DEPARTMENT OF KNOWLEDGE TECHNOLOGIES
Jožef Stefan Institute

AUTOMATED SCIENTIFIC MODELLING: INTEGRATING DATA- AND KNOWLEDGE- DRIVEN APPROACHES

SCI. DISCOVERY AS PROBLEM SOLVING

Many philosophers of science have held mystical views of discovery, believing it to be ‘immune to logical analysis’, ‘inherently irrational’ and ‘beyond understanding’

Herbert Simon (1966) put forward the view that

- Problem solving can be viewed as heuristic search
- Scientific discovery is a kind of problem solving, involving
  - *Search* through a space of *problem states*
  - Generated by applying mental *operators*
  - Guided by *heuristics* to make it tractable

This paved the way for understanding and automating sci. disc.

COMPUTATIONAL SCIENTIFIC DISCOVERY

Scientific Discovery is the process by which scientists create or find hitherto unknown knowledge, such as

- A new class of objects (e.g., new class of celestial objects, say quasars, or a new species of living organisms)
- An empirical law, e.g., Kepler's law of planetary motion
- An explanatory theory, e.g., Newton's theory of gravity

Computational scientific discovery aims to

- Provide computational support for and
- Automate certain aspects of this process

SCIENTIFIC KNOWLEDGE STRUCTURES

In the process, of scientific discovery, scientists generate and manipulate/revise scientific knowledge structures

- Observations
- Taxonomies
- Laws
- Theories
- Models, Predictions, Explanations (derived from the above)

Taxonomies

- Define or describe concepts for a domain, along with specialization relations among them
- Specify the concepts and terms used to state laws and theories

SCIENTIFIC KNOWLEDGE STRUCTURES

Laws: Summarize relations among observed variables, objects or events

Theories:

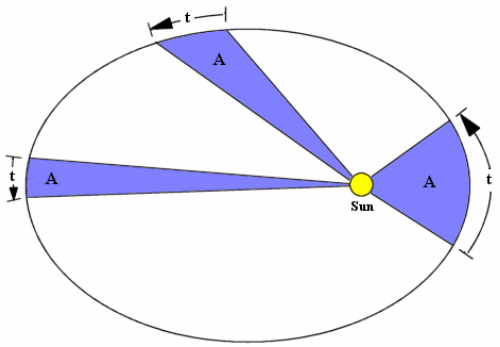
- Statements about the structures or processes that arise in the environment
- Stated using terms from the domain's taxonomy
- Interconnect laws into a unified theoretical account

Models: More specific than laws, adapt a general law to a specific situation/system

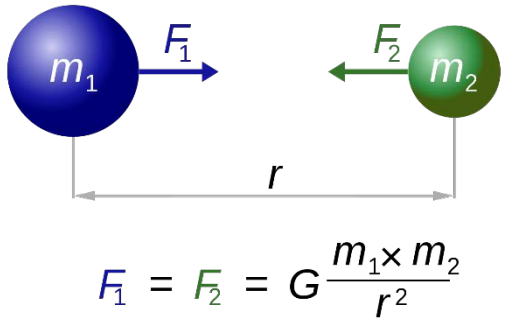
- E.g., F = m x a (Newton's second law)
- For a specific object of mass 2.3 kg, we would have the model F = 2.3 x a

SCIENTIFIC KNOWLEDGE: REPRESENTATION

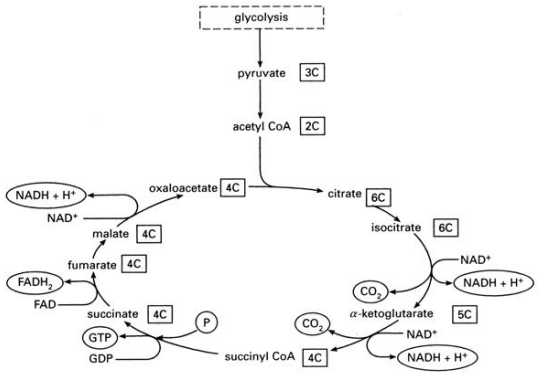
Scientific knowledge is typically represented in scientific formalisms (e.g., equations, pathways), introduced and routinely used by scientists



Kepler's laws of planetary motion



Newton's theory of gravitation



Krebs' citric acid cycle

It involves domain expertise, e.g., concepts from the studied scientific domain

Is accessible to/communicable among domain scientists

COMPUTATIONAL SCIENTIFIC DISCOVERY: FINDING EQNS IN DATA

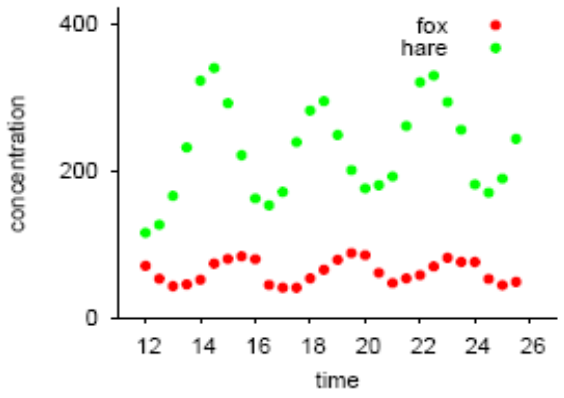
- Input: Observations of distances between moons and Jupiter (d) and periods of their orbits (p)
- Output: Kepler's third law $d^3/p^2 = k$
- Process of discovery
  - BACON (Langley 1978; Langley et al. 1987)
  - Carries out heuristic search through the space of numeric terms, looking for constant values and linear relations

Moon	d	p	d/p	d^2/p	d^3/p^2
A	5.67	1.77	3.20	18.15	58.15
B	8.67	3.57	2.43	21.04	51.06
C	14.00	7.16	1.96	27.40	53.16
D	24.67	16.69	1.48	36.46	53.89

- Proceeds from observed variables to constant theoretical term

AUTOMATED MODELLING OF DYNAMIC SYSTEMS: FINDING ODEs IN DATA

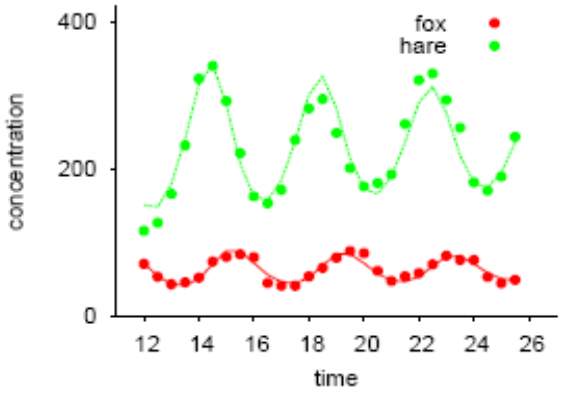
- Input: Observed behavior of dynamic system



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

- Output: System of ODEs

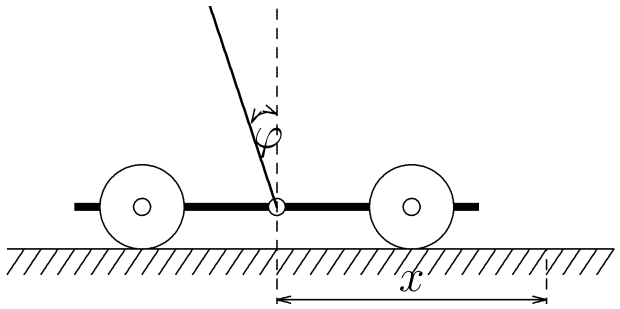
$$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$$
$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$



FINDING POLYNOMIAL ODEs: LAGRANGE

Parametric specification of the search space of polynomial ODEs (Dzeroski and Todorovski 1993)

- Introducing time derivatives up to order **o** (numerically)
- Introducing new terms with multiplication of system variables and their derivatives (up to degree **d**)
- Exhaustive search: Generating and testing equations using linear regression with max **r** independent variables



$$(M+m)\ddot{x} + \tfrac{1}{2}ml(\ddot{\varphi}\cos\varphi - \dot{\varphi}^2\sin\varphi) = F$$

$$\ddot{x}\cos\varphi + \tfrac{2}{3}l\ddot{\varphi} = g\sin\varphi$$

$$x_0 = 0,\ \varphi_0 = 3\pi/16,\ \dot{x}_0 = 0,\ \dot{\varphi}_0 = 0$$
$$M = 1,\ m = 0.1,\ l = 1,\ F = 7.5$$

FINDING POLYNOMIAL ODEs: CIPER

Constrained induction of polynomial equations (Todorovski, Ljubic & Dzeroski 2003)

Input : training data D, dependent variable vd (derivative), number of equations b

Output: the b polynomial equations for vd that are best wrt the data D, according to the MDL heuristics

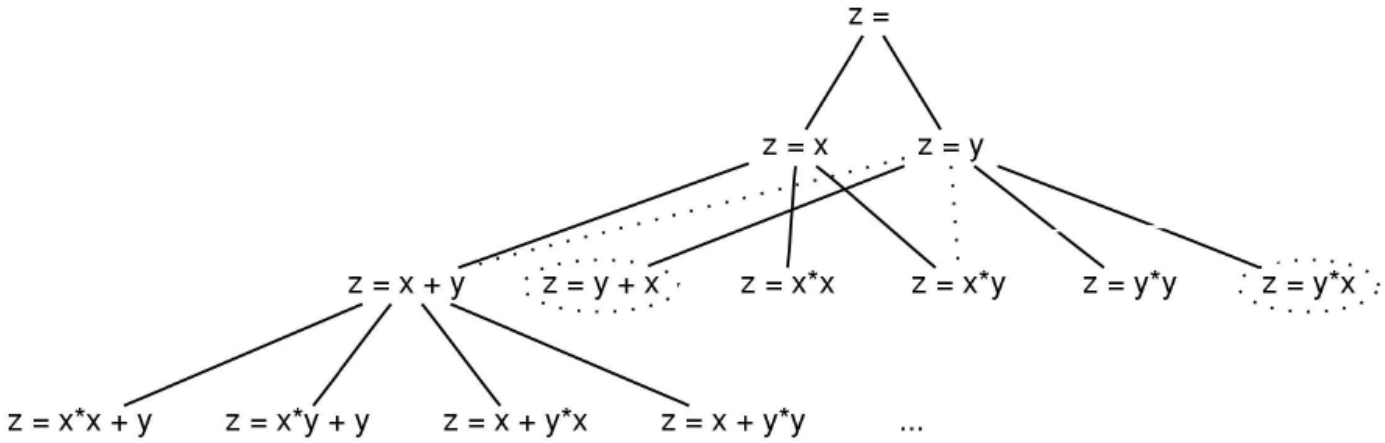
$$\mathrm{MDL}(v_d = P) = \mathrm{len}(P) \log m + m \log \mathrm{MSE}(v_d = P)$$

Heuristic (beam) search through the space of eqns

- Starts with simplest equation vd == const
- Uses a refinement operator on polynomial eqns
- Refinements are super-polynomials of parents

FINDING POLYNOMIAL ODEs: CIPER

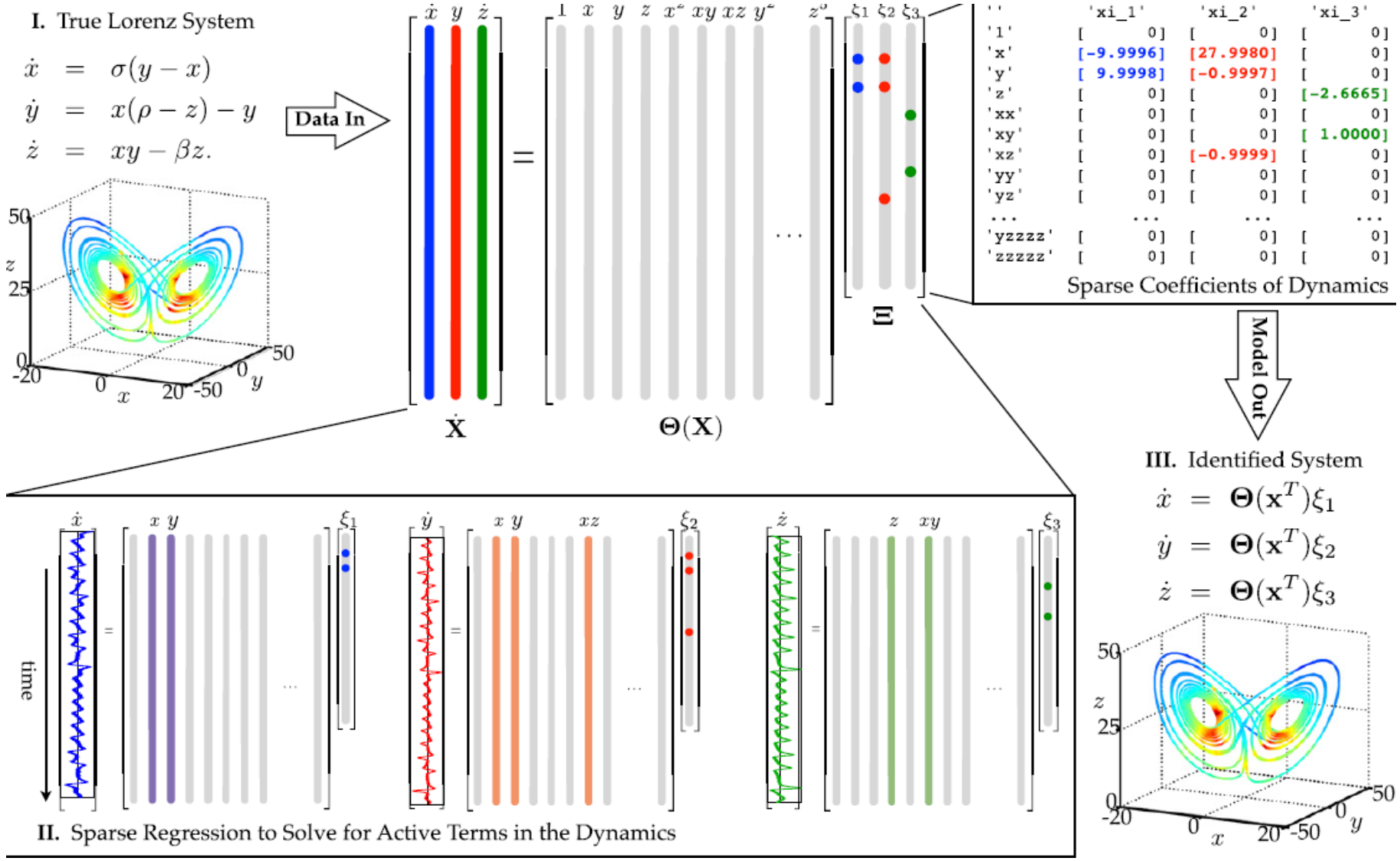
Heuristic search through the space of polynomial equations



