Declarative Bias in Equation Discovery

Ljupčo Todorovski
University of Ljubljana, Faculty of Medicine
Vrazov trg 2
1105 Ljubljana, Slovenia
Ljupco.Todorovski@mf.uni-lj.si

Sašo Džeroski
Jozef Stefan Institute
Jamova 39
1111 Ljubljana, Slovenia
Saso.Dzeroski@ijs.si

Abstract

Declarative bias plays an important role when learning in potentially huge hypothesis spaces. While scientific discovery systems, which perform equation discovery as a subtask, consider such potentially huge hypothesis spaces, few (if any) employ declarative (as opposed to hard-coded) bias to define and restrict their hypothesis space. We present an equation discovery system LAGRAMGE that uses grammars to define and restrict its hypothesis space. These grammars can make use of functions defined as domain specific knowledge, in addition to common mathematical operators. LAGRAMGE was successfully applied to three artificial domains, rediscovering the correct equations. It was also applied to a real-world problem, discovering equations that make sense in terms of domain knowledge and produce accurate predictions.

1 INTRODUCTION

The term bias refers to any kind of basis for choosing one generalization over another, other than strict consistency with the observed training examples [Mitchell 1980]. Bias is an unavoidable component of machine learning systems. Stronger bias leads to better performance on unseen cases. Several kinds of bias exist, including language, search and validation bias.

There is a trend in machine learning to make language bias declarative, i.e., to make the hypothesis language considered a parameter or input of the learning system. This trend has been most obvious in areas of machine learning concerned with very expressive (and therefore large) hypothesis languages, such as subsets of first-order logic. Many systems thus exist within the area of inductive logic programming (ILP) that implement and use various forms of declarative bias (for an overview see [Nédellec et al. 1996]).

Different formalisms for declarative language bias have been used, such as clause templates/schemata [Morik et al. 1993], clause sets [Bergadano and Gunetti 1995] or combinations of both [Dehaspe and DeRaedt 1995]. [Cohen 1992] uses grammars to describe the clauses in the hypothesis language of GRENDEL. Grammars have been also used to bias search in genetic programming [Whigham 1995].

Scientific discovery, which subsumes equation discovery, also deals with large hypothesis spaces. Given a fixed set of operators and variables, the number of equations that can be constructed is infinite. To restrict the size of the search space, equation discovery systems typically employ a depth limit (see e.g. [Zembowitz and Żytkow 1992], [Džeroski and Todorovski 1993]). Few, if any, use declarative bias to specify and restrict their hypothesis space.

Declarative bias can limit the search in a potentially huge space to a manageable level. Also, declarative language bias can be viewed as a way of providing domain knowledge to the learning system, so that it can adapt better to the task at hand. It is thus clearly desirable to have a declarative bias facility in a system for equation discovery, where the search space is potentially huge and the problems faced very diverse.

In the paper, we describe an equation discovery system that uses declarative bias in the form of context free grammars. LAGRAMGE takes as input a grammar and a set of data, then (heuristically) searches the equation space defined by the grammar. It can use the usual mathematical operators defined in C, and additional functions defined by the grammar at hand. LAGRAMGE was successfully applied to three artificial problems and one real-world problem, proving the utility of declarative bias in equation discovery.

The remainder of the paper is organized as follows. Section 2 defines the problem of equation discovery as addressed by LAGRAMGE. It then proceeds to describe the declarative bias formalism used and the way expressions are derived within it. Section 3 describes the algorithm itself, including the refinement operator employed, the search strategy and the search heuristics. Section 4 describes the experiments performed and Section 5 concludes with a summary and a discussion of related and further work.

2 A DECLARATIVE BIAS FOR EQUATION DISCOVERY

The problem of equation discovery is to find an equation that describes a given set of (measured) data. We use context free grammars to specify the language of equation structures that can be used.

2.1 PROBLEM DEFINITION

Given:

- context free grammar $G = (N, T, P, S)$ and
- input data $D = (V, v_d, M)$, where
  - $V = \{v_1, v_2, \ldots v_n\}$ is a set of domain variables,
  - $v_d \in V$ is dependent variable and
  - M is a set of one or more measurements. Each measurement is a table of measured values of the domain variables at successive time points:

time	v_1	v_2	$\ldots$	v_n
t_0	$v_{1,0}$	$v_{2,0}$	$\ldots$	$v_{n,0}$
t_1	$v_{1,1}$	$v_{2,1}$	$\ldots$	$v_{n,1}$
t_2	$v_{1,2}$	$v_{2,2}$	$\ldots$	$v_{n,2}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$
t_N	$v_{1,N}$	$v_{2,N}$	$\ldots$	$v_{n,N}$

Find an equation that expresses the dependent variable v_d in terms of variables from V. This equation is expected to minimize the discrepance between the measured and calculated values of the dependent variable. The equation can be:

- differential, i.e., of the form $dv_d/dt = \dot{v_d} = E$, or
- ordinary, i.e., of the form $v_d = E$.

where E is an expression that can be derived from the context free grammar G.

