

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

MODELS AND MODELLING

What are models?

- A model is a simplified representation of reality (an observed or studied system)
- Captures some key features of the system, not all of them

Models are like (geographical) maps

- Different maps for different purposes; focus on different objects; different scales
- Never contain all the details

Two basic kinds of models

- Physical models — e.g., a car or plane in a wind tunnel
- Mathematical models — equations, computer programs

Modelling is the construction (and use) of models

Model building/synthesis and model analysis



WHY DO WE BUILD MODELS?

Models are an omnipresent analytical tool,
because of the following properties

- **Data integration:** Models integrate potentially vast collections of observations/ measurements into a single entity
- **Prediction:** Models allow for simulation and prediction of the future behavior of the studied system under varying conditions
- **Explanation:** Models can provide explanations, i.e., reveal the structure of processes and phenomena that govern the behavior of the observed system



HOW DO WE BUILD MODELS?

Modelling from first principles vs. empirical modelling

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

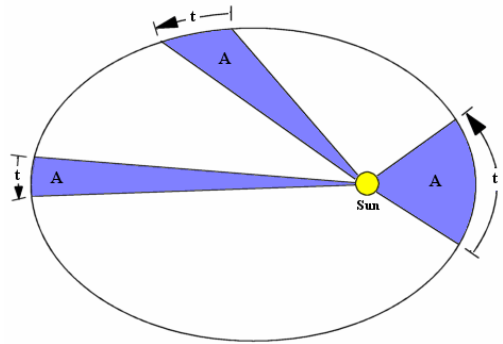
Data-driven (empirical) modeling focusses on data:

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

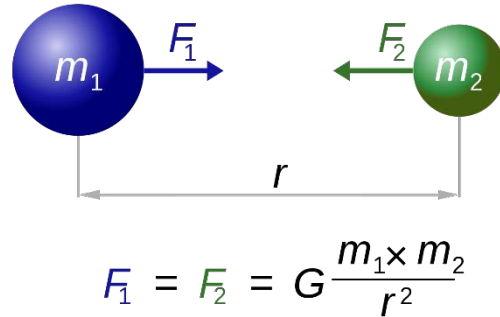


MODELS IN SCIENCE: REPRESENTATION OF SCIENTIFIC KNOWLEDGE

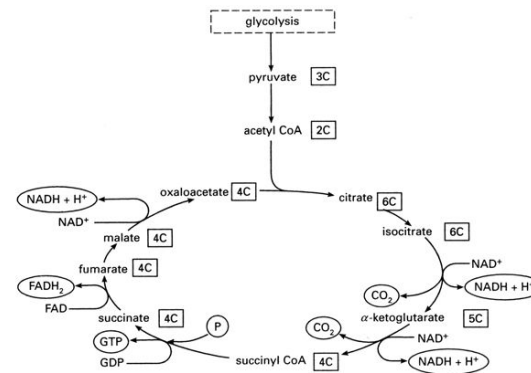
Scientific knowledge is typically represented in scientific formalisms (e.g., equations, pathways), introduced & routinely used by scientists



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

Scientific modelling is related to scientific discovery



SCIENTIFIC DISCOVERY AS PROBLEM SOLVING

Many philosophers of science have held mystical views of scientific 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 SD



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., $F = m \times a$ (Newton's second law)
- For a specific object of mass 2.3 kg, we would have the *model* $F = 2.3 \times a$



COMPUTATIONAL SCIENTIFIC DISCOVERY: FINDING EQUATIONS 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

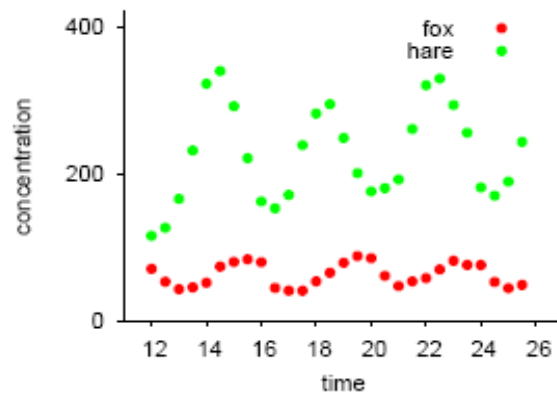
Moon	d	p	d/p	d^2/p	d^3/p^2
A	5.67	1.77	3.20	18.15	58.15
B	8.67	3.57	2.43	21.04	51.06
C	14.00	7.16	1.96	27.40	53.16
D	24.67	16.69	1.48	36.46	53.89

- Proceeds from observed variables to constant theoretical term



AUTOMATED MODELLING OF DYNAMIC SYSTEMS: FINDING ODEs IN DATA

- Input: Observed behavior of dynamic system

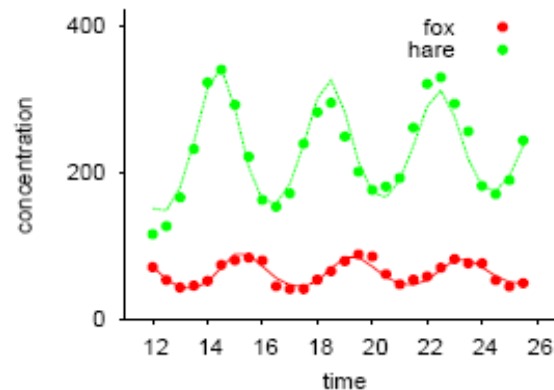


Time	System variables			
	v_1	v_2	$\dots$	v_n
t_0	$v_{1,0}$	$v_{2,0}$	$\dots$	$v_{n,0}$
t_1	$v_{1,1}$	$v_{2,1}$	$\dots$	$v_{n,1}$
$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$
t_m	$v_{1,m}$	$v_{2,m}$	$\dots$	$v_{n,m}$

- Output: System of ODEs

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

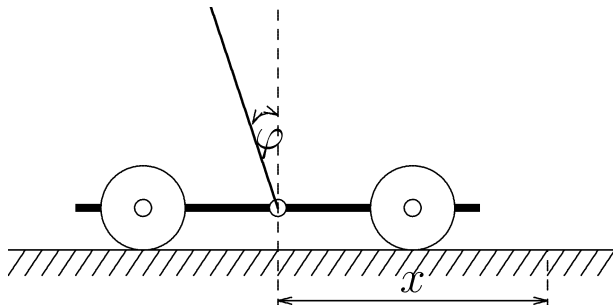
$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$



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



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

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

$$x_0 = 0, \varphi_0 = 3\pi/16, \dot{x}_0 = 0, \dot{\varphi}_0 = 0$$

$$M = 1, m = 0.1, l = 1, F = 7.5$$



FINDING POLYNOMIAL ODEs: CIPER

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

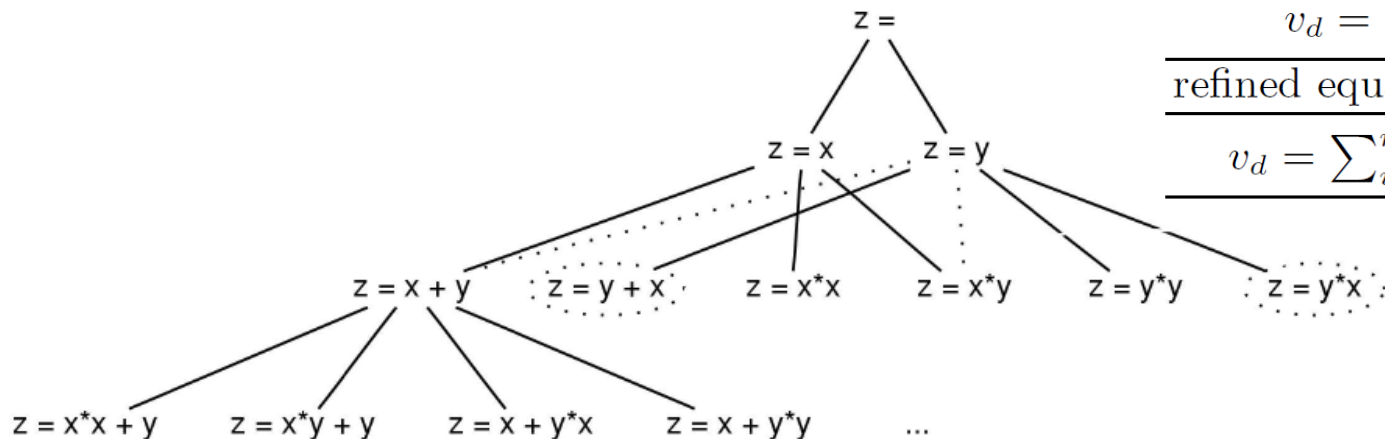
Input: training data D , dependent variable vd (derivative), number of equations b

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

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

Heuristic (beam) search through the space of eqns

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



original (current) equation

$$v_d = \sum_{i=1}^r \text{const}_i \cdot T_i$$

refined equations that increase r (one for each $v \in V \setminus v_d$)

$$v_d = \sum_{i=1}^r \text{const}_i \cdot T_i + \text{const}_{r+1} * v, \text{ where } \forall i : v \neq T_i$$