Refinement operators

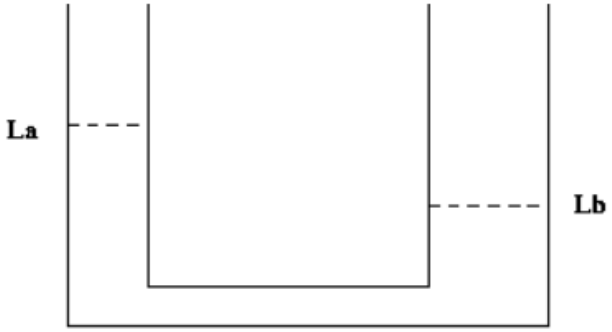
original (current) equation
$v_d = \sum_{i=1}^{r} \text{const}_i \cdot T_i$
refined equations that increase r (one for each $v \in V \setminus v_d$)
$v_d = \sum_{i=1}^{r} \text{const}_i \cdot T_i + \text{const}_{r+1} * v$, where $\forall i : v \neq T_i$
refined equations that increase d (one for each T_j and $v \in V \setminus v_d$)
$v_d = \sum_{i=1, i \neq j}^{r} \text{const}_i \cdot T_i + T_j * v$, where $\forall i \neq j : T_j * v \neq T_i$

SINDY: POLYNOMIAL ODEs WITH SPARSE LR (BRUNTON, PROCTOR & KUTZ 2016)



REPRESENTATION: QUALITATIVE MODELS

- The U-tube system



- Differential equations

$$\dot{l_A} = c(l_B - l_A)$$
$$\dot{l_B} = -\dot{l_A}$$

- Qualitative differential equations (qualitative constraints valid on the system vars. & their derivs.)

$$deriv(l_B, \dot{l_B}) \quad deriv(l_A, \dot{l_A}) \quad minus(\dot{l_A}, \dot{l_B})$$
$$M^+(l_A, \dot{l_B}) \quad M^+(l_B, \dot{l_A}) \quad M^-(l_A, l_B)$$
$$M^-(l_A, \dot{l_A}) \quad M^-(l_B, \dot{l_B}) \quad M^-(\dot{l_A}, \dot{l_B})$$

DATA-DRIVEN & KNOWLEDGE-DRIVEN MODELLING APPROACHES

Data-driven (empirical) modeling focusses on data:

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, which doesn't rely on domain knowledge and is most often black box

Knowledge-driven modelling focuses on domain knowledge:

1. Expert derives proper model, based on knowledge of domain and modelling formalism

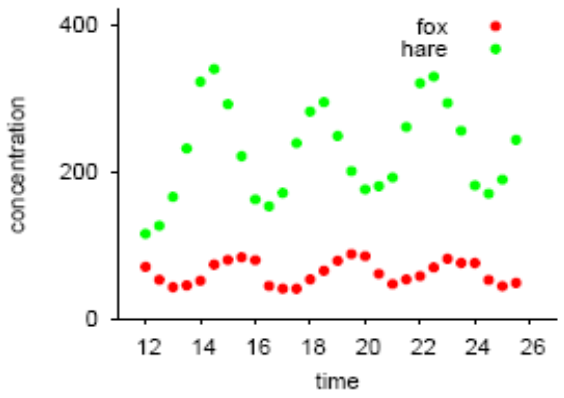
2. Typically, both the structure and parameters of the models are derived by the expert from knowledge about processes and process rates/parameters

INTEGRATING DATA- AND KNOWLEDGE-DRIVEN MODELLING

- Allows flexible trade-off between data and domain knowledge and handles
  - Lots of knowledge and little data or
  - Lots of data and little knowledge, as the case may be
- Produces models that fit the data well, but also make sense from the domain point of view
- Domain knowledge can be about
  - The basic building blocks of systems/ models in the domain
  - Existing models in the domain (that need to be revised)
  - Incomplete models, then need to be completed
- All of these can be expressed with grammars

GRAMMAR-BASED EQUATION DISCOVERY

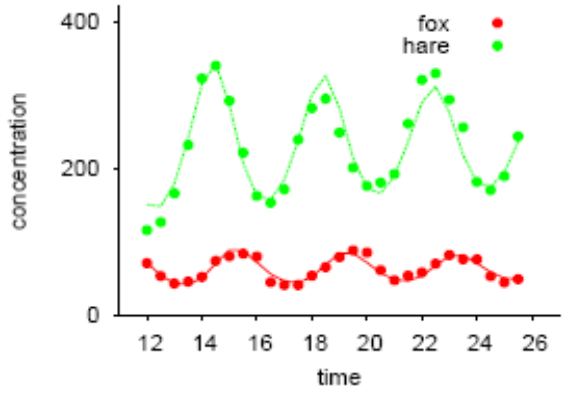
- Input: Observed behavior of dynamic system + Grammar



$$
\begin{array}{rcl}
PPModel & \rightarrow & PreyChange,\ PredatorChange \\
 & & \\
PreyChange & \rightarrow & PreyGrowth - Interaction \\
PredatorChange & \rightarrow & const * Interaction + PredatorLoss \\
 & & \\
PreyGrowth & \rightarrow & const * V_{Prey} \\
PreyGrowth & \rightarrow & const * V_{Prey} * (1 - V_{Prey}/const) \\
 & & \\
Interaction & \rightarrow & const * V_{Predator} * V_{Prey} \\
Interaction & \rightarrow & const * V_{Predator} * V_{Prey}/(V_{Prey} + const) \\
 & & \\
PredatorLoss & \rightarrow & -const * V_{Predator} \\
 & & \\
V_{Prey} & \rightarrow & hare \\
V_{Predator} & \rightarrow & fox
\end{array}
$$

- Output: System of ODEs

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



EXAMPLE CFGS FOR EQUATIONS

- Universal grammar (arithmetic expressions); Polynomials

$$E \rightarrow E + F \mid E - F \mid F$$
$$F \rightarrow F * T \mid F/T \mid T$$
$$T \rightarrow const \mid v \mid (E)$$

$$E \rightarrow const \mid const \cdot F \mid E + const \cdot F$$
$$F \rightarrow v \mid v \cdot F$$

- Context free grammar used for environmental dynamic systems:

$$E \rightarrow const \mid const \cdot F \mid E + const \cdot F$$
$$F \rightarrow v \mid Y \mid v \cdot Y$$
$$Y \rightarrow \texttt{monod}(const, v)$$

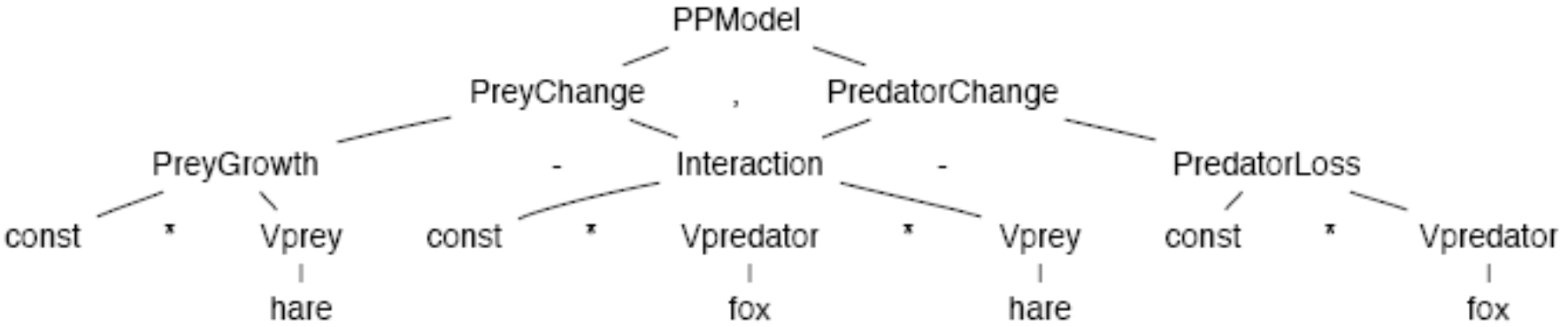
$$Y : \frac{v}{v + const}$$

- User defined function `monod`:

```
double monod(double c, double v) {
   return(v / (v + c));
}
```

GRAMMARS ARE GENERATIVE MODELS

- Parse trees derive expressions from a grammar



- Refinement of parse trees: To get more complex equations, use more complex production rules instead of simpler ones
- E.g., use second instead of first rule below

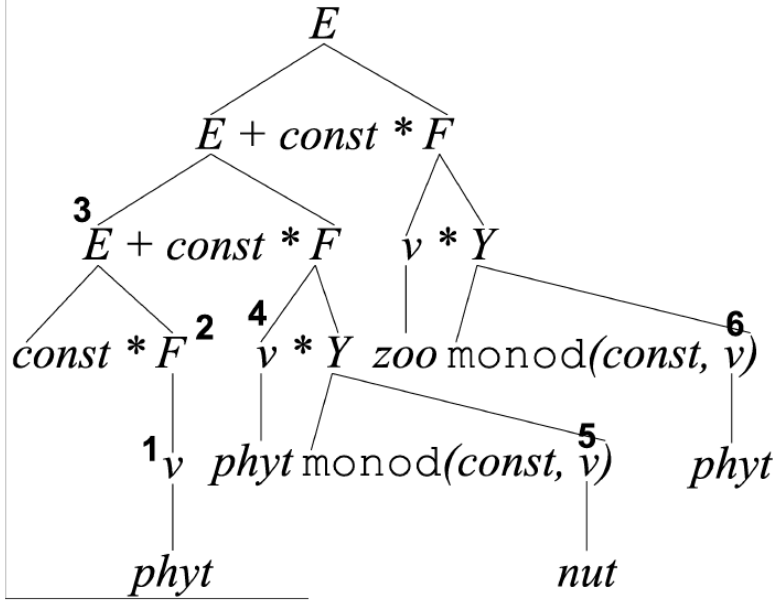
$$PreyGrowth \longrightarrow const * V_{Prey}$$
$$PreyGrowth \longrightarrow const * V_{Prey} * (1 - V_{Prey}/const)$$

EXAMPLE PARSE TREE & REFINEMENT

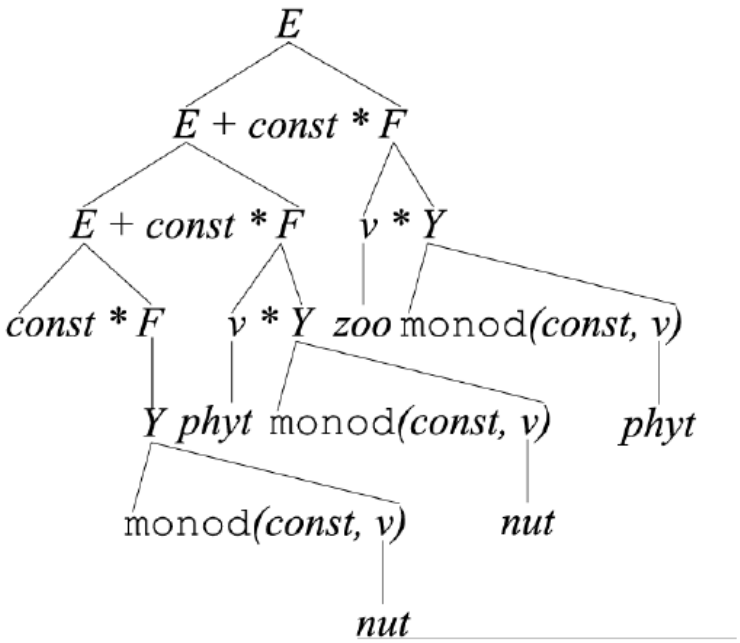
- Expression:

$$const \cdot phyt + const \cdot phyt \cdot \frac{nut}{nut+const} + const \cdot zoo \cdot \frac{phyt}{phyt+const}$$

- Parse tree:



Refined tree (refinement) (2)



SEARCHING THE SPACE OF EQUATION STRUCTURES DEFINED BY THE GRAMS.

LAGRAMGE (Todorovski & Dzeroski 1997)

- Exhaustive search, generate and test equations corresponding to parse trees of limited depth
- Heuristic search (beam search)

1. start with simplest parse tree, then repeat
2. evaluate parse trees, place them in beam, retain k best
3. if beam not changed, stop & report beam content
4. otherwise expand/refine each parse tree on the beam

Parameter estimation (gradient descent, RRR; diff. evolution)

Search can also be performed with grammar-based genetic programming (that's how we got the idea for LAGRAMGE)

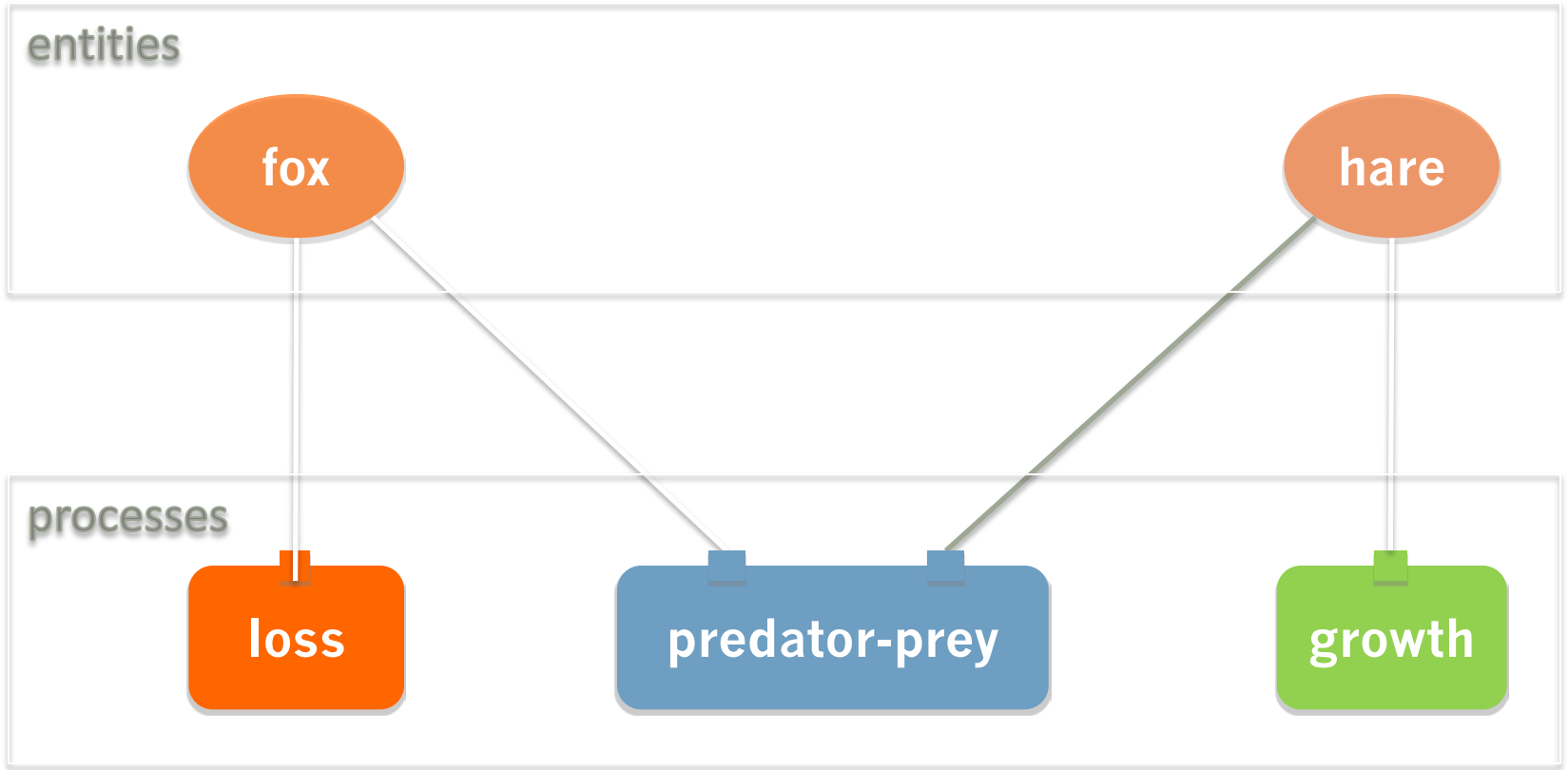
MULTI-LAYER REPRESENTATIONS: PROCESS-BASED MODELS

