Integrating Domain Knowledge in Equation Discovery

Ljupčo Todorovski and Sašo Džeroski

Department of Knowledge Technologies
Jožef Stefan Institute, Ljubljana, Slovenia
Ljupco.Todorovski@ijs.si, Saso.Dzeroski@ijs.si

Abstract. In this chapter, we focus on the equation discovery task, i.e., the task of inducing models based on algebraic and ordinary differential equations from measured and observed data. We propose a methodology for integrating domain knowledge in the process of equation discovery. The proposed methodology transforms the available domain knowledge to a grammar specifying the space of candidate equation-based models. We show here how various aspects of knowledge about modeling dynamic systems in a particular domain of interest can be transformed to grammars. Thereafter, the equation discovery method LAGRAMGE can search through the space of models specified by the grammar and find ones that fit measured data well. We illustrate the utility of the proposed methodology on three modeling tasks from the domain of Environmental sciences. All three tasks involve establishing models of real-world systems from noisy measurement data.

1 Introduction

Scientists and engineers build mathematical models to analyze and better understand the behavior of real-world systems. Establishing an acceptable model for an observed system is a very difficult task that occupies a major portion of the work of the mathematical modeler. It involves observations and measurements of the system behavior under various conditions, selecting a set of variables that are important for modeling, and formulating the model itself. This chapter addresses the task of automated modeling, i.e., the task of formulating a model from the observed behavior of the selected system variables.

There are at least three properties of mathematical models that make them an omnipresent analytic tool. The first is the *data integration* aspect — mathematical models are able to integrate large collections of observations and measurements into a single entity. Second, models allow for simulation and *prediction* of the future behavior of the observed system under varying conditions. Finally, mathematical models can provide *explanations*, i.e., reveal the structure of processes and phenomena that govern the behavior of the observed system.

Although the aforementioned properties of models are equally important to scientists and engineers, most of the current automated modeling approaches based on machine learning and data mining methods are mainly concerned with

S. Džeroski and L. Todorovski (Eds.): Computational Discovery, LNAI 4660, pp. 69–97, 2007.

the first two: data integration and accurate prediction of future behavior. In this chapter, we focus on the third, explanatory aspect of induced models. We address two issues that are related to the explanatory aspect. The first is the issue of modeling notations and formalisms used by scientists and engineers. While machine learning and data mining methods focus on inducing models based on notations introduced by researchers in these areas (such as decision trees and Bayesian networks), we induce models based on algebraic and differential equations, i.e., standard notations, already established in many scientific and engineering domains. The second is the issue of integrating knowledge from the specific domain of interest in the induction process. This is in analogy with the manual modeling process — to obtain an explanatory *white-box* model of an observed system, scientists and engineers rely on and make use of this knowledge.

We present a methodology for integrating domain-specific knowledge in the process of model induction. The methodology allows integration of several different types of knowledge. The first type includes knowledge about basic processes that govern the behavior of systems in the particular domain and the equation templates that are typically used by domain experts for modeling these processes. The second type includes knowledge about models already established and used in the domain. The method can take into account a fully specified initial model, or a partially specified model of the observed system. We relate the integration of knowledge in the induction process to the notion of inductive language bias, which defines the space of candidate model structures considered by the induction method. We show that the three different types of domain-specific knowledge mentioned above can be transformed to inductive language bias, i.e., a specification of the space of models. We illustrate the utility of the presented methodology on three practical tasks of establishing models of real-world systems from measurement data in the domain of Environmental sciences.

The chapter is organized as follows. In Section 2, we start with a discussion of the relation between domain-specific knowledge and language bias in induction methods. In the following Section 3, we introduce the equation discovery task, i.e., the task of inducing equation-based models and propose methods for specifying language bias for equation discovery. Section 4 illustrates how to use the knowledge about basic processes that govern the behavior of population dynamics systems for establishing explanatory models thereof. In Section 5, we focus on the integration of (full or partial) initial model specifications in the model induction process. Section 6 discusses related research. Finally, in Section 7 we summarize and propose directions for further research.

2 Domain-Specific Knowledge and Language Bias

Many studies of machine learning methods and especially their applications to real-world problems show the importance of background knowledge for the quality of the induced models. Pazzani and Kibler (1992) show that the use of domain-specific knowledge improves the predictive performance of induced models on test examples unseen in the induction phase. Domain knowledge is

also important for the acceptance of the induced concepts by human experts. As Pazzani et al. (2001) show, the consistency of the induced concepts with existing knowledge in the domain of interest is an important factor that influences the opinion of human experts about the credibility of the concepts and their acceptability for further use. This is especially true in complex scientific and engineering domains, where a vast amount of knowledge is being systematically collected and well documented.

Although these results are well known, most state-of-the-art machine learning methods do not allow for *explicit* integration of domain knowledge in the induction process. Knowledge is usually *implicitly* involved in the phases that precede or follow the induction process, that is the data preparation or preprocessing phase, when the set of variables important for modeling of the observed phenomena are identified, or in the model interpretation phase. A notable exception are learning methods developed within the area of inductive logic programming (ILP) (Lavrač & Džeroski, 1994). The notion of (background) knowledge and its integration in the induction process is made explicit there and background knowledge is a part of the learning task specification. ILP methods deal with the induction of first-order logic programs (or theories) from examples and domain knowledge. Background knowledge predicates allow user to integrate knowledge from the domain under study in ILP. They define concepts (or building blocks) that may be used in the induced theories. Note, however, that background knowledge predicates specify building blocks only and do not specify how to combine them into proper programs or theories.

Another way to integrate domain knowledge in the induction process is through the use of inductive bias, which refers to any kind of mechanism or preference used by the induction algorithm to choose among candidate hypotheses (Utgoff, 1986; Mitchell, 1991). The preference usually goes beyond the mere consistency (or degree-of-fit) of the hypothesis with the training examples and determines in which region of the candidate hypotheses space we are more likely to find a solution. Thus, the term bias is related to domain knowledge, since knowledge can help constrain the space of candidate hypotheses. Nédellec et al. (1996) define three types of inductive bias. The first type, language bias, is used to specify the space of hypotheses considered in the induction process. In our case, language bias would specify the space of candidate equation-based models. Note that language bias does not only specify a set of building blocks for establishing models (as background knowledge predicates do), but it also specifies recipes for combining them into proper candidate models (or hypotheses). The second type is search bias, which specifies the order in which the hypotheses are considered during the induction process. The third type is validation bias that specifies the acceptance or stopping criteria for the induction process.