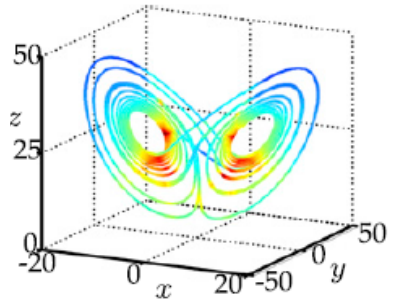
refined equations that increase d (one for each T_j and $v \in V \setminus v_d$)

$$v_d = \sum_{i=1, i \neq j}^r \text{const}_i \cdot T_i + T_j * v, \text{ where } \forall i \neq j : T_j * v \neq T_i$$

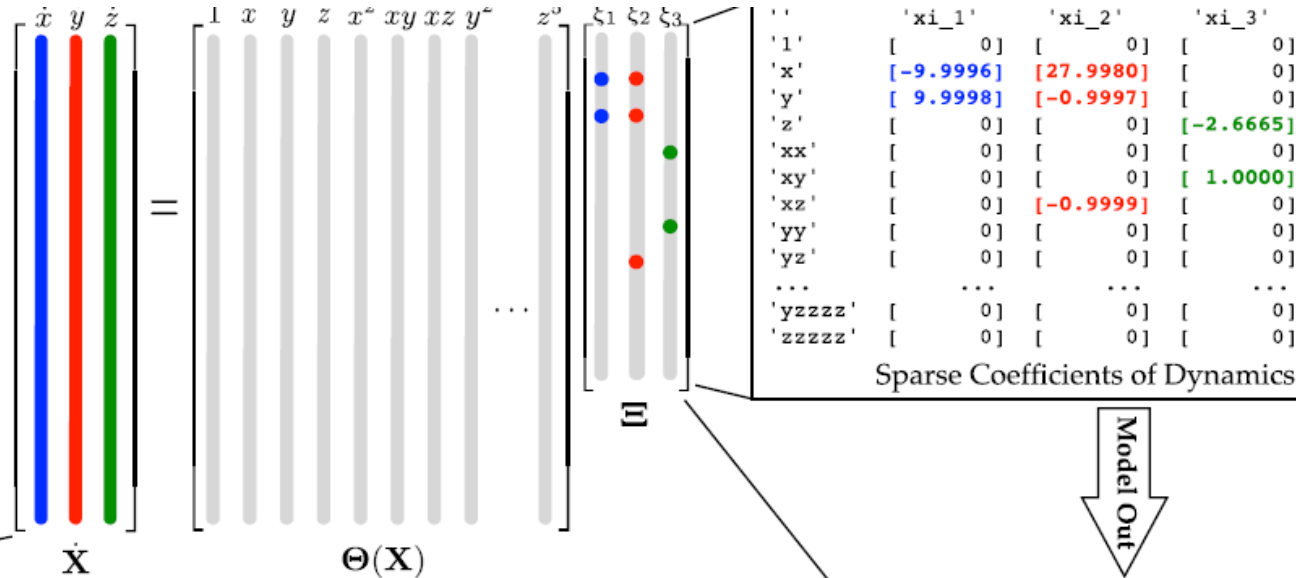
SINDY: POLYNOMIAL ODEs WITH SPARSE LR (BRUNTON, PROCTOR & KUTZ 2016)

I. True Lorenz System

$$\begin{aligned}\dot{x} &= \sigma(y - x) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z.\end{aligned}$$



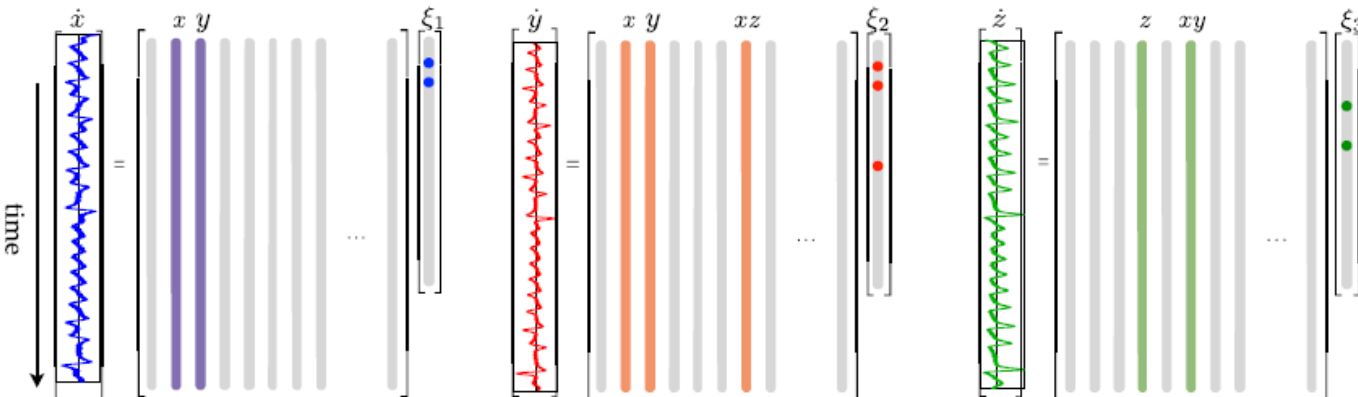
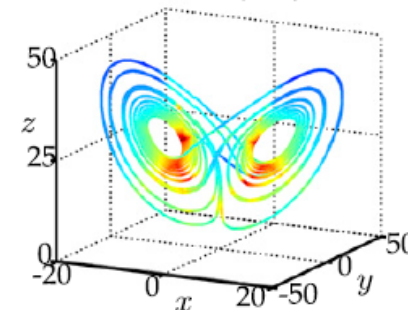
Data In



Model Out

III. Identified System

$$\begin{aligned}\dot{x} &= \Theta(\mathbf{x}^T)\xi_1 \\ \dot{y} &= \Theta(\mathbf{x}^T)\xi_2 \\ \dot{z} &= \Theta(\mathbf{x}^T)\xi_3\end{aligned}$$



II. Sparse Regression to Solve for Active Terms in the Dynamics

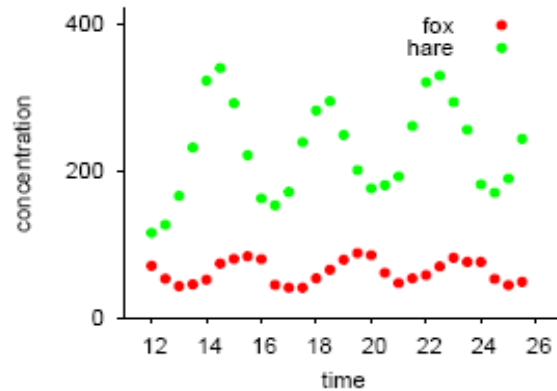
INTEGRATING DATA- AND KNOWLEDGE-DRIVEN MODELLING

- 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



$$PPModel \rightarrow PreyChange, PredatorChange$$

$$PreyChange \rightarrow PreyGrowth - Interaction$$

$$PredatorChange \rightarrow const * Interaction + PredatorLoss$$

$$PreyGrowth \rightarrow const * V_{Prey}$$

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

$$Interaction \rightarrow const * V_{Predator} * V_{Prey}$$

$$Interaction \rightarrow const * V_{Predator} * V_{Prey} / (V_{Prey} + const)$$

$$PredatorLoss \rightarrow -const * V_{Predator}$$

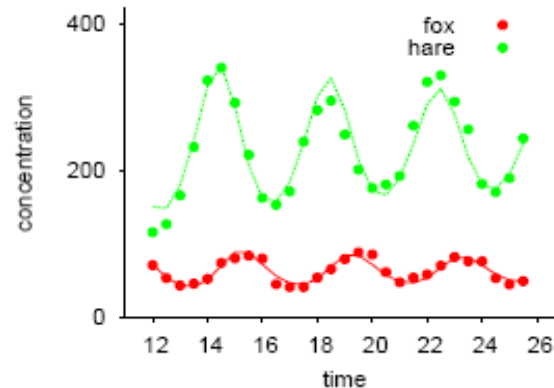
$$V_{Prey} \rightarrow hare$$

$$V_{Predator} \rightarrow fox$$

- Output: System of ODEs

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

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



EXAMPLE CFGS FOR EQUATIONS

- Universal grammar (arithmetic expressions); Polynomials

$$\begin{array}{ll} E \rightarrow E + F \mid E - F \mid F & E \rightarrow \text{const} \mid \text{const} \cdot F \mid E + \text{const} \cdot F \\ F \rightarrow F * T \mid F / T \mid T & F \rightarrow v \mid v \cdot F \\ T \rightarrow \text{const} \mid v \mid (E) \end{array}$$

- Context free grammar used for environmental dynamic systems:

$$\begin{array}{l} E \rightarrow \text{const} \mid \text{const} \cdot F \mid E + \text{const} \cdot F \\ F \rightarrow v \mid Y \mid v \cdot Y \\ Y \rightarrow \text{monod}(\text{const}, v) \end{array}$$