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

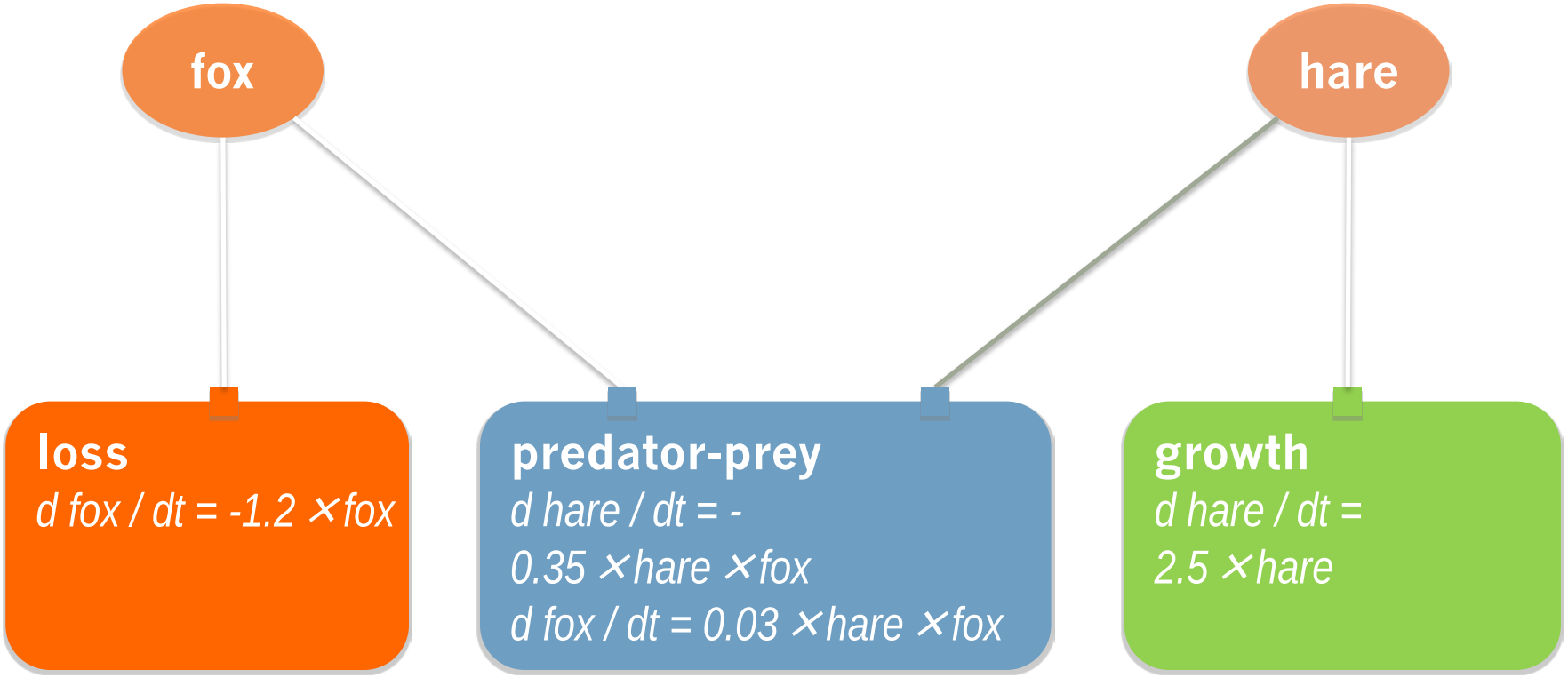
$$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$$
$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$

- 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

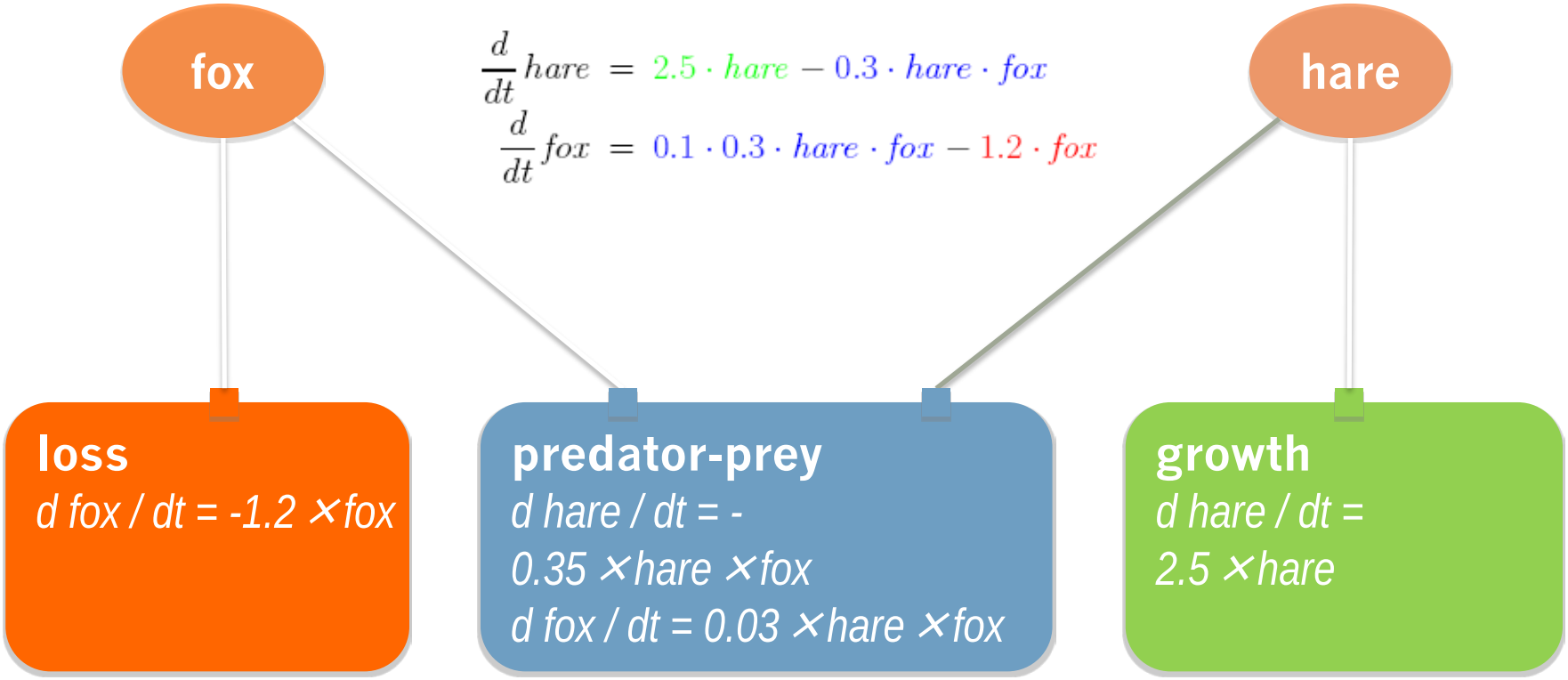
PBM: QUALITATIVE ASPECT



PBM: QUANTITATIVE ASPECT

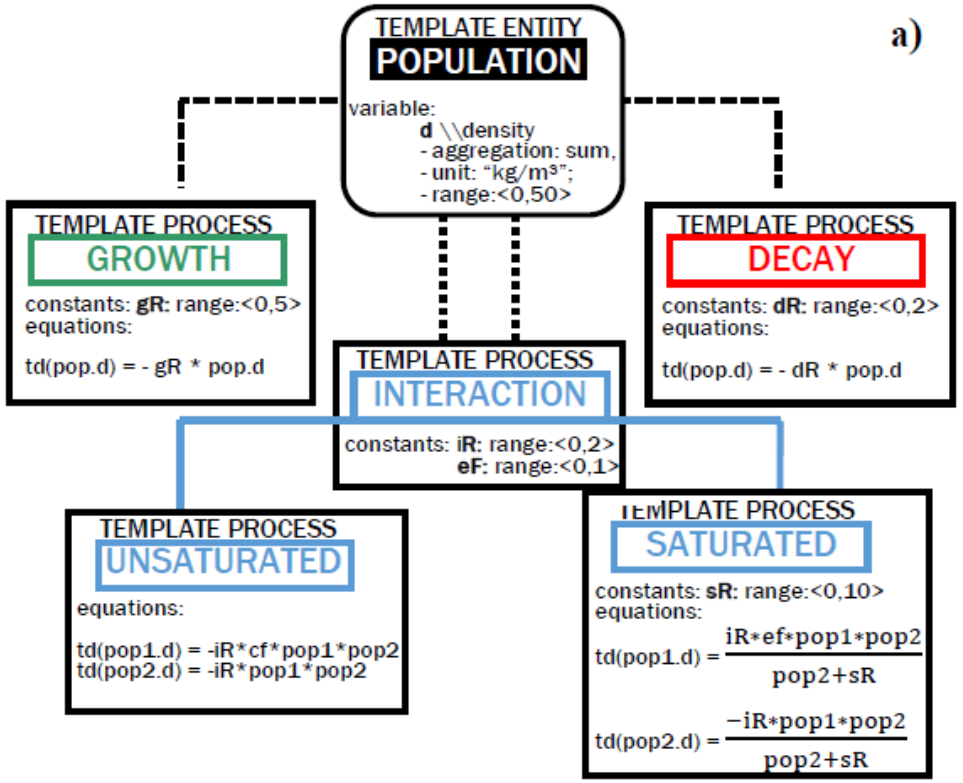


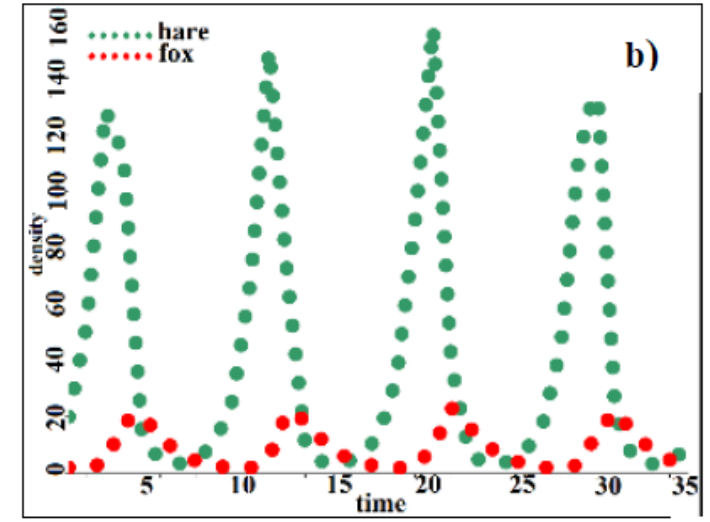
QUAL. & QUANT. ASPECT OF PBMS

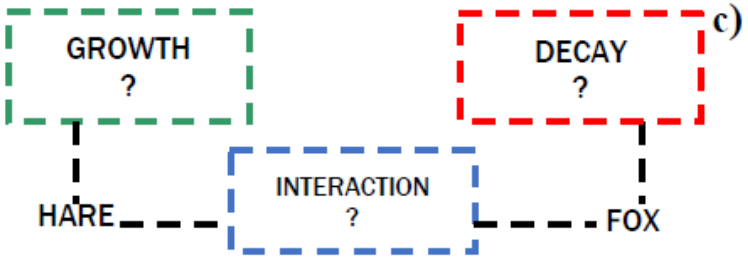


LEARNING PROCESS-BASED MODELS: INPUTS

a) Library of domain knowledge; b) Data (obs. beh.); c) Incomplete model (incl. some template processes)







LEARNING PROCESS-BASED MODELS: INPUTS

- In computer-readable format

```
template entity Population { \\ d = density
    vars: d {aggregation:sum, unit: "kg/m$^3$"; range:<0,500>};
template process Growth (pop : Population) {
    consts:      gR {range:<0,5>};
    equations: td(pop.d) = gR *pop.d; }
template process Decay (pop : Population) {
    consts:      dR{range:<0,2>};
    equations: td(pop.d) = – dR*pop.d; }
template process Interaction (pop1 : Population, pop2 : Population){
    consts:      iR {range: <0,2>}, eF {range: <0,1>};}
template process UnsaturatedPP: Interaction {
    equations: td(pop.d) =iR * eF * pop1.d * pop2.d,
               td(pop.d) = – iR * pop1.d * pop2.d; }
template process SaturatedPP: Interaction {
    consts:      sR {range: <0,10>};
    equations: td(pop.d) =R * eF * pop1.d * pop2.d / (pop2.d + sR),
               td(pop.d) = – iR * pop1.d * pop2.d / (pop2.d + sR); }
```

a)

time	hare	fox
0	20	2
1	50	4
2	120	10
3	70	20
4	12	15
5	5	10
6	2	3
7	15	2
8	65	2

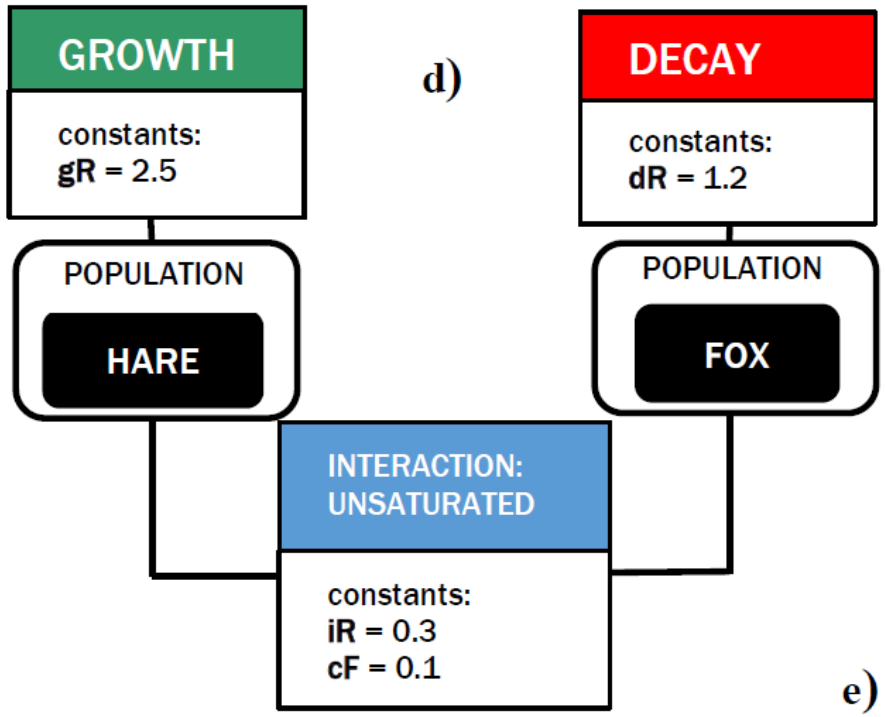
b)

```
entity hare : Population;
entity fox : Population;
process growth (hare): Growth;
process decay (fox): Decay;
process predator_prey (fox,hare): Population;
```

c)

