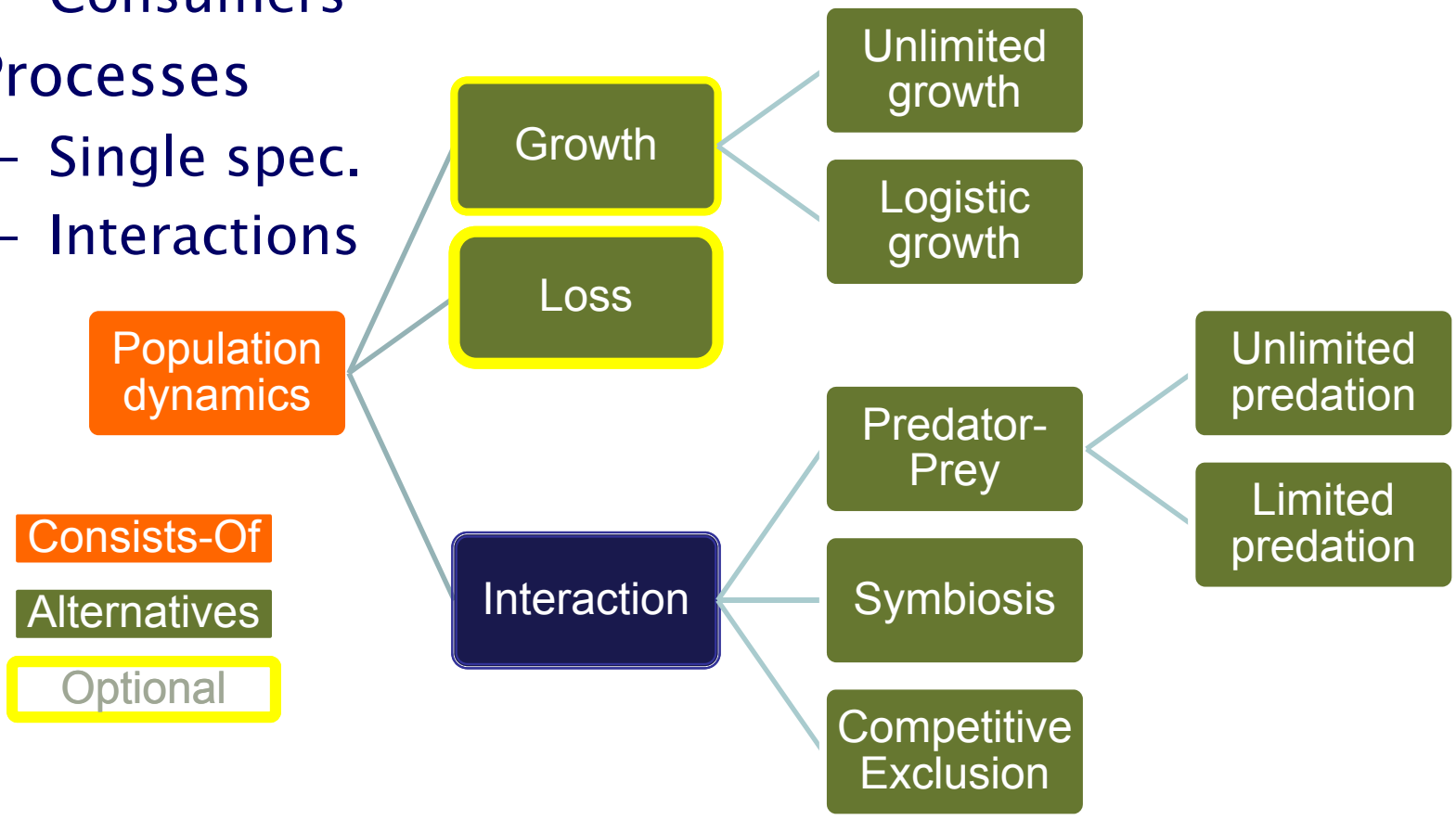


Libraries of Domain Knowledge

- A catalogue of kinds of entities and processes encountered in the domain of discourse
- Template entities/ processes
- Hierarchies of entities and processes
- Mathematical formulations of processes
  - Equation fragments
  - Alternative formulations allowed

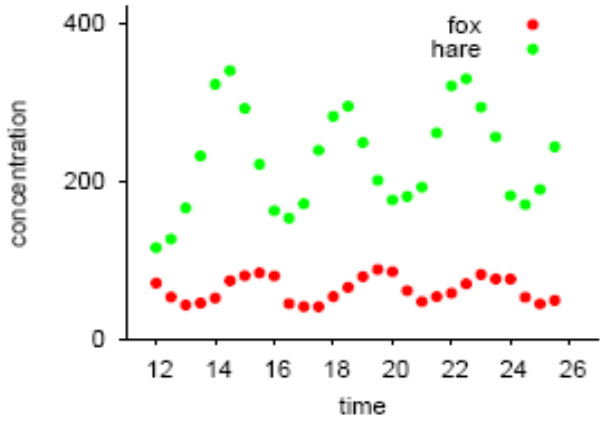
Hierarchies of Species and Processes

- Taxa (Species, Genera, …)
  - Primary producers
  - Consumers
- Processes
  - Single spec.
  - Interactions



Learning PBMs: Inductive Process Modelling

- Input: Observed behavior+ Task+ Template processes



process predator_prey_interaction
variables $Prey\{species\}, Pred\{species\}$
parameters $r[0, \inf], e[0, \inf]$
equations
$\frac{d}{dt} Prey = -1 \cdot r \cdot Prey \cdot Pred$
$\frac{d}{dt} Pred = e \cdot r \cdot Prey \cdot Pred$

- Output: Instance processes + ODEs

process exponential_growth
$\frac{d}{dt} hare = 2.5 \cdot hare$

process exponential_loss
$\frac{d}{dt} fox = -1.2 \cdot fox$

process predator_prey_interaction
$\frac{d}{dt} hare = -0.3 \cdot hare \cdot fox$
$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox$

$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$
$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$

Modeling Task Specification

- Specifies the entities present in the system, as well as the measurements available and their relation to the entities
- Specifies high-level processes expected to take place
- Allows for flexible use of the domain knowledge in the library

- For example, we can have quite a complete task specification, leaving only constant parameters to fit
- Or, we can have very high-level processes, requiring the search through a large set of combinations of alternative formulations for each of these

A Machine Learning Approach to Learning Models of Dynamic Systems

- Heuristically search the space of possible **model structures**
  - What is the space?
    - The space of structures considered is defined by the **task specification** and **library of domain knowledge** (template entities and processes)
  - What is the heuristic?
    - Takes error/degree of fit to the **observed behavior(s)** into account, possibly additional factors (such as model complexity)

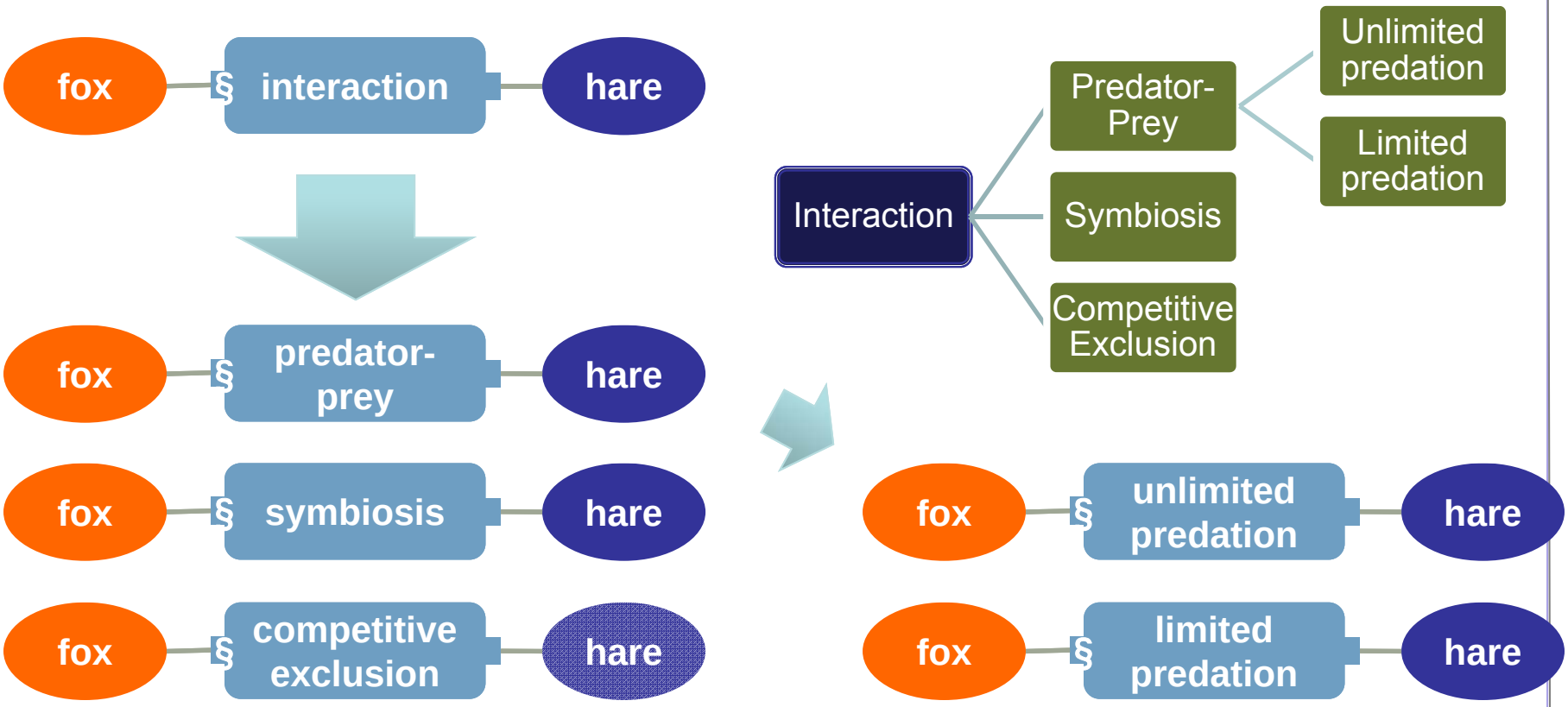
A machine learning approach to IPM

- Consider different sets of (high-level) processes
- Consider different sub-processes and alternative model formulations for each process:
  These correspond to different ODE structures
- To evaluate candidate model structures
  - Parameters are calibrated (nonlinear optimization)
  - Goodness of fit between measured and simulated values is considered