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

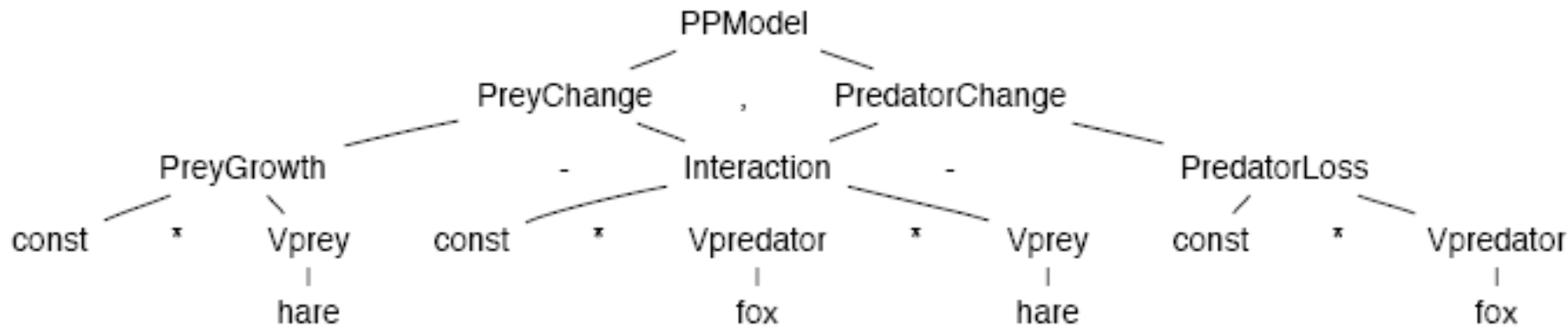
- User defined function `monod`:

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

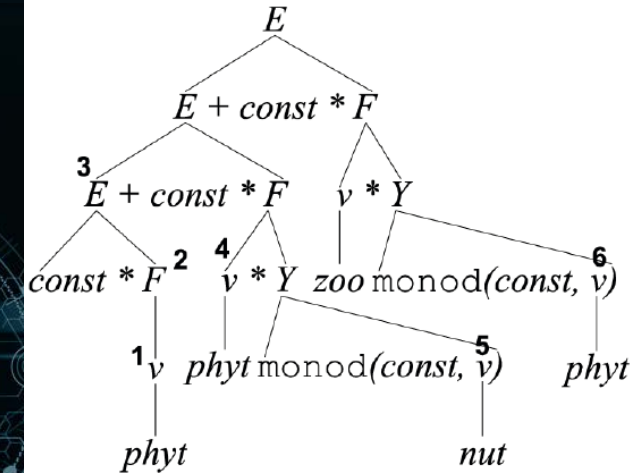


GRAMMARS ARE GENERATIVE MODELS

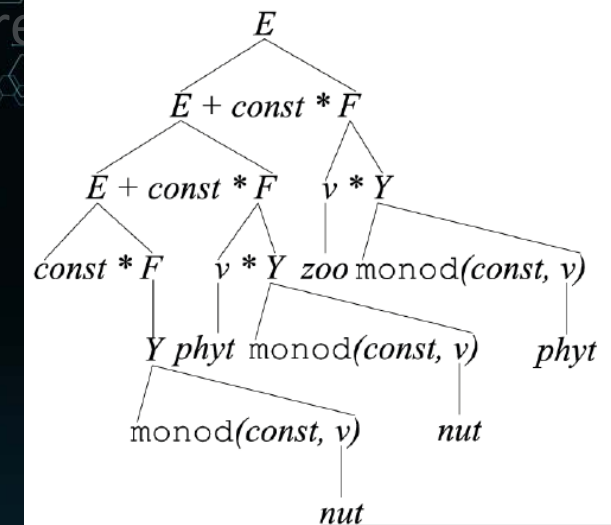
- Parse trees derive expressions from a grammar



- Parse tree:



Refined tree (refinement) (2)



- 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

$$PreyGrowth \rightarrow const * V_{Prey}$$

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

SEARCHING THE SPACE OF EQUATION STRUCTURES DEFINED BY THE GRAMMAR

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

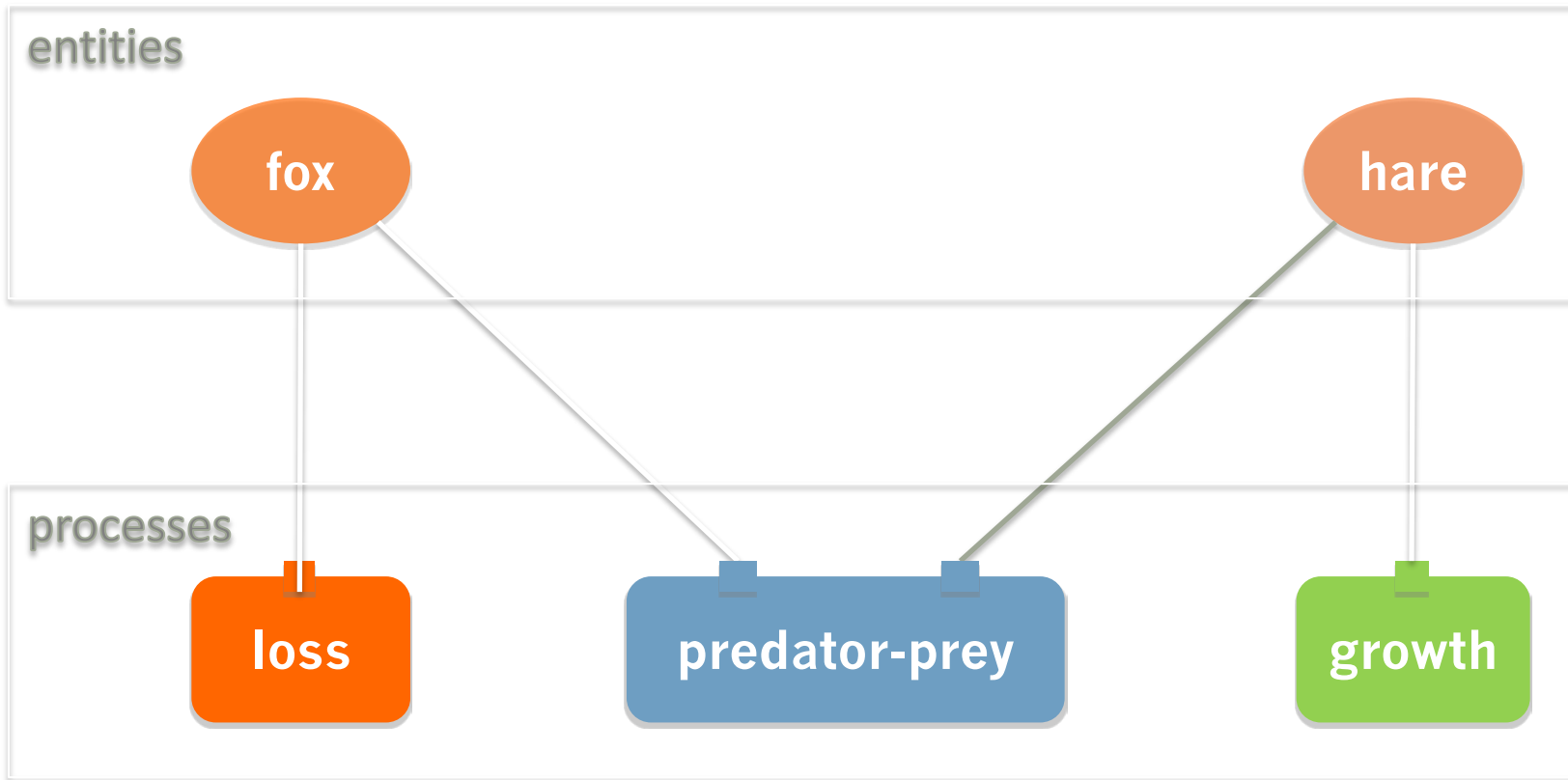
- Integrate two aspects of equation-based models
- Quantitative aspect: ODEs with given structure and parameter values that allow simulation

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

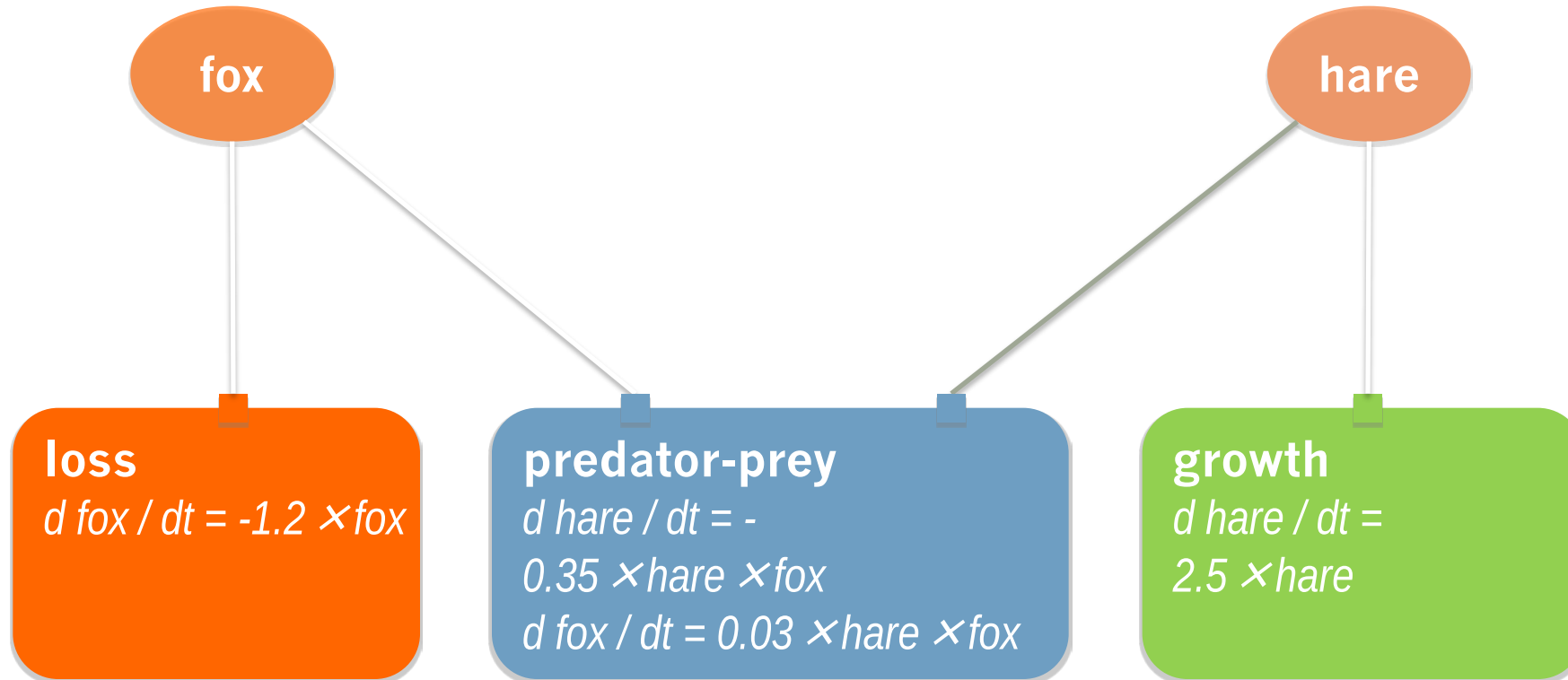
- Qualitative aspect: explanation of the structure of the modeled systems in terms of entities and processes
- In equations: Variables correspond to entities, terms correspond to processes
 - Exponential growth of hare population
 - Exponential loss of fox population
 - Predator-prey interaction between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



