Integrating Data- and Knowledge-Driven Approaches to Automated Modelling

Sašo Džeroski
Jozef Stefan Institute (JSI), Ljubljana, Slovenia

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:

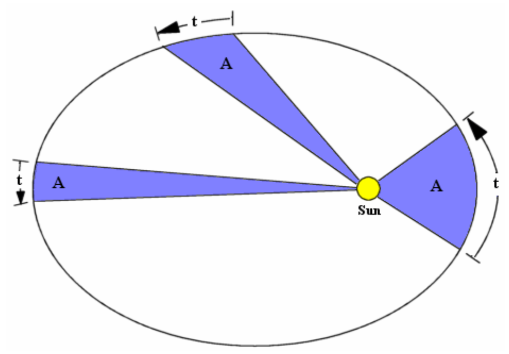
- Statements about the structures or processes that arise in the environment
- Stated using terms from the domain's taxonomy
- Interconnect laws into a unified theoretical account

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

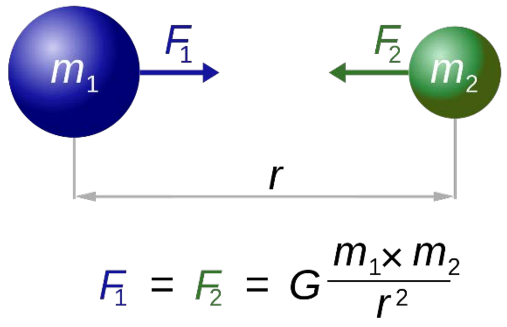
- E.g., $F = m \times a$ (Newton's second law)
- For a specific object of mass 2.3 kg, we would have the model $F = 2.3 \times a$

Scientific knowledge: representation

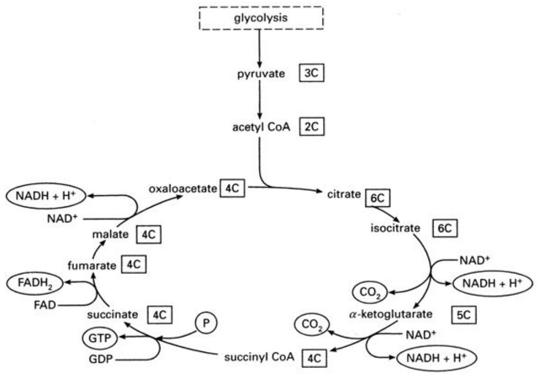
Scientific knowledge is typically represented in scientific formalisms (e.g., equations, pathways), introduced and routinely used by scientists



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

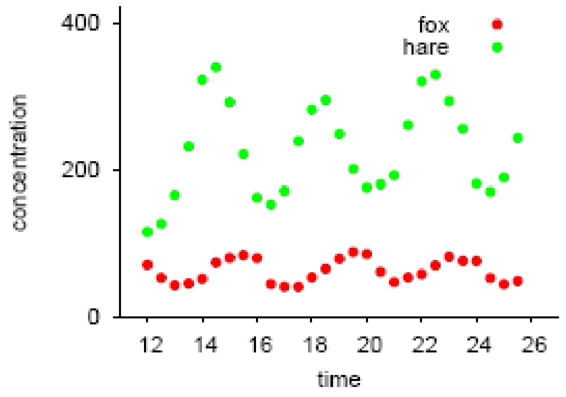
- Input: Observations of distances between moons and Jupiter (d) and periods of their orbits (p)
- Output: Kepler's third law $d^3/p^2 = k$
- Process of discovery
    - BACON (Langley 1978; Langley et al. 1987)
    - Carries out heuristic search through the space of numeric terms, looking for constant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

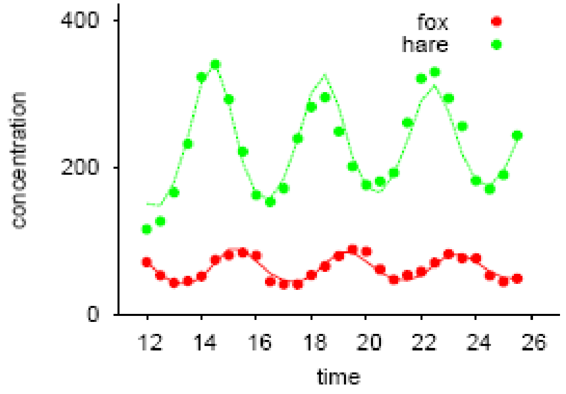
- Input: Observed behavior of dynamic system



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

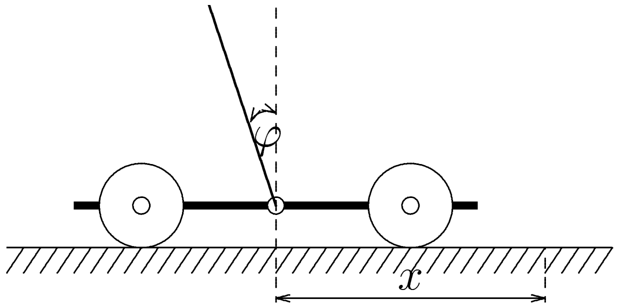
$$
\begin{aligned}
\frac{d}{dt} hare &= 2.5 \cdot hare - 0.3 \cdot hare \cdot fox \\
\frac{d}{dt} fox &= 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox
\end{aligned}
$$