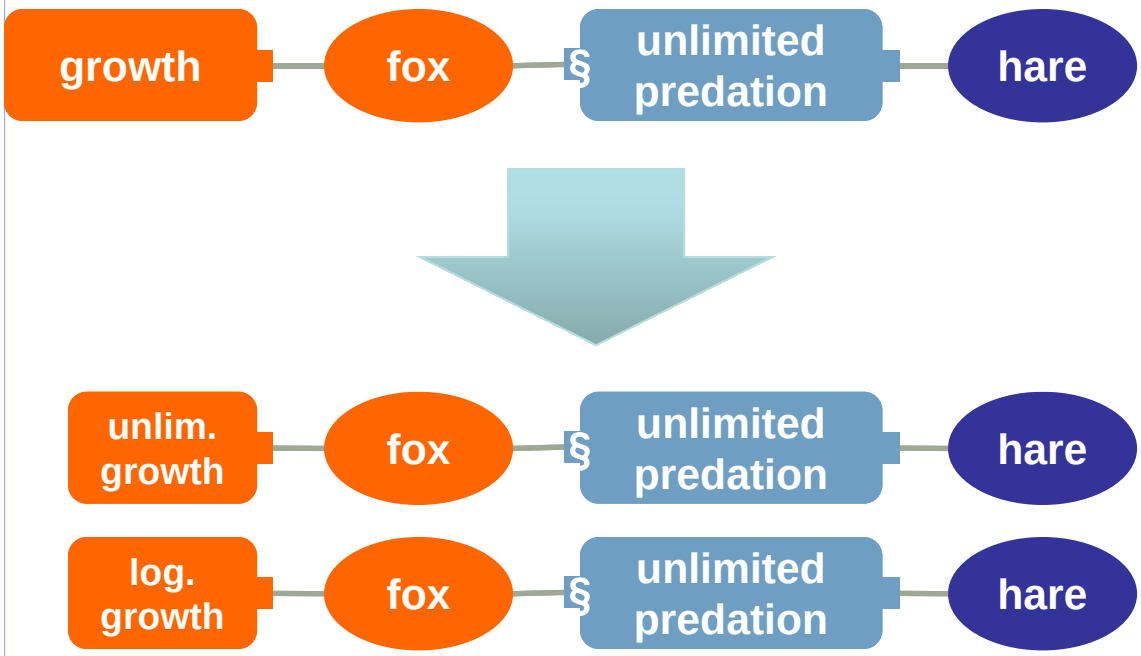
IPM: Direct Search for Process Models

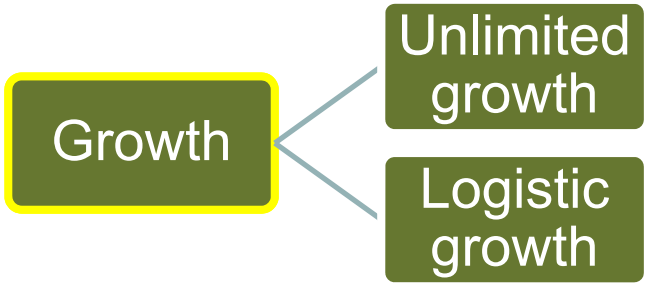
- Consider different sets of (high-level) processes
- Consider different sub-processes and alternative model formulations for each process:
  These correspond to different ODE structures
- To evaluate candidate model structures
  - Parameters are calibrated (nonlinear optimization)
  - Goodness of fit between measured and simulated values is considered

IPM: Generate Models



IPM: Generate Models (2)

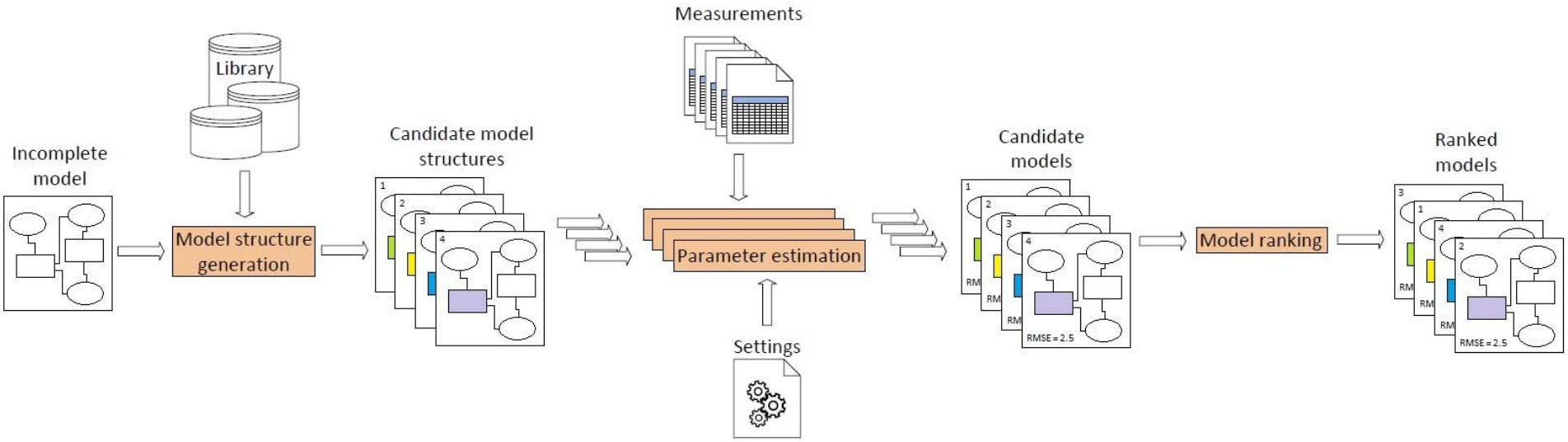




etc.

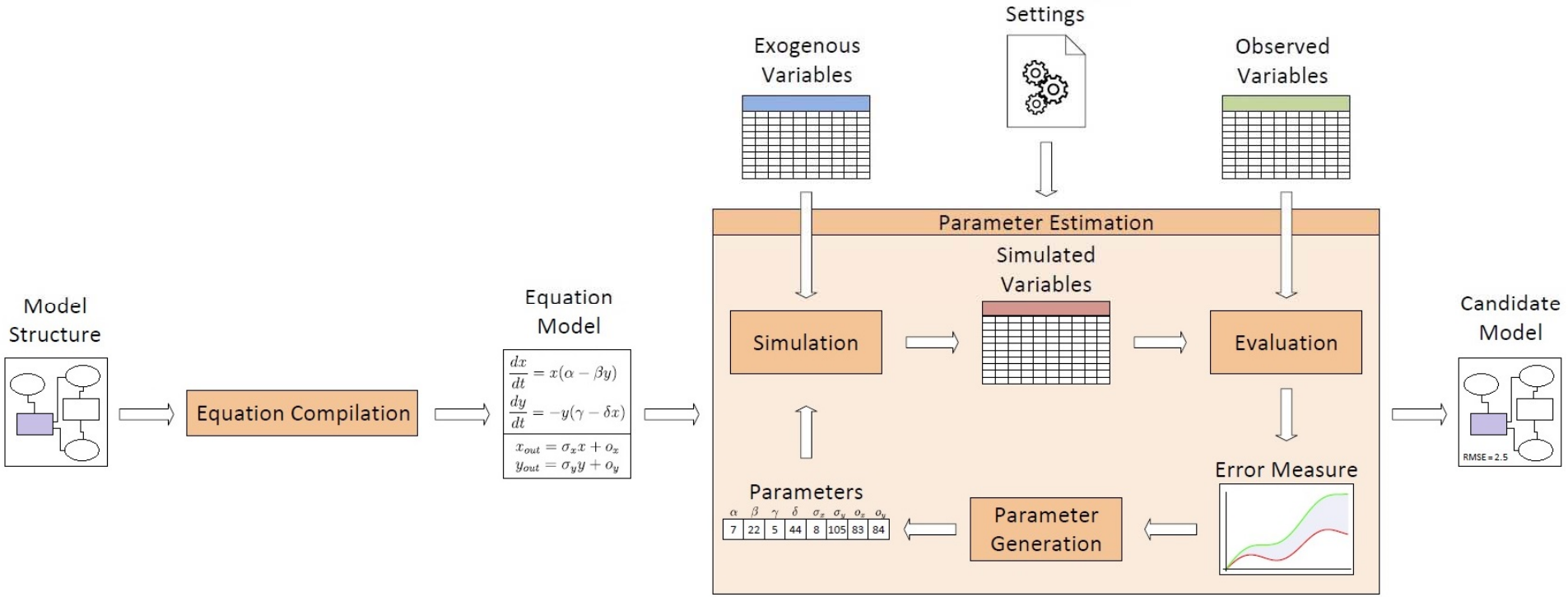
ProBMoT: A SW Platform for IPM

- Process-Based Modeling Tool (D. Čerepnalkoski)
- Given library of domain knowledge, conceptual model (task specification), measured data
- Generates (exhaustively) model structures
- Fits model parameters and
- Finds best candidate (process-based) model(s)

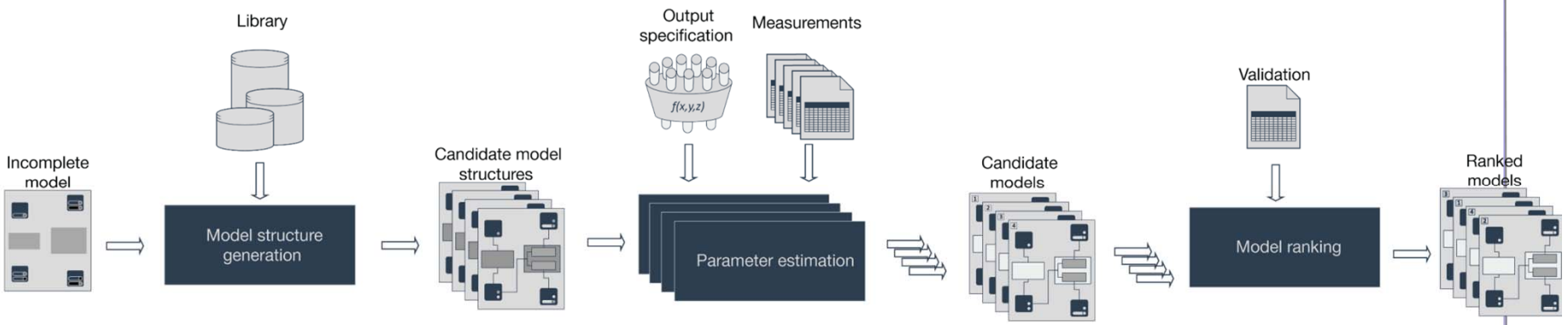


Parameter Estimation in ProBMoT

- Supports the use of different optimization methods for parameter fitting (incl. gradient descent with RRRs)
  - ACO/DASA, Differential Evolution, Multi-objective DE
- Supports the use of different fitness functions (and not just sum of squared errors)



ProBMoT: More accurate workflow



PROCESS-BASED MODELING TOOL
PROBMOT
WWW.PROBMOT.IJS.SI

Recent Advances in IPM Methods

Meta-heuristic optimization for parameter estimation:

- Different and **multiple** objective functions
- Different optimization methods from a general library

Formalism(s) for representing domain knowledge:
Stochastic reaction models

Search model structures: Heuristic (evolutionary) search

Learning ensemble ODE models: Bagging/Boosting IPMs

Meta-learning about IPM: Learning constraints on models

Applications of Inductive Process Modeling

Application areas

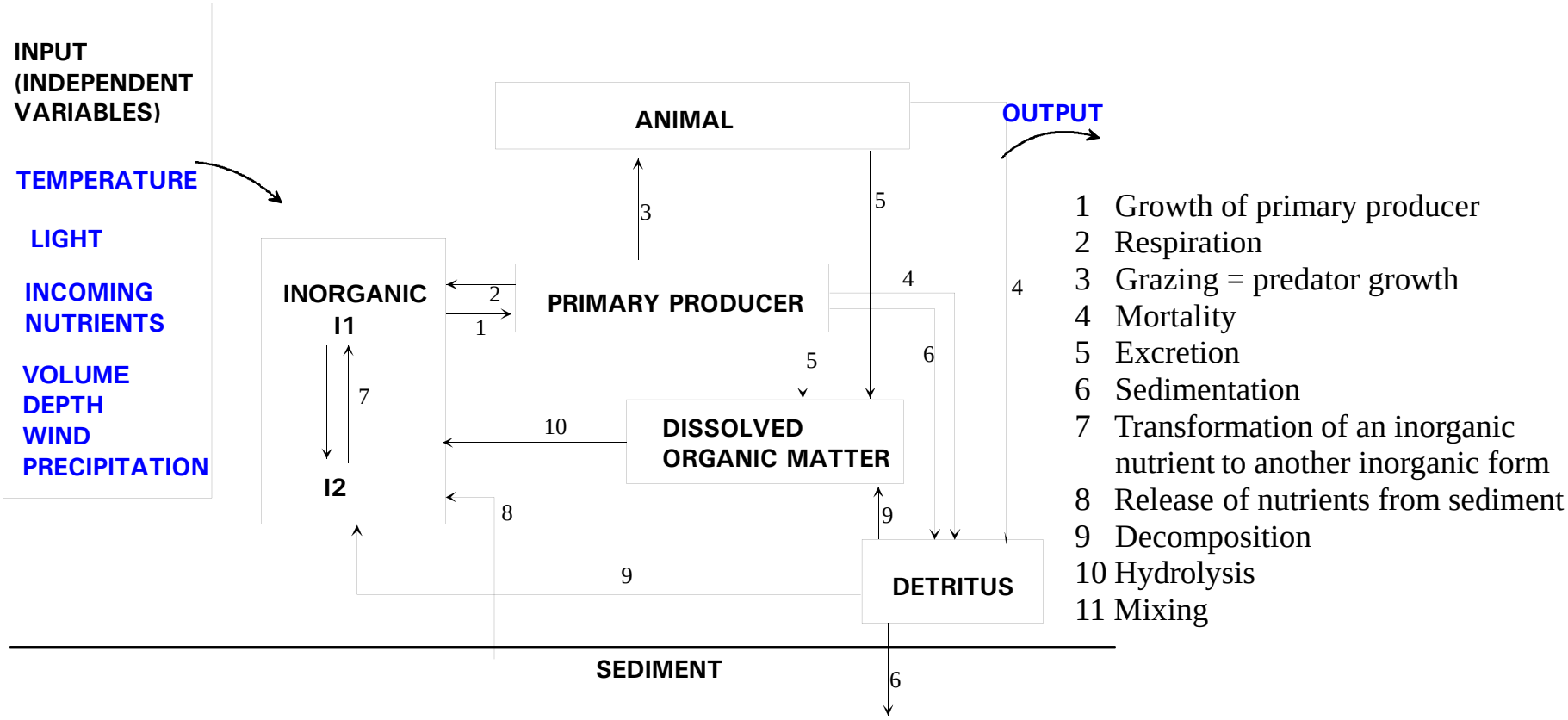
- Systems Ecology
- Systems Biology
- Synthetic Biology