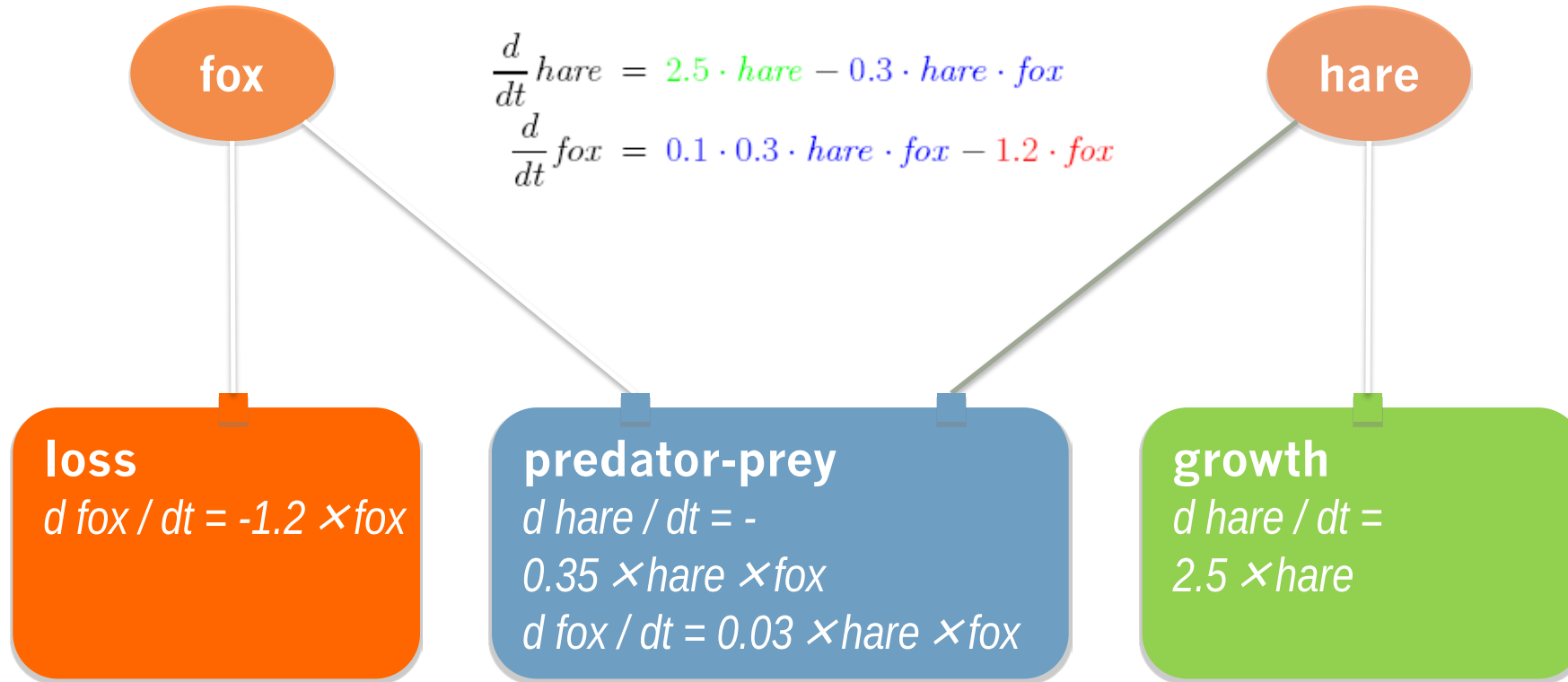
PBM: QUALITATIVE ASPECT



PBM: QUANTITATIVE ASPECT

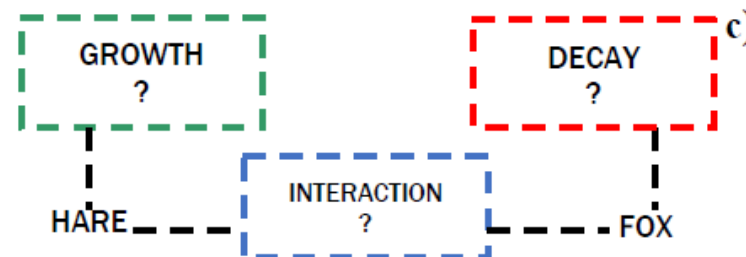
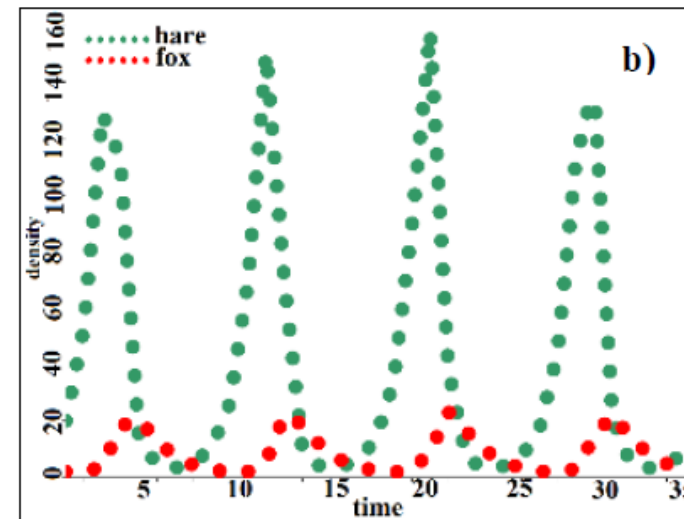
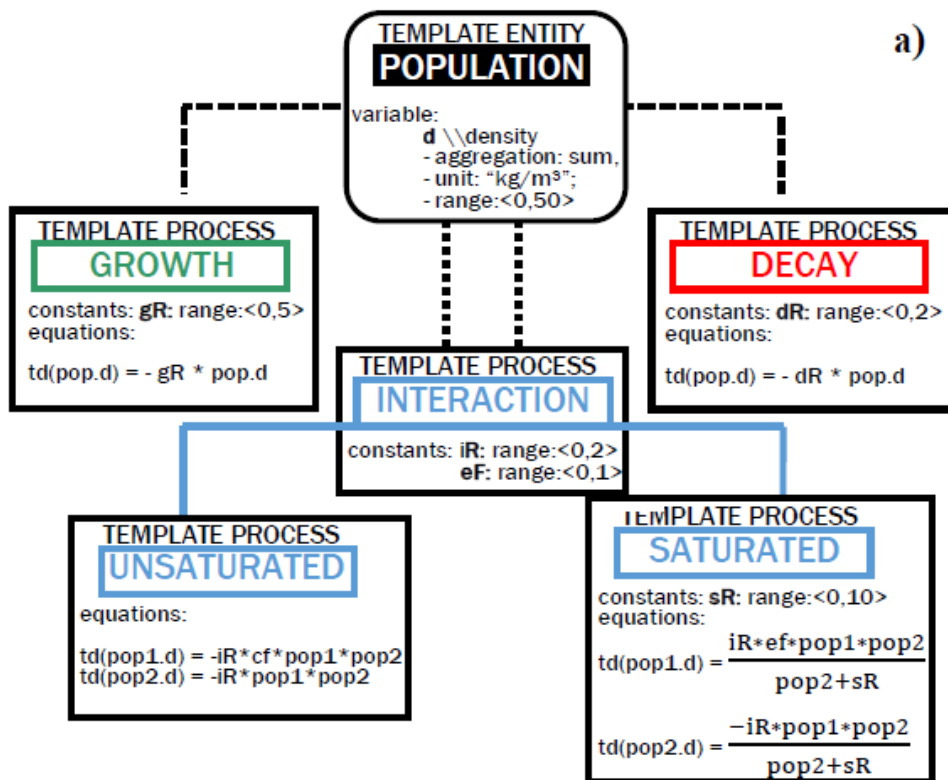