2.2 THE DECLARATIVE BIAS FORMALISM

The syntax of the expressions on the right-hand side of the equation is prescribed with the context free grammar $G = (N, T, P, S)$. N, T and P are the sets of nonterminals, terminals and productions, respectively, and $S \in N$ is the starting nonterminal. Productions in P are of the form $A \rightarrow \alpha$, where $A \in N$ is called the left side and $\alpha \in (N \cup T)^*$ the right side of the production. The set of all productions with nonterminal A on their left side is termed P_A.

The grammar used to describe the declarative bias for equation finding has several symbols with special meanings. The terminal *const* $\in T$ is used to denote a constant parameter in an equation that has to be fitted to the input data. The terminals v_i are used to denote variables from the input domain D. Finally, the nonterminal $v \in N$ denotes any variable from the input domain. Productions connecting this symbol to the terminals v_i are attached to v automatically, i.e., $\forall v_i \in V : v \rightarrow v_i \in P$.

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

$$
\begin{array}{rcl}
N & = & \{E, F, M, v\} \\
T & = & \{+, \mathit{const}, *, \texttt{monod}, (, ",",), N, P, Z\} \\
P & = & \left\{ \begin{array}{lcl} E & \rightarrow & \mathit{const} \mid \mathit{const} * F \\ & \mid & E + \mathit{const} * F \\ F & \rightarrow & v \mid M \mid v * M \\ M & \rightarrow & \texttt{monod}(\mathit{const}, v) \end{array} \right\} \\
S & = & E
\end{array}
$$

Table 1: Example Context Free Grammar G Used as Declarative Bias for Equation Discovery

The only restriction on the grammar G is that it has to generate expressions that are legal in the C programming language. This means that it can use all C built-in operators and functions. Additional functions, representing background knowledge about the addressed domain can be used, as long as they are defined in conjunction with the grammar. Note that the derived equations may be nonlinear in both the parameters and the system variables.

For example, consider the grammar defined for the aquatic ecosystem domain (see Section 4.1) in Table 1. The productions $v \to N \mid P \mid Z$ connecting the nonterminal v with the domain variables are automatically added to the grammar. The definition of the background knowledge function `monod` in the C programming language is also attached to the grammar.

2.3 GENERATING EXPRESSIONS FROM THE GRAMMAR

Expressions are derived from the grammar $G = (N, T, P, S)$ using derivation trees as described in Table 2.

begin with derivation tree $\mathcal{T}$ consisting of the node S
repeat
    choose a non-terminal leaf node A in $\mathcal{T}$
    choose a production $p = A \to A_1 A_2 \ldots A_l \in P_A$
    expand A with the successors $A_1, A_2, \ldots A_l$
until all leaf nodes in $\mathcal{T}$ are terminals

Table 2: Algorithm for Deriving Expressions from a Context Free Grammar

The height of a derivation tree $h(\mathcal{T})$ is the maximal length of a path from the root S to some of the leaf nodes.

Figure 1 shows derivation trees for the expressions $const * N/(const + N)$ and $const * N * N/(const + N)$, generated with the grammar from Table 1. The height of both derivation trees is $h(\mathcal{T}_a) = h(\mathcal{T}_b) = 4$.

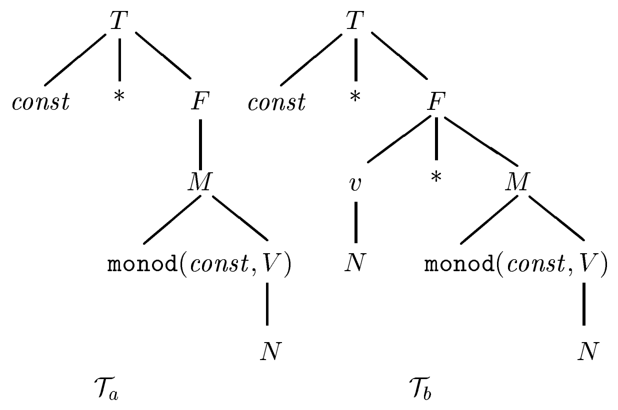

Figure 1: Derivation Trees

2.4 SEARCH SPACE COMPLEXITY

The complexity of the search space is based on the number of expressions that can be derived with the grammar $G = (N, T, P, S)$. Let $n_G(A, h)$ be the number of derivation trees in grammar G with root symbol A and height h, and $N_G(A, h)$ the number of derivation trees with root symbol A and height up to h. Then, we can use the following reccursive formulas to calculate these numbers:

- $A \in T$:

  $n_G(A, 0) = 1$
  $h \geq 1: \; n_G(A, h) = 0$

- $A \in N$:

  $n_G(A, 0) = 0$
  $n_G(A, 1) = \text{number of prods } A \to w : w \in T^*$

  $h \geq 2 : n_G(A, h) = \sum_{A \to A_1 \ldots A_l \in P_A}$
  $\left[\prod_{i=1}^{l} N_G(A_i, h-1) - \prod_{i=1}^{l} N_G(A_i, h-2)\right]$

- $N_G(A, h) = \sum_{k=0}^{h} n_G(A, k)$

The complexity of the search space is the number $N_G(S, h_{max})$ of derivation trees with the starting nonterminal symbol S in the root and height up to h_{max}.

Using the formulas above, we can compare the number of expressions that can be derived using the example grammar G from Table 1 and the number of expressions derived using the universal grammar U:

$$\begin{array}{lcl} E & \to & E + F \mid E - F \mid F \\ F & \to & F * T \mid F/T \mid T \\ T & \to & const \mid v \mid (E) \end{array}$$

Table 3 lists the numbers of equations at different heights for each of the two grammars.

h	$N_G(E, h)$	$N_U(E, h)$
1	1	0
2	1	0
3	1	7
4	121	36
5	**1831**	7300
6	27481	14674005
7	412231	$2.3607 \cdot 10^{12}$
8	6183481	$5.5481 \cdot 10^{21}$
9	92752231	$\mathbf{3.8267 \cdot 10^{38}}$
10	$1.3913 \cdot 10^{9}$	$1.2462 \cdot 10^{68}$

Table 3: Search Space Size for Two Grammars

All arithmetical expressions involving common arithmetical operators can be generated using the universal grammar. The expression for the second (most complicated) equation in the aquatic ecosystem domain from Section 4.1 can be derived with a derivation tree of height 9 using the universal grammar. On the other hand, using the example grammar from Table 1, a derivation tree of height 5 can be used to generate the same expression. Therefore, by using the grammar G as declarative bias the complexity of the search space is reduced by a factor of 10^{35}, as compared to using the universal grammar.

3 LAGRAMGE - THE ALGORITHM

3.1 REFINEMENT OPERATOR ON DERIVATION TREES

The height of a production p is the minimal height of a derivation tree with production p at its root node. The height of the production $p = A \to A_1 A_2 \dots A_l$ can be calculated as follows:

$h(p) = 1 + \max_{i=1}^{l} \{h(A_i)\}$, where

$$h(A) = \begin{cases} \min_{q \in P_A} \{h(q)\} & ; \quad A \in N \\ 0 & ; \quad A \in T \end{cases} .$$

procedure `RefinementOperator`$(\mathcal{T}, h_{max})$
  choose a nonterminal node A in $\mathcal{T}$
  let $p_{A,i}$ be the production applied at that node
  $l :=$ length of the path from A
    to the root node of $\mathcal{T}$
  delete the subtrees of node A
  if $i + 1 \leq |P_A|$ **and** $l + h(p_{A,i+1}) \leq h_{max}$ **then**
    let $p_{A,i+1} = A \to A_1 A_2 \dots A_l$
    replace $p_{A,i}$ with $p_{A,i+1}$
    expand the node A with the successors
      $A_1, A_2, \dots A_l$
    repeat
      choose non-terminal leaf node B in $\mathcal{T}$
      let $p_{B,1} = B \to B_1 B_2 \dots B_m \in P_B$
      expand the node B with the successors
        $B_1, B_2, \dots B_m$
    until all leaf nodes in $\mathcal{T}$ are terminals
  endif

Table 4: Refinement Operator

Using the height of productions we order the productions for each nonterminal A, i.e. the sets P_A, in ascending order. Furthermore, $p_{A,i}$ denotes the i-th production in the ordered set P_A.

Now we can define a refinement operator as shown in Table 4. Optionally, the height of a refinement can be limited by parameter h_{max}.