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

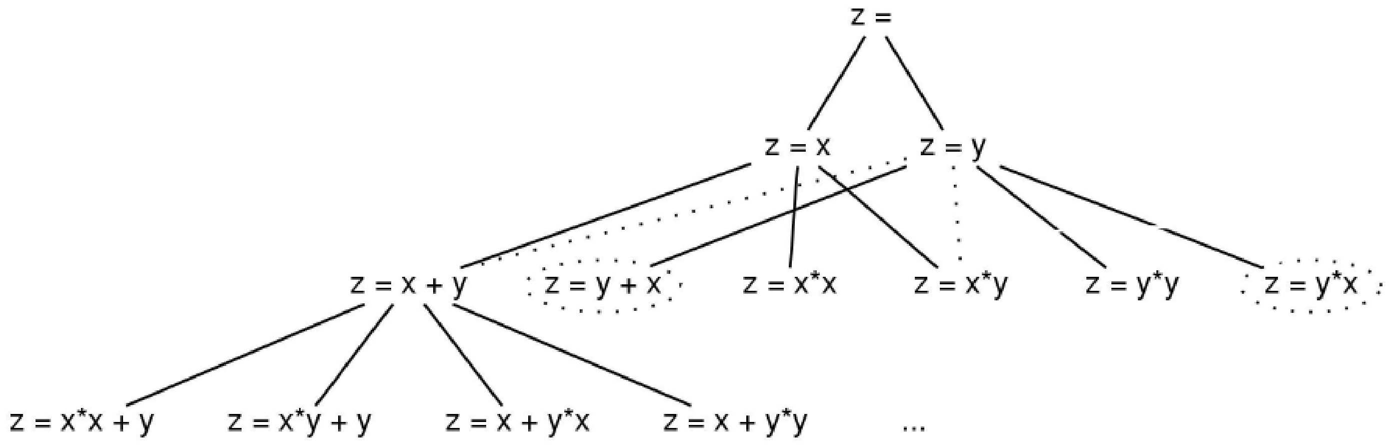
$$\mathrm{MDL}(v_d = P) = \mathrm{len}(P) \log m + m \log \mathrm{MSE}(v_d = P)$$

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

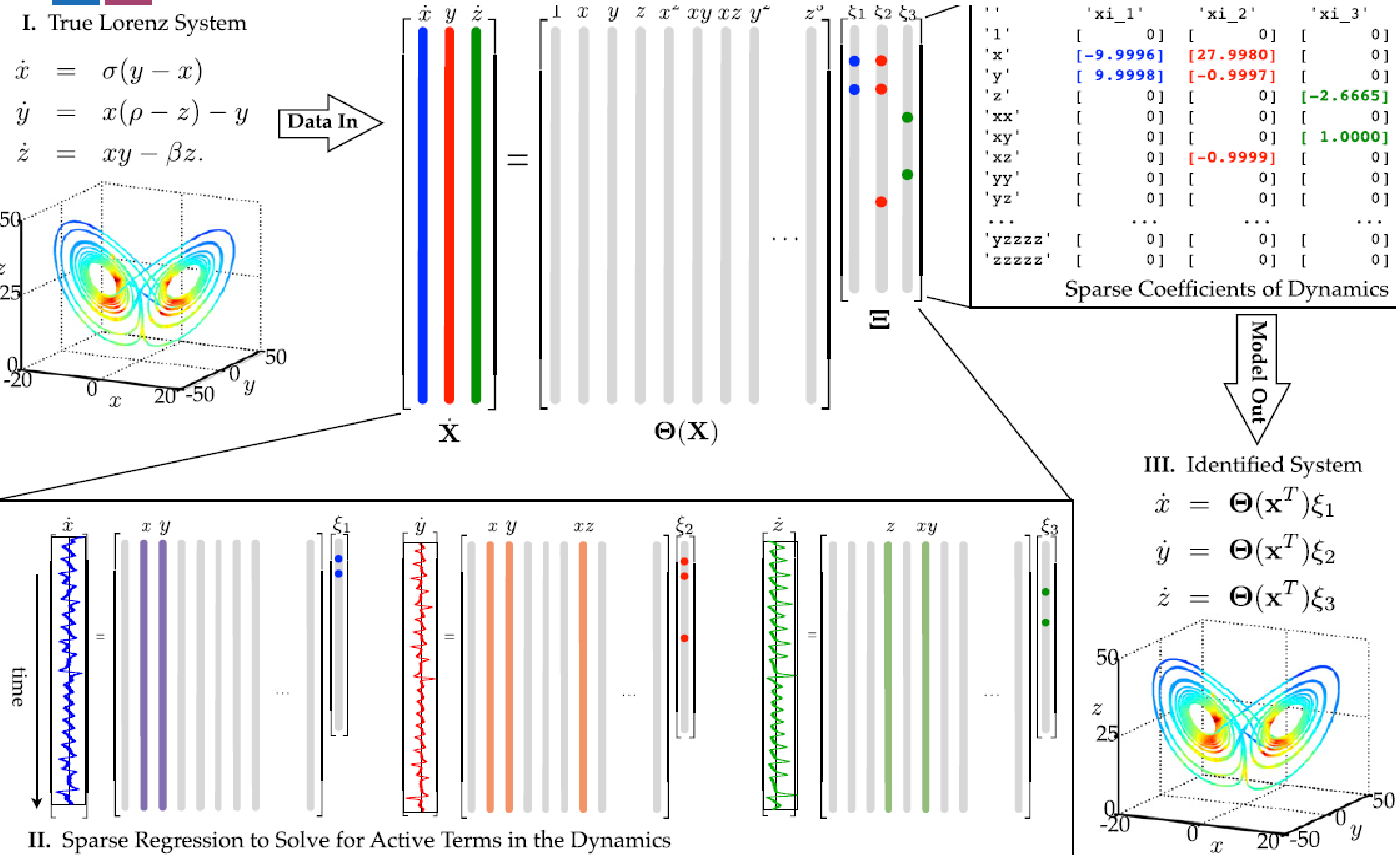
Heuristic search through the space of polynomial equations