QUALITATIVE & QUANTITATIVE ASPECTS OF PBMS



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)

```

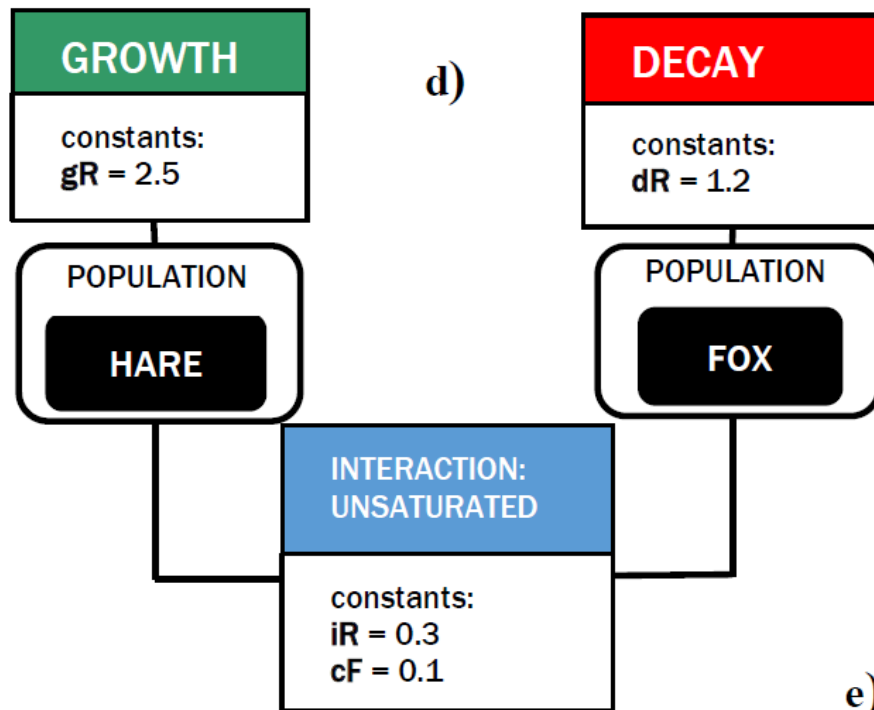
entity hare : Population;
entity fox : Population;
process growth (hare): Growth;
process decay (fox): Decay;
process predator_pre (fox,hare): Population;
    
```

c)



LEARNING PROCESS-BASED MODELS: OUTPUTS

- In human & computer-readable format



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)

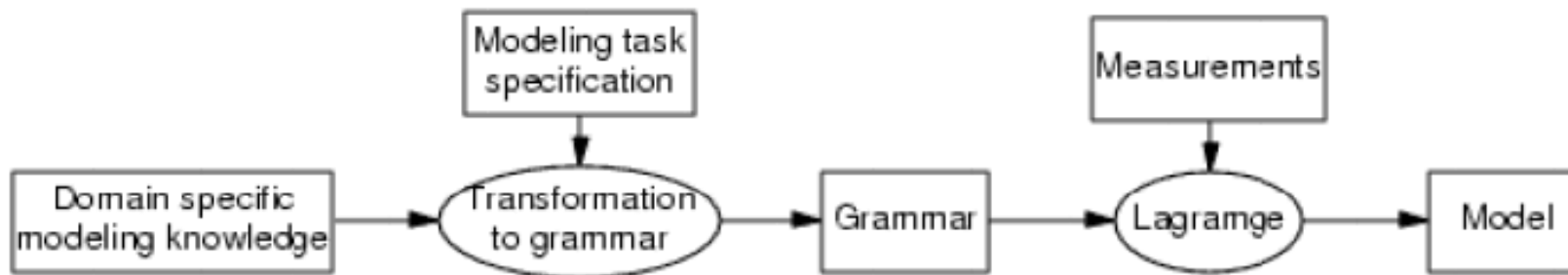
$$\frac{d}{dt} hare_d = 2.5 * hare_d - 0.3 - hare_d * fox_d$$

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

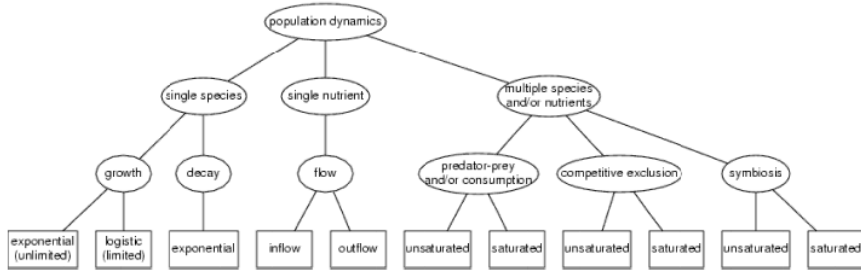


THE LAGRAMGE APPROACH TO IPM

- 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)

$$\frac{d}{dt}i = + \text{Flow}(i) - \sum_{\text{food}, i \in \text{food}} \text{const}(_, 0, 1, \text{Inf}) * \text{Feeds_on}(p, \text{food})$$

combining scheme Population_dynamics(Population p)

$$\frac{d}{dt}p = + \text{Growth}(p) - \text{Decay}(p) + \sum_{\text{food}} \text{const}(_, 0, 1, \text{Inf}) * \text{Feeds_on}(p, \text{food}) - \sum_{\text{pred}, \text{food}, p \in \text{food}} \text{const}(_, 0, 1, \text{Inf}) * \text{Feeds_on}(\text{pred}, \text{food})$$

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 * $\prod_{c \in cs} \text{Saturation}(c)$

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

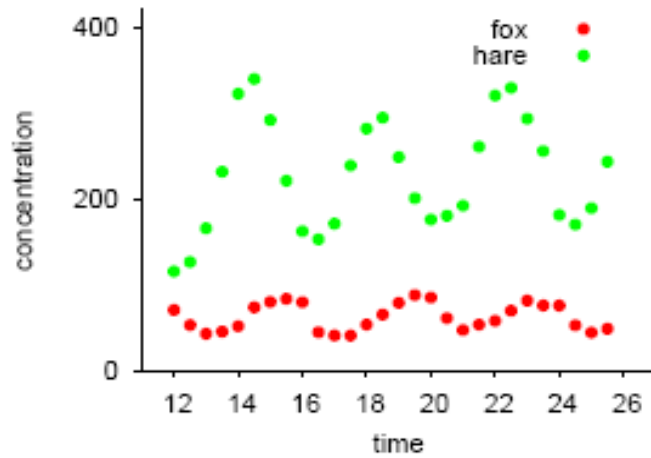
Feeds_on(lynx, hare) -> lynx * Saturation(hare)

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

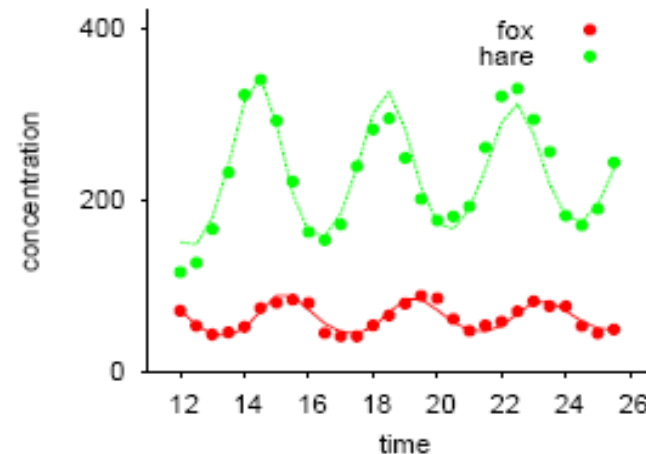


$PPModel$	$\rightarrow$	$PreyChange, PredatorChange$
$PreyChange$	$\rightarrow$	$PreyGrowth - Interaction$
$PredatorChange$	$\rightarrow$	$const * Interaction + PredatorLoss$
$PreyGrowth$	$\rightarrow$	$const * V_{Prey}$
$PreyGrowth$	$\rightarrow$	$const * V_{Prey} * (1 - V_{Prey}/const)$
$Interaction$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