Systems Ecology

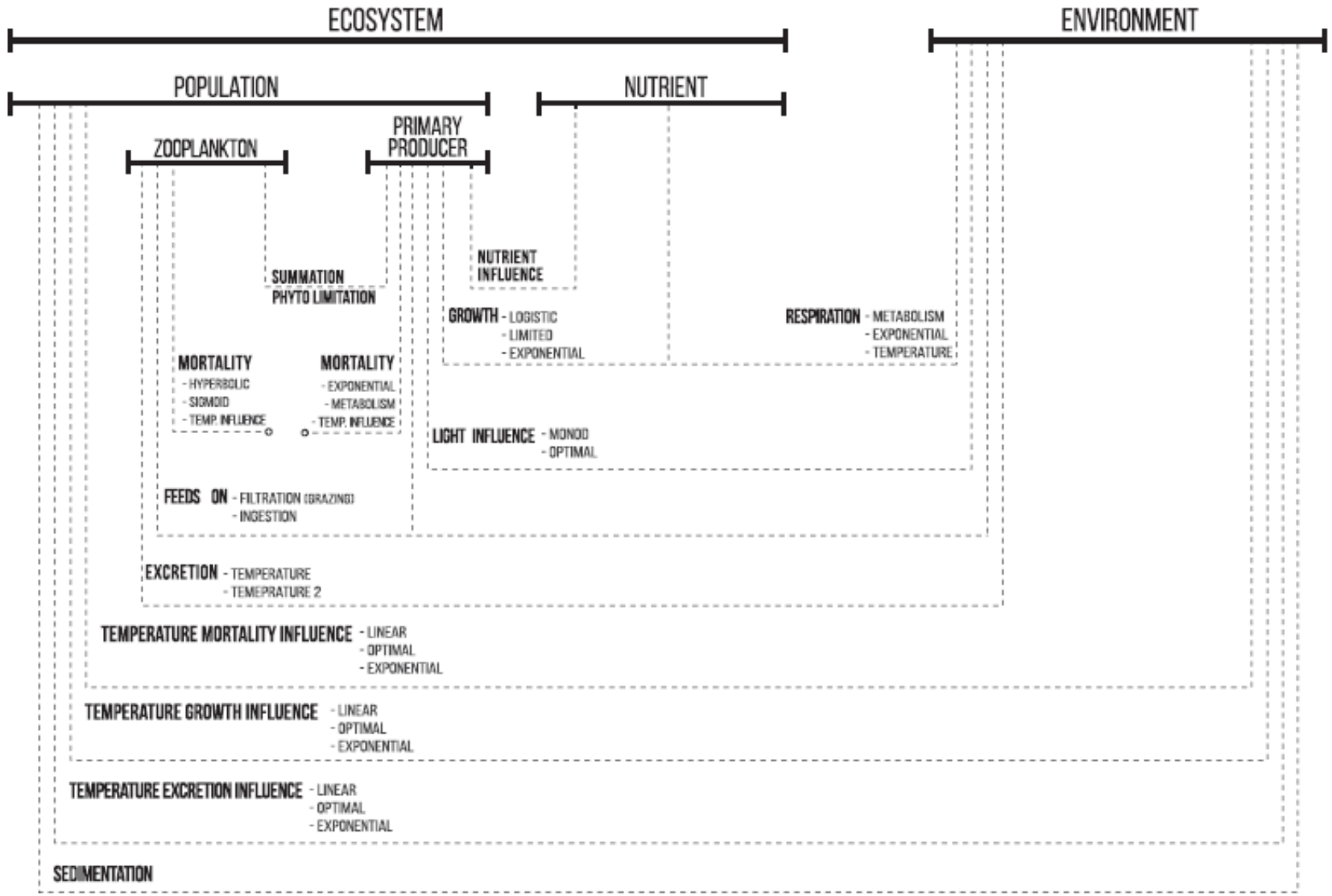
- Population ecology, esp. population dynamics
- Mostly for aquatic ecosystems
- Library of domain knowledge (Atanasova et al. 2006)
- Used for modeling many different aquatic ecosystems
  - Lakes
  - Lagoon
  - Sea

Knowledge for Modeling Aquatic Ecosystems

- An overview of entities and processes in the library



Knowledge for Modeling Aquatic Ecosystems



Modeling Aquatic Ecosystems

Venice lagoon

$$\dot{\texttt{biomass}} = 4.79 \cdot 10^{-5} \cdot \texttt{biomass} \cdot (1 - \frac{\texttt{biomass}}{0.844}) + 0.406 \cdot \texttt{biomass} \cdot (1 - e^{-0.216 \cdot \texttt{temp}}) \cdot (1 - e^{-0.413 \cdot \texttt{DO}}) \cdot \frac{\texttt{NH3}}{\texttt{NH3} + 1} - 0.0343 \cdot \texttt{biomass}$$

Ross sea, Antarctica

Lake Glumsoe, Denmark

$$\dot{\texttt{phyto}} = 0.553 \cdot \texttt{temp} \cdot \frac{\texttt{phosp}}{0.0264 + \texttt{phosp}} - 4.35 \cdot \texttt{phyto} - 8.67 \cdot \texttt{phyto} \cdot \texttt{zoo}$$

Automated Modeling of Lake EcoSystems

Lake ecosystems

- Lake Bled

- Lake Kasumigaura

- Lake Greifensee, Switzerland
- Lake Kinneret, Israel
- Lake Zurich, Switzerland

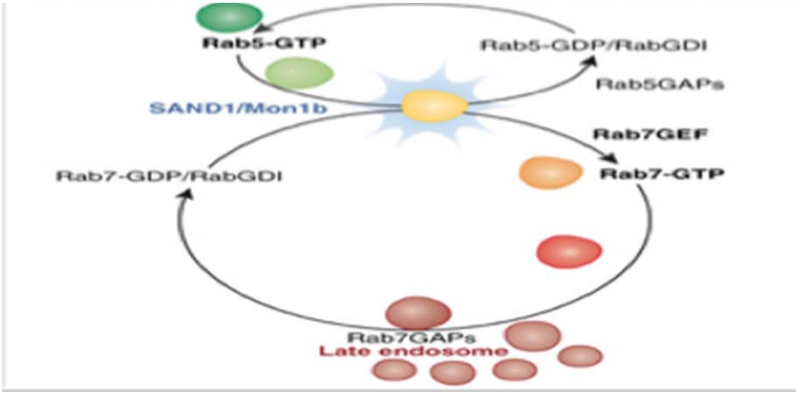
Systems Biology: 'Reconstructing' Biological Networks

- Reconstructing networks is of central interest in SB
  - Formulating network models
  - That capture the dynamics of the studied systems
  - From time course data
- Need to determine structure and dynamics of the net
  - Structure (nodes/species, arcs/reactions)
  - Dynamics: behavior in time, captured by ODEs
    - Functional form of ODEs, including reaction rates (e.g., Michaelis-Menten vs. Hill kinetics)
    - Constant parameters in the ODE (e.g., kinetic or reaction rates constants, e.g.,

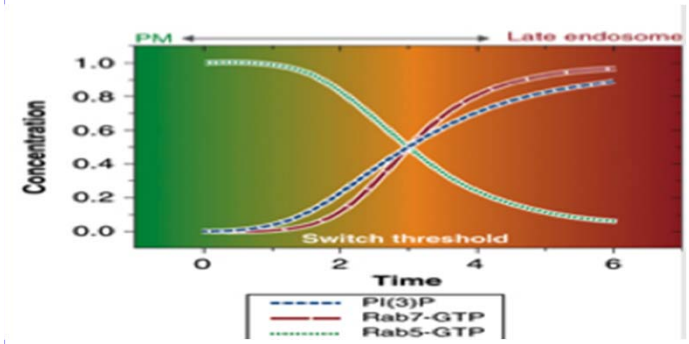
      $dP / st = reaction\_rate \times S \times S / (S + modulation\_rate)$

‘Reconstructing’ Biological Networks

- Structure



- Dynamics



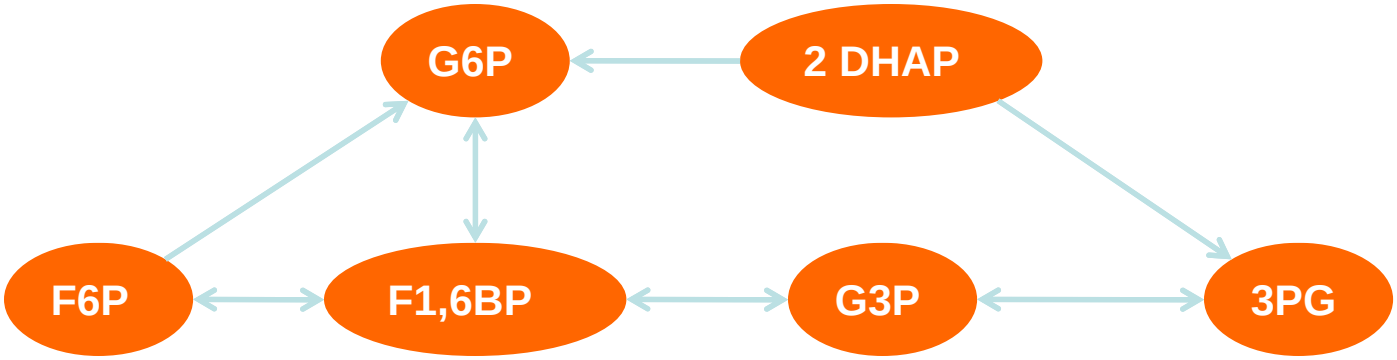
$$\frac{d}{dt}r_5 = v_1 + v_7 + v_9 - v_2 - v_3$$
$$\frac{d}{dt}R_5 = v_2 - v_7 - v_9$$
$$\frac{d}{dt}r_7 = v_4 + v_{10} - v_5 - v_6 - v_8$$
$$\frac{d}{dt}R_7 = v_5 + v_6 - v_{10}.$$