Depending on the kind of formalism available for specifying the bias for an induction method, it may be non-declarative (built-in), parametrized, or declarative (Nédellec et al., 1996). Methods for inducing decision trees (Quinlan, 1993) are typical examples of methods with non-declarative or parametrized language bias, since they explore the fixed hypothesis space of decision trees using

variables from the given data set. Parametrized bias lets the user influence the set of candidate hypotheses by setting some of its parameters, such as the depth (or complexity) of the induced decision trees. On the other hand, declarative language bias provides the user with complete control over the space of candidate hypotheses. Thus, declarative language bias provides a powerful mechanism for integrating domain knowledge in the process of induction.

Different formalisms can be used for specifying the space of candidate hypotheses. Nédellec et al. (1996) provide an overview of declarative bias formalisms used in ILP. Note, however, that these formalisms are developed for concepts and hypotheses expressed in first-order logic and are not directly applicable to the task of building equation-based models. In the next section, we propose a declarative bias formalism for equation discovery that will allow for integrating different aspects of domain-specific modeling knowledge in the process of inducing models from data.

3 Equation Discovery and Language Bias

Before we propose an appropriate declarative bias formalism for equation-based models, we define the task of inducing equations from data. The task is usually referred to as the task of equation discovery. Equation discovery is the area of machine learning that develops methods for automated discovery of quantitative laws, expressed in the form of equations, in collections of measured data (Langley et al., 1987; Langley & Żytkow, 1989; Džeroski & Todorovski, 1995; Washio & Motoda, 1997). In the rest of this section, we then provide an overview of typical language biases used in existing equation discovery methods, propose a declarative bias formalism for equation discovery, and provide an example bias based on domain knowledge for modeling population dynamics.

3.1 Task Definition

The equation discovery task can be formalized as follow. Given:

- a set of (numeric) variables V that are important for modeling the observed system, where $V = S \cup E$, $S = \{s_1, s_2, \ldots s_n\}$ is a set of *system* (or dependent) variables that we want to explain (or be able to predict) with the induced model, while $E = \{e_1, e_2, \ldots e_m\}$ is a set of *exogenous* (or input/output) variables that are not explained by the induced model, but may be used in the model equations;
- a designated time variable t, in cases when we deal with a task of modeling a dynamic system;
- one or more tables with measured/observed values of the variables from V (in consecutive time points or independent experiments),

find a set of n equations, one for each variable in S, that can explain and accurately predict the values of the system variables and take one of the two forms:

- differential $\frac{d}{dt}s_i = f_i(s_1, s_2, \ldots s_n, e_1, e_2, \ldots e_m)$ or
- algebraic $s_i = f_i(s_1, s_2, \ldots s_{i-1}, s_{i+1}, \ldots s_n, e_1, e_2, \ldots e_m)$.

Table 1. Two example context free grammars that specify distinct classes of arithmetical expressions. The one on the left-hand side is the "universal" grammar that specifies the space of all arithmetical expressions that can be built using the four basic arithmetical operators (and parentheses). The grammar on the right-hand side specifies the space of polynomials.

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

3.2 A Language Bias Formalism for Equation Discovery

Once provided with the task definition, we can identify the language bias of an equation discovery method as the space or class of candidate arithmetical expressions f_i that can appear on the right-hand sides of the induced model equations. Thus, the declarative language bias formalism for equation discovery should allow us to specify classes of arithmetical expressions. We argue here that context free grammars (Hopcroft & Ullman, 1979) provide an ideal framework for this purpose. Chomsky (1956) introduced context free grammars as formal models of natural or synthetic languages and since then they have became a standard formalism for specifying the syntax of programming languages.

Table 1 provides two example grammars that specify two distinct classes of arithmetical expressions. The grammar on the left-hand side of the table specifies the space of all arithmetical expressions that can be built using the four basic arithmetical operators and parentheses. Formally, a context free grammar consists of a finite set of variables, usually referred to as *nonterminals* or syntactic categories, each of them representing a subclass of subexpressions or phrases in the language represented by the grammar.

In the particular example, the nonterminals E, F, T, and V refer to arithmetical expressions, factors, terms, and variables, respectively. The class of factors F represents arithmetical expressions that are built using multiplication and division only[1]. Terms represent atomic arithmetical expressions that are further combined into factors. A term T can be either (1) an arbitrary expression E in parentheses, (2) a variable V, or (3) a constant parameter. We specify these three classes of terms using three grammar *productions* (also named rewrite rules). The first rule is formalized as $T \rightarrow (E)$, the second as $T \rightarrow V$, and the third as $T \rightarrow const$. In short, this is written as $T \rightarrow (E) \mid V \mid const$, where the symbol "|" separates alternative productions for the same nonterminal.

The symbol *const* is a *terminal* grammar symbol that represents a constant parameter in the arithmetical expression with an arbitrary value, which is usually fitted against training data in the process of equation discovery. The terminals are primitive grammar symbols that can not be further rewritten, i.e., no

[1] Note that the organization of the grammar follows the relative priority of the basic arithmetic operators—multiplication and division are always performed before addition and subtraction.

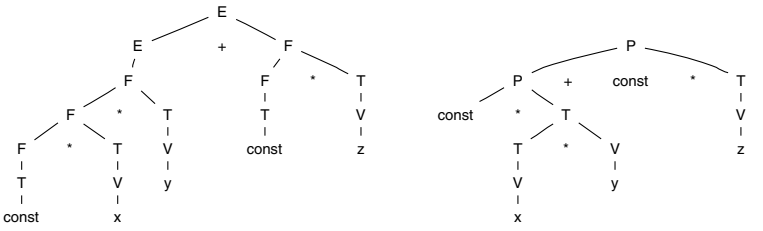

Fig. 1. Two parse trees that derive the same polynomial expression $const * x * y + const * z$. The one on the left-hand side corresponds to the universal grammar, while the parse tree on the right-hand side corresponds to the polynomial grammar from Table 1.

productions are affiliated with them. Other terminals in the example grammar are "+", "-", "*", "/", "(", and ")".

Similarly, the grammar on the right-hand side of Table 1 specifies the space of polynomials over the variables in V. Productions for the nonterminal P specify that each polynomial is a linear combination of one or more terms T multiplied by constants. Each term, in turn, is a multiplication of one or more variables from V. Note that the productions for the nonterminal V depend on the particular equation discovery task at hand. Typically, a single production $V \rightarrow v$ will be added to the grammar for each observed/measured variable v. For example, if we deal with an equation discovery task where we measure three variables x, y, and z, the corresponding productions will be $V \rightarrow x \mid y \mid z$.