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)

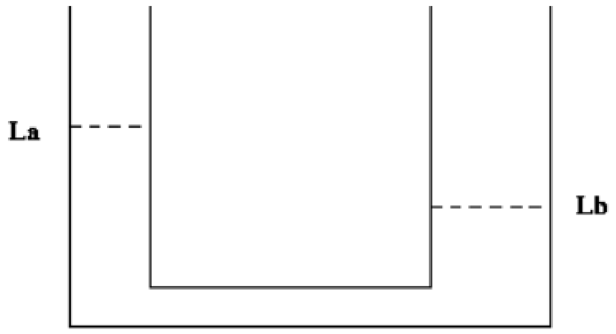


I. True Lorenz System
$\dot{x} = \sigma(y - x)$
$\dot{y} = x(\rho - z) - y$
$\dot{z} = xy - \beta z.$
50
z
25
0
-20
0
x
20
-50
0
50
y
Data In
$\dot{x}$ y $\dot{z}$
$=$
1 x y z x^2 xy xz y^2 z^5
ξ_1 ξ_2 ξ_3
$\dot{\mathbf{X}}$
$\Theta(\mathbf{X})$
Ξ
Sparse Coefficients of Dynamics
Model Out
III. Identified System
$\dot{x} = \Theta(\mathbf{x}^T)\xi_1$
$\dot{y} = \Theta(\mathbf{x}^T)\xi_2$
$\dot{z} = \Theta(\mathbf{x}^T)\xi_3$
50
z
25
0
-20
0
x
20
-50
0
50
y
time
$\dot{x}$
x y
ξ_1
$\dot{y}$
x y xz
ξ_2
$\dot{z}$
z xy
ξ_3
II. Sparse Regression to Solve for Active Terms in the Dynamics

```
''         'xi_1'        'xi_2'        'xi_3'
'1'      [       0]   [       0]   [       0]
'x'      [-9.9996]    [27.9980]    [       0]
'y'      [ 9.9998]    [-0.9997]    [       0]
'z'      [       0]   [       0]   [-2.6665]
'xx'     [       0]   [       0]   [       0]
'xy'     [       0]   [       0]   [ 1.0000]
'xz'     [       0]   [-0.9999]    [       0]
'yy'     [       0]   [       0]   [       0]
'yz'     [       0]   [       0]   [       0]
...         ...          ...          ...
'yzzzz'  [       0]   [       0]   [       0]
'zzzzz'  [       0]   [       0]   [       0]
```

Representation: Qualitative models

- The U-tube system



- Differential equations

$$\begin{aligned} \dot{l}_A &= c(l_B - l_A) \\ \dot{l}_B &= -\dot{l}_A \end{aligned}$$

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

$$\begin{array}{lll} deriv(l_B, \dot{l}_B) & deriv(l_A, \dot{l}_A) & minus(\dot{l}_A, \dot{l}_B) \\ M^+(l_A, \dot{l}_B) & M^+(l_B, \dot{l}_A) & M^-(l_A, l_B) \\ M^-(l_A, \dot{l}_A) & M^-(l_B, \dot{l}_B) & M^-(\dot{l}_A, \dot{l}_B) \end{array}$$

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

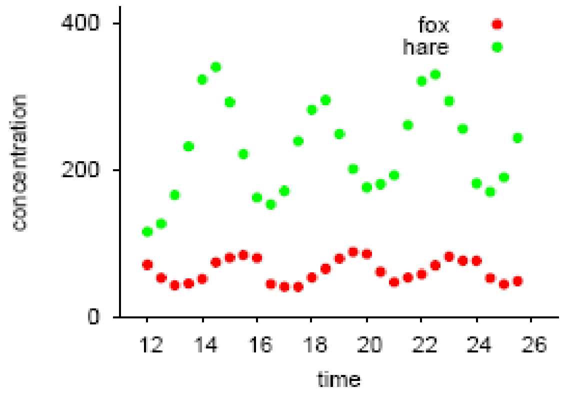
2. Typically, both the structure and parameters of the models are derived by the expert from knowledge about processes and process rates/parameters

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, than need to be completed
- All of these can be expressed with grammars

Grammar-based equation discovery

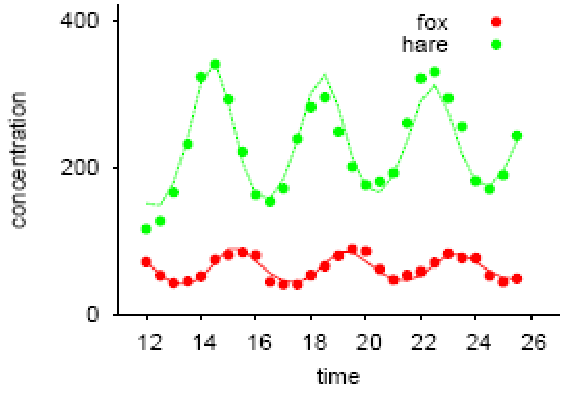
- Input: Observed behavior of dynamic system + Grammar