LEARNING PROCESS-BASED MODELS: OUTPUTS

- In human & computer-readable format



d)

e)

d)

```
entity hare : Population; {
    vars: d {role endogenous; initial:20}; }
entity fox : Population {
    vars: d {role endogenous; initial:2}; }
process growth (hare): Growth
  { consts: gR=2.5;}
process decay (fox): Decay
  { consts: dR=1.2;}
process predator_prey (fox,hare): UnsaturatedPP
  { consts: eF = 0.1, iR = 0.3;}
```

e)

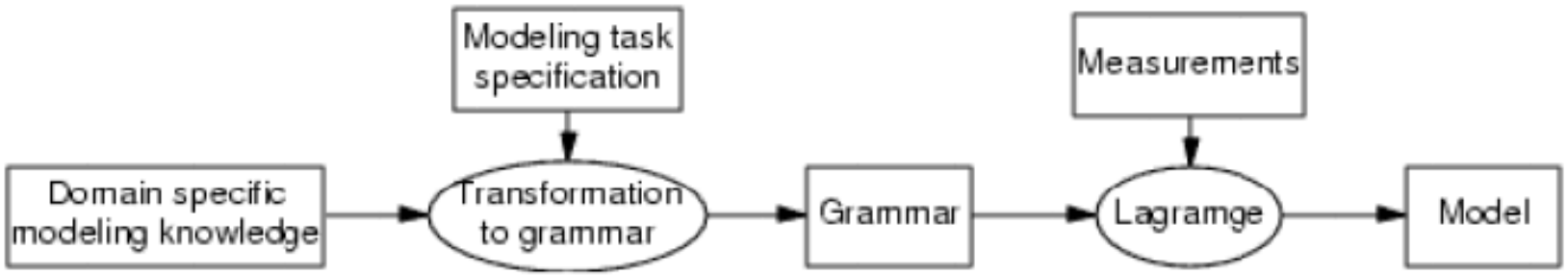
$$\frac{d}{dt} hare_d = 2.5 * hare_d - 0.3 - hare_d * fox_d$$

$$\frac{d}{dt} fox_d = 0.1 * 0.3 * hare_d * fox_d - 1.2 * fox_d$$

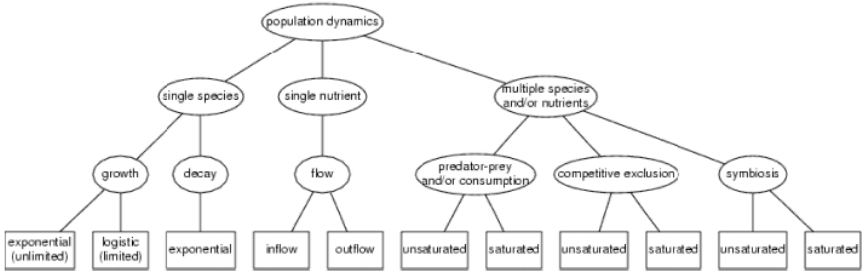
THE LAGRAMGE APPROACH TO IPM

- Take domain knowledge and task specification
- Generate a grammar from these

- Take the grammar and observed data and give these as input to LAGRAMGE



TRANSFORM PBM PROBLEM INTO FORM SUITABLE FOR LAGRAMGE




```
combining scheme Population_dynamics(Inorganic i)
  d/dt i = + Flow(i)
    - Σ_{food,i∈food} const(_,0,1,Inf) * Feeds_on(p, food)
```

```
combining scheme Population_dynamics(Population p)
  d/dt p = + Growth(p) - Decay(p)
    + Σ_food const(_,0,1,Inf) * Feeds_on(p, food)
    - Σ_{pred,food,p∈food} const(_,0,1,Inf) * Feeds_on(pred, food)
```

```
variable Population hare
variable Population lynx

process Growth(hare)
process Decay(lynx)
process Feeds_on(hare, lynx)
```

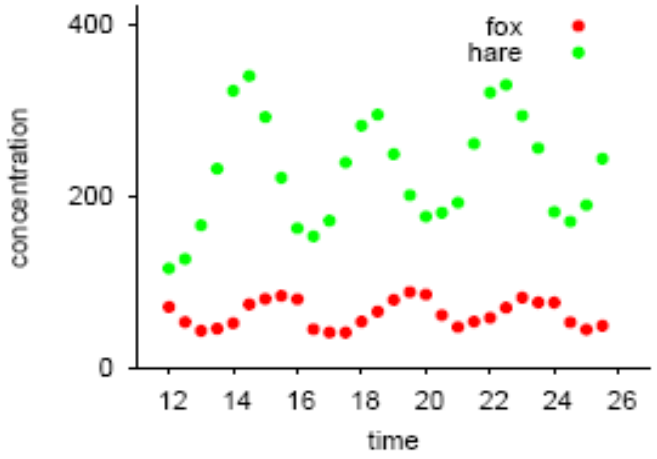
```
process class Growth(Population p)

process class Exponential_growth is Growth
  expression const(growth_rate,0,1,Inf) * p

process class Logistic_growth is Growth
  expression const(growth_rate,0,1,Inf) * p * (1 - p / const(capacity,0,1,Inf))
```

```
process class Feeds_on(Population p, set of Concentration cs)
  condition p ∉ cs
  expression p * Π_{c∈cs} Saturation(c)
```

```
general_lotka_volterra ->
    time_deriv(hare) = Growth(hare) - const[0:1:]  * Feeds_on(lynx,hare)
    time_deriv(lynx) = const[0:1:]  * Feeds_on(lynx,hare) - Decay(lynx)

Growth(hare) -> const[0:1:]  * hare
Growth(hare) -> const[0:1:]  * hare * (1 - hare / const[0:1:])

Decay(lynx) -> const[0:1:]  * lynx

Feeds_on(lynx,hare) -> lynx * Saturation(hare)

Saturation(hare) -> hare
Saturation(hare) -> hare / (hare + const[0:1:])
```

THE TASK OF GRAMMAR-BASED INDUCTIVE PROCESS MODELLING

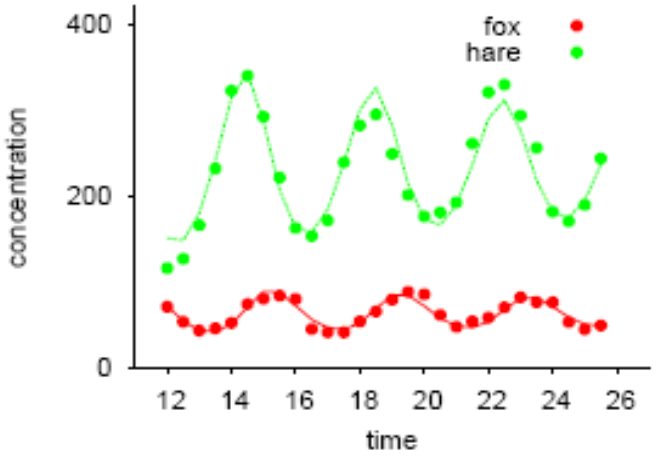
- Input: Observed behavior of dynamic system + Grammar



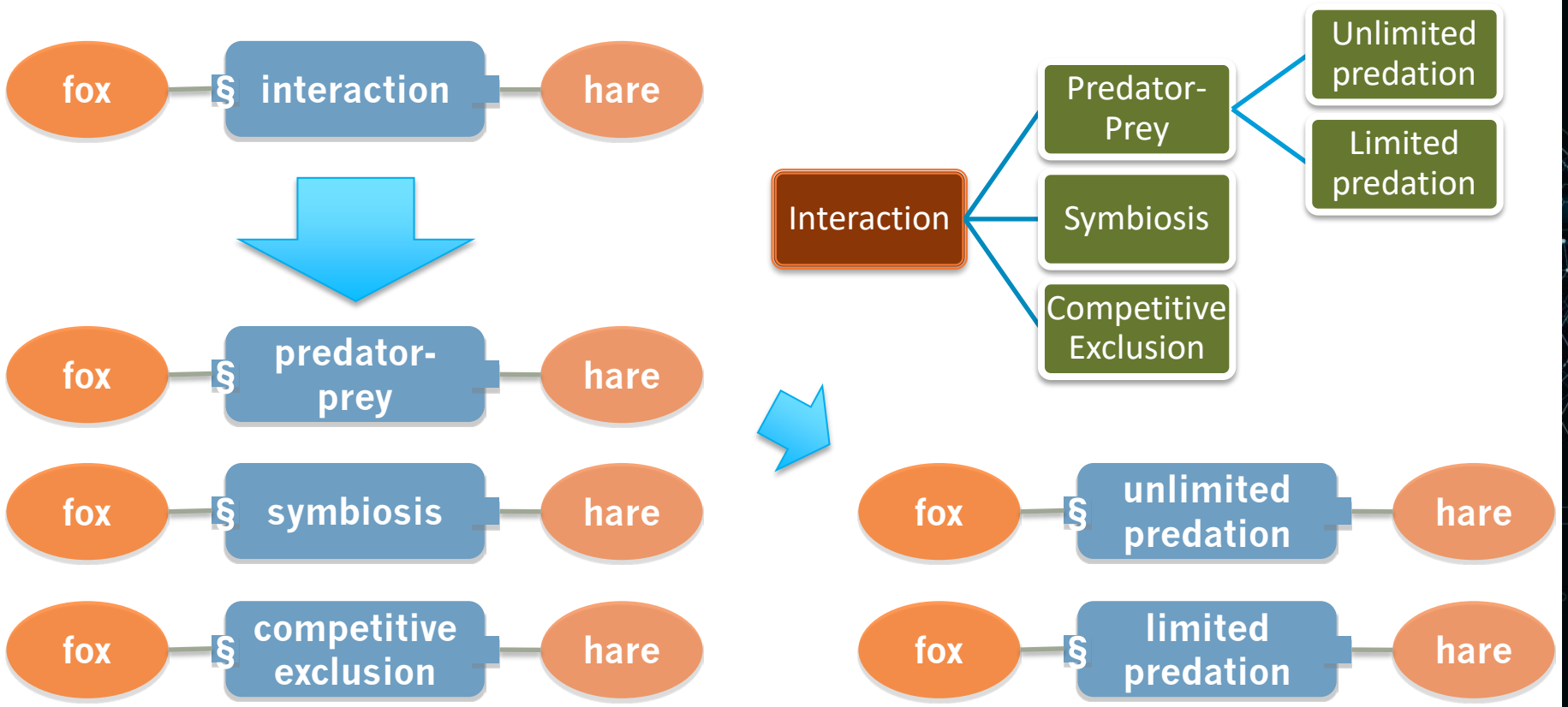
$$
\begin{array}{rcl}
PPModel & \rightarrow & PreyChange,\ PredatorChange \\
 & & \\
PreyChange & \rightarrow & PreyGrowth - Interaction \\
PredatorChange & \rightarrow & const * Interaction + PredatorLoss \\
 & & \\
PreyGrowth & \rightarrow & const * V_{Prey} \\
PreyGrowth & \rightarrow & const * V_{Prey} * (1 - V_{Prey}/const) \\
 & & \\
Interaction & \rightarrow & const * V_{Predator} * V_{Prey} \\
Interaction & \rightarrow & const * V_{Predator} * V_{Prey}/(V_{Prey} + const) \\
 & & \\
PredatorLoss & \rightarrow & -const * V_{Predator} \\
 & & \\
V_{Prey} & \rightarrow & hare \\
V_{Predator} & \rightarrow & fox
\end{array}
$$

- Output: Set of ordinary differential equations(ODEs)

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

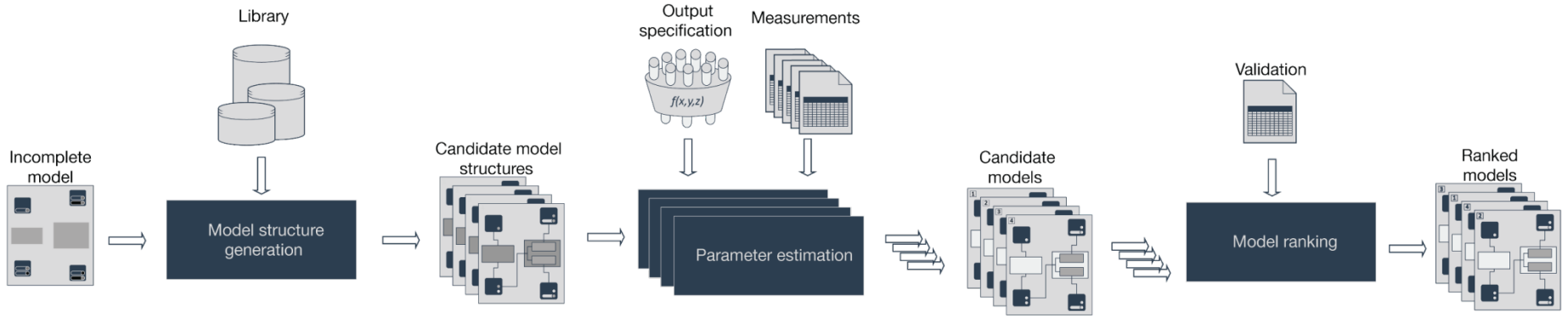


DIRECT SEARCH IN THE SPACE OF PBMS



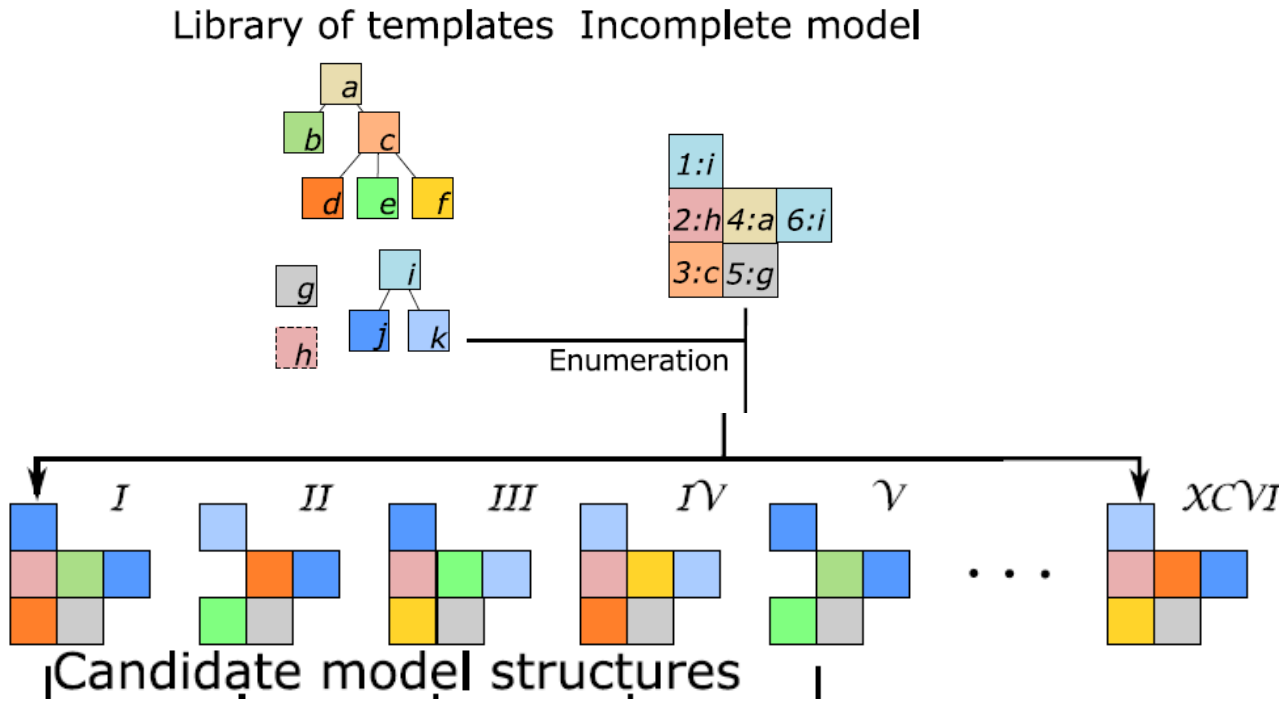
PROBMOT: SW PLATFORM FOR LEARNING PROCESS-BASED MODELS