Given a context free grammar, we can check whether a certain expression (string) belongs to the language defined by the grammar (parse task) or generate expressions that belong to the language (generate task). For both purposes, we use the notion of a parse tree, which describes the way a certain expression can be derived using the grammar productions. Figure 1 presents two example parse trees that derive the same (polynomial) expression $const * x * y + const * z$. The left-hand tree presents the derivation of this expression using the universal grammar, while the one on the right-hand side of the figure presents its derivation using the polynomial grammar. Each internal node in a parse tree corresponds to a nonterminal symbol N from the grammar. Its children nodes $n_1, n_2, \ldots n_k$, ordered from left to right, always correspond to a grammar production $N \rightarrow n_1\ n_2 \ldots n_k$. The leaf nodes of a parse tree are terminal symbols: reading them from left to right, we get the expression derived with the parse tree.

Most equation discovery methods would adopt one of the language biases based on the grammars from Table 1. While the BACON (Langley et al., 1987) and SDS (Washio & Motoda, 1997) methods for equation discovery can induce equations based on arbitrary arithmetical expressions (i.e., use the universal grammar as a built-in bias), FAHRENHEIT (Zembowicz & Żytkow, 1992) and LAGRANGE (Džeroski & Todorovski, 1995) limit their scope to inducing polynomial equations. Note that the language biases of the aforementioned equation discovery methods are usually non-declarative and often take the form of a

relatively small (pre-defined or built-in) class of possible equations. The user is allowed to influence the built-in bias using a number of parameters. LAGRANGE would let user specify the maximal degree of polynomials or maximal number of multiplicative terms in the polynomial (Džeroski & Todorovski, 1995). Additionally, Zembowicz and Żytkow (1992) let the user specify functions that can be used to transform the variables before they are incorporated in the polynomials.

Another type of language constraints used for equation discovery is based on information about the measurement units of the observed system variables. COPER (Kokar, 1986) uses dimensional analysis theory (Giordano et al., 1997) in order to constrain the space of equations to those that properly combine variables and terms taking care about the compatibility of their measurement units. The SDS method (Washio & Motoda, 1997) extends this approach to cases in which the exact measurement units of the system variables are not known. In such cases, SDS employs knowledge about the type of the measurement scale for each system variable, which is combined with knowledge from measurement theory to constrain the space of possible equations.

Note, however, that knowledge about measurement units or the scale types thereof is domain independent. Experts from a specific domain of interest can usually provide much more modeling knowledge about the system at hand than merely enumerating the measurement units of the system variables. Many textbooks on mathematical modeling give comprehensive overviews of the modeling knowledge for specific domains, such as biology (Murray, 1993) or biochemistry (Voit, 2000). In the next section, we will show how this kind of knowledge can be transformed to a grammar that provides an appropriate knowledge-based bias for equation discovery in the domain of population dynamics.

3.3 An Example from Modeling Population Dynamics

The domain of population dynamics falls within the field of population ecology, which studies the structure and dynamics of populations. A population is a group of individuals of the same species that inhabit a common environment. More specifically, we consider modeling the dynamics of populations, especially how their concentrations change through time (Murray, 1993). Here, we focus on population dynamics in aquatic systems, where we are mainly concerned with interactions between dissolved inorganic nutrients (e.g., nitrogen and phosphorus), primary producers (or plants, such as phytoplankton and algae), and secondary producers (or animals, such as zooplankton and fish).

Models of population dynamics are based on the mass conservation principle, where influences of different interactions and processes on a single variable are summed up. Following that principle, the change of a primary producer population can be modeled as:

$$\frac{dPP}{dt} = growth(PP) - grazing(SP, PP) - mortality(PP), \tag{1}$$

where the *growth* and *mortality* are expressions modeling the influence of growth and mortality processes on the concentration of the primary producer PP, while

Table 2. An example language bias (context free grammar) for equation discovery based on knowledge about modeling population dynamics in aquatic ecosystems

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

$$
\begin{aligned}
PPChange &\rightarrow PPGrowth - Grazing - PPMortality \\
\\
PPGrowth &\rightarrow const * V_{PP} \mid const * Limitation * V_{PP} \\
Limitation &\rightarrow V_{Nutrient} \mid \texttt{monod}(V_{Nutrient}, const) \\
\\
Grazing &\rightarrow const * V_{PP} * V_{SP} \\
\\
PPMortality &\rightarrow const * V_{PP} \\
\\
V_{PP} &\rightarrow phytoplankton \\
V_{SP} &\rightarrow zooplankton \\
V_{Nutrient} &\rightarrow phosphorus \mid nitrogen
\end{aligned}
$$

the *grazing* expression models the influence of secondary producer (zooplankton) grazing on *PP*. Note that the growth process positively influences the change of primary producer concentration, while mortality and grazing influence it negatively. Murray (1993) enumerates different expressions typically used by ecologists to model these processes. The grammar presented in Table 2 captures this kind of knowledge about modeling population dynamics.

The first production in the grammar follows the mass conservation equation template (1) and introduces three nonterminals *PPGrowth*, *Grazing*, and *PPMortality*, one for each of the processes of growth, grazing, and mortality. Productions for these three nonterminals, further in the grammar, define alternative models for these processes. The productions for the growth process define unlimited and limited growth. The first model assumes unlimited exponential growth of the primary producer population in the absence of interactions with secondary producers. The second model corresponds to the more realistic situation where the growth of the primary producer population is limited by the supply of nutrients. The nonterminal *Limitation* defines two alternative limitation terms. The first assumes unlimited capacity of the primary producer for nutrient absorption. The second (Monod) term is used when the absorption capacity of the primary producer saturates to a certain value, no matter how plentiful the nutrient supply is. Ecologists use different expressions for modeling saturation of the absorption capacity — one possibility is Monod (also known as Michaelis-Menten's) term defined as:

$$\texttt{monod}(V_{Nutrient}, const) = \frac{V_{Nutrient}}{V_{Nutrient} + const},$$

where the constant parameter is inversely proportional to the saturation rate. The graphs in Figure 2 depict the difference between the unsaturated and saturated model of the nutrient absorption capacity.