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

- 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

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

$$\begin{aligned} E &\rightarrow const \mid const \cdot F \mid E + const \cdot F \\ F &\rightarrow v \mid v \cdot F \end{aligned}$$

- Context free grammar used for environmental dynamic systems:

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

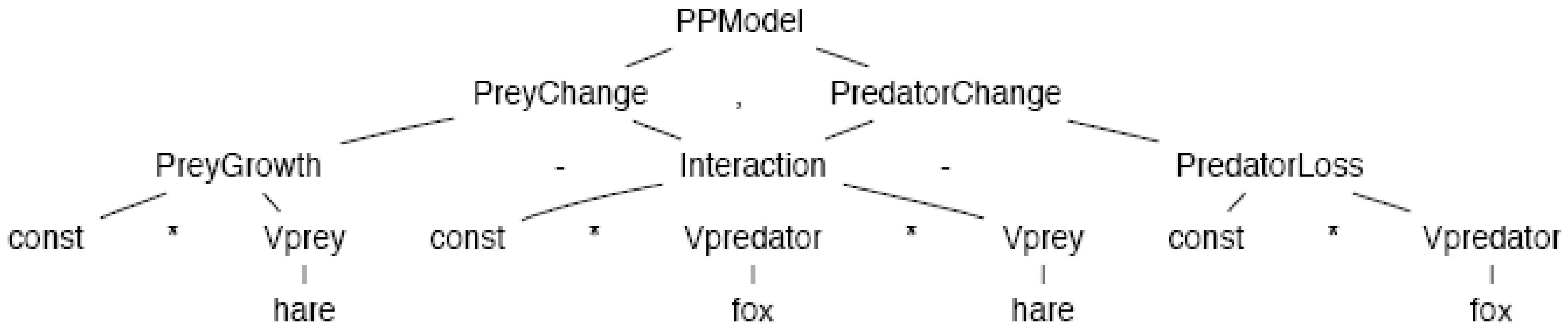
$$Y : \frac{v}{v + const}$$

- 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

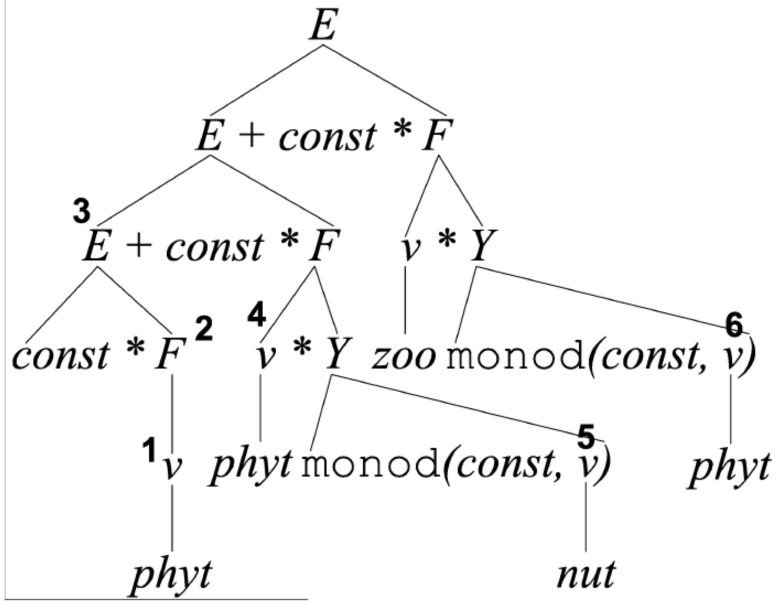
$$PreyGrowth \rightarrow const * V_{Prey}$$
$$PreyGrowth \rightarrow const * V_{Prey} * (1 - V_{Prey}/const)$$

Example parse tree & refinement

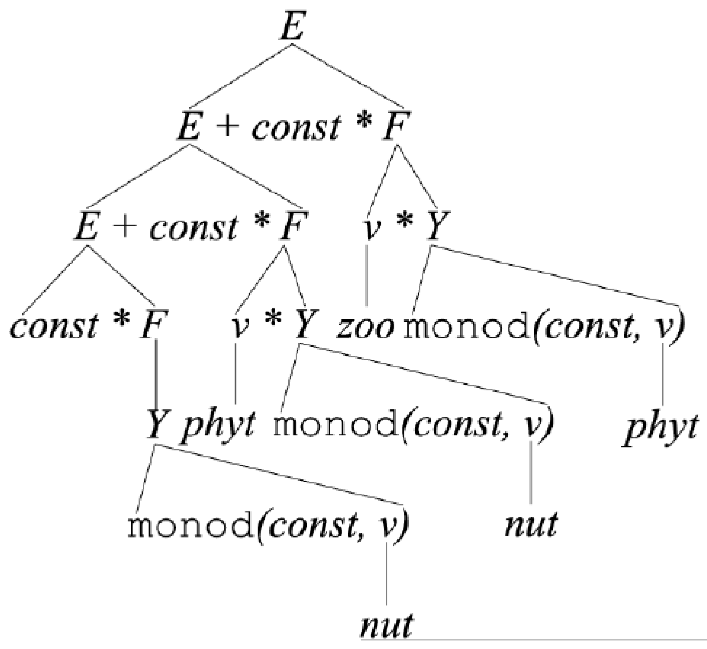
- Expression:

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

- Parse tree:



Refined tree (refinement) (2)



Searching the space of equation structures defined by the grams.