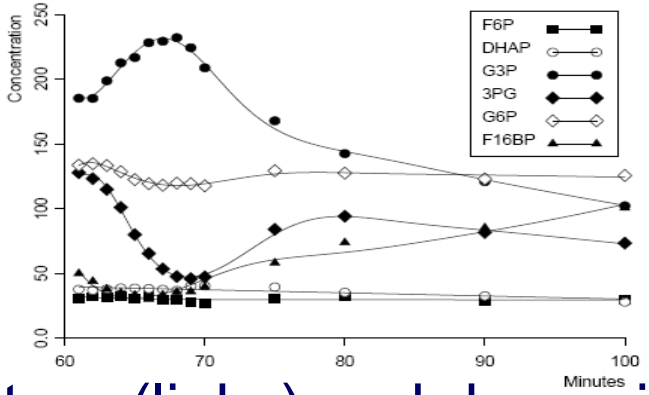
$$v_1 = c_1$$
$$v_2 = \frac{c_2 r_5(t)}{1 + e^{(c_3 - R_5(t))c_4}} \frac{t}{100 + t}$$
$$v_3 = c_5 r_5(t)$$
$$v_4 = c_6$$
$$v_5 = \frac{c_7 r_7(t) R_7(t)^{c_8}}{c_9 + R_7(t)^{c_8}}$$
$$v_6 = \frac{c_{10} r_7(t)}{1 + e^{(c_{11} - R_5(t))c_{12}}}$$
$$v_7 = \frac{c_{13} R_5(t)}{1 + e^{(c_{14} - R_7(t))c_{15}}}$$
$$v_8 = c_{16} r_7(t)$$
$$v_9 = c_{17} R_5(t)$$
$$v_{10} = c_{18} R_7(t).$$

‘Reconstructing’ Networks is an IPM task

Input

- Domain knowledge (partial models, basic processes)
- Time course data



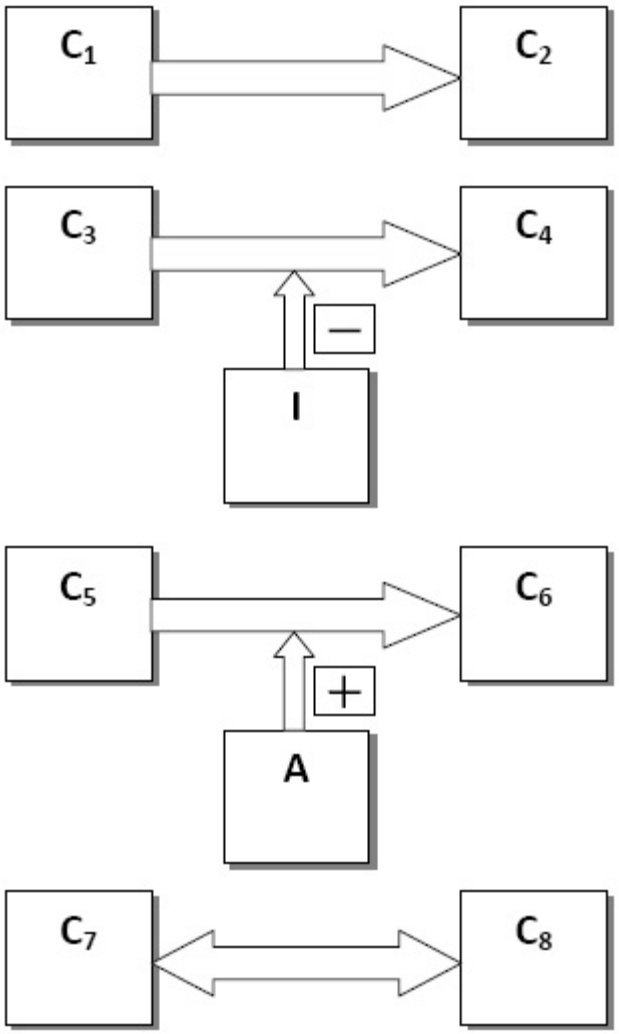


Output: Network structure (links) and dynamics (ODEs)

Metabolic Networks: Library for IPM

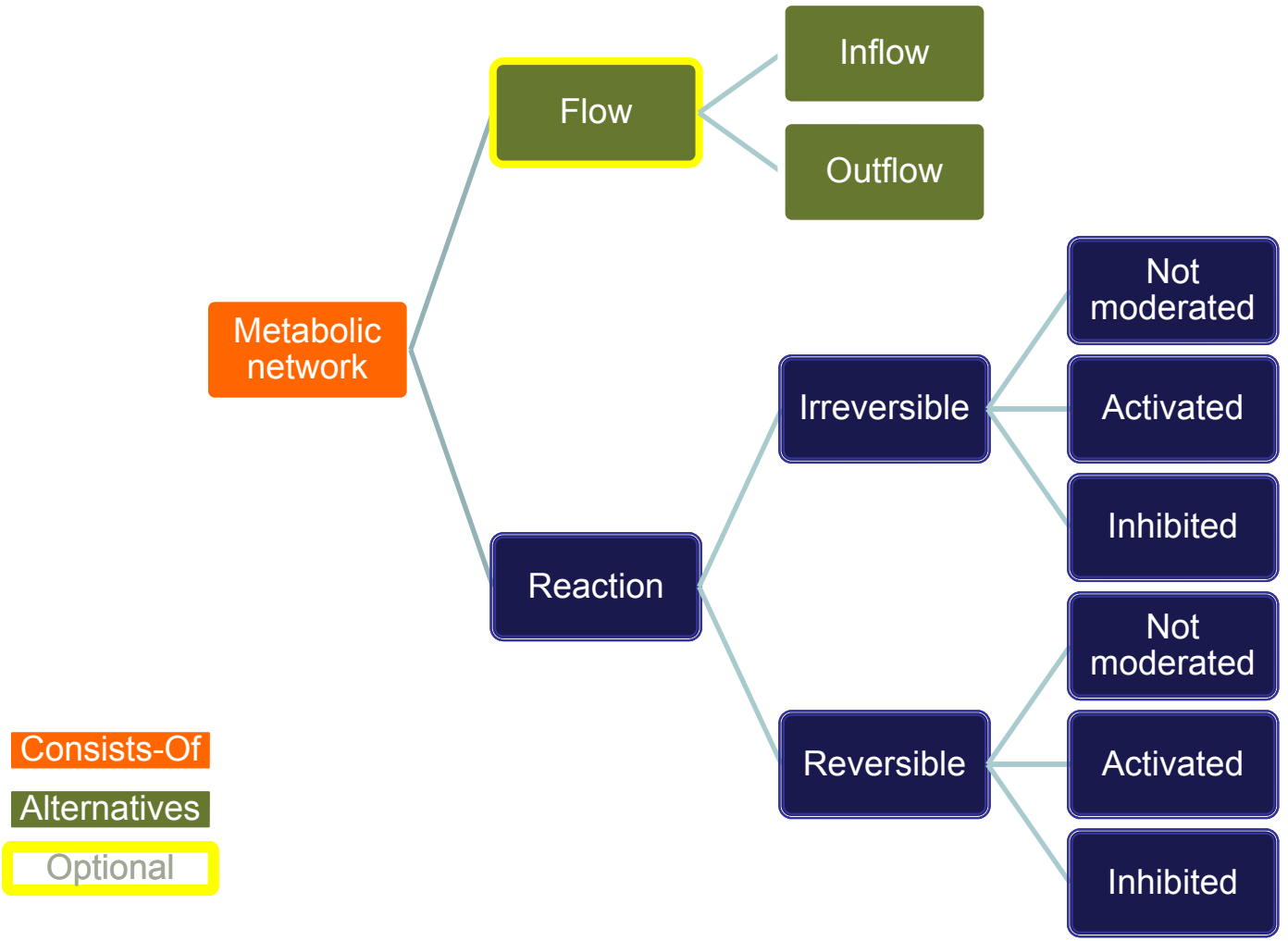
- Entities = chemical compounds
- Processes = chemical reactions
- Entities can have different roles in reactions
  - Substrates are input compounds
  - Products are output compounds
  - Modulates are enzymes that activate/inhibit the reaction

Types of Reactions in Reaction Networks



- Irreversible
- Inhibition
- Activation
- Reversible

Modeling Knowledge for Metabolic Networks



Template Processes: Irreversible

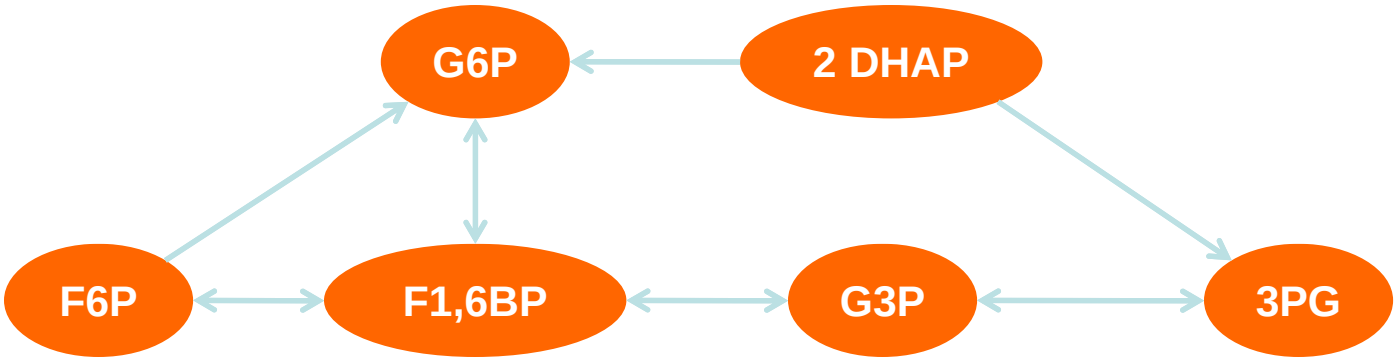
- template process Irreversible_not_modulated
  - variables *S{compound}, P{compound}*
  - constants *reaction_rate(0, Inf)*
  - equations
    - *dS / dt = –1 × reaction_rate × S*
    - *dP / st = reaction_rate × S*

- template process Irreversible_activated
  - variables *S{compound}, P{compound}, M{compound}*
  - constants *reaction_rate(0, Inf), modulation_rate(0, Inf)*
  - equations
    - *dS / dt = –1 × reaction_rate × S × S / (S + modulation_rate)*
    - *dP / st = reaction_rate × S × S / (S + modulation_rate)*