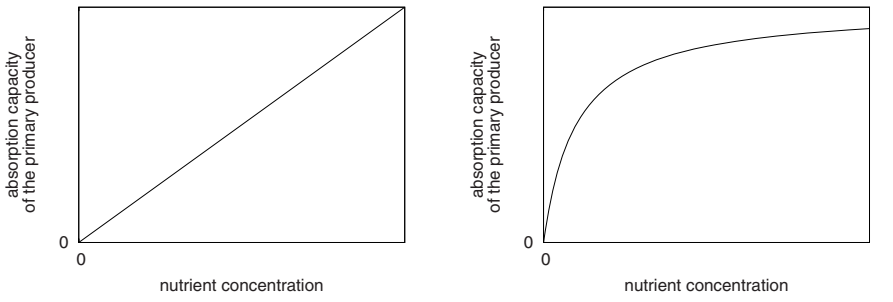

Fig. 2. A comparison of two alternative terms used to model the limitation of primary producer growth due to limited supply of nutrients. The one on the left-hand side assumes unlimited nutrient absorption capacity, while the model on the right-hand side corresponds to situations where the absorption capacity saturates to a certain value.

The remaining productions in the grammar provide models for the other two processes of grazing and mortality. Note that the grammar includes a single model for each process. One can add modeling alternatives for these two processes by adding alternative productions to the appropriate nonterminal symbol.

3.4 Lagramge

Once provided with domain knowledge encoded in the form of a grammar and measurement data about the variables of the observed system, we can employ the Lagramge induction method to perform search through the space of candidate equations defined by the grammar. Lagramge follows a beam strategy to search through the space of parse trees generated by the grammar in the order from simplest (shallowest) trees to more complex (deeper) ones (Todorovski & Džeroski, 1997). In addition to the grammar that constrains the space of candidate models, the user can specify a maximal depth of the parse tree and further limit the complexity of the induced equations.

Each parse tree generated by the grammar is evaluated with respect to its fit to the provided measurement data. To this end, Lagramge first fits the values of the constant parameters against data using a nonlinear least-squares algorithm (Bunch et al., 1993) that carries out second-order gradient descent through the parameter space. To avoid entrapment in local minima, the procedure involves multiple restarts with different initial parameter values. Once Lagramge finds the optimal parameter values, it measures the discrepancy between the observed values of the system variables and the values predicted by the equation based on the parse tree and the optimal parameter values. Note that prediction in the case of dynamic systems requires simulating the model over the full time span. To improve the convergence rate of the gradient search, we employ *teacher forcing* simulation (Williams & Zipser, 1989), which finds parameters that best predict observations at the next time point based solely on those at the present

time point. The discrepancy is measured as mean squared error (MSE) and this measure is used as a heuristic function that guides the search. An alternative heuristic function MDL (which stands for minimal description length) takes into account the complexity of the parse tree and introduces a preference towards simpler models (Todorovski & Džeroski, 1997). At the end of the beam search procedure, LAGRAMGE returns a user specified number of models with the lowest values of the heuristic function selected by the user (among the two choices — MSE and MDL).

Todorovski and Džeroski (1997) show that LAGRAMGE is capable of integrating many different types of domain knowledge through grammars. For example, knowledge about the measurement units of system variables has been used to build a grammar for modeling a mechanical pole on cart system. In another example, knowledge about the basic processes that govern population dynamics has been used for automated modeling of phytoplankton growth in Lake Glumsø in Denmark from a sparse and noisy set of real-world measurements (Todorovski et al., 1998). A drawback of LAGRAMGE is that it is difficult for domain experts to express or encode their knowledge in the form of a grammar. Furthermore knowledge-based grammars such as the one from Table 2 are usually task specific, i.e., a grammar built for modeling one system (e.g., phytoplankton growth in Lake Glumsø) cannot be reused for modeling other systems from the same domain (e.g., population dynamics in another lake with a slightly different set of observed variables). These limitations can be addressed by the transformational approach, which is the topic we focus on in the rest of this chapter. In the next two sections, we show that different types of higher level domain knowledge encoded in various formalisms can be easily transformed to grammars and thus integrated in the process of equation discovery.

4 A Language Bias Formulation of Process-Based Modeling Knowledge

This section presents a flexible and high-level formalism for encoding domain knowledge that is more accessible to domain experts than the grammar-based formalism discussed above. The formalism organizes knowledge in a taxonomy of process classes, each of which represents an important class of basic processes that influence the behavior of entities in the domain of study. For each process class, a number of alternative equation models, usually used by modeling experts in the domain, can be specified. The formalism also encodes knowledge on how to combine the models of individual basic processes into a single model of the whole observed system. We illustrate the use of the formalism by encoding knowledge on modeling population dynamics.

We designed the formalism for knowledge representation so that the resulting library of knowledge for a particular domain is independent of the modeling task at hand. Provided with such a library, the user only needs to specify the modeling task by stating the types of the observed variables along with a list of processes that are most likely to influence the system behavior. While the

library of domain-specific knowledge should be provided by a modeling expert with extensive modeling experience, the task specification can written by a naïve user who is familiar with the domain but does not have much modeling expertise.

4.1 Process-Based Knowledge for Modeling Population Dynamics

In this section, we reconsider the population dynamics domain introduced in Section 3.3. As we already stated there, population dynamics studies the structure and dynamics of species sharing a common environment and is mainly concerned with the study of interactions between different species. While Section 3.3 focuses on the absorption of inorganic nutrients by the primary producer species (plants) and its effect on the change of primary producer concentration only, we extend our focus here to other types of interactions and their effect on the concentrations of all the species involved.

Research in the area of population dynamics modeling was pioneered by Lotka and Volterra and their well-known Volterra-Lotka model of predator-prey interaction (Murray, 1993), which can be schematized as:

$$\dot{N} = \mathit{growth\_rate}(N) - \mathit{feeds\_on}(P, N)$$
$$\dot{P} = \mathit{feeds\_on}(P, N) - \mathit{decay\_rate}(P),$$

where P and N denote the concentrations of predator and prey, respectively. The model combines the influences of three fundamental processes: prey growth, predation, and predator decay. Each of the processes is modeled following one of the following three assumptions. The first assumption is that the growth rate of the prey population in the absence of predation is proportional to its density, i.e., $\mathit{growth\_rate}(N) = aN$. This means that the growth of the population is exponential and unlimited, which is unrealistic in many cases. Natural environments often have a limited carrying capacity for the species. In such cases, one can use the alternative logistic growth model $\mathit{growth\_rate}(N) = aN(1 - N/K)$, where the constant parameter K determines the carrying capacity of the environment for species N.