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



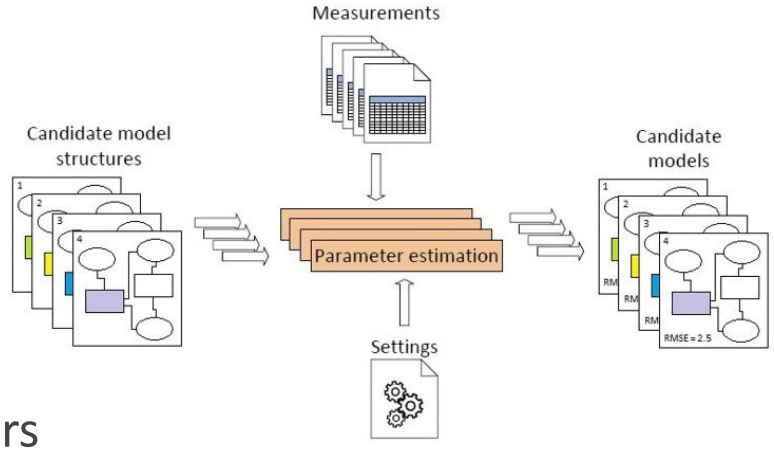
PROBMOT: GENERATING MODEL STRUCTURES

- Input:
  - Library of domain knowledge +
  - Incomplete model (task specification)
- Output: A set of model structures



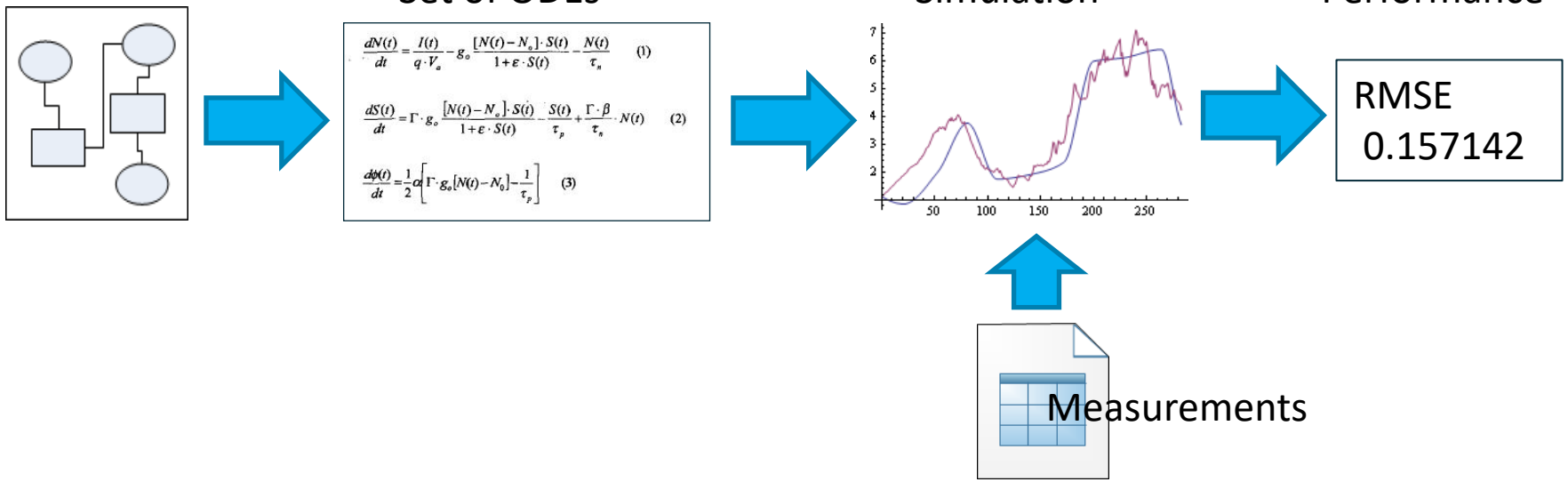
PROBMOT: PARAMETER ESTIMATION

- Input:
  - Model structures
  - Objective function(s), e.g., RMSE
  - Incl. observed data
- Output: Complete models, incl. parameters
- Parameter estimation approaches
  - Local derivative-based methods, e.g., gradient-descent
    - With rapid random restarts to avoid local optima
    - Minimize the sum of squared errors
  - Global (meta-heuristic approaches)
    - Differential evolution, Ant-colony optimization
    - Can use different cost functions!



EVALUATING INDIVIDUAL COMPLETE MODELS

- A complete model contains both a model structure and its parameter values (from instance processes)
- A complete model can be translated into a set of ODEs
- A set of ODEs, given measurements from exogenous (input) variables, yields a simulation
- Measurements of output variables compared to their simulated values yield a model quality estimate

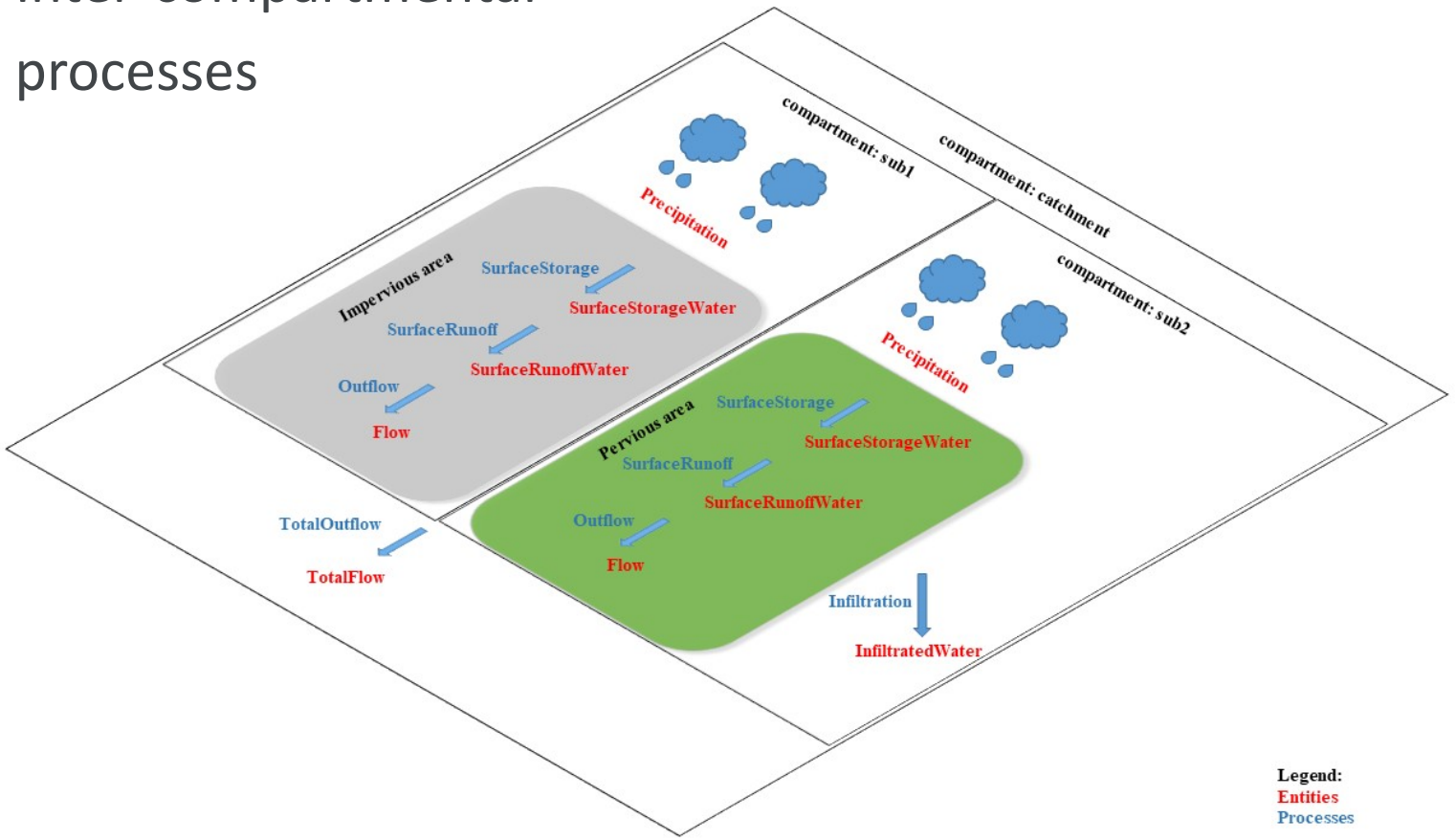


RECENT ADVANCES IN PBM METHODS

- Representing models & Knwldg: Compartmental models
- 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
- Process-based design: No measurements, just constraints
- Search model structures: Heuristic (evolutionary) search
- Learning ensembles of process-based models

COMPARTMENTAL MODELS

- Modular representation: Each compartment is a system/model with its own entities & processes
- Inter-compartmental processes



DIFFERENT OBJECTIVE FUNCTIONS IN LEARNING PB MODELS

- By default, SSE (sum-of-squared errors) used

$$\sum_i (Rab_{5,i} - \widehat{Rab}_{5,i})^2$$

- SSE can be used in gradient-based opt. approaches

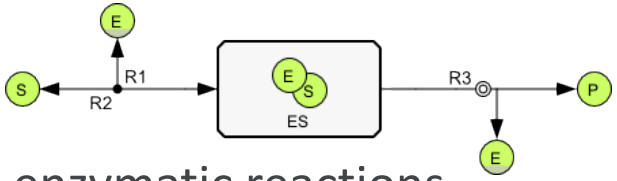
- In ProBMoT, other objective functions can be used
- E.g., avg. root relative error $E(m) = \frac{1}{2} \cdot \left(\sqrt{\frac{\sum_i (Rab_{5,i} - \widehat{Rab}_{5,i})^2}{\sum_i (Rab_{5,i} - \overline{Rab}_5)^2}} + \sqrt{\frac{\sum_i (Rab_{7,i} - \widehat{Rab}_{7,i})^2}{\sum_i (Rab_{7,i} - \overline{Rab}_7)^2}} \right)$

- or correlation between measured and simulated values
- Incl. multiple objectives, scalarized $ER(m) = \alpha \cdot E(m) + (1 - \alpha) \cdot R(m)$

- Or in a truly multi-objective fashion, considering Pareto fronts (and HVUPF)

STOCHASTIC MODEL OF ENZYMATIC RS

- Michaelis – Menten: 4 molecules (E-enzyme, S-substrate, P-product, ES-complex), 3 processes
- $E + S \underset{k1,k2}{\longleftrightarrow} ES \underset{k3}{\rightarrow} P + E$



- Process-based model of the enzymatic reactions

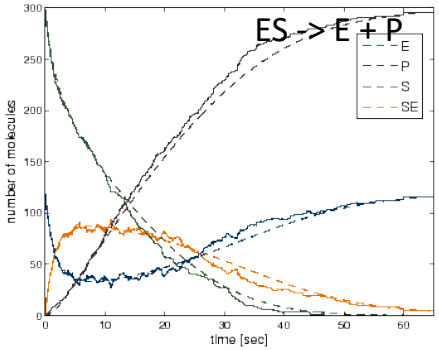
```
entity e : molecule {
    vars: conc { role: endogenous; initial: 100;};}
entity s : molecule {
    vars: conc { role: endogenous; initial: 100;};}
entity es : molecule {
    vars: conc { role: endogenous; initial: 0;};}
entity p : molecule {
    vars: conc { role: endogenous; initial: 0;};}
```

```
process ESBind(e,s,es):Binding{
    consts: k1;}
process ESSeparate(e,s,es):Separation{
    consts:  k2;}
process ProductFormation(e,p,es):Separation{
    consts: k3;}
```

E + S -> ES

ES -> E + S

ES -> E + P



- Entities -> distinct molecules/species
- Processes -> interactions via reaction channels (stochastic kinetic rates)

EQUATION- & REACTION-BASED PROCESS TEMPLATES

```
library Michaelis;
template entity molecule{
      vars: conc {range:<0,1000>};}
template process Binding(e : molecule,
s : molecule, es : molecule){
      consts: k1{range: <0, 10>};
      equations:
            td(es.conc) =
k1*e.conc*s.conc,
            td(e.conc) = -
k1*e.conc*s.conc,
            td(s.conc) = -
k1*e.conc*s.conc;}
template process Separation(e :
molecule, s : molecule, es : molecule){
      consts: k2{range: <0, 10>};
      equations:
            td(es.conc) = -k2*es.conc,
            td(e.conc) = k2*es.conc,
            td(s.conc) = k2*es.conc;}
```