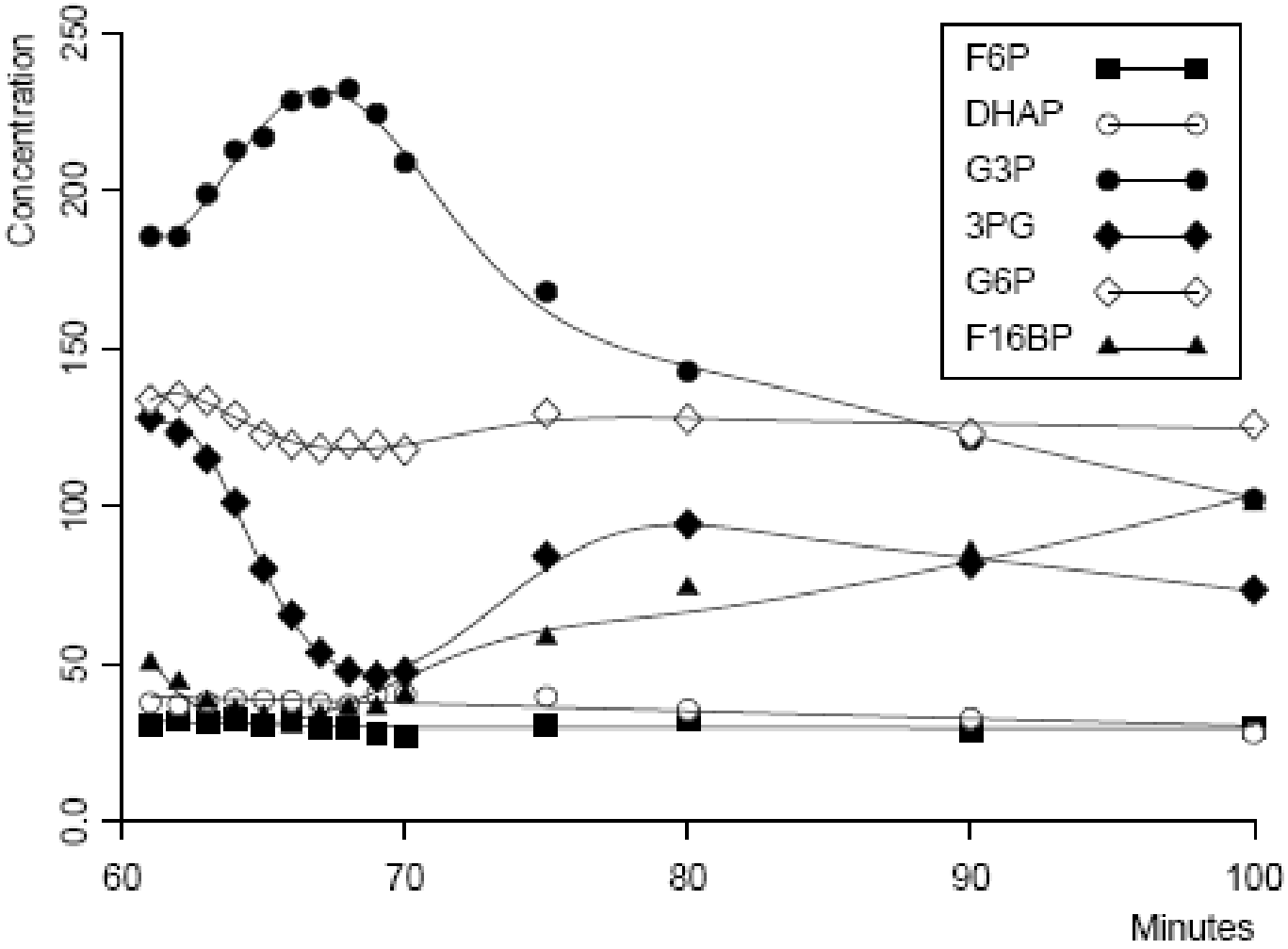
Example Application: Glycolisys

- Inducing (partial) chemical network of glycolisys
  - Data: temporal responses of species to pulse changes (14 time points)
  - From: Torralba et al. (2003) PNAS 100(4): 1494-1498
- Responses of six chemical compounds:
  - G6P (glucose 6-phosphate)
  - F6P (fructose 6-phosphate)
  - F1,6BP (fructose 1,6-bisphosphate)
  - G3P (glycerol 3-phosphate)
  - 3PG (3-phosphoglycerate)
  - DHAP (dihydroxyacetone 3-phosphate)
- Library of domain knowledge as above

Induced Glycosis Network

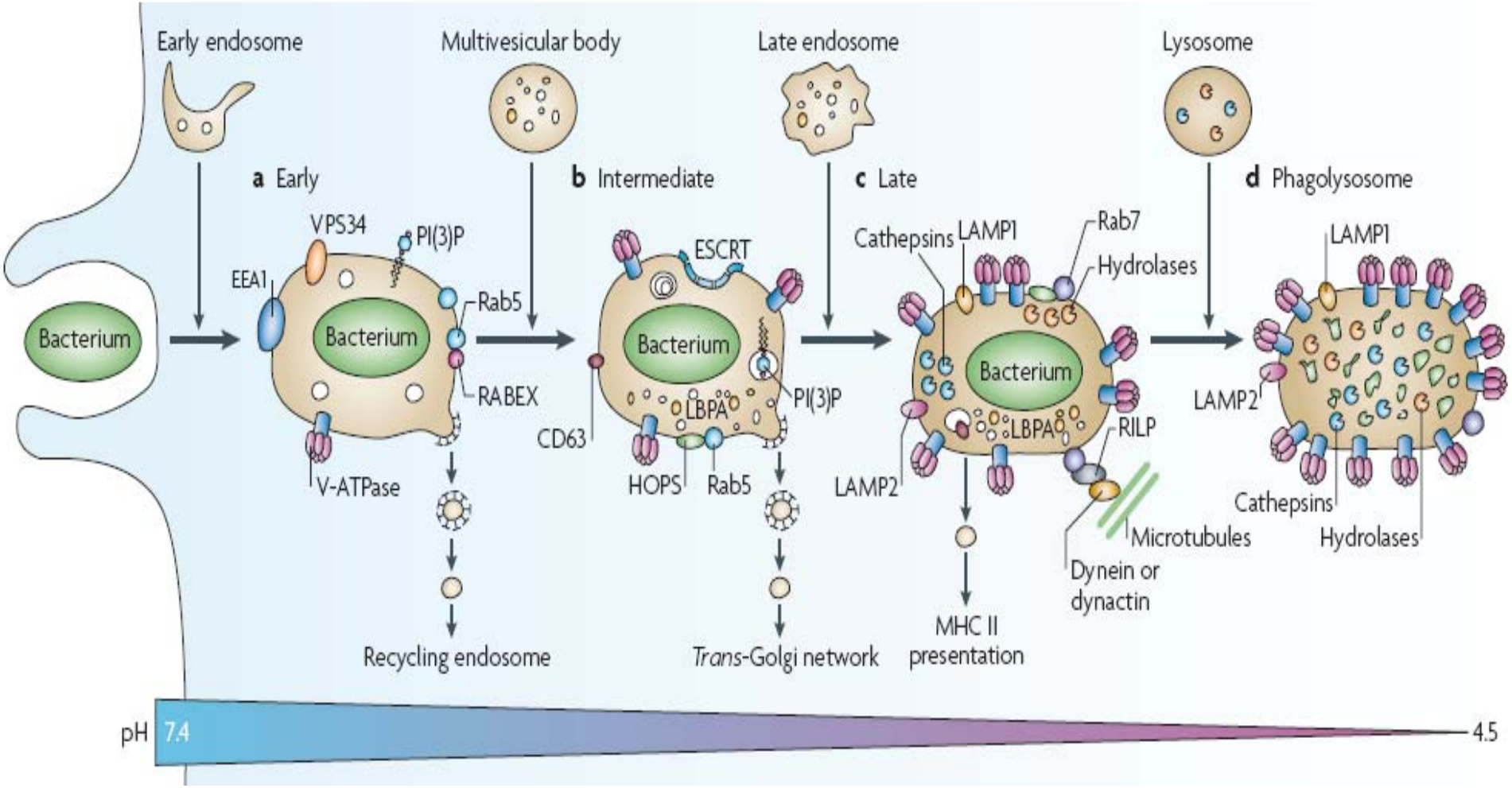


Apps in SB/Glycolisys: Measured and predicted



Systems Biology: Modelling phagocytosis

- The endocytic/phagocytic pathway

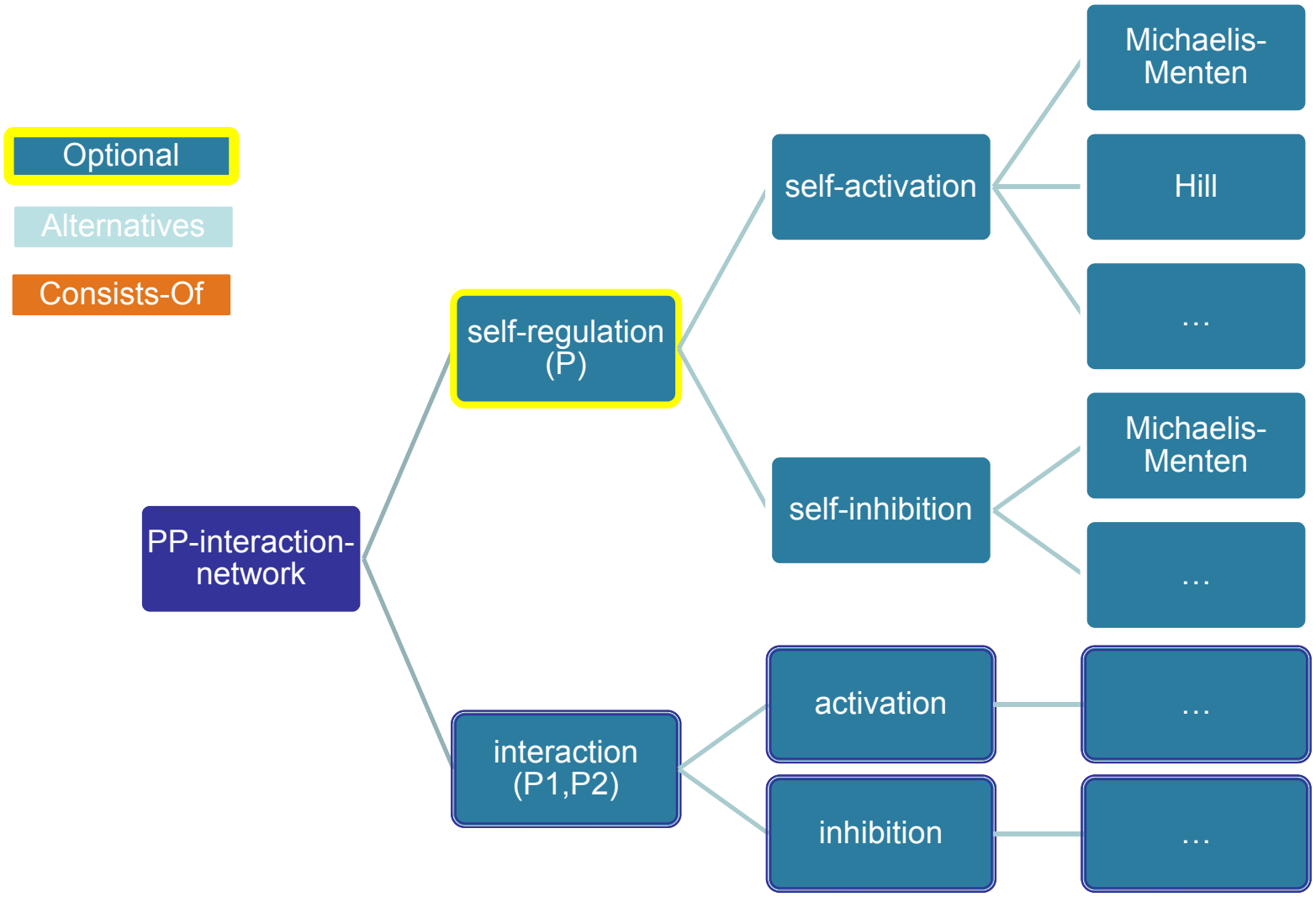


Endosome maturation

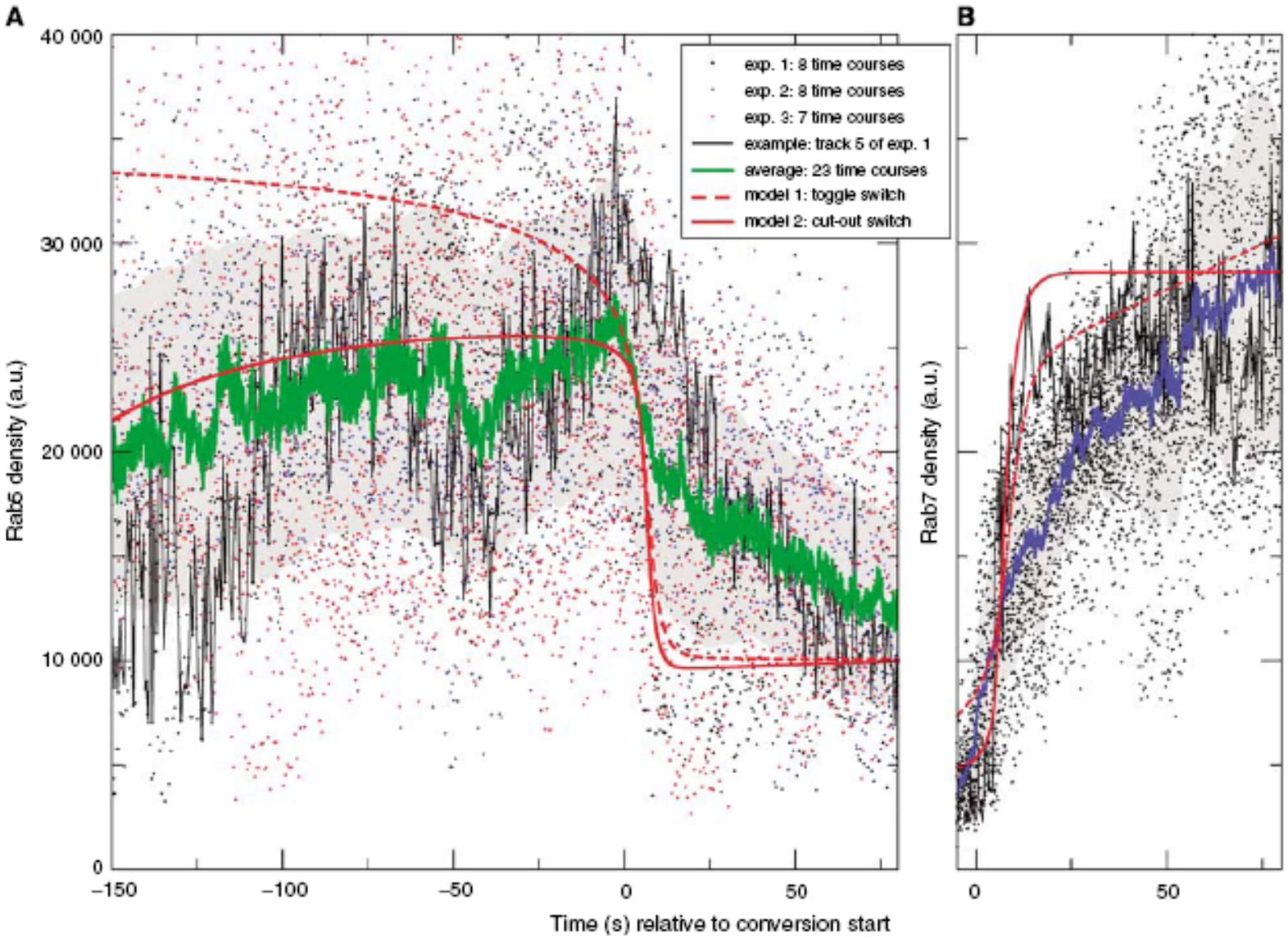
- Early endosomes: pH 6.0-6.5, **rich in Rab5**, EEA1, syntaxin 13, endobrevin, PI(3)P
- Late endosomes: pH 5.0-6.0, **rich in Rab 7**, M6PR, VAMP7, syntaxin 7, vATPase, LAMP-1/-2, lusosbisphospatidic acid