The second assumption made in the simple Volterra-Lotka model is that the predation rate is proportional to the densities of both the predator and the prey populations, i.e., $\mathit{feeds\_on}(P, N) = bPN$. In analogy with growth, this means that the predation capacity grows exponentially and is unlimited. Again, in some cases the predators have limited predation capacity. When the prey population density is small the predation rate is proportional to it, but when the prey population becomes abundant, the predation capacity saturates to a certain limit value. Several different terms can be used to model the predator saturation response to the increase of prey density (Murray, 1993):

$$(a)\ P\frac{N}{N+B}; \quad (b)\ P\frac{N^2}{N^2+B}; \quad (c)\ P(1-e^{-BN}),$$

where B is the constant that determines the saturation rate.

Table 3. A taxonomy of process classes encoding knowledge about alternative models of the predation process

```
process class Feeds_on(Population p, Populations ns)
   condition p ∉ ns
   expression p * ∏_{n∈ns} Saturation(p, n)

process class Saturation(Population p, Population n)
process class Type_0 is Saturation
   expression n
process class Type_A is Saturation
   expression n / (n + const(saturation_rate,0,Inf))
process class Type_B is Saturation
   expression n * n / (n * n + const(saturation_rate,0,Inf))
process class Type_C is Saturation
   expression 1 - exp(const(saturation_rate,0,Inf) * n)
```

Table 4. A combining scheme specifies how to combine the models of individual population dynamics processes into a model of the entire system

```
combining scheme Population_dynamics(Population p)
   ṗ = + Growth(p) - Decay(p)
       + ∑_{food} const(_,0,Inf) * Feeds_on(p, food)
       - ∑_{predator} const(_,0,Inf) * Feeds_on(predator, p)
```

The third assumption is that the predator decay rate is proportional to its concentration, which leads to exponential decay in the absence of interactions with other species. Relaxing the three assumptions made in the simple Volterra-Lotka model, we can build different, more complex and more realistic models of predator-prey population dynamics.

The modeling knowledge about the different modeling alternatives for the predation process described above can be encoded as shown in Table 3. The `Feeds_on` process class refers to predation processes, while `Saturation` refers to different saturation models. The first line specifies that each predation process relates a single predator population `p` with one or more prey populations `ns`. The next line specifies the constraint that `p` can not predate on itself (i.e., specifying non-cannibalistic behavior). The last line of the `Feeds_on` process declaration specifies the expression template that is used to model the influence of the predation process. The expression defines this influence as a product of predator concentration `p` and the influences of saturation processes for each prey population $\texttt{n} \in \texttt{ns}$. Similarly, the saturation process specifies alternative expressions for modeling the limited (or unlimited) predation capacity of `p` on `n`. Each modeling alternative is encoded as a subclass of the `Saturation` process class.

The second part of the process-based knowledge encodes a recipe for combining the models of individual processes into a model of the entire observed system. The combining scheme presented in Table 4 provides such a recipe for

Table 5. A formal specification of predator-prey interaction between two species. Given this specification and the library of knowledge presented in Tables 3 and 4, one can reconstruct all candidate models of predator-prey interactions between two species.

```
system variable Population predator, prey

process Growth(prey)
process Feeds_on(predator, prey)
process Decay(predator)
```

building models of population dynamics. The recipe is based on the "*conservation of mass*" principle, explained earlier in Section 3.3. It specifies that the change of a population concentration is positively influenced by the process of population growth `Growth(p)` processes and negatively influenced by population decay `Decay(p)`. Similarly, `Feeds_on` processes can positively or negatively influence the change of `p`, depending on the role of the species in the predator-prey interaction. The predation processes that involve `p` as a predator positively influence the change of `p`, while the processes where `p` is involved as a prey negatively influence the change of `p`.

Given this kind of knowledge encoded in a library, one can specify the simple Volterra-Lotka model as illustrated in Table 5. The task specification contains declarations of the two variables involved along with the set of three processes of growth, decay, and predation. An automated modeling system, as the one presented in (Todorovski, 2003), can transform this specification to the candidate model equations. Note that this abstract specification accounts for more than one model, since there are many alternative models of predator saturation that can be used for the `Feeds_on` process. In such cases, the system would transform the specification to a grammar enumerating all possible models of predator-prey interaction between two species. We will not provide further details about the transformation method, that can be found in (Todorovski, 2003), but rather illustrate its use on a practical task of modeling population dynamics.

4.2 Modeling Algal Growth in the Lagoon of Venice

The Lagoon of Venice measures 550 km^2, but is very shallow, with an average depth of less than 1 meter. It is heavily influenced by anthropogenic inflow of nutrients – 7 mio kg/year of nitrogen and 1.4 mio kg/year of phosphorus (Bendoricchio et al., 1994). These (mainly nitrogen) loads are above the Lagoon's admissible trophic limit and generate its dystrophic behavior, which is characterized by excessive growth of algae, mainly Ulva rigida. Coffaro et al. (1993) present four sets of measured data available for modeling the growth of algae in the Lagoon. The data was sampled weekly for slightly more than one year at four different locations in the Lagoon. Location 0 was sampled in 1985/86, locations 1, 2, and 3 in 1990/91. The sampled quantities are nitrogen in ammonia NH_3, nitrogen in nitrate NO_3, phosphorus in orthophosphate PO_4 (all in [μg/l]),

Table 6. A task specification used for modeling algae growth in the Lagoon of Venice

```
exogenous variable Inorganic temp, DO, NH3, NO3, PO4
system variable Population biomass

process Growth(biomass) biomass_growth
process Decay(biomass) biomass_decay
process Feeds_on(biomass, *) biomass_grazing
```

dissolved oxygen *DO* (in percentage of saturation), temperature T ([degrees C]), and algal biomass B (dry weight in [g/m^2]).

In a previous study, Kompare and Džeroski (1995) apply the equation discovery method GoldHorn (Križman, 1998) to the task of modeling algal growth in the Lagoon of Venice. Since GoldHorn could not find an accurate model based on the set of measured variables, two additional variables were calculated and added to this set — growth and mortality rates, which are known quantities in ecological modeling and were calculated according to the simplified version of an existing model of algal growth in the lagoon proposed by Coffaro et al. (1993). From the extended set of system variables and data measured at Location 0, GoldHorn discovered a difference equation for predicting biomass that, due to the large measurement errors (estimated at the level of 20-50%), does not fit the data perfectly, but still predicts most of the peaks and crashes of the biomass concentration correctly (Kompare & Džeroski, 1995). Although the equation model involves the mortality rate, as calculated by domain experts, the model itself is still a black-box model that does not reveal the limiting factors for the biomass growth in the lagoon.