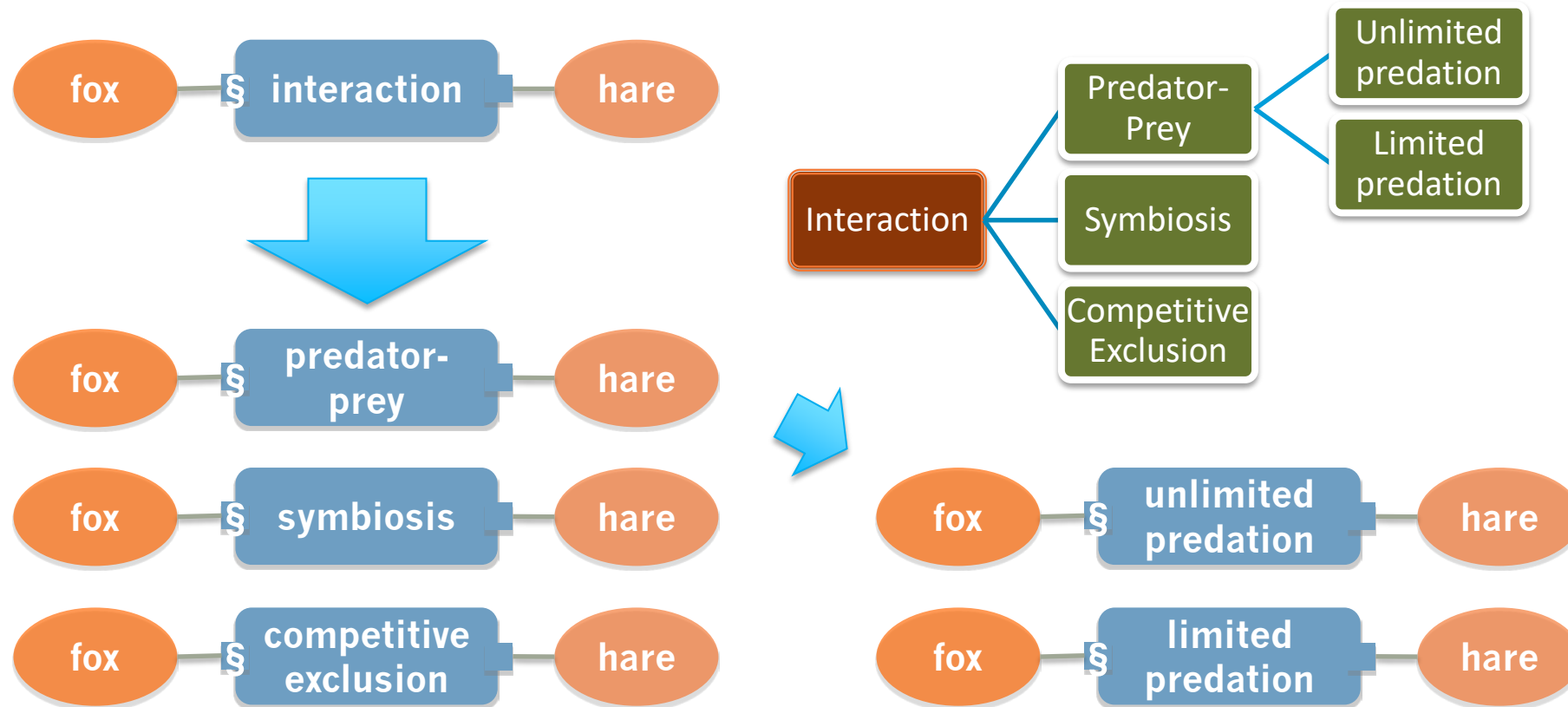
- Output: Set of ordinary differential equations(ODEs)

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

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

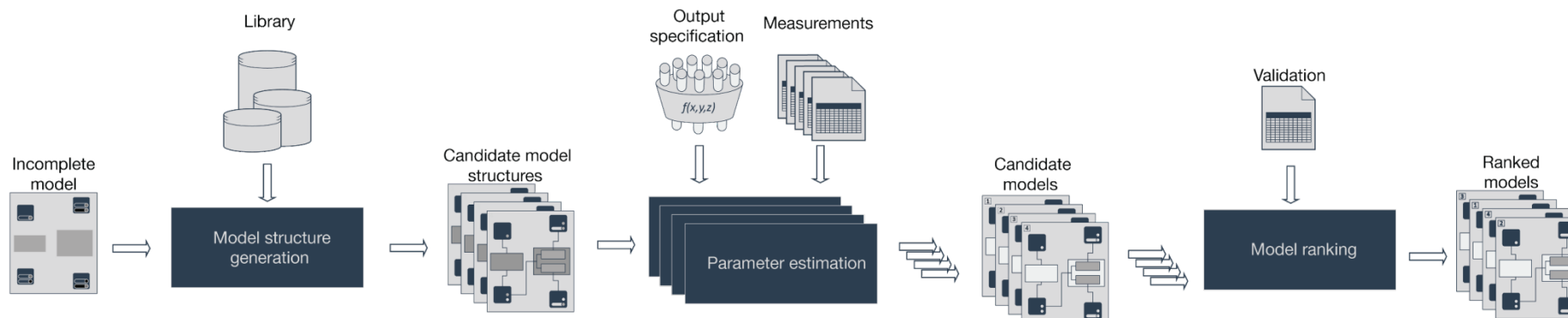


DIRECT SEARCH IN THE SPACE OF PBMS



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/heuristically) model structures
- Fits model parameters and
- Finds best candidate (process-based) model(s)



EQUATION DISCOVERY WITH PCFGS (BRENCÉ 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

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



PROBABILISTIC CFGS FOR EQUATION DISCOVERY: 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 EQUATION DISCOVERY

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

```

1: procedure GENERATE_SAMPLE( $G, A$ )
2:    $(s, p) = ([ ], 1)$ 
3:   Choose a random rule  $(A \rightarrow \alpha) \in \mathcal{R} : \alpha = A_1 A_2 \dots A_k, A_i \in \mathcal{N} \cup \mathcal{T}$ 
4:   for  $i = 1, i \leq k$  do
5:     if  $A_i \in \mathcal{T}$  then
6:        $s = s.append(A_i)$ 
7:     else
8:        $(s_i, p_i) = GENERATE\_SAMPLE(G, A_i)$ 
9:        $s = s.append(s_i)$ 
10:       $p = p \cdot p_i$ 
11:   return  $(s, p)$ 

```

```

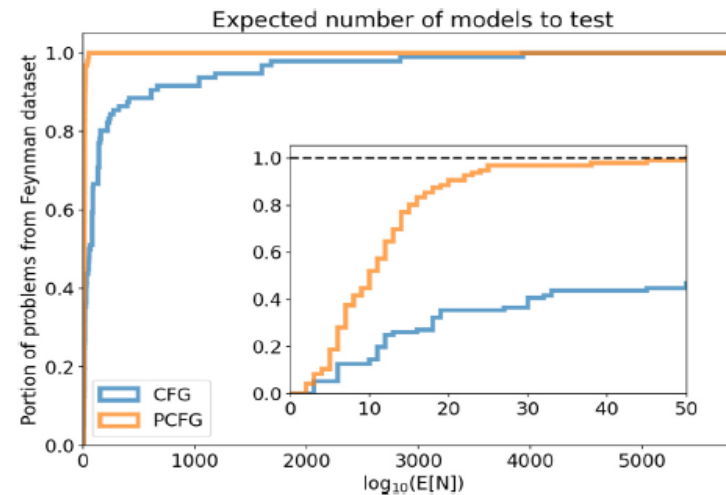
1: procedure MC_GBED( $G, N, D, e$ )
2:    $eqns = [ ]$ 
3:   for  $i = 1, i \leq N$  do
4:      $(e, p) = GENERATE\_SAMPLE(G, S)$ 
5:      $e_c = canonical\_form(e)$ 
6:      $eqn = fit\_parameters(e_c, v, D)$ 
7:      $error = ReRMSE(eqn, D)$ 
8:      $eqns.append(eqn, p, error)$ 
9:   return  $eqns.sort(key = error, order = increasing)$ 

```



THE UTILITY OF PCF GRAMMARS IN EQUATION DISCOVERY

- CF Grammars can 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 (PROBABILITIES IN PCF) GRAMMARS FROM CORPORA OF EQUATIONS (CHAUSHEVSKA EL AL. 2023)

Initial grammar

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

Parse

Corpus of equations

$$\frac{n_rho^{*}mom^{*}\tanh(mom^{*}B/(kb^{*}I))}{mom^{*}H/(kb^{*}T)+(mom^{*}\alpha)/}$$