Consider the grammar in Table 1 again. The heights of the productions of the grammar are given in Table 5. Note that the derivation tree $\mathcal{T}_b$ on Figure 1 is one of the refinements of the derivation tree $\mathcal{T}_a$: the node F was chosen in the procedure `RefinementOperator`, and the production $F \to M$ was replaced with $F \to v * M$.

h	productions with height h
1	$E \to const; v \to N; v \to P; v \to Z$
2	$M \to \texttt{monod}(const, v); F \to v$
3	$F \to M; F \to v * M; E \to const * F$
	$E \to E + const * F$

Table 5: Heights of Productions in Example Grammar

3.2 TOP-LEVEL ALGORITHM

LAGRAMGE uses the beam search procedure shown in Table 6 on the search space defined by the context free grammar G. In addition to the grammar G and the domain definition D, LAGRAMGE take as input the maximal height h_{max} of the derivation trees used for deriving expressions, the beam width w_b, a fitting method $\mathcal{F}$, a heuristic function F_h and a stopping criterion S.

procedure LAGRAMGE$(D, G, h_{max}, w_b, \mathcal{F}, F_h, S)$
  let $\mathcal{T}_0$ be the shallowest derivation tree
  $Q_1 := \{\mathcal{T}_0\}$
  repeat
    $Q := Q_1$
    $R :=$ set of all refinements
      of expressions in Q
    fit the constant parameter values
      of expressions in R with $\mathcal{F}$
    evaluate expressions in R according to F_h
    $Q_1 := Q \cup R$
    retain the w_b best expressions in Q_1
  until $Q = Q_1$ **or** S

Table 6: Top-level LAGRAMGE Algorithm

Expressions generated by the context free grammar G contain one or more special terminal symbols *const*. A nonlinear fitting method $\mathcal{F}$ is applied to determine the values of these parameters. The fitting method minimizes the value of the error function $\mathit{Error}(\mathbf{c})$, i.e., if $\mathbf{c}$ is the vector of constant parameters in expression E, then the result of the fitting algorithm is a vector of parameter values $\mathbf{c}^*$, such that $\mathbf{c}^* =$

$\text{argmin}_{\mathbf{c}\in R^{dim(c)}}\{Error(\mathbf{c})\}$. The error function *Error* is a sum of squared errors function, defined in the following manner:

- for a differential equation of the form $dv_d/dt = \dot{v_d} = E$:
  $Error(\mathbf{c}) =$
  $\sum_{i=0}^{N}\left[v_{d,i} - \left(v_{d,0} + \int_{t_0}^{t_i} E(\mathbf{c}, v_1, \ldots v_n)\right)\right]^2$, and
- for an ordinary equation of the form $v_d = E$:
  $Error(\mathbf{c}) =$
  $\sum_{i=0}^{N}(v_{d,i} - E(\mathbf{c}, v_{1,i}, \ldots v_{d-1,i}, v_{d+1,i}, \ldots v_{n,i}))^2$.

The downhill simplex and Levenberg-Marquart [Press et al. 1986] algorithms are used for minimization of the error function.

After fitting, the value of the heuristic function of the expression is evaluated. It is based on the sum of squared errors value calculated *SSE* by the fitting method ($SSE(E) = Error(\mathbf{c}^*)$). An alternative heuristic function *MDL* can be used, that takes into account the length l of E:

$$MDL(E) = SSE(E) + \frac{l}{10 \cdot l_{max}}\sigma_{v_d},$$

where l_{max} is the length of the largest expression generated by the grammar with height of the derivation tree up to h_{max} and σ_{v_d} is the standard deviation of the measurements for the dependent variable v_d.

The stopping criterion S is a logical condition which is either always false or is equal to $F_h(E) \le T$, where E is the expression with the best F_h value derived so far and T is a user defined treshold.

4 EXPERIMENTS

LAGRAMGE was applied to three synthetic and one real-world problem of equation discovery. The four problems were chosen to illustrate the capabilities of LAGRAMGE. They all concern the behavior of dynamic systems (differential equations were thus sought by LAGRAMGE) and involve nonlinearities. These can be nonlinearities in the system variables or parameters, or nonlinearities arising from the use of logical conditions in the model of the dynamic system. The two poles on cart is a complex system that was out of reach (complexity-wise) of existing machine discovery systems and the phytoplankton growth domain is a real domain, which is not too complex judged by the equations structure, but was not successfuly modeled by existing discovery systems. Two of the domains involve background knowledge in the form of predefined functions that can be used in the equations.

For the synthetic domains, the original equations were taken and simulated for ten randomly chosen initial states for a certain number of time steps. The generated data were then given to LAGRAMGE. For all problems, the data were accompanied with an appropriate bias, i.e., grammar, for the problem at hand. Derivation tree height was limited, the limit varying between 4 and 6.