The task of modeling algal growth in the Lagoon of Venice from Table 6 specifies the types of the observed system variables and the processes that are important for the growth of the biomass (algae) in the lagoon. Note that the specification of the `biomass_grazing` process leaves the nutrient parameter of the `Feeds_on` process class unspecified (denoted using the symbol `*`). Since ecologists did not know the limiting factors for the biomass growth, they let Lagramge search for the model that would reveal them. The search space consists of 6248 candidate models with generic constant parameters.

The model induced from Location 0 data reveals that the limiting factors for biomass growth in the lagoon are dissolved oxygen (`DO`) and nitrogen in nitrate (`NO3`). The model induced from Location 2 data reveals that the limiting factors for the biomass growth are temperature (`temp`), dissolved oxygen (`DO`), and nitrogen in ammonia (`NH3`). Although the two models are not completely consistent, they both identify dissolved oxygen and nitrogen based nutrients to be limiting factors for biomass growth. The differences between the two models may be due to the fact that the measurements were taken during two different time periods.

In the experiments with the data measured at the other two locations (1 and 3), Lagramge did not find an accurate model of biomass growth. Note that these results still compare favorably with the results obtained by GoldHorn, which discovered an acceptable model for Location 0 only.

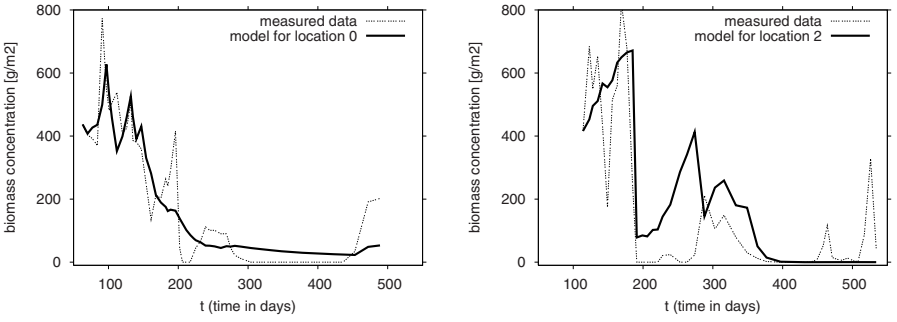

Fig. 3. Simulations of the two models of biomass growth in the Lagoon of Venice, discovered by LAGRAMGE, compared to the measured biomass concentration (left-hand side: Location 0, right-hand side: Location 2)

Figure 3 compares the measured and simulated values of the biomass for both models. We ran long-term simulations of the models from the initial value of the biomass without restarting the simulation process at each measurement point. For values of all other system variables needed during the simulation, we used the measurement at the nearest time point in the past. As in the GOLDHORN experiments, due to the high measurement errors of the order 20-50%, the models discovered by LAGRAMGE did not fit the measured data perfectly. However, they correctly predict most of the peaks and crashes of the biomass concentration. These events are more important to ecologists than the degree of fit. Note however another important advantage of these models over the one discovered by GOLDHORN. While the GOLDHORN model is black-box, the models discovered by LAGRAMGE identify the most influential limiting factors for biomass growth in the Lagoon of Venice.

5 Language Bias Based on Existing Models

Another type of domain-specific knowledge that is typically neglected by equation discovery methods is contained in the existing models already established in the domain of interest. Rather than starting the search with an existing model, current equation discovery methods start their search from scratch. In contrast, theory revision methods (Ourston & Mooney, 1994; Wrobel, 1996) start with an existing theory and use heuristic search to revise it in order to improve its fit to data. However, research on theory revision has been mainly concerned with the revision of models expressed in propositional or first-order logic. Therefore, existing methods for theory revision are not directly applicable to the task of revising models based on equations.

In this section, we present a flexible, grammar-based methodology for revising equation-based models. To support the revision of existing models, we first transform the given model into an initial grammar that can be used to derive the given model only. The nonterminals in the grammar and their productions

reflect the structure of the initial model. Next, we extend the initial grammar with alternative productions that specify the possible modeling alternatives. The modeling alternatives can be specified by a domain expert or can be determined from modeling knowledge about the domain at hand. The extended grammar built in this manner specifies the space of possible revisions of the initial model. In the last step, we employ LAGRAMGE to search through the space of possible revisions and find those that fit the data better than the initial model.

5.1 Revising the CASA Model

The CASA model, developed by Potter and Klooster (1997) at NASA Ames, accounts for the global production and absorption of biogenic trace gases in the Earth atmosphere. It also allows us to predict changes in the geographic patterns of major vegetation types (e.g., grasslands, forest, tundra, and desert) on the land. CASA predicts annual global fluxes in trace gas production as a function of surface temperature, moisture levels, soil properties, and global satellite observations of the land surface. It operates on gridded input at different levels of resolution, but typical usage involves grid cells that are eight kilometers square, which matches the resolution for satellite observations of the land surface.

The overall CASA model is quite complex, involving many variables and equations. We decided to focus on one portion that is concerned with modeling net production of carbon by terrestial plants (NPPc). Table 7 presents the NPPc portion of the CASA model. The model predicts the NPPc quantity as the product of two unobservable variables, the photosynthetic efficiency, E, at a site (grid cell) and the solar energy intercepted, *IPAR*, at that site.

Photosynthetic efficiency is in turn calculated as the product of the maximum efficiency (0.56) and three stress factors that reduce this efficiency. The stress term *T2* takes into account the difference between the optimum temperature, *topt*, and actual temperature, *tempc*, for a site. The stress factor *T1* involves the nearness of *topt* to a global optimum for all sites. The third term, W, represents stress that results from lack of moisture as reflected by *eet*, the estimated water loss due to evaporation and transpiration, and *PET*, the water loss due to these processes given an unlimited water supply. In turn, *PET* is defined in terms of the annual heat index, *ahi*, for a site, and *pet_tw_m*, a modifier on *PET* to account for day length at differing locations and times of year.

The energy intercepted from the sun, *IPAR*, is computed as the product of *FPAR_FAS*, the fraction of energy absorbed through photo-synthesis for a given vegetation type, *monthly_solar*, the average radiation for a given month, and *SOL_CONV*, the number of days in that month. *FPAR_FAS* is a function of *fas_ndvi*, which indicates overall greenness at a site as observed from space, and *srdiff*, an intrinsic property that takes on different numeric values for different vegetation types.