LAGRAMGE (Todorovski & Dzeroski 1997)

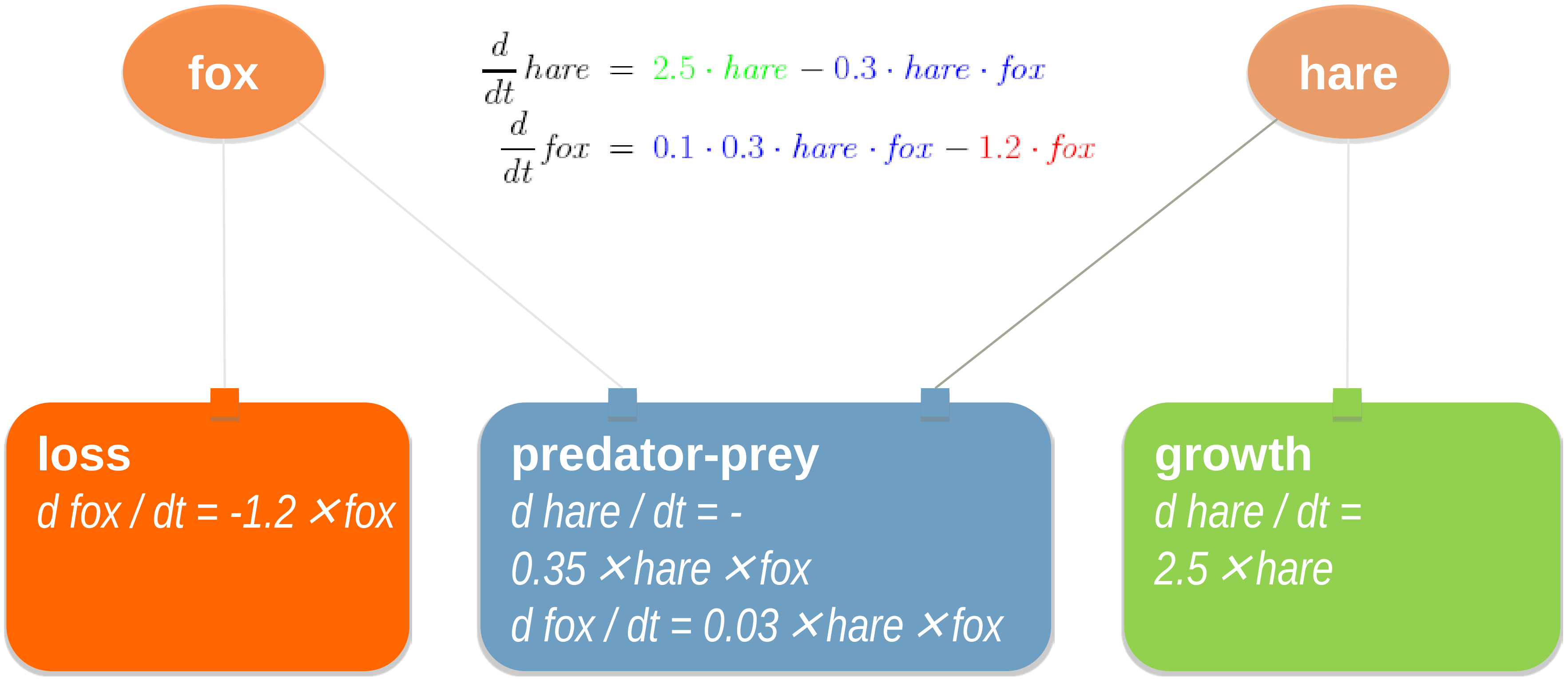
- Exhaustive search, generate and test equations corresponding to parse trees of limited depth
- Heuristic search (beam search)

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: Qual. & quant. Aspect of PBMs