Beam search with beam width 25 was used in conjunction with the downhill simplex method for nonlinear parameter optimization. The sum of squared errors was used as the heuristic function. Both the alternative of using a stopping criterion and the alternative of not using one were explored. The stopping criterion employed had the treshold value $T = 10^{-5}$.

For each experiment, the grammar used is listed, together with the number of expressions at each derivation height, up to the maximum allowed height. For each dependent variable, we record the number of expressions visited during the search. For the synthetic domains, we indicate whether the correct equation was discovered.

4.1 AQUATIC ECOSYSTEM

A simple system of differential equations describes the evolution of the concentrations of nutrient N, phytoplankton P and zooplankton Z in an aquatic environment:

$$\begin{aligned}
\dot{N} &= -\frac{NP}{k_N + N} \\
\dot{P} &= \frac{NP}{k_N + N} - r_P P - \frac{PZ}{k_P + P} \\
\dot{Z} &= \frac{PZ}{k_P + P} - r_Z Z.
\end{aligned}$$

It was simulated with the parameters $k_N = k_P = 1$, $r_P = r_Z = 0.05$, from ten randomly selected initial states, for 100 time steps of 0.01.

The grammar from Table 1 was used. Note that the `monod` function is defined, which represents background knowledge about population growth that comes from the area of ecological modeling. The number of derivation trees $N_G(E, h)$ with height below or equal to h is given in the following table:

h	1	2	3	4	5
$N_G(E,h)$	1	1	7	121	1831

With no stopping criterion applied, LAGRAMGE found the three correct equations, considering 475 expressions for $\dot{N}$, 532 for $\dot{P}$, and 518 for $\dot{Z}$. With the stopping criterion applied, LAGRAMGE found the correct equation for $\dot{N}$ after considering 20 expressions, and equations similar to but simpler than the original ones after considering 355 expressions for $\dot{P}$, and 58 for $\dot{Z}$.

4.2 PIECEWISE-LINEAR CIRCUIT

A simple piecewise-linear circuit wich exhibits a chaotic attractor similar to that of the Lorenz equations can be described by the following three state equations:

$$\begin{aligned}
\dot{x} &= \sigma(y-x) \\
\dot{y} &= \mu_1(r-z)-qy \\
\dot{z} &= \mu_2 x - bz
\end{aligned}$$

where $\mu_1 = \text{sgn}\ x$, $\mu_2 = \text{sgn}\ y$.

We simulated the equations for 100 time steps of $h = 0.1$ from ten randomly chosen initial states. The parameter values were $\sigma = 1.2$, $r = 7$, $q = 0.1$ and $b = 0.1$.

The following grammar was used:

```
double If(double v, double e1, double e2) {
return((v > 0) ? e1 : e2);
}
```

$$\begin{aligned}
\mathit{IfE} &\rightarrow E \mid \mathtt{If}(v, E, E) \\
E &\rightarrow \mathit{const} \mid \mathit{const} * v \mid E + \mathit{const} * v
\end{aligned}$$

h	1	2	3	4
$N_G(\mathit{IfE},h)$	0	4	154	1900

Note that the `If` function is defined as background knowledge. This example illustrates the possibility to use logical conditions in the grammars and equations.

LAGRAMGE found the three correct equations both with and without applying the stopping criterion, visiting 510 expressions for $\dot{x}$, 498 for $\dot{y}$ and 603 for $\dot{z}$ in both cases.

4.3 TWO POLES ON CART

The task of balancing the inverted pendulum (pole on cart) is a standard benchmark problem for testing classical and modern approaches to the control of dynamical systems [Geva and Sitte 1993], [Urbančič and Bratko 1994]. The inverted pendulum consists of a cart that can move along a horizontal track, and a pole hinged on top of the cart, so that it can rotate in the vertical plane defined by the track and its fixed point. A force parallel to the track can be applied to the cart. The inverted pendulum is the most complicated system whose dynamics were successfully modeled by LAGRANGE [Džeroski and Todorovski 1993].

The two poles on cart system complicates the inverted pendulum by hinging a second pole on top of the first. The top pole can rotate in the vertical plane around the point where it is attached to the bottom pole. The system variables in this case are x, the distance of the cart from the origin point on the track, φ, the inclination angle of the bottom pole relative to the vertical line through its fixed point, and ϑ, the inclination angle of the top pole. The dynamics of this system can be described by a system of second order differential equations.