Of the variables we have mentioned, *NPPc*, *tempc*, *ahi*, *monthly_solar*, *SOL_CONV*, and *fas_ndvi*, are observable. Two additional terms, *eet* and *pet_tw_m*, are defined elsewhere in the model, but we assume their definitions are correct and thus we can treat them as observables. The remaining variables are unobservable

Table 7. The NPPc portion of the CASA model that accounts for the net production of carbon by terrestial plants

$$
\begin{aligned}
&NPPc = \max(0, E \cdot IPAR)\\
&\quad E = 0.56 \cdot T1 \cdot T2 \cdot W\\
&\qquad T1 = 0.8 + 0.02 \cdot topt - 0.0005 \cdot topt^2\\
&\qquad T2 = 1.1814/((1 + \exp(0.2 \cdot (TDIFF - 10))) \cdot (1 + \exp(0.3 \cdot (-TDIFF - 10))))\\
&\qquad\quad TDIFF = topt - tempc\\
&\qquad W = 0.5 + 0.5 \cdot eet/PET\\
&\qquad\quad PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^A \cdot pet\_tw\_m\\
&\qquad\qquad A = 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239\\
&\quad IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5\\
&\qquad FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95)\\
&\qquad\quad SR\_FAS = (1 + fas\_ndvi/1000)/(1 - fas\_ndvi/1000)\\
&\qquad SOL\_CONV = 0.0864 \cdot days\_per\_month
\end{aligned}
$$

and must be computed from the others using their definitions. This portion of the model also contains a number of numeric parameters, as shown in Table 7.

The ecologists that developed the CASA model pointed out which parts of the initial CASA-NPPc model are likely candidates for revision. Their confidence in the equations used to calculate the values of the four intermediate variables E, $T1$, $T2$, and SR_FAS is low, so they considered them to be "weak" parts of the model. Thus, we focused our revision efforts to the equations corresponding to these four intermediate variables. The data set for revising the CASA-NPPc model contains measurements of the observed variables at 303 sites. We assess the quality of the revised models using root mean squared error evaluated on training data and cross-validated. The RMSE of the initial model on the training data set is 465.213.

Grammars Used for Revision. A set of equations defining a target variable through a number of intermediate variables can easily be turned into a grammar. The staring nonterminal symbol represents the dependent variable *NPPc*, while other nonterminals correspond to the intermediate variables that appear on the left-hand sides of the equations in the initial model. Each nonterminal has a single production that is obtained by replacing the equality ("=") sign in the equation with "→". In such a grammar, the terminals denote observed variables, the model's constant parameters, and the arithmetical operators involved in the equations. Such a grammar generates a single model which is exactly the initial CASA-NPPc model from Table 7.

The grammar obtained from the initial model lets us specify an arbitrary number of alternative models for any intermediate variable by adding productions to the corresponding nonterminal. These additional productions specify alternative modeling choices, only one of which will eventually be chosen to revise the initial model. In general, there are two classes of alternative productions. The productions from the first class replace one or more constant parameter values with a generic constant parameter *const*. In our experiments with the CASA-NPPc

Table 8. Grammar productions specifying modeling alternatives that can be used to revise the initial CASA-NPPc model

`Ec-100`	$E \rightarrow const[0:1.12] \cdot T1 \cdot T2 \cdot W$
`Es-exp`	$E \rightarrow const[0:1.12] \cdot T1^{const[0:]} \cdot T2^{const[0:]} \cdot W^{const[0:]}$
`T1c-100`	$T1 \rightarrow const[0:1.6] + const[0:0.04] \cdot topt - const[0:0.001] \cdot topt^2$
`T1s-poly`	$T1 \rightarrow const \mid const + (T1) * topt$
`T2c-100`	$T2 \rightarrow const[0:2.3628]/((1+\exp(const[0:0.4] \cdot (TDIFF - const[0:20]))) \cdot (1+\exp(const[0:0.6] \cdot (-TDIFF - const[0:20]))))$
`T2s-poly`	$T2 \rightarrow const \mid const + (T2) \cdot TDIFF$
`SR_FASc-25`	$SR\_FAS \rightarrow (1 + fas\_ndvi/const[750:1250]) / (1 - fas\_ndvi/const[750:1250])$

model, we use alternative productions that allow for a 100% relative change of the initial value of a constant parameter. This can be specified by replacing the fixed value constant parameter v with a terminal symbol $const[0 : 2 \cdot v]$. Thus, the lower bound for the newly introduced constant parameter is set to $v - 100\% \cdot v = 0$, while the upper bound is set to $v + 100\% \cdot v = 2 \cdot v$. Productions in the second class are slightly more complex and allow for *structural* revision of the model.

As we pointed out above, the focus of the model revision is on the equations corresponding to the four intermediate variables E, $T1$, $T2$, and SR_FAS. Table 8 presents the modeling alternatives we used to revise the equations for these four variables.

Alternative productions for E

`Ec-100` allows refitting the value of the constant parameter in the equation for E.

`Es-exp` enables structural revision of the E equation. It replaces the initial $T1 \cdot T2 \cdot W$ product with an expression that allows for arbitrary exponents of the three multiplied variables (i.e., $T1^{e_1} \cdot T2^{e_2} \cdot W^{e_3}$). The obtained values of the exponents in the revised model would then correspond to the relative magnitude of the influences of $T1$, $T2$, and W on E.

Alternative productions for T1

`T1c-100` allows refitting the values of the constant parameters in the *T1* equation.

`T1s-poly` replaces the initial second degree polynomial for calculating *T1* with an arbitrary degree polynomial of the same variable *topt*.

Alternative productions for T2

`T2c-100` enables 100% relative change of the constant parameter values in the *T2* equation.

`T2s-poly` replaces the initial equation for *T2* with an arbitrary degree polynomial of the variable *TDIFF*.

Table 9. The root squared mean error (RMSE) of the revised models and the percentage of relative error reduction (RER) over the initial CASA-NPPc model. We evaluated RMSE of the revisions on training data (training - the left-hand side of the table) and using 30-fold cross-validation (the right-hand side of the table).

	training		cross-validation	
alternative production(s)	RMSE	RER(%)	RMSE	RER(%)
`Ec-100`	458.626	1.42	460.500	1.01
`Es-exp`	443.029	4.77	443.032	4.77
`T1c-100`	458.301	1.49	460.799	0.95
`T1s-poly`	450.265	3.21	457.370	1.69
`T2c-100`	457.018	1.76	459.633	1.20
`T2s-poly`	450.972	3.06	461.642	0.77
`SR_FASc-25`	453.157	2.59	455.281	2.13
`ALL`	414.739	10.85	423.684	8.93

Alternative productions for SR_FAS

`SR_FASc-25` allows for a 25% relative change of the constant parameter values in the *SR_FAS* equation. We used 25% instead of usual 100% to avoid values of the constant parameters below 750, which would cause singularity (division by zero) problems when applying the equation.