- Endosome maturation: early -> late endosome
  - Rab conversion crucial in the process (Rab5 to Rab7)
  - Expected behavior: switch from high Rab5/low Rab7 to low Rab5/high Rab7
- Different possible switches: toggle vs. cut-out

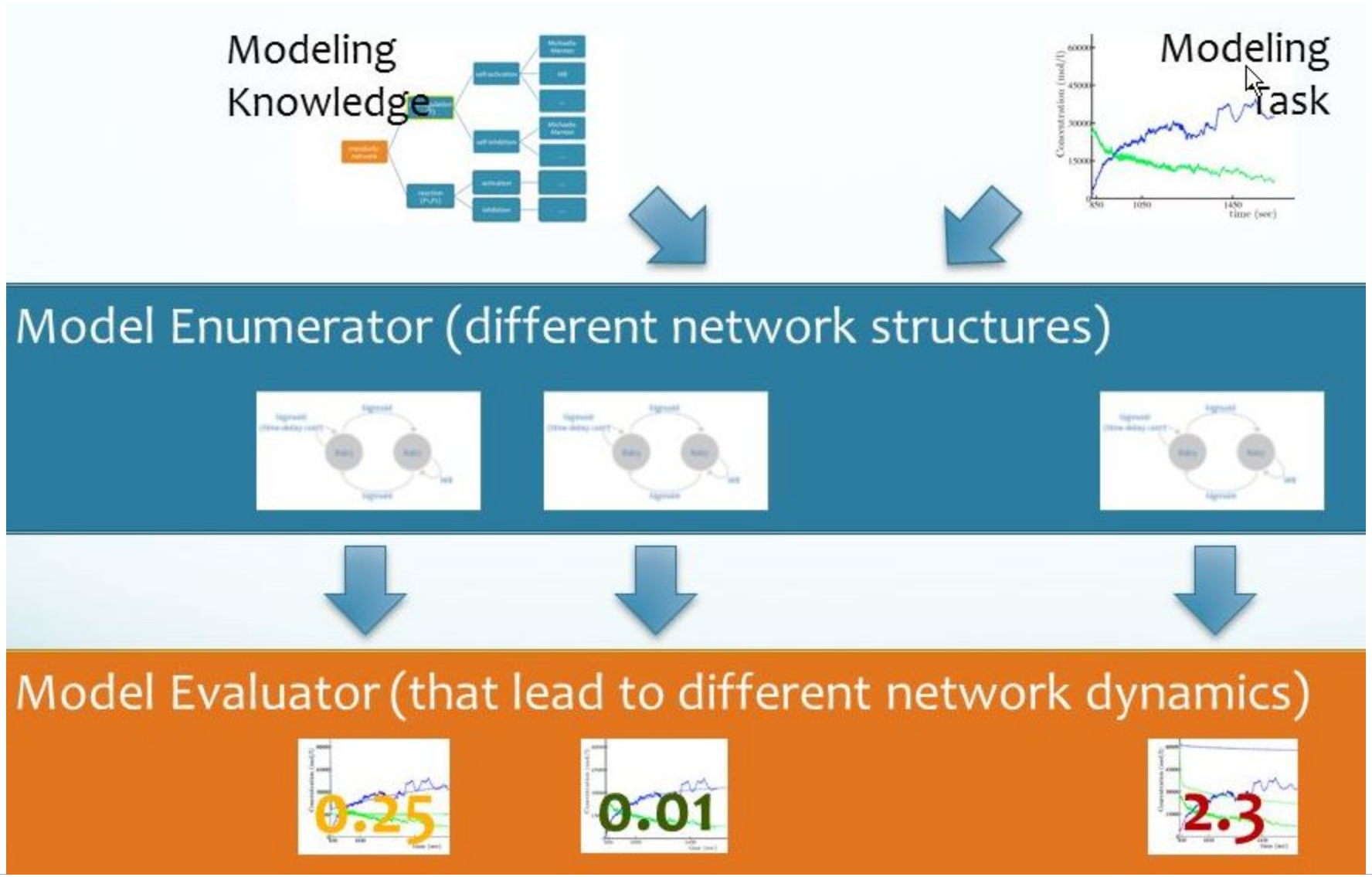
The GTPase modeling library: Processes



The Data


A
B
40 000
30 000
20 000
10 000
0
Rab5 density (a.u.)
Rab7 density (a.u.)
exp. 1: 8 time courses
exp. 2: 8 time courses
exp. 3: 7 time courses
example: track 5 of exp. 1
average: 23 time courses
model 1: toggle switch
model 2: cut-out switch
–150
–100
–50
0
50
0
50
Time (s) relative to conversion start

Modeling endosome maturation

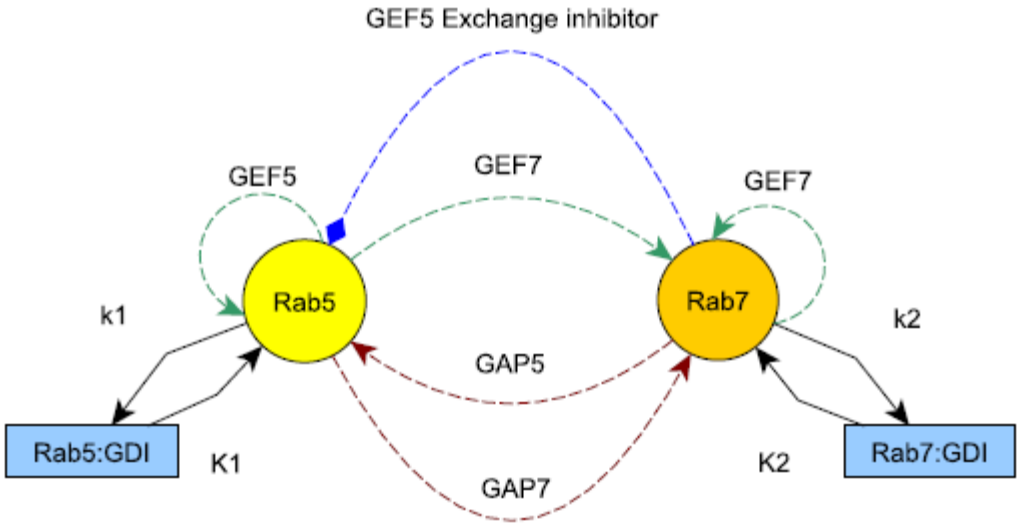


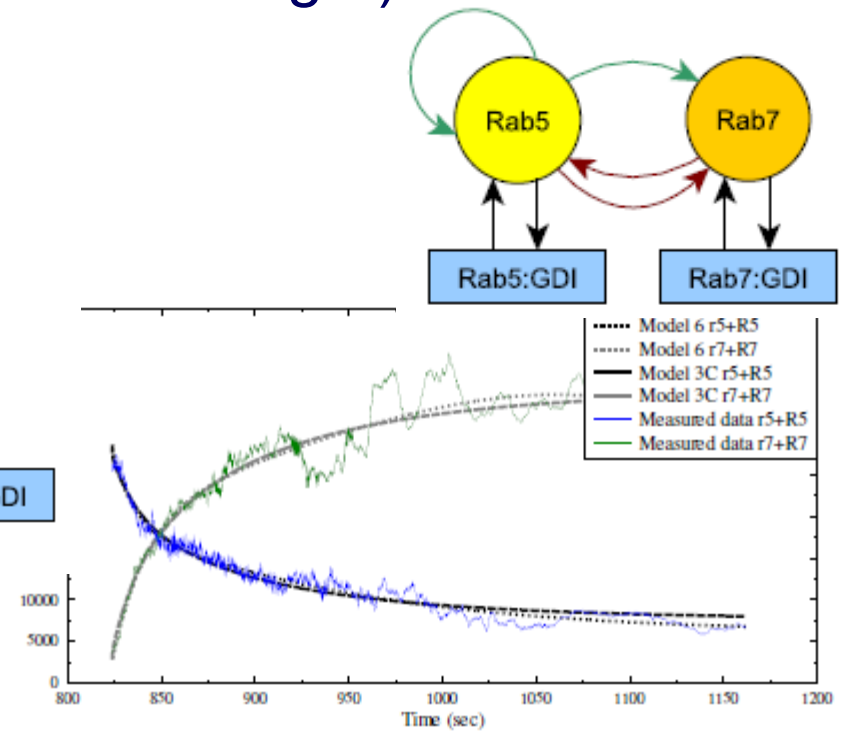
Model by del Conte Zerial et al. and alternative proposed by ProBMoT

- Criteria: RMSE and BIC (Bayesian Information Crit.)

$$RMSEObjective(m) = \sum_{i=1}^{2} \sqrt{\frac{1}{N} \sum_{j=1}^{N} (x_i[j] - y_i[j])^2} \qquad BIC(m) = N \ln(MSE(m)) + k \ln(N)$$

- The structures (dCZ left, ProbMoT right)





Systems vs. Synthetic Biology

Systems Biology: Re-constructing biological networks

- From **observed** behavior
- Reverse engineering

Synthetic biology: Constructing biological circuits (nets)

- That would produce **desired** behavior
- Design / engineering of biological circuits

What is common to both?