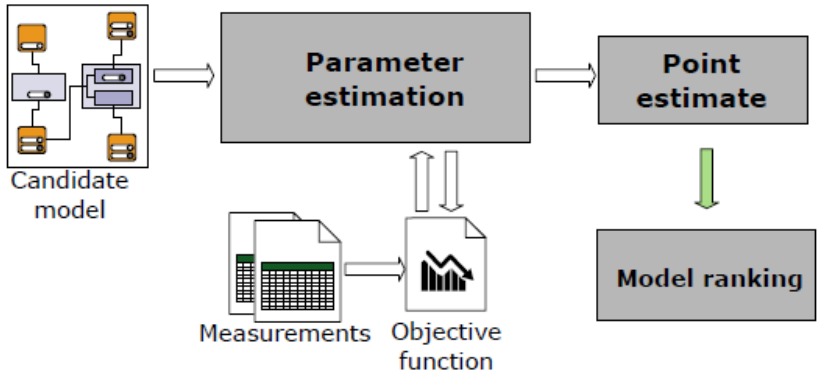
E + S -> ES

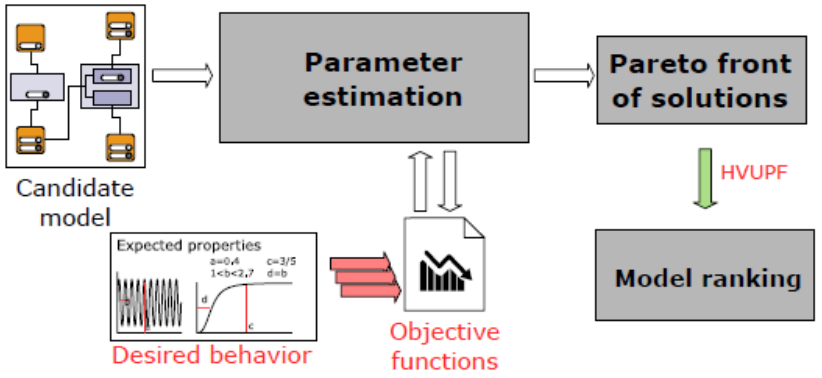
ES -> E + S
ES -> E + P

```
library Michaelis;
template entity molecule{
      vars: conc {range:<0,1000>};}
template process Binding(e : molecule, s :
molecule, es : molecule){
      consts: k1 {range: <0, 10>};
      equations: e.conc + s.conc -> es.conc [k1];
}
```

```
template process Separation(e : molecule, s :
molecule, es : molecule){
      consts: k2 {range: <0, 10>};
      equations: es.conc -> e.conc + s.conc [k2];
}
```

AUTOMATED MODELING: IDENTIFICATION VS. DESIGN





Process-Based Modeling

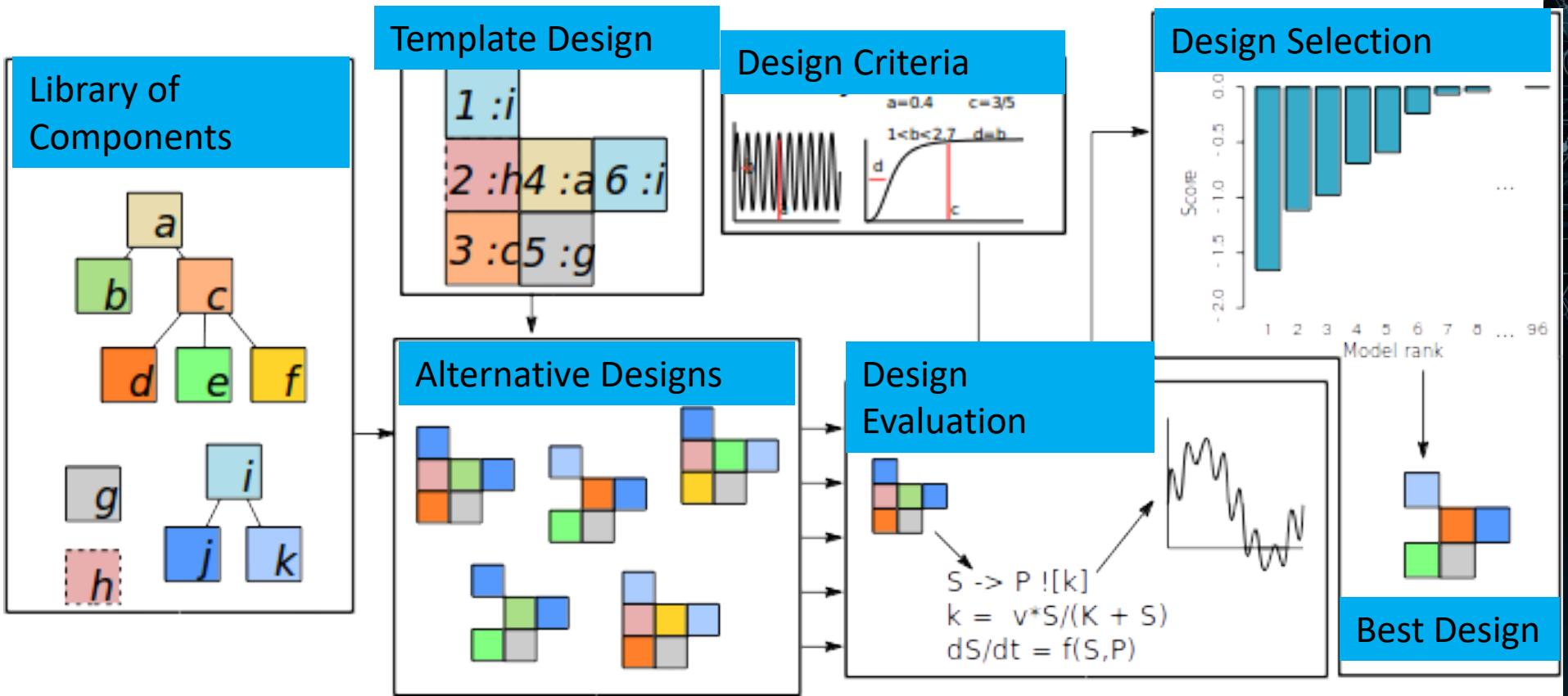
- Data-driven (measurements of target system variables).
- Typically single objective function (fit to measurements).
- Single objective optimization — point estimate.

Process-Based Design

- No measurements available. Driven by desired behavior.
- May require multiple, potentially conflicting objective functions.
- Multi-objective optimization — Pareto front of solutions.

ProBMoT FOR DESIGN: THE WORKFLOW

- Same formalism for representing domain knowledge/ library of components and incomplete model (template design) as in process-based identification

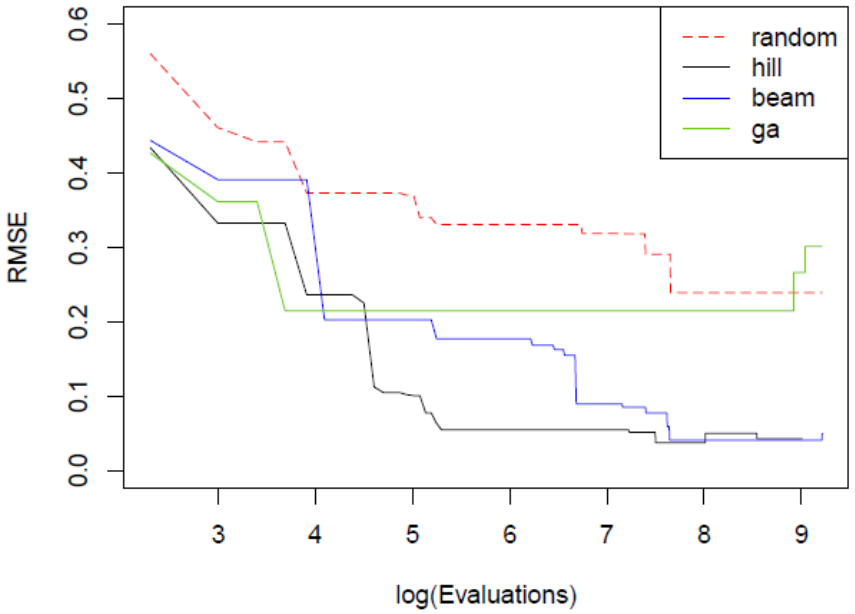


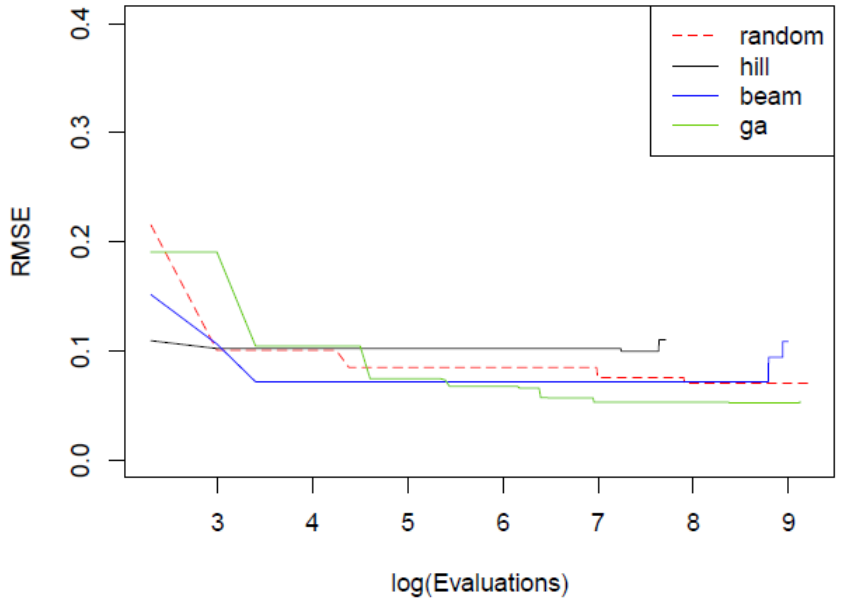
HEURISTIC (INSTEAD OF EXHAUSTIVE) SEARCH IN LEARNING PB MODELS

Recent comparison of greedy, beam and evolutionary srch

On 8 problems of automated modeling of dynamic. syst.

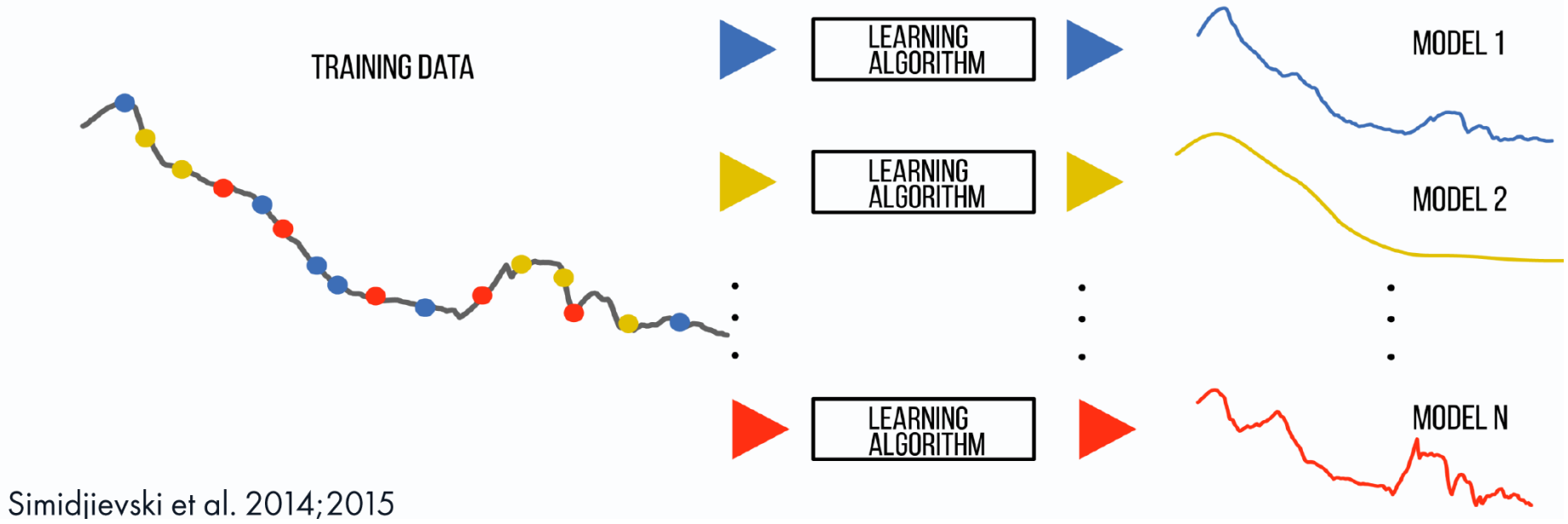
Solutions very close to optimal obtained by greedy search





LEARNING ENSEMBLES OF ODE MODELS

- Sampling data points when learning about dynamical systems
- Takes into account the temporal ordering of the data points