Note that we can add an arbitrary combination of these alternative productions to the initial grammar. If all of them are added at the same time, then LAGRAMGE will search for the most beneficial combination of revisions. In the latter case, LAGRAMGE searches the space of 384 possible revisions of the original CASA-NPPc model (note that we limit the parse tree depth so that the maximal degree of the polynomials corresponding to $T1$ and $T2$ is five).

Results. Table 9 summarizes the results of the revision. When we allow only a single of the seven alternatives, revising the structure of the E equation leads to the largest (almost 5%) reduction of the initial model error. However, we obtain the largest error reduction of almost 11% on the training data and 9% when cross-validated with a combination of revision alternatives. The optimal combination of revisions is `Es-exp`, `T1c-100`, `T2s-poly`, and `SR_FASc-25` , which leads to the model presented in Table 10. It is worth noticing that the reductions obtained with single alternative productions nearly add up to the error reduction obtained with the optimal combination of revisions.

The most surprising revision result proposes that the water stress factor W does not influence the photosynthetic efficiency E, i.e., $E = 0.402 \cdot T1^{0.624} \cdot T2^{0.215} \cdot W^0$. Ecologists that developed the CASA model suggested that this surprising result might be due to the fact that the influence of the water stress factor on E is already being captured by the satellite measurements of the relative greenness, *fas_ndvi*, and thus, W becomes obsolete in the E equation.

Furthermore, the revised model replaces the initial second-degree polynomial for calculating *T1* with a linear equation. The structural revision `T2s-poly` replaced the complex initial equation structure for calculating *T2* with a relatively

Table 10. The best revision of the initial CASA-NPPc model. The parts that were left intact in the revision process are printed in gray.

$$
\begin{aligned}
&NPPc = \max(0, E \cdot IPAR)\\
&\quad E = 0.402 \cdot T1^{0.624} \cdot T2^{0.215} \cdot W^{0}\\
&\qquad T1 = 0.680 + 0.270 \cdot topt - 0 \cdot topt^{2}\\
&\qquad T2 = 0.162 + 0.0122 \cdot TDIFF + 0.0206 \cdot TDIFF^{2} - 0.000416 \cdot TDIFF^{3}\\
&\qquad\quad -0.0000808 \cdot TDIFF^{4} + 0.000000184 \cdot TDIFF^{5}\\
&\qquad\quad TDIFF = topt - tempc\\
&\qquad W = 0.5 + 0.5 \cdot eet/PET\\
&\qquad\quad PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^{A} \cdot pet\_tw\_m\\
&\qquad\qquad A = 0.000000675 \cdot ahi^{3} - 0.0000771 \cdot ahi^{2} + 0.01792 \cdot ahi + 0.49239\\
&\quad IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5\\
&\qquad FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95)\\
&\qquad\quad SR\_FAS = (1 + fas\_ndvi/750)/(1 - fas\_ndvi/750)\\
&\qquad SOL\_CONV = 0.0864 \cdot days\_per\_month
\end{aligned}
$$

simple fifth degree polynomial. While the initial form of the *T2* equation is fairly well grounded in first principles of plant physiology, it has not been extensively verified from field measurements. Therefore, both empirical improvements are considered plausible by the ecologists.

5.2 Completing a Partial Model of Water Level Change in the Ringkøbing Fjord

In the last series of experiments, we illustrate that the proposed methodology can be also used for completing a partial model specification. In such a case, human experts specify only some parts of the model structure and leaves others unspecified or partly specified. The goal is then to determine both the structure and constant parameter values of the unspecified parts.

An example of such a task is modeling water level variation in Ringkøbing fjord, a shallow estuary located at the Danish west coast, where it experiences mainly easterly and westerly winds.[2] Wind forcing causes large short term variation of the water level (h) measured at the gate between the estuary and the North Sea. Domain experts specified the following partial model for the temporal variation of the water level in the estuary:

$$\dot{h} = \frac{f(a)}{A}(h_{sea} - h + h_0) + \frac{Q_f}{A} + g(W_{vel}, W_{dir}). \tag{2}$$

The water level response to the wind forcing depends on both wind speed (variable W_{vel}, measured in [m/s]) and direction (W_{dir}, measured in degrees) and is

[2] The task was used as an exercise within a post-graduate course on modeling dynamic systems organized in 2000. Since the Web page of the course is no longer available, we cannot provide a proper reference to the original task specification. Note also that we could not consult domain experts and therefore could not obtain expert comments on the induced models.

Table 11. The grammar for modeling water level variation in the Rinkøbing fjord that follows the partial model specification from Equation 2

$$WaterLevelChange \rightarrow (F/A) * SaltWaterDrive + FreshWaterFlow + G$$

$$SaltWaterDrive \rightarrow (h_{sea} - h + const)$$
$$FreshWaterFlow \rightarrow Q_f/A$$

modeled by an unknown function g. Apart from wind forcing, fresh water supply (Q_f, measured in [m^3/s]) influences the water level change. When the gate is closed, fresh water is accumulated in the estuary causing a water level rise of Q_f/A, where A is the surface area of the estuary measured in squared meters. During periods when the gate is open, the stored fresh water is emptied in the North Sea. The gate is also opened in order to maintain sufficient water level in the estuary, in which case the water rise is driven by the difference between the water level in the open sea (variable h_{sea}, measured in meters), the water level in the estuary (h, measured in meters), and the constant parameter (h_0). The flow is restricted by the friction of the flow, modeled by an unknown function f of the number of gate parts being open (a). Namely, the gate consists of 14 parts and allows for opening some parts and closing others. The value of A is not directly observed, but a function that calculates A on the basis of h is provided, so A can be also treated as an observed variable.

Given the partial model specification and measurements of the observed variables, the task of model completion is to find the structure and parameters of the unknown functions f and g. The data set contains hourly measurements of all the observed variables within the period from 1st of January to 10th of December 1999.

Grammars for Completion. In order to apply our methodology to the task of model completion, we first recode the partial model specification into a grammar. The grammar presented in Table 11 follows the partial model formula along with the explanations of its constituent terms. The grammar contains two nonterminal symbols F and G that correspond to the unknown functions f and g, respectively.

In the second step, we add productions for F and G to the grammar. They specify modeling alternatives for the completion task. In absence of domain-specific