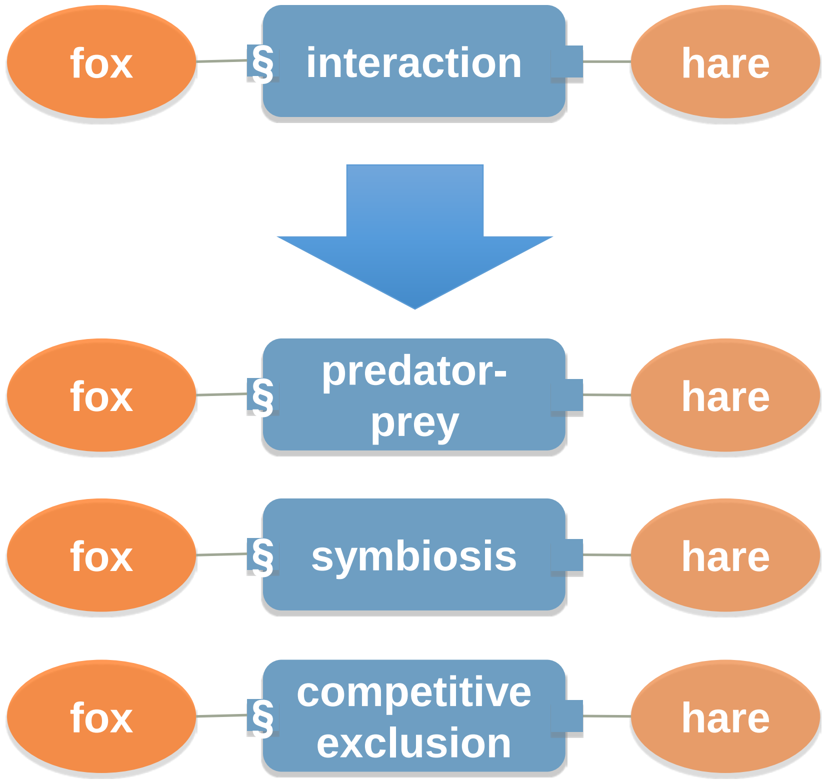


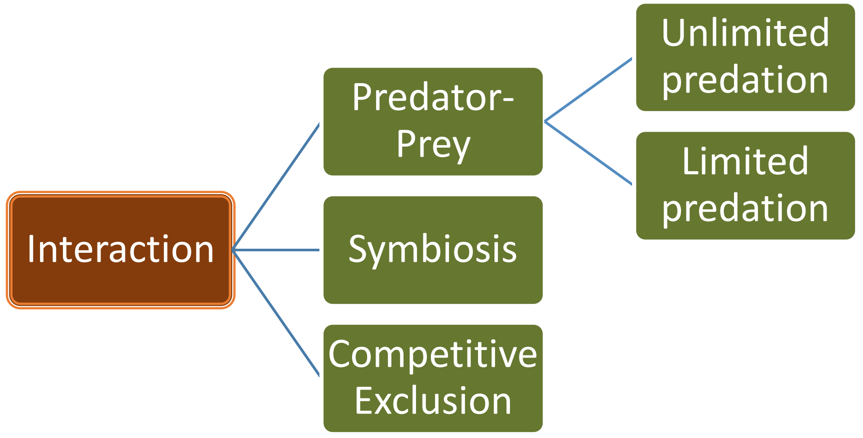
$$\frac{d}{dt} hare = 2.5 \cdot hare - 0.3 \cdot hare \cdot fox$$

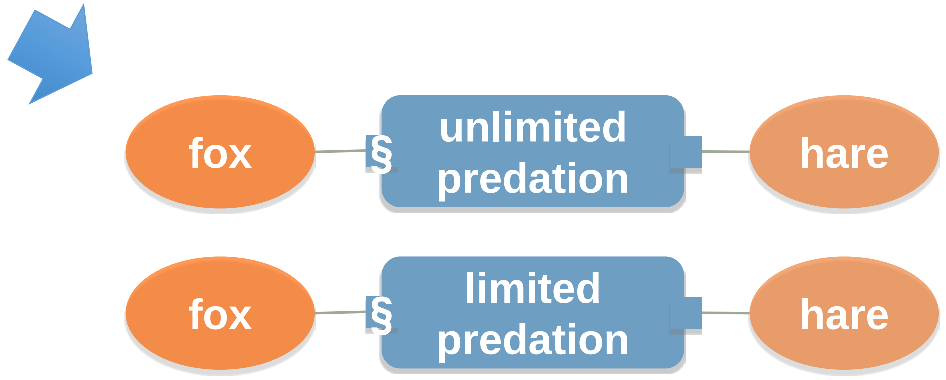
$$\frac{d}{dt} fox = 0.1 \cdot 0.3 \cdot hare \cdot fox - 1.2 \cdot fox$$

We can transform the problem of PBM into a problem of grammar-based equation discovery (Todorovski 2003) or…

Direct search in the space of PBMs







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. $\mathcal{R} =$

$$E \to E + F \mid E - F \mid F$$
$$F \to F * T \mid F/T \mid T$$
$$T \to (E) \mid V$$
$$V \to x \mid y.$$

4. $S = E.$

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

$$E \to E + F\ [0.2] \mid E - F\ [0.2] \mid F\ [0.6]$$
$$F \to F * T\ [0.2] \mid F/T\ [0.2] \mid T\ [0.6]$$
$$T \to R\ [0.2] \mid V\ [0.4] \mid c\ [0.4]$$
$$R \to (E)\ [0.6] \mid \sin(E)\ [0.1] \mid \cos(E)\ [0.1]$$
$$\mid \sqrt{E}\ [0.1] \mid \exp(E)\ [0.1]$$
$$V \to x\ [0.5] \mid y\ [0.5]$$