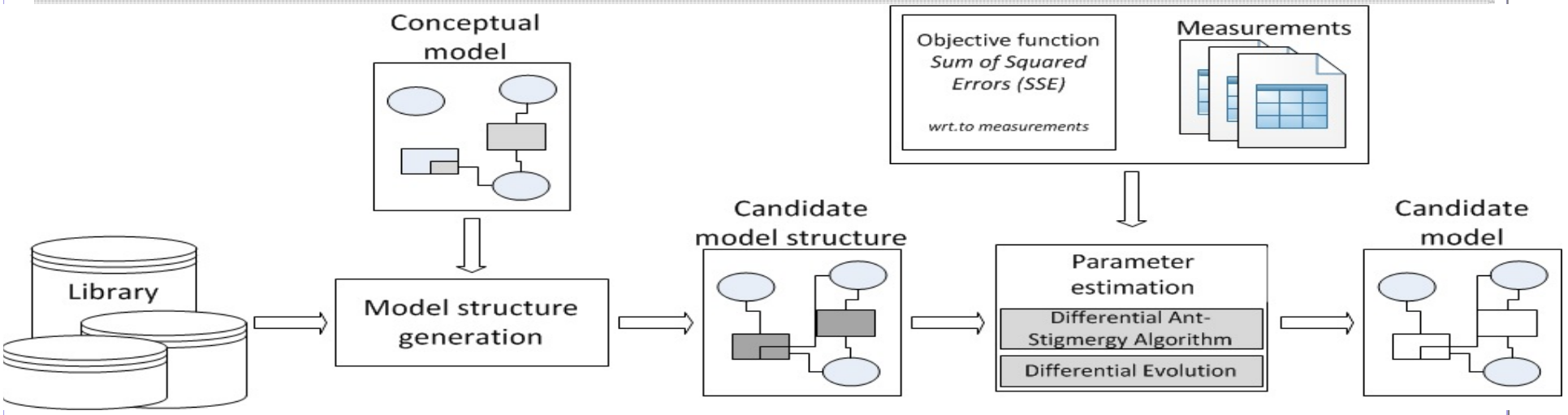
- The use of models of the dynamics of the circuits
- To investigate (in-silico) the behavior of the circuits

IPM for Synthetic Biology

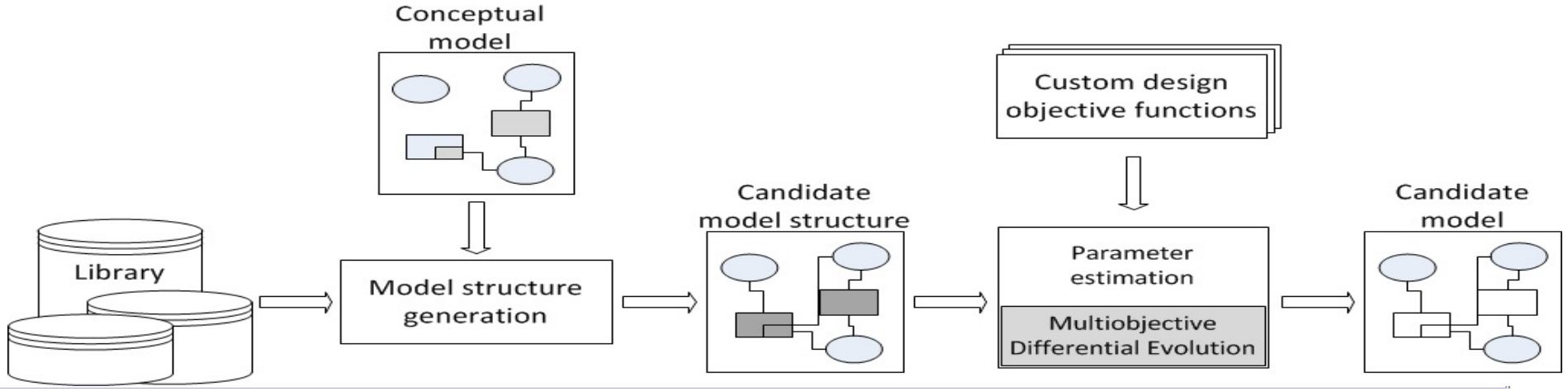
- No observed data. Instead, formalized expected behavior in the form of **custom objective functions**
- **Multi-objective** optimization needed
- Model parameters are fitted so that the candidate model exhibits desired behavior (objective functions)

- Input:
  - Library of domain knowledge, conceptual model
  - List of behavior objectives (objective functions)
- Output:
  - Suitable models and corresponding sets of parameters

ProBMoT for Systems Design vs. Identification



Instead of data, formalized desired behavior in the form of **custom design objective functions.** Multi-obj. opt. needed.



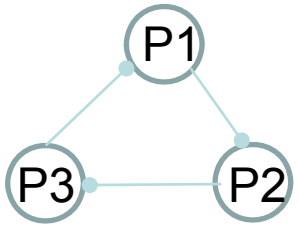
Design of biological circuits with complex behaviours

- Case studies:
  - Repressilator (Elowitz and Leibler)
  - Coupled repressilators (Gao et al)
- Task:
  - Propose design/model of the desired circuit and its parameters to optimize the custom objective function
- Objective function:
  - Largest Lyapunov exponent (λ_1)

    as an indicator of dynamical behavior
    - If $\lambda_1 = 0$ the system is oscillatory
    - If $\lambda_1 > 0$ the system is chaotic
    - If $\lambda_1 < 0$ the system is stable

Case study: Repressilator

- Repressilator
  - Synthetic genetic regulatory network
  - 3 repressor proteins & corresponding mRNA
  - Connected in a repression loop

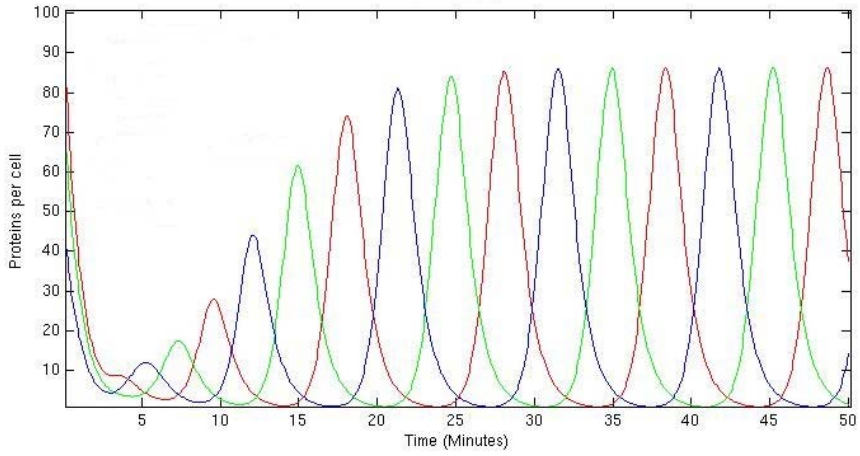


- Objective function – minimization of the absolute value of the largest Lyapunov exponent (targeting oscillatory beh.)

$$\frac{dm_i}{dt} = -m_i + \frac{\alpha}{1 + p_j^n} + \alpha_0$$

$$\frac{dp_i}{dt} = -\beta(p_i - m_1)$$

- (i,j) varies trough (1,3), (2,1), (3,2)



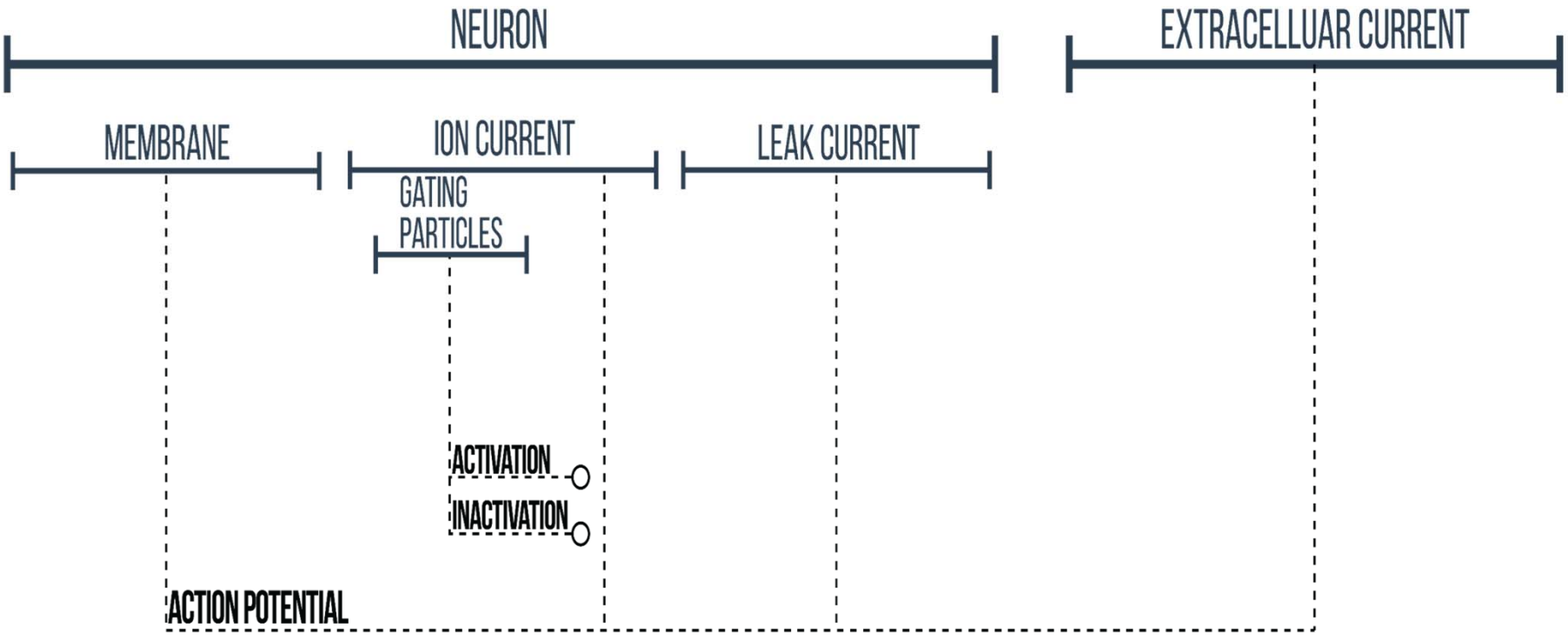
Case study: Coupled Repressilators

- Coupled repressilators
  - Two repressilators (x and y).
  - Coupling represented by modifying one equation in each repressilator (γ – coupling strength).
- Objective function – maximization of the largest Lyapunov exponent (targeting chaotic behavior)

$$\frac{dx_{m_3}}{dt} = -x_{m_3} + \frac{\alpha}{1 + \left(x_{p_2} - \gamma y_{p_2}\right)^n} + \alpha_0,$$

$$\frac{dx_{m_3}}{dt} = -y_{m_3} + \frac{\alpha}{1 + \left(y_{p_2} - \gamma x_{p_2}\right)^n} + \alpha_0$$

Library of domain knowledge for modeling neurons

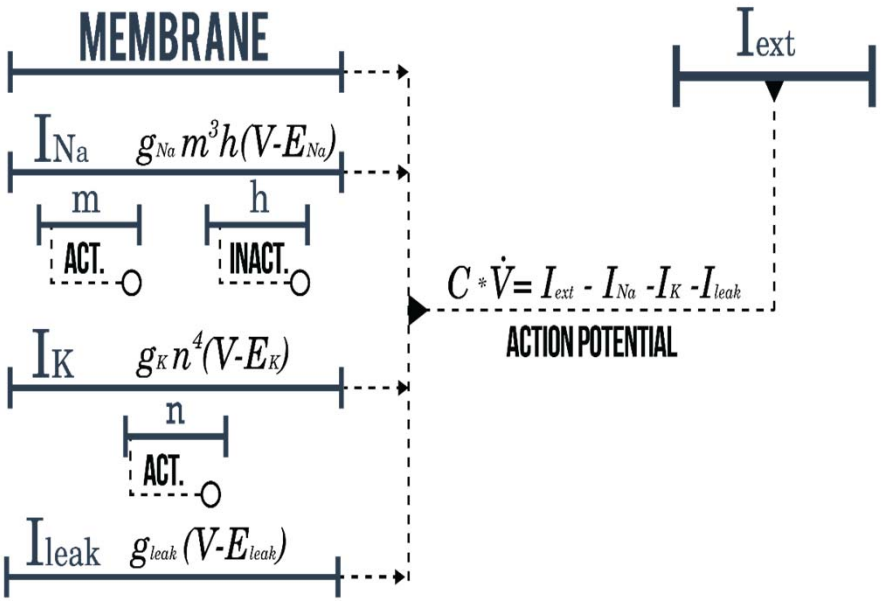


Library of domain knowledge for modeling neurons

```
template entity membrane{
     consts:
          C { range: <0,1>; unit:"uF/cm^-2"};
     vars:
          V {aggregation:sum; unit:"mV"; range:<-100,100>};}
template entity IonCurrent {
     consts:
          g { range: <0,200>; unit:"mS/cm^-2"},
          E { range: <-100, 100>; unit:"mV"};
     vars:
          Ic { aggregation:sum; range:<-100,100>}; }

template process Current ( m:membrane, iC:IonCurrent){}
template process ionCurrentProcess ( gP:gatingParticle) : Current{
     equations:
          iC.Ic = iC.g*pow(gP.gp,gp.n)*(m.V - iC.E); }

template process leakCurrent: Current{
     equations:
          iC.Ic = iC.g *(m.V - iC.E); }

template process membranePotential (m:membrane, iCs:IonCurrent<1,10>,
                                    eC: ExCurrent){
     equations:
          td(m.V) = (eC.I_ex - <iC:iCs>.Ic)/m.C; }
```