$$\begin{gathered}
2(M+m_1+m_2)\ \ddot{x} + l_1(m_1+2m_2)\ \ddot{\varphi}\cos\varphi + \\
m_2 l_2\ \ddot{\vartheta}\cos\vartheta = \\
2F + (m_1+2m_2)l_1\ \dot{\varphi}^2\sin\varphi + m_2 l_2\ \dot{\vartheta}^2\sin\vartheta
\end{gathered}$$

$$\begin{gathered}
(m_1+2m_2)\ \ddot{x}\cos\varphi + 2l_1(m_1/3+m_2)\ \ddot{\varphi} + \\
m_2 l_2\ \ddot{\vartheta}\cos(\vartheta-\varphi) = \\
m_2 l_2\ \dot{\vartheta}^2\sin(\vartheta-\varphi) + (m_1+2m_2)g\ \sin\varphi
\end{gathered}$$

$$\begin{gathered}
\ddot{x}\cos\vartheta + l_1\ \ddot{\varphi}\cos(\vartheta-\varphi) + 2/3\ \ddot{\vartheta} = \\
g\ \sin\vartheta - l_1\ \dot{\varphi}^2\sin(\vartheta-\varphi)
\end{gathered}$$

In the equations, M, m_1 and m_2 are the masses of the cart, the bottom, and the top pole, respectively, l_1 and l_2 the lengths of the poles, F the force applied to the cart, and $g = 9.81$ is the gravitational acceleration.

Ten simulations runs were performed, each of 50 time steps of $h = 0.01$, starting from a randomly chosen initial state. No control force was applied, i.e., $F = 0$. The parameters' values were chosen as follows: $M = 2[kg]$, $m_1 = 1[kg]$, $m_2 = 0.5[kg]$, $l_1 = 1[m]$, $l_2 = 0.5[m]$.

The grammar below was used for $\ddot{x}$. The grammar for $\ddot{\varphi}$ replaces the production for $Facc_1$ with $Facc_1 \rightarrow \ddot{x} * \cos A \mid \ddot{x}$, while the grammar for $\ddot{\vartheta}$ replaces the production for $Facc_2$ with $Facc_2 \rightarrow \ddot{x} * \cos A \mid \ddot{x}$.

$$
\begin{array}{lcl}
E & \rightarrow & Eacc + Evel \\
Eacc & \rightarrow & const * Facc_1 + const * Facc_2 \\
Facc_1 & \rightarrow & \ddot{\varphi} * \cos A \mid \ddot{\varphi} \\
Facc_2 & \rightarrow & \ddot{\vartheta} * \cos A \mid \ddot{\vartheta} \\
Evel & \rightarrow & Evel + const * Fvel \mid const * Fvel \\
Fvel & \rightarrow & Vvel_2 * \sin A \mid Vvel_2 \mid \sin A \\
Vvel_2 & \rightarrow & \dot{\varphi} * \dot{\varphi} \mid \dot{\vartheta} * \dot{\vartheta} \\
A & \rightarrow & \varphi \mid \vartheta \mid \varphi - \vartheta
\end{array}
$$

h	1	2	3	4	5	6
$N_G(E,h)$	0	0	0	0	2	528

While the above grammar looks complicated, it can be constructed easily with a modest knowledge of physics, considering the fact that the target equations are equations of Lagrangian dynamics.

The target equations were found both with and without applying the stopping criterion, visiting 292 expressions for $\ddot{x}$, 283 for $\ddot{\varphi}$ and 350 for $\ddot{\vartheta}$ in both cases. LAGRAMGE thus successfully identified the dynamics of a system much more complicated than systems considered so far in equation discovery, due to the use of declarative bias in the form of a grammar.

4.4 PHYTOPLANKTON GROWTH IN LAKE GLUMSOE

The final problem we consider is a real-world problem of modeling the dynamics of phytoplankton growth in the Danish lake Glumsoe [Jorgensen et al. 1986]. The system variables considered are the concentrations of phytoplankton *phyt*, zooplankton *zoo*, as well as the concentrations of nitrogen NS and phosphorus PS.

The grammar used was constructed based on background knowledge on algal growth. Phosphorus and nitrogen are nutrients for phytoplankton and can thus appear in Monod terms. Other terms model the decay of phytoplankton and the feeding of zooplankton on phytoplankton. There are some constraints on the values of the parameters in the equations specified by the grammar: they have to be positive, except for the ones in front of the decay term and the term describing the feeding of zooplankton on phytoplankton, which should be negative.

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

$$
\begin{array}{lcl}
E & \rightarrow & F * phyt + const * phyt \\
 & \mid & F * phyt + const * phyt + \\
 & & const * V\_phyt * zoo \\
F & \rightarrow & const * M * M \mid const * M + const * M \\
 & \mid & const * M \\
M & \rightarrow & \mathtt{monod}(const, VM) \\
VM & \rightarrow & PS \mid NS
\end{array}
$$

h	1	2	3	4	5
$N_G(E,h)$	0	0	0	0	72

We used one set of measured data, smoothed in different ways by three human experts, giving three experimental data sets. The discovered differential equations were used to forecast the concentration of phytoplankton (*phyt*) one step ahead. We used a 'leave one out' testing method: LAGRAMGE was given two sets of data for equation discovery, and the best equation discovered was then tested on the remaining data set.