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 $A \in \mathcal{N}$

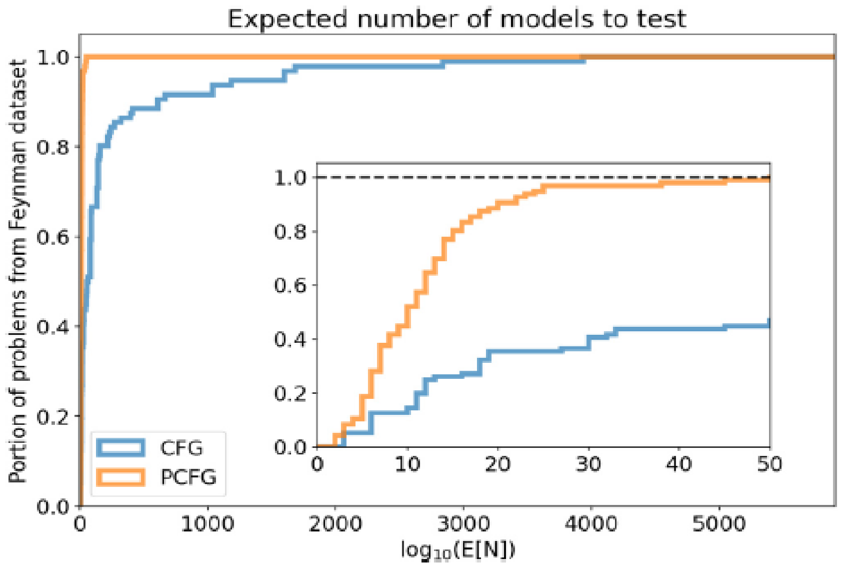
Ensure: Sentence s corresponding to a randomly sampled parse tree ψ from G with root node A, probability p of ψ

```
1:  procedure GENERATE_SAMPLE($G$, $A$)
2:      $(s, p) = ([\ ], 1)$
3:      Choose a random rule $(A \rightarrow \alpha) \in \mathcal{R} : \alpha = A_1 A_2 \ldots A_k,\ A_i \in \mathcal{N} \cup \mathcal{T}$
4:      for $i = 1, i \leq k$ do
5:          if $A_i \in \mathcal{T}$ then
6:              $s = s.\text{append}(A_i)$
7:          else
8:              $(s_i, p_i) = $ GENERATE_SAMPLE($G$, $A_i$)
9:              $s = s.\text{append}(s_i)$
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:          $e_c = \text{canonical\_form}(e)$
6:          $eqn = \text{fit\_parameters}(e_c, v, D)$
7:          $error = ReRMSE(eqn, D)$
8:          $eqns.\text{append}(eqn, p, error)$
9:      return $eqns.\text{sort}(\text{key} = error, \text{order} = \text{incr}$
```