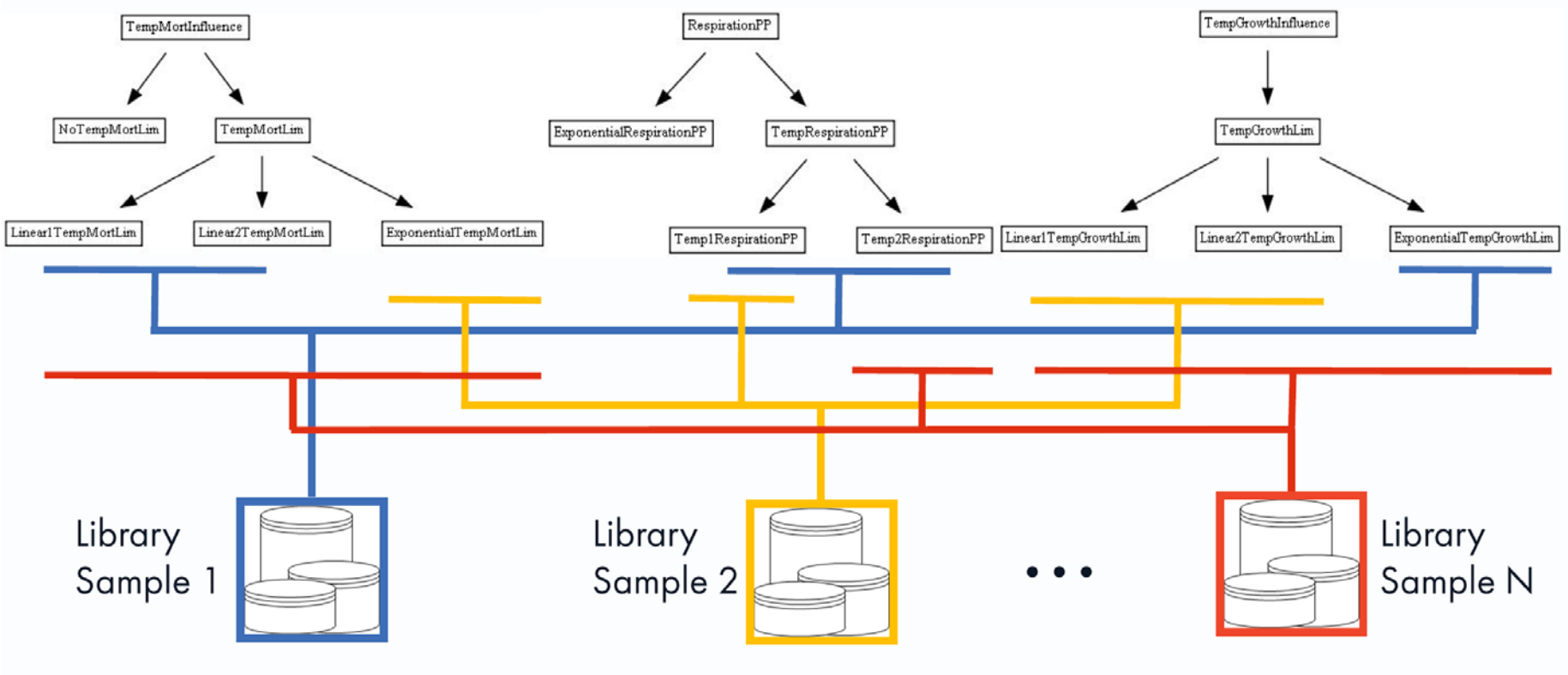


Simidjievski et al. 2014;2015

- Learn a set of ODE models, one for each sampled subtrajectory

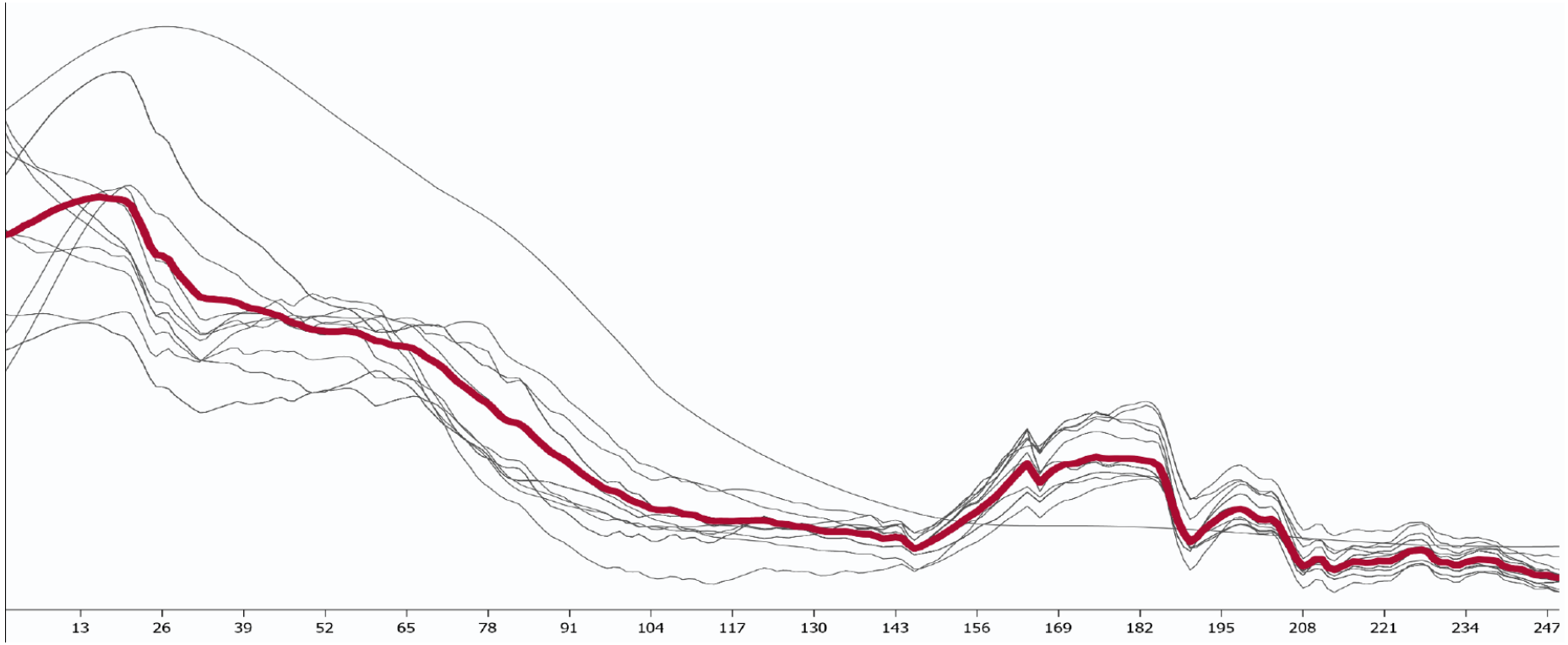
RANDOM LIBRARY SUBSETS

- Instead of randomly selecting subsets of the data
- We randomly select subsets of the library of domain knowledge



COMBINING PREDICTED TRAJECTORIES

- Use standard approaches for aggregating numerical values
  - Average, Weighted average
  - Median, Weighted median



EQUATION DISCOVERY WITH PCFGS (BRENCE ET AL.)

- Basic idea: Instead of CFGs, use Probabilistic CFGs

- Probabilistic CFGs are much like CFGs
- They consist, in essence, of production rules

- In PCFGs, each production rule has a probability
- For each nonterminal symbol A, the probabilities of all production rules with A on the LHS sum up to 1

$$\sum_{(A \to \alpha) \in \mathcal{R}} P(A \to \alpha) = 1.$$

PROBABILISTIC CFGS FOR ED: EXAMPLE

- A PCFG for arithmetic expressions
- Probabilities for each non-terminal symbol sum up to 1

1. $\mathcal{N} = \{E, F, T, V\}$
2. $\mathcal{T} = \{x, y, +, -, *, /, (,)\}$
3. $\mathcal{R} =$

$E \rightarrow E + F \mid E - F \mid F$

$F \rightarrow F * T \mid F/T \mid T$

$T \rightarrow (E) \mid V$

$V \rightarrow x \mid y.$

4. $S = E.$

1. $\mathcal{N} = \{E, F, T, R, V\}$
2. $\mathcal{T} = \{c, x, y, +, -, *, /, (,), \sin, \cos, \sqrt{}, \exp\}$
3. $\mathcal{R} =$

$E \rightarrow E + F\ [0.2] \mid E - F\ [0.2] \mid F\ [0.6]$

$F \rightarrow F * T\ [0.2] \mid F/T\ [0.2] \mid T\ [0.6]$

$T \rightarrow R\ [0.2] \mid V\ [0.4] \mid c\ [0.4]$

$R \rightarrow (E)\ [0.6] \mid \sin(E)\ [0.1] \mid \cos(E)\ [0.1]$

$\mid \sqrt{E}\ [0.1] \mid \exp(E)\ [0.1]$

$V \rightarrow x\ [0.5] \mid y\ [0.5]$

4. $S = E.$

SAMPLING EXPRESSIONS FROM PCFGS & MONTE-CARLO ALGORITHM FOR ED

Algorithm 1 Sample a sentence from a probabilistic context-free grammar.

Require: Probabilistic grammar $G = (\mathcal{N}, \mathcal{T}, \mathcal{R}, S)$, non-terminal $A \in \mathcal{N}$

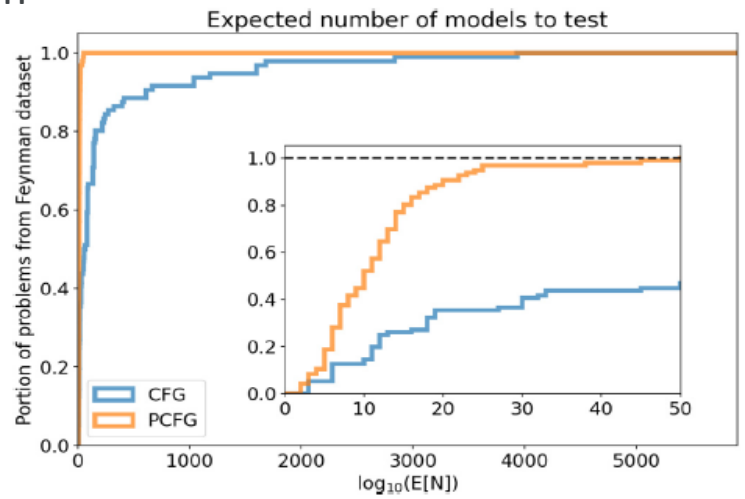
Ensure: Sentence s corresponding to a randomly sampled parse tree ψ from G with root node A, probability p of ψ

procedure GENERATE_SAMPLE(G, A)
  $(s, p) = ([\ ], 1)$
  Choose a random rule $(A \rightarrow \alpha) \in \mathcal{R} : \alpha = A_1 A_2 \ldots A_k$, $A_i \in \mathcal{N} \cup \mathcal{T}$
  for $i = 1, i \leq k$ **do**
    if $A_i \in \mathcal{T}$ **then**
      $s = s$.append(A_i)
    else
      (s_i, p_i) = GENERATE_SAMPLE(G, A_i)
      $s = s$.append(s_i)
      $p = p \cdot p_i$
  return (s, p)

procedure MC_GBED(G, N, D, e)
  $eqns = [\ ]$
  for $i = 1, i \leq N$ **do**
    (e, p) = GENERATE_SAMPLE(G, S)
    e_c = canonical_form(e)
    eqn = fit_parameters(e_c, v, D)
    $error = ReRMSE(eqn, D)$
    $eqns$.append(eqn, p, $error$)
  return $eqns$.sort(key = $error$, order = increasing)

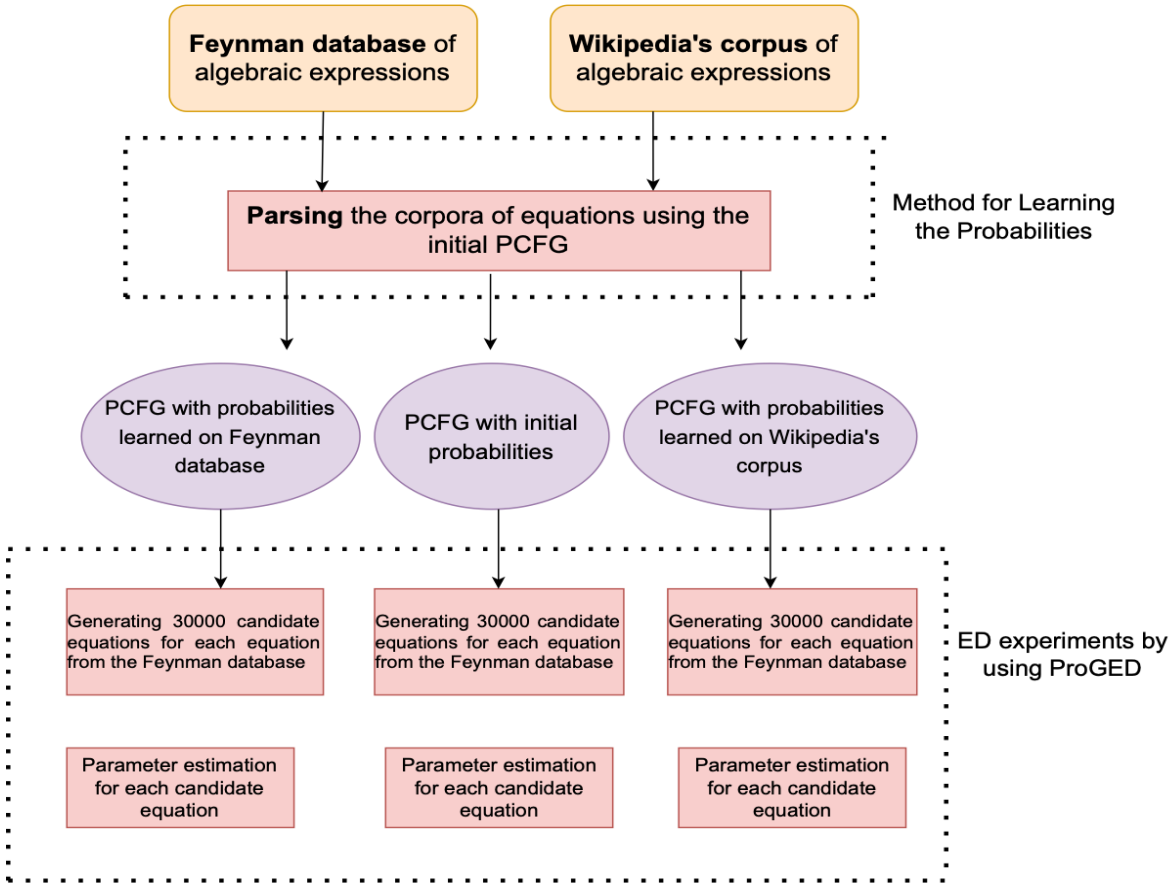
THE UTILITY OF GRAMMARS IN ED

- CF Grammars ca be used to represent different kinds of domain knowledge
  - Basic building blocks of systems/ models in the domain
  - Existing models in the domain (that need to be revised)
  - Incomplete models, than need to be completed
- PCFGs can be used to express parsimony preferences
- PCFGs have been recently used to represent dimensional constraints



LEARNING GRAMMARS FROM CORPORA OF EQUATIONS (CHAUSHEVSKA EL AL.)

- For a start, we can learn the probabilities in a PCFG



LEARNING GRAMMARS FROM CORPORA OF EQUATIONS (CHAUSHEVSKA EL AL.)