In experiments with the MDL heuristic (all possible 72 equations were considered) and the best equation discovered by LAGRAMGE was chosen that satisfied the constraints for the parameter values. The three equations obtained have the following form: $\dot{phyt} = const_1 * phyt * PS/(const_2 + PS) + const_3 * phyt$

The constant parameter values, as well as the correlation between the measured and forecasted values of phytoplankton (on the testing set) for each of the three equations are shown below:

Training sets	$const_1$	$const_2$	$const_3$	R
1, 2	0.6168	0.1014	-0.4422	0.9994
1, 3	0.7629	0.0797	-0.5916	0.9989
2, 3	0.3831	0.4444	-0.1553	0.9996

The equations discovered make sense from an ecological point of view. They tell us that phosphorus is a limiting factor for phytoplankton growth in the lake. They also give accurate short term predictions for phytoplankton growth.

5 DISCUSSION

Declarative language bias is a way of providing domain knowledge to the learning system. In this way, the learning system can adapt better to the task at hand. We have presented an equation (machine) discovery system LAGRAMGE that uses declarative bias. Background knowledge in the form of function definitions can also be used, including logical conditions / predicates. LAGRAMGE (heuristically) searches the space of equation structures defined by a grammar, which is further ordered by a refinement operator.

The equations considered can be nonlinear in both the system variables and the parameters. LAGRAMGE thus uses nonlinear optimization procedures, such as downhill simplex and Levenberg-Marquardt, to fit equation parameters. The search heuristic used is based on the fit of equations to measured data, but can also take into account the length of equations (MDL). LAGRAMGE can find both ordinary and differential equations, in both implicit and explicit form. It can also take into account more than one behavior of a dynamic system.

On the declarative bias side, LAGRAMGE is related to learning systems in the area of inductive logic programming (ILP), which often use strong declarative bias [Nédellec et al. 1996]. Such systems include, for example, GRENDEL [Cohen 1992], MOBAL [Morik et al. 1993], TRACY [Bergadano and Gunetti 1995], and CLAUDIEN [Dehaspe and DeRaedt 1995]. ILP systems, however, rarely deal with numerical data by producing equations, with the notable exception of FORS [Karalič and Bratko 1997] system for relational regression. Related is also the genetic programming system of [Whigham 1995], which uses grammars to bias its search, but does not deal with the problem of equation discovery.

On the equation discovery side, LAGRAMGE is related to machine discovery systems. These include BACON [Langley et al. 1987], EF [Zembowitz and Żytkow 1992], E* [Schaffer 1993], LAGRANGE [Džeroski and Todorovski 1993] and GOLDHORN [Križman et. al 1995]. Except for LAGRANGE and GOLDHORN, none of the mentioned systems can deal with differential equations. All the mentioned systems include some form of bias: the particular biases used are however either hard-coded or parametrized and thus much less flexible that the declarative bias facility of LAGRAMGE. For example, E* chooses among several (six) built-in simple equation templates and is limited to equations involving two input variables. EF provides a declarative bias facility for specifying the set of built-in functions for introducing new terms, but these can only be combined by polynomial regression, producing equations that are linear in their parameters. The depth of new terms and the degree of polynomials considered in regression are specified as parameters. The depth of new terms which can be introduced by multiplication is a parameter in LAGRANGE, as is the number of additive terms involved in an equation. GOLDHORN considers the same equation space as LAGRANGE, that is the space of multivariate polynomials on the system variables.

On the experimental side, LAGRAMGE reconstructed from simulated data three models of dynamic systems, that are out of reach for existing machine discovery systems, e.g. LAGRANGE [Džeroski and Todorovski 1993]. The first includes piece-wise linear equations, i.e., logical conditions. In the second domain, LAGRAMGE makes use of background knowledge defining a building block useful in ecological modeling, stemming from the Monod expression. The pole on cart domain was the most complicated domain still within reach of LAGRANGE. The two poles on cart domain, however, is well out of its reach, due to the size of the search space. The declarative bias facility of LAGRAMGE enabled us to specify a grammar based on some physics knowledge about Lagrangian dynamics, which greatly reduced the equation space. Finally, LAGRAMGE was applied to the practical domain of predicting phytoplankton growth in the Danish lake Glumsoe. While GOLDHORN [Križman et. al 1995], a successor of LAGRANGE, failed to produce satisfactory equations in this domain, LAGRAMGE generated sensible equations that make use of domain knowledge (the Monod block) and produce accurate predictions.