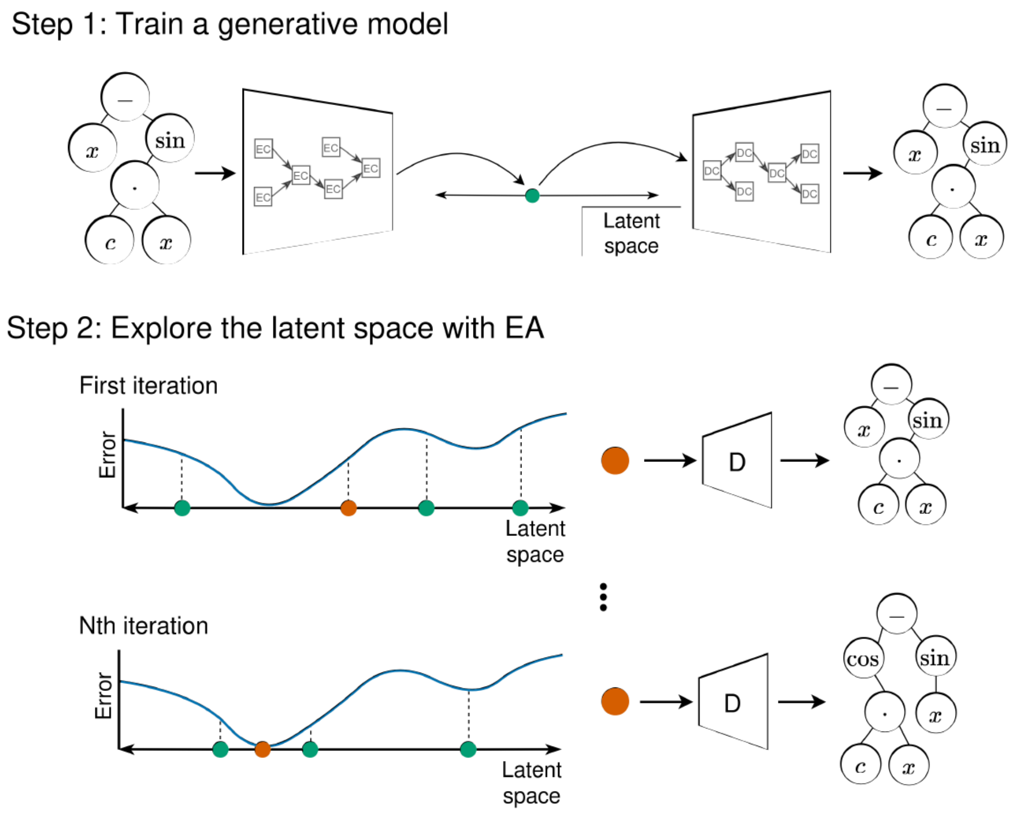
The utility of grammars in ED

- 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



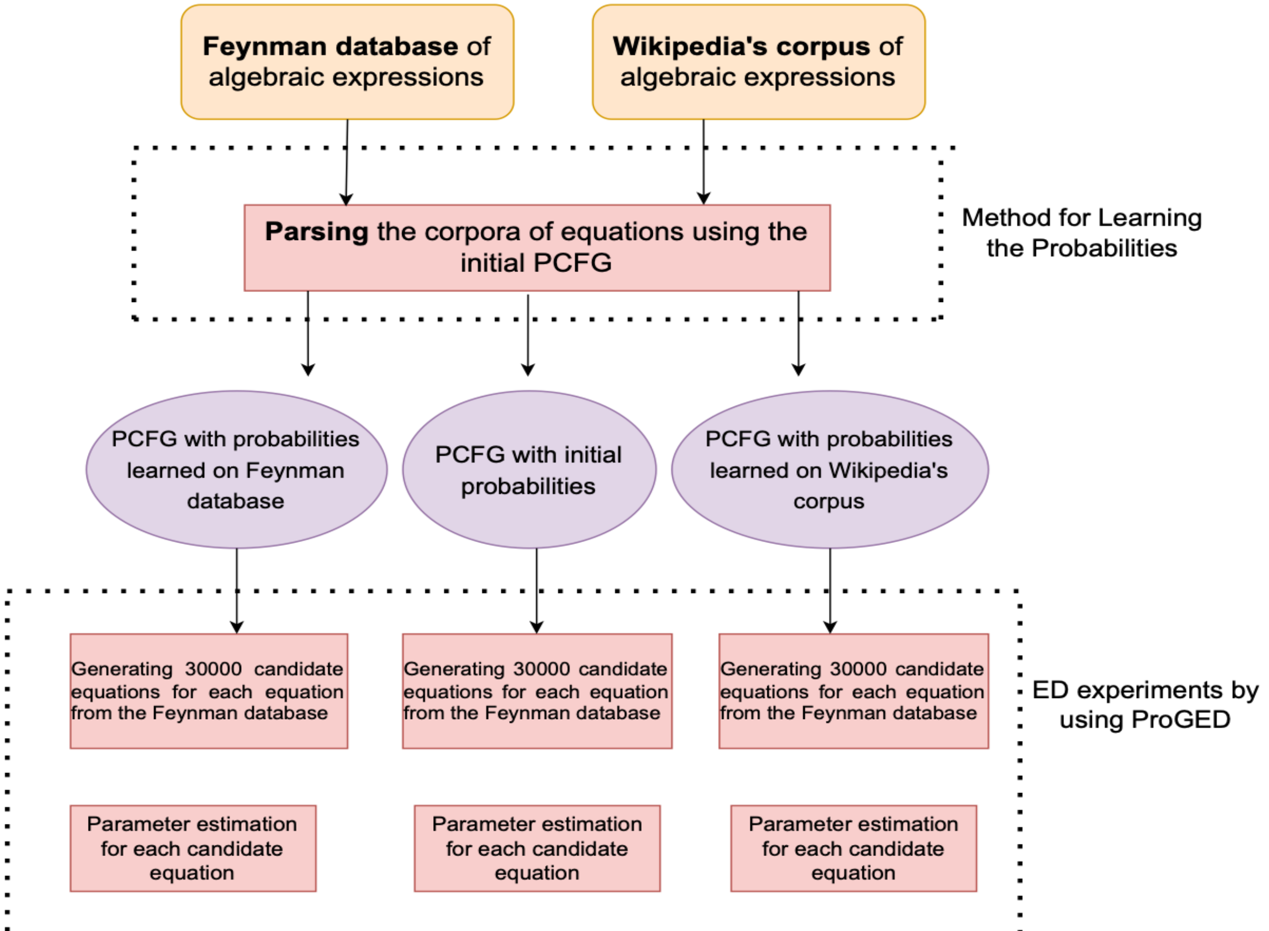
Using grammars to train other generative models (Meznar et al.)

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



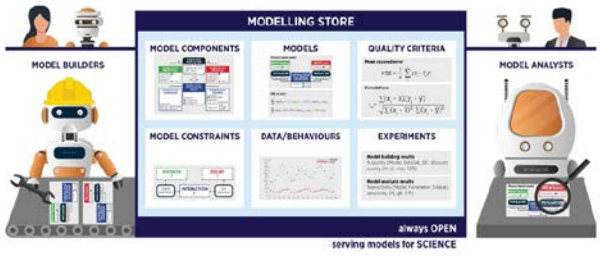
Learning grammars from corpora of equations (Chaushevska el al.)

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



Feynman database of algebraic expressions
Wikipedia's corpus of algebraic expressions
Parsing the corpora of equations using the initial PCFG
Method for Learning the Probabilities
PCFG with probabilities learned on Feynman database
PCFG with initial probabilities
PCFG with probabilities learned on Wikipedia's corpus
Generating 30000 candidate equations for each equation from the Feynman database
Generating 30000 candidate equations for each equation from the Feynman database
Generating 30000 candidate equations for each equation from the Feynman database
ED experiments by using ProGED
Parameter estimation for each candidate equation
Parameter estimation for each candidate equation
Parameter estimation for each candidate equation

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

Thank you for your attention!

Thanks also to members of my research group and external collaborators on the topics covered here

- Ljupco Todorovski
- Pat Langley, Will Bridewell
- Natasa Atanasova, Mateja Skerjanec, Matej Radinja
- Darko Cherepnalkoski, Kate Tashkova
- Nikola Simidjievski, Jovan Tanevski
- Gjorgji Peev, Marija Chaushevska
- Jure Brence, Nina Omejc, Bostjan Gec, Sebastian Mežnar