Hodgkin-Huxley Model

Process-based Model



Ordinary Differential Equations

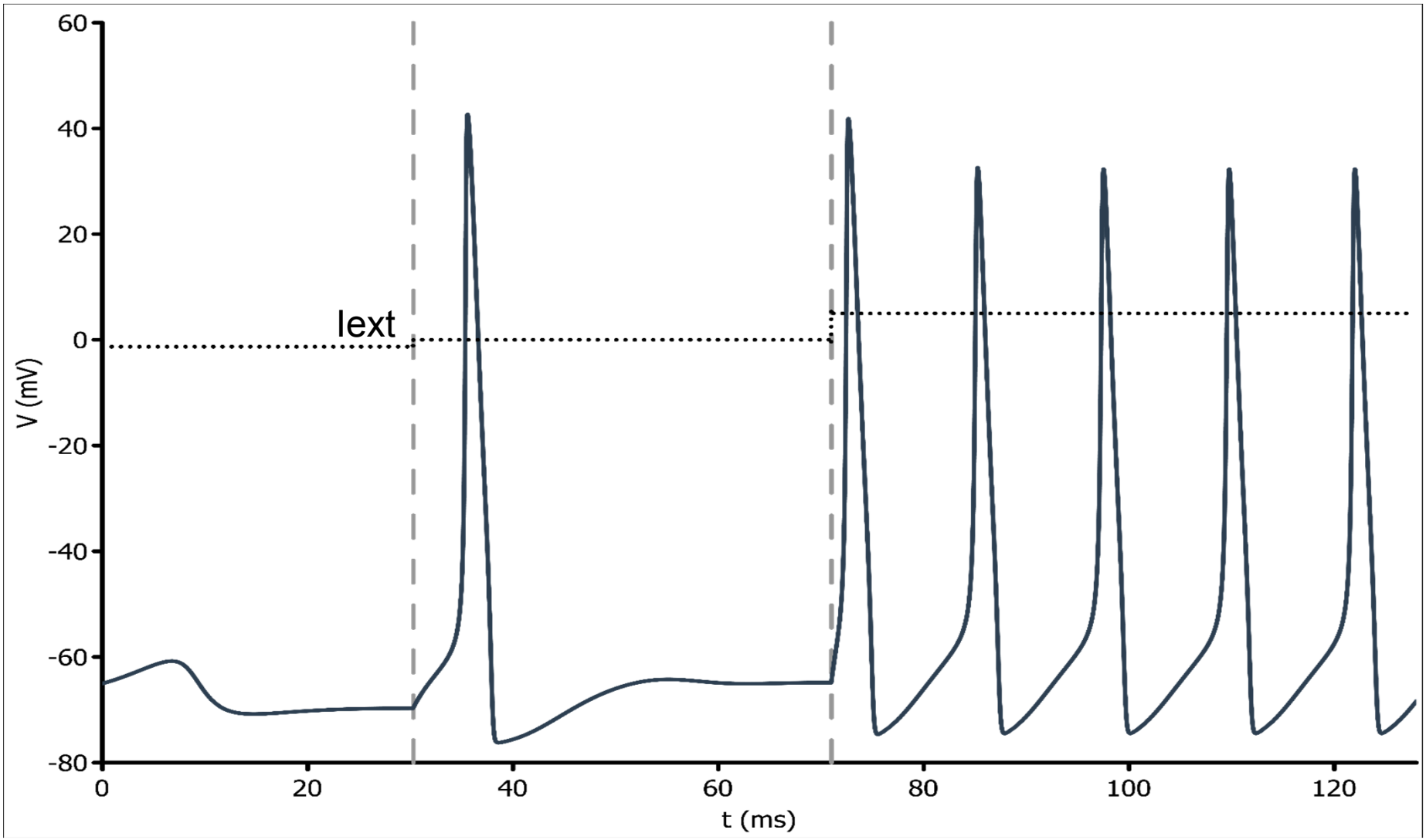
$$C * \dot{V} = I_{ext} - g_{Na} m^3 h(V-E_{Na}) - g_K n^4(V-E_K) - g_{leak}(V-E_{leak})$$

$$\dot{m} = [m - m_\infty(V)] / \tau_m(V)$$

$$\dot{h} = [h - h_\infty(V)] / \tau_h(V)$$

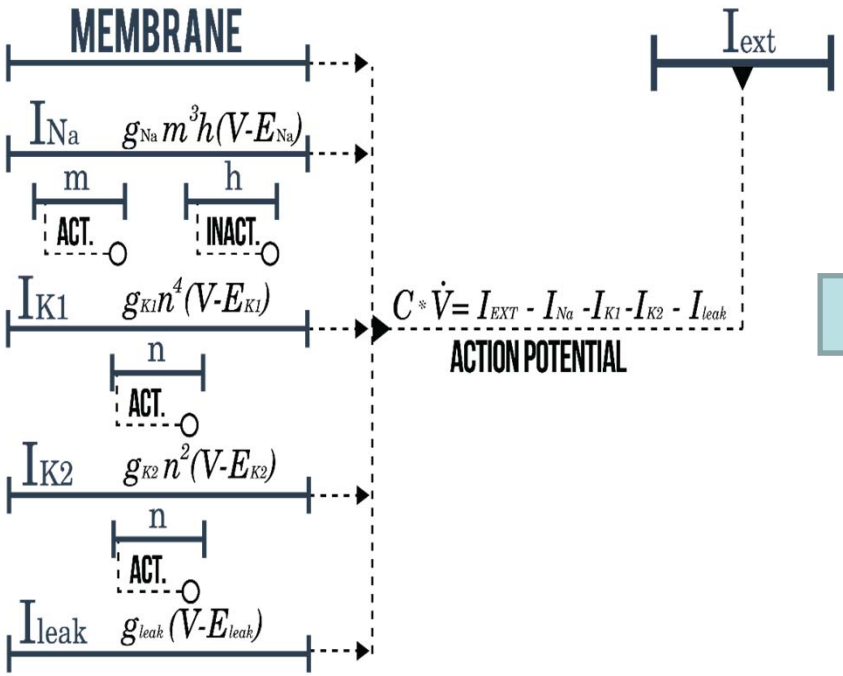
$$\dot{n} = [n - n_\infty(V)] / \tau_n(V)$$

Hodgkin-Huxley Model



Fast-spiking Cortical Interneuron

Process-based model



Ordinary Differential Equations

$$C * \dot{V} = I_{ext} - g_{Na} m^3 h(V-E_{Na}) - g_{K1} n_1^4 (V-E_{K1}) - g_{K2} n_2^2 (V-E_{K2}) - g_{leak} (V-E_{leak})$$

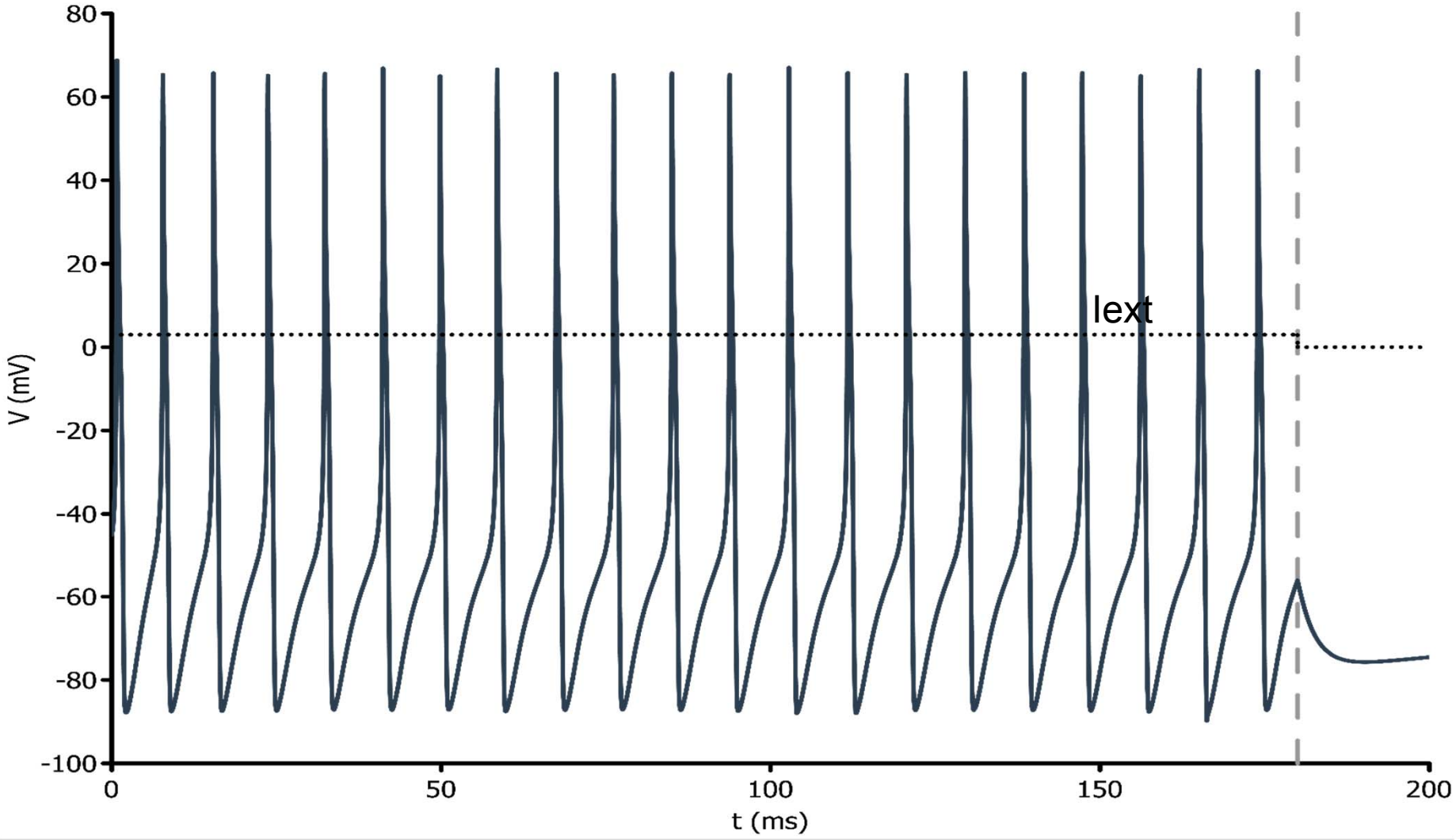
$$\dot{m} = [m - m_\infty(V)] / \tau_m(V)$$

$$\dot{h} = [h - h_\infty(V)] / \tau_h(V)$$

$$\dot{n}_1 = [n_1 - n_{1\infty}(V)] / \tau_{n1}(V)$$

$$\dot{n}_2 = [n_2 - n_{2\infty}(V)] / \tau_{n2}(V)$$

Fast-spiking Cortical Interneuron



Summary & Outlook

- Proposed process-based representation of models and domain knowledge
- Natural, understandable, ontologically grounded
- Includes both quantitative and qualitative aspects
- Proposed methods for learning process-based models

- Applications in systems biosciences: Systems ecology, Systems Biology, Synthetic Biology

- Summary: Automated modeling/identification, design

- Outlook: **Automating Systems Sciences**
  Using machine learning also for analysis & control