$$(\epsilon* c^{*}2*kb^{*}T)*M$$

$$mom^{*}(1+\chi)*B$$

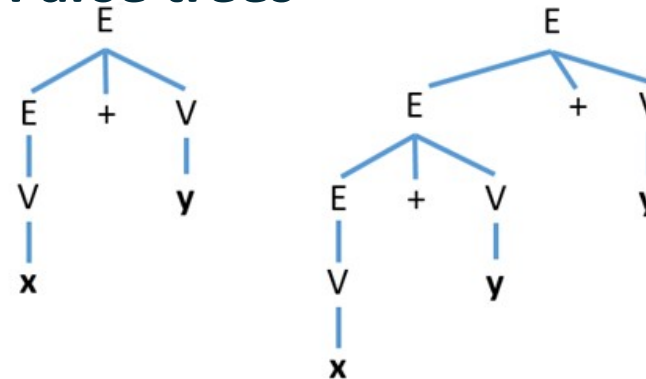
$$Y*A^{*}x/d$$

$$Y/(2*(1+\sigma))$$

$$1/(\exp((h/(2*\pi))^{*}\omega/(kb^{*}T))-1)$$

$$(h/(2*\pi))^{*}\omega/(\exp((h/(2*\pi))^{*}\omega/($$

Parse trees



Production frequencies

Updated grammar

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

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

$$\exp(-\theta^2/2)/\sqrt{2\pi}$$

$$\exp(-(\theta/\sigma)^2/2)/(\sqrt{2\pi}\sigma)$$

$$\exp(-((\theta-\theta_1)/\sigma)^2/2)/(\sqrt{2\pi}\sigma)$$

$$\sqrt{(x_2-x_1)^2+(y_2-y_1)^2}$$

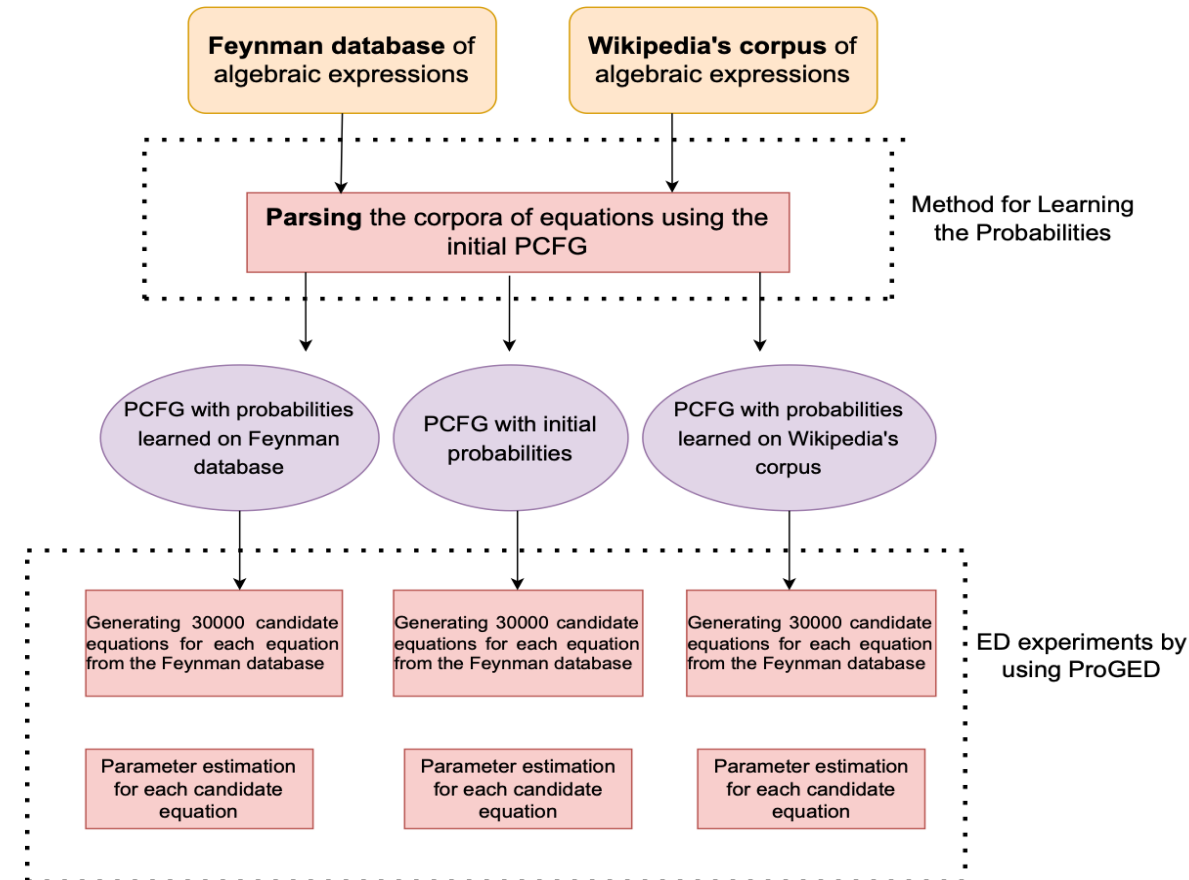
$$G m_1 m_2 / ((x_2-x_1)^2 + (y_2-y_1)^2 + (z_2-z_1)^2)$$

$$m_0 / \sqrt{1-v^2/c^2}$$

$$x_1 y_1 + x_2 y_2 + x_3 y_3$$

$$\mu N n$$

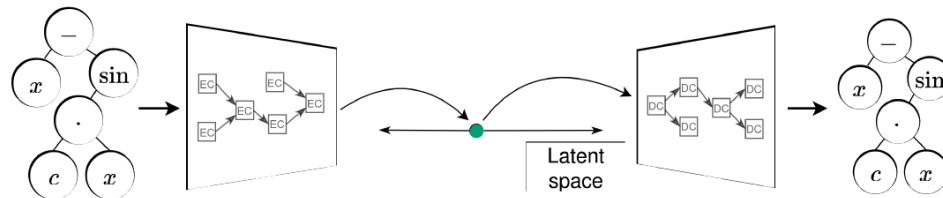
$$q_1 q_2 r / (4\pi \epsilon r^3)$$

$$q_1 r / (4\pi \epsilon r^3)$$


USING GRAMMARS TO TRAIN OTHER GENERATIVE MODELS (MEZNAR ET AL. 2023)

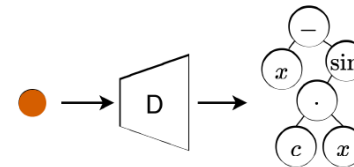
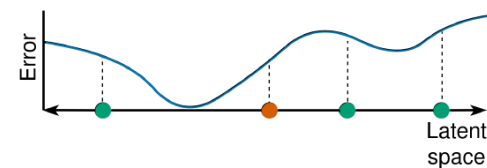
- Generate many expressions from a grammar
- Train a hierarchical variational autoencoder
- Explore the HVAE latent space with Evolutionary Algorithms

Step 1: Train a generative model



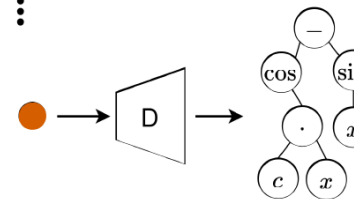
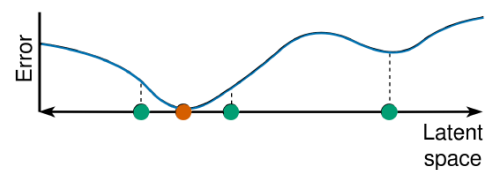
Step 2: Explore the latent space with EA

First iteration



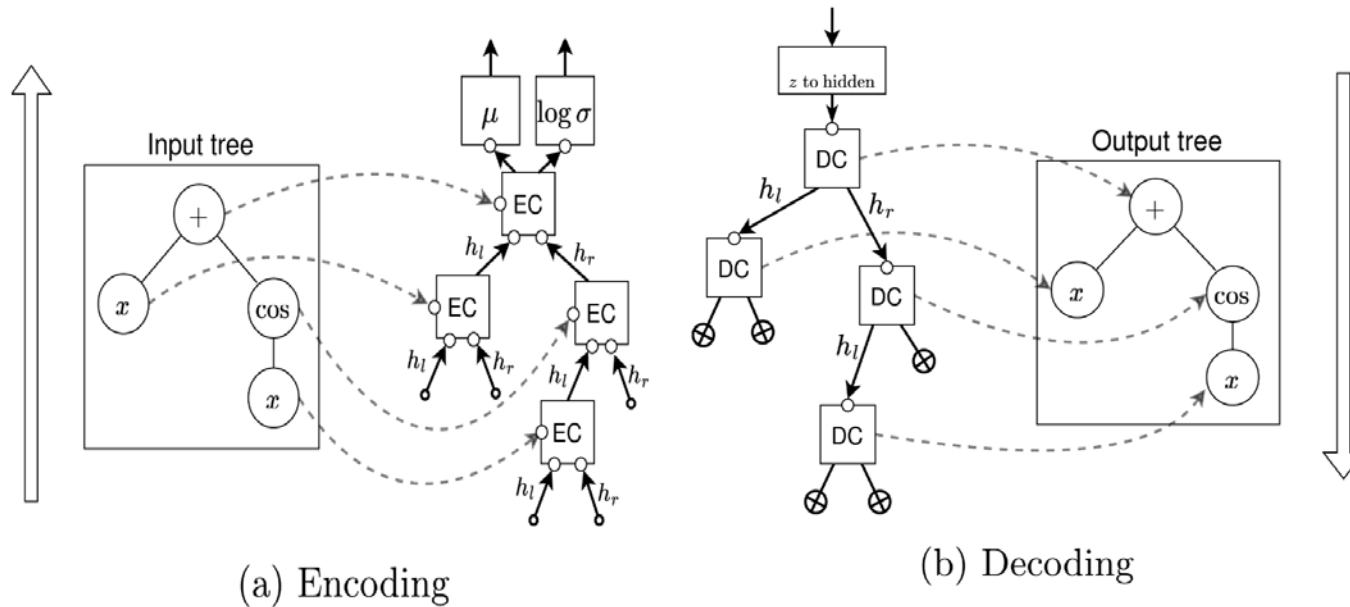
⋮

Nth iteration



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)



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

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

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	$c \cdot x \cdot \cos \frac{x}{c} + c$

- We decode expressions and expect a smooth transition between expressions



EVOLUTIONARY ALGORITHM (EDHiE)

- EDHiE (Eq. disc. with Hierarchical variational autoEncoders)
- Crossover: Offspring as 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

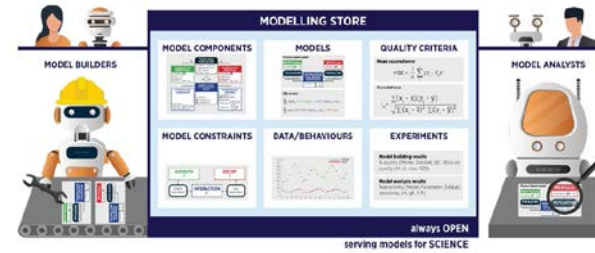
$$\mathcal{N}(a \cdot \mu_z, a \cdot \sigma_z + (1 - a) \cdot I)$$
- Nguyen benchmark results

Name	EDHiE (our)			DSO (Petersen et al., 2021)			PySR (Cranmer, 2023)	
	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 ($\pm 15,613$)	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 ($\pm 10,264$)	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 ($\pm 28,197$)	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)