Initial grammar

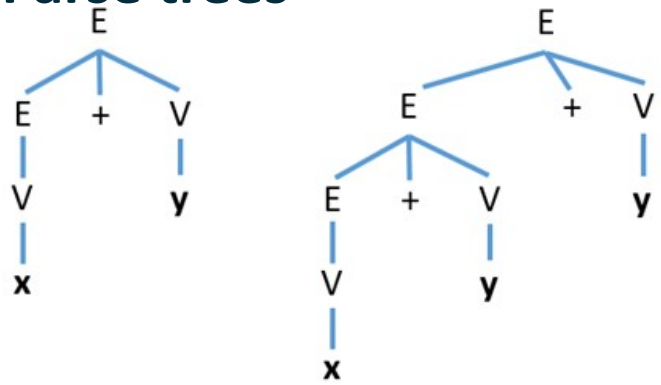
$E \rightarrow E + F$ [0.2] | $E - F$ [0.2] | F [0.6]
$F \rightarrow F * T$ [0.2] | F / T [0.2] | T [0.6]
$T \rightarrow R$ [0.2] | V [0.4] | c [0.4]
$R \rightarrow (E)$ [0.6] | sin(E) [0.1] | cos(E) [0.1]
$\rightarrow \sqrt{E}$ [0.1] | exp(E) [0.1]

Parse

Corpus of equations

n_rho*mom*tanh(mom*B/(kb*T))
mom*H/(kb*T)+(mom*alpha)/(epsilon*c**2*kb*T)*M
mom*(1+chi)*B
Y*A*x/d
Y/(2*(1+sigma))
1/(exp((h/(2*pi))*omega/(kb*T))-1)
(h/(2*pi))*omega/(exp((h/(2*pi))*omeg

Parse trees



Updated grammar

$E \rightarrow E + F$ [0.10] | $E - F$ [0.15] | F [0.75]
$F \rightarrow F * T$ [0.36] | F / T [0.24] | T [0.40]
$T \rightarrow R$ [0.15] | V [0.72] | c [0.13]
$R \rightarrow (E)$ [0.56] | sin(E) [0.12] | cos(E) [0.09]
$\rightarrow \sqrt{E}$ [0.14] | exp(E) [0.09]

Production frequencies

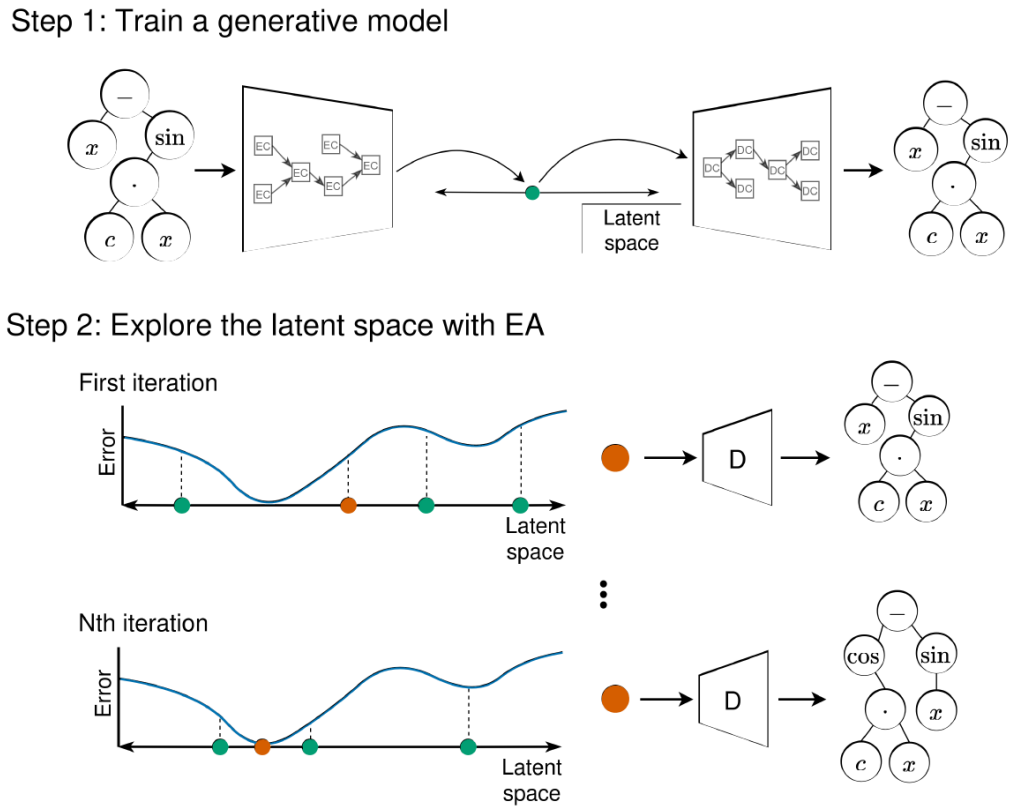
USING LEARNED GRAMMARS IN EQUATION DISCOVERY

- Probabilities in the universal grammar used earlier now learned from the 100 algebraic equations from R. Feynman‘s textbooks
- Everything else kept the same
- **Successfully reconstructed equations: 43**
- As opposed to 36 when using probabilities specified manually

```
exp(-theta**2/2)/sqrt(2*pi)
exp(-(theta/sigma)**2/2)/(sqrt(2*pi)*sigma)
exp(-((theta-theta1)/sigma)**2/2)/(sqrt(2*pi)*sigma)
sqrt((x2-x1)**2+(y2-y1)**2)
G*m1*m2/((x2-x1)**2+(y2-y1)**2+(z2-z1)**2)
m_0/sqrt(1-v**2/c**2)
x1*y1+x2*y2+x3*y3
mu*Nn
```

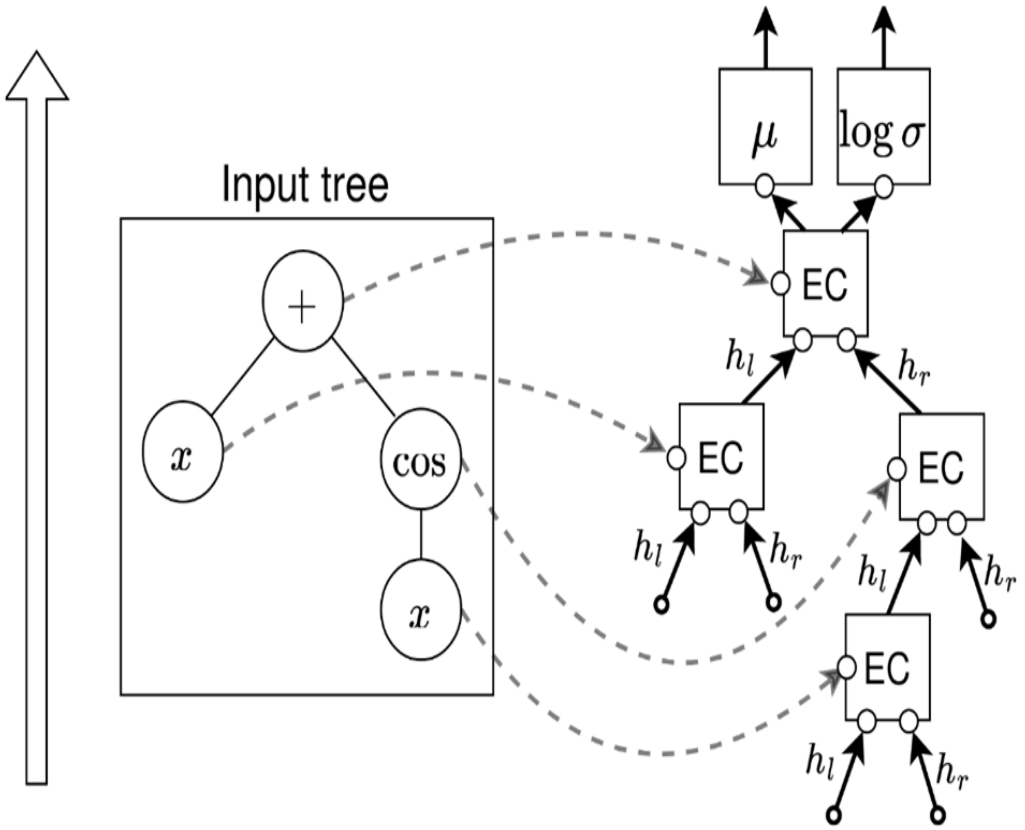
USING GRAMMARS TO TRAIN OTHER GENERATIVE MODELS (MEZNAR ET AL.)

- Generate many expressions from a grammar
- Train a hierarchical variational autoencoder
- Explore the HVAE latent space with Evolutionary Algorithms



RECURSIVE ENCODING AND DECODING OF EXPRESSION TREES

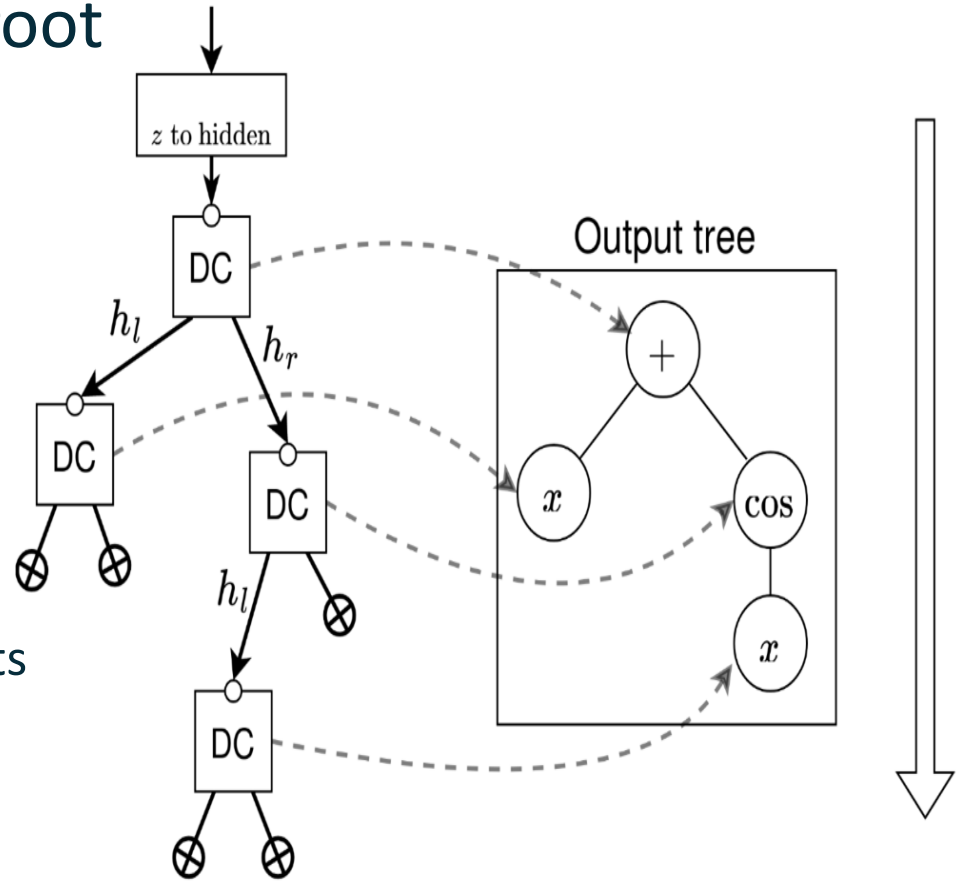
- Recursive encoding from leaf to root nodes
- Use zero-vectors for missing child nodes
- Encoding cell: modification of the GRU cell (GRU21)



(a) Encoding

RECURSIVE ENCODING AND DECODING OF EXPRESSION TREES

- Recursive decoding from root to leaf nodes
- Number of children depends on the symbol in the generated node:



(b) Decoding

- Operator: generate both descendants
- Function: generate left descendant
- Variable, constant: finish decoding this branch

LINEAR INTERPOLATION

- Syntactically similar expressions are close together in latent space,
- We encode two expressions and generate a sequence of points with the formula:

$$z_\alpha = (1 - \alpha) \cdot z_A + \alpha \cdot z_B$$

- We decode expressions and smoothly transition between expressions

Expression A	$x - \sin(c \cdot x)$
$\alpha = 0$	$x - \sin(c \cdot x)$
$\alpha = 0.25$	$x \cdot \sin c - \sin(c \cdot x)$
$\alpha = 0.5$	$c \cdot \sin x + x$
$\alpha = 0.75$	$c \cdot \cos \frac{x}{c} + c$
$\alpha = 1$	$c \cdot x \cdot \cos \frac{x}{c} + c$
Expression B	$c \cdot x \cdot \cos \frac{x}{c} + c$

EVOLUTIONARY ALGORITHM

- Crossover: Create an offspring as a convex combination of two parents,

$$z_O = (1 - a) \cdot z_A + a \cdot z_B \quad a \in [0, 1]$$

- Mutation: Sample a point from the Gaussian distribution with bias toward the vicinity of the individual,

$$\mathcal{N}(a \cdot \mu_z, a \cdot \sigma_z + (1 - a) \cdot I)$$

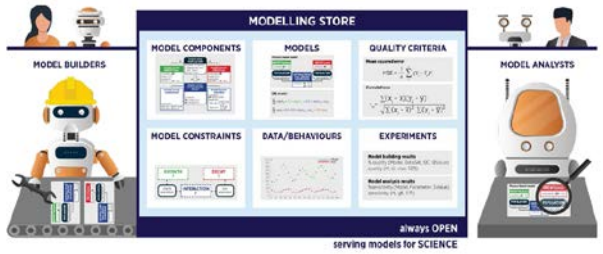
- EDHiE (Equation Discovery with Hierarchical variational autoEncoders)

COMPARISON ON NGUYEN BENCHMARK

ID	Expression
NG-1	$x^3 + x^2 + x$
NG-2	$x^4 + x^3 + x^2 + x$
NG-3	$x^5 + x^4 + x^3 + x^2 + x$
NG-4	$x^6 + x^5 + x^4 + x^3 + x^2 + x$
NG-5	$\sin x^2 \cdot \cos x - 1$
NG-6	$\sin x + \sin(x + x^2)$
NG-7	$\ln(x + 1) + \ln(x^2 + 1)$
NG-8	$\sqrt{x}$
NG-9	$\sin x + \sin y^2$
NG-10	$2 \sin x \cdot \cos y$

	EDHiE (our)			DSO (Petersen et al., 2021)			PySR (Cranmer, 2023)	
Name	Successful	Mean R^2	Evaluated	Successful	Mean R^2	Evaluated	Successful	Mean R^2
NG-1	**10**	**1.00 (± 0.00)**	**573 (± 261)**	**10**	**1.00 (± 0.00)**	4565 (± 327)	**10**	1.00 (± 0.00)
NG-2	**10**	**1.00 (± 0.00)**	**5803 (± 4148)**	**10**	**1.00 (± 0.00)**	12,206 (± 9186)	**10**	1.00 (± 0.00)
NG-3	6	1.00 (± 0.01)	20,931 (± 4858)	**10**	**1.00 (± 0.00)**	**8053 (± 3766)**	2	1.00 (± 0.01)
NG-4	3	1.00 (± 0.01)	**21,346 (± 4479)**	**8**	**1.00 (± 0.01)**	32,946 (± 15,613)	0	0.99 (± 0.01)
NG-5	**3**	**0.32 (± 0.45)**	**20,615 (± 8394)**	0	0.00 (± 0.00)	NA	0	0.16 (± 0.15)
NG-6	**8**	**0.88 (± 0.14)**	**12,772 (± 7923)**	1	0.59 (± 0.15)	49,599 (± 0)	4	0.86 (± 0.13)
NG-7	8	1.00 (± 0.01)	**19,203 (± 3595)**	**10**	**1.00 (± 0.00)**	22,579 (± 10,264)	7	0.99 (± 0.01)
NG-8	**10**	**1.00 (± 0.00)**	**405 (± 174)**	**10**	**1.00 (± 0.00)**	5521 (± 1779)	**10**	**1.00 (± 0.00)**
NG-9	8	0.95 (± 0.15)	**7041 (± 3933)**	2	0.60 (± 0.20)	39,786 (± 28,197)	**10**	**1.00 (± 0.00)**
NG-10	**1**	0.70 (± 0.17)	**31863 (± 6970)**	0	0.56 (± 0.10)	NA	**1**	**0.80 (± 0.16)**
Total/mean	**66**	**0.89 (± 0.21)**		61	0.78 (± 0.31)		54	0.88 (± 0.26)

THE VISION:
AUTOMATED SCIENTIFIC MODELING



- AI/Machine learning approaches capable of functioning within the scientific knowledge ecosystem and contributing to the pool of scientific knowledge
  - Using both observations and existing knowledge/models
  - Producing new knowledge (models) that can be used further (by both humans and machines)
- To this end, we need to
  - Integrate knowledge-driven and data-driven modeling
  - Data and knowledge need both to be first-class citizens, so that one can store, query/find, retrieve and reuse them
  - Representations for models and knowledge should be close to those used by humans in scientific modeling