Promising directions for further work concern mostly background knowledge and declarative bias acquisition. This might include the creation of domain specific grammar libraries, e.g., for the domain of ecological modeling, as well as the use of intermediate concepts constructed by decomposition of real functions [Demsar et al. 1997]. The possibility to constrain parameter values according to domain knowledge seems also worth to explore. Such constraints could be a part of the declarative bias.

Acknowledgements

This work was supported in part by the Slovenian Ministry of science and technology and in part by the European Union through the ESPRIT IV Project 20237 Inductive Logic Programming 2.

References

[Bergadano and Gunetti 1995]
Bergadano, F., and Gunetti, D. (1995) *Indictive logic programming: from Machine Learning to Software Engineering.* MIT Press, Cambridge, MA.

[Cohen 1992]
Cohen, W. (1992) Compiling prior knowledge into an explicit bias. In *Proc. Ninth International Conference on Machine Learning*, pages 102–110. Morgan Kaufmann, San Mateo, CA.

[Dehaspe and DeRaedt 1995]
Dehaspe, L., and De Raedt, L. (1995) A declarative language bias for concept-learning algorithms. *The Knowledge Engineering Review*, 7(3): 251–269.

[Demsar et al. 1997]
Demšar, J., Zupan, B., Bohanec, M., and Bratko, I. (1997) Constructing intermediate concepts by decomposition of real functions. In *Proc. Ninth European Conference on Machine Learning.* Springer, Berlin.

[Džeroski and Todorovski 1993]
Džeroski, S., and Todorovski, L. (1993) Discovering dynamics In *Proc. Tenth International Conference on Machine Learning*, pages 97–103. Morgan Kaufmann, San Mateo, CA.

[Geva and Sitte 1993]
Geva, S., and Sitte, J. (1993) A cartpole experimental benchmark for trainable controllers. *IEEE Control Systems*, 13(5): 40–51.

[Jorgensen et al. 1986]
Jorgensen, S., Kamp-Nielsen, L., Chirstensen, T., Windolf-Nielsen, J., and Westergaard, B. (1986) Validation of a prognosis based upon a eutrophication model. *Ecological modelling*, 32: 165–182.

[Karalič and Bratko 1997]
Karalič, A., and Bratko, I. (1997) First order regression. *Machine Learning*, 26: 147–176.

[Križman et. al 1995]
Križman, V., Džeroski, S., and Komare, B. (1995) Discovering dynamics from measured data. *Electrotechnical Review*, 62: 191–198.

[Langley et al. 1987]
Langley, P., Simon, H., and Bradshaw, G. (1987) Heuristics for empirical discovery. In Bolc, L., editor, *Computational Models of Learning.* Springer, Berlin.

[Mitchell 1980]
Mitchell, T. (1980) The need for biases in learning generalizations. Technical Report TR-117, Department of Computer Science, Rutgers University.

[Morik et al. 1993]
Morik, K., Wrobel, S., Kietz, J.-U., and Emde, W. (1993) *Knowledge Acquisition and Machine Learning: Theory, Methods and Applications.* Academic Press.

[Nédellec et al. 1996]
Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F., and Tausend, B. (1996) Declarative bias in ILP. In De Raedt, L., editor, *Advances in Inductive Logic Programming*, pages 82–103. IOS Press, Amsterdam.

[Press et al. 1986]
Press, W. H., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T. (1986) *Numerical Recipes.* Cambridge University Press, Cambridge, MA.

[Schaffer 1993]
Schaffer, C. (1993) Bivariate scientific function finding in a sampled, real-data testbed. *Machine Learning*, 12: 167–183.

[Urbančič and Bratko 1994]
Urbančič, T., and Bratko, I. (1994) Learning to control dynamic systems. In Michie, D., Spiegelhalter, D., and Taylor, C., editors, *Machine Learning, Neural and Statistical Classification.* Ellis Horwood, Chichester.

[Whigham 1995]
Whigham, P. (1995) Grammatically-based genetic programming. In *Proc. ML'95 Workshop on Genetic Programming: From Theory to Real-World Applications*, pages 33–41. Lake Tahoe, CA.

[Zembowitz and Żytkow 1992]
Zembowitz, R., and Żytkow, J. (1992). Discovery of equations: experimental evaluation of convergence. In *Proc. Tenth National Conference on Artificial Intelligence.* Morgan Kaufmann, San Mateo, CA.