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	$2 \sin x \cdot \cos y$

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



ARTIFICIAL INTELLIGENCE FOR SCIENCE @ JSI

National projects, Gravity framework

Artificial Intelligence for Science (GC-001): AI4Sci

The project focuses on the development of AI approaches along four major directions and their use in materials science, life sciences, environmental sciences

AI & Materials Science (GC-004): CHRONOSTORE

Chemical Energy Storage Solutions Across Temporal Scales for Climate-Resilient Renewable Energy Systems

AI & Life Sciences/ Personalized Medicine (GC-005): BOOST

Bridging oligonucleotide-based therapeutic upregulation of translation

AI & Environmental/Earth Sciences (GC-006): GeoAI

Geospatial information technologies for resilient and sustainable society



ARTIFICIAL INTELLIGENCE FOR SCIENCE (GC-0001)

National project, Gravity framework

Develop Artificial Intelligence methods suitable for use in Science

Structured in four workpackages

- *Explainable ML for science*
- *Foundation models for science,*
- *Automated scientific modelling, and*
- *Semantic technologies for open science.*

Applications in the Sciences

- *Materials Science*
- *Life Sciences*
- *Environmental Sciences*
- *Mathematics*
- *Computer science (ML, optimization): AutoML/AutoOpt*



oi deepseek-r1:14b 14.8B

oi deepseek-r1:7b 7.6B

oi llama3.3:latest 70.6B

oi gemma-slo-vlm:latest 5.0B

oi gemma3:27b 27.4B

oi gemma3:27b-it-qat 27.4B

oi hf.co/tknez/GaMS-9B-Instruct

oi QwQ-32B-GGUF:latest 32.8B

oi llama4:scout 108.6B

oi GaMS-27B-Instruct 27.2B

oi cogito:70b 70.6B

oi hf.co/gabriellarson/NatureLM-8

oi magistral:24b-small-2506-fp16

oi mathstral:7b-v0.1-fp16 7.2B

oi mistral-small3.2:24b-instruct-2506-

oi phi4-reasoning:14b-plus-fp16 14.7B

oi qwen3:32b 32.8B

AI FOR SCIENCE: INSTITUTIONAL INFRASTRUCTURE (llm.ijs.si)

Local infrastructure at JSI, serving LLMs to researchers at JSI

At the end of March 2024, we released a preliminary service, based on the LLM model Mistral7B

By October 2024, we had added a few additional LLMs

We have a local setup of open-sourced LLMs, hosted on JSI infrastructure, available via a chat assistant & programmatic API

- Fast and scalable implementation using Paged Attention
- Up to few thousand concurrent connections

Different LLMs available at llm.ijs.si for different purposes

- Multiple models are implemented concurrently, including DeepSeek and MathModels – Mathstral, which are specialized in custom domains.
- Current list (left) includes GAMS, a LLM for Slovene
- This specialization allows them to deliver tailored solutions for specific applications, enhancing their effectiveness and performance in niche areas.

SLAIF: SLOVENIAN AI FACTORY

AI HPC, SERVICES & ACTIVITIES



Combination of CPU and GPU resources, supporting both traditional HPC and AI computing:

- CPU: approx. 10 petaFLOPS
- GPU: 30 petaFLOPS (Phase 1) +70 petaFLOPS (Phase 2) = 100 petaFLOPS in total

Access to Computing and Data Infrastructure

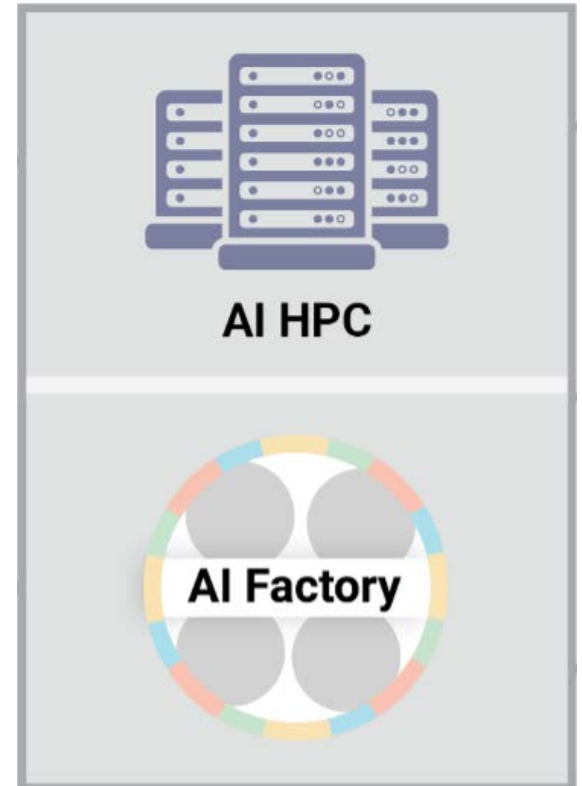
General Workflows for Training and Fine-Tuning Models

- Large language models
- Large speech models
- Vision-language models
- Multimodal models

Domain-Specific Workflows and Models for Different Verticals

- Green transition
- Health and biotechnology
- Digital society
- AI for science

AI Education/ Skills Development



ARTIFICIAL INTELLIGENCE FOR SCIENCE @ JSI

EU funded projects

AIREA: *European Network of Excellence for AI-Enabled Agricultural & Environmental Pollution Science*

DIAMOND: *Digital Innovation for Advanced Materials and Novel Design*

ELAI: *European Laboratory for Artificial Intelligence*

ELIAS: *European Lighthouse of Artificial Intelligence for Sustainability*

LLMs4EU: *Large Language Models for the European Union*

AutoLearn-SI: *Leveraging Benchmarking Data for Automated Machine Learning and Optimization*



**Funded by
the European Union**

Artificial Intelligence for Science

Computational Approaches to Scientific Discovery

Table of Contents

I Automating Scientific Discovery	
Knowledge Engineering for Automated Scientific Discovery	3
<i>Larisa N. Soldatova</i>	
The Relevance of Natural Kinds to the Computational Discovery of Novel Scientific Variables	21
<i>Benjamin C. Jantzen</i>	
Data-Driven and Knowledge-Based Approach to Scientific Discovery	43
<i>Mitsuhiro Odaka, Morgan Magnin, and Katsumi Inoue</i>	
Automated Scientific Discovery of Mind and Behavior	61
<i>Sebastian Musslick, Younes Strittmatter, and Marina Dubova</i>	
The Use of AI-Robotic Systems for Scientific Discovery	99
<i>Alexander H. Gower, Konstantin Korovin, Daniel Brunnsäker, Filip Kronström, Gabriel K. Reder, Ievgeniia A. Tiukova, Ronald S. Reiserer, John P. Wikswo, and Ross D. King</i>	
II Equation Discovery and Symbolic Regression	
Expediting Symbolic Regression for Science Using Scientific Approaches	125
<i>Md Nasim, Nan Jiang, and Yezhang Xue</i>	
Deep Symbolic Optimization: Reinforcement Learning for Symbolic Mathematics	147
<i>Conor F. Hayes, Felipe Leno Da Silva, Jiachen Yang, T. Nathan Mundhenk, Chak Shing Lee, Jacob F. Pettit, Claudio Santiago, Sookyoung Kim, Joanne T. Kim, Ignacio Aravena Solis, Ruben Glatt, Andre R. Goncalves, Alexander Ladd, Ahmet Can Solak, Thomas Desautels, Daniel Faissol, Brenden K Petersen, and Mikel Landajuela</i>	
Neural-Guided Equation Discovery	178
<i>Jannis Brugger, Mattia Cerrato, David Richter, Cedric Derstroff, Daniel Maninger, Sajjad Shumaly, Mira Mezini, and Stefan Kramer</i>	
Generative Models for Equation Discovery	216
<i>Sebastian Mežnar, Jure Brence, Sašo Džeroski, and Ljupčo Todorovski</i>	

XVIII Table of Contents

Ramanujan Machine – Generating Conjectures on Fundamental Constants	240
<i>Itay Beit-Halachmi, Rotem Elimelech, Ofir David, Ofir Razon, Michael Shalyt, and Ido Kaminer</i>	
III Discovering Models of Dynamical Systems	
Computational Discovery of Quantitative Process Models	261
<i>Pat Langley</i>	
Discovering Polynomial and Quadratic Structure in Nonlinear Ordinary Differential Equations	284
<i>Boris Kramer and Gleb Pogudin</i>	
A Comprehensive Review of Latent Space Dynamics Identification Algorithms for Intrusive and Non-Intrusive Reduced-Order-Modeling	303
<i>Christophe Bonneville, Xiaolong He, April Tran, Jun Sur R. Park, William Fries, Daniel A. Messenger, Siu Wun Cheung, Yeonjong Shin, David M. Bortz, Debojyoti Ghosh, Jiun-Shyan Chen, Jonathan Belof, and Youngsoo Choi</i>	
Progressive Learning for Subsurface Physics	336
<i>Teeratorn Kadeethum, Daniel O'Malley, Youngsoo Choi, Hari S. Viswanathan, and Hongkyu Yoon</i>	
Process-Based Modelling of Natural and Urban Catchments	364
<i>Matej Radinja, Nataša Atanasova, Sašo Džeroski, and Mateja Škerjanec</i>	
IV Artificial Intelligence for Science	
Artificial Intelligence for Science	397
<i>Sašo Džeroski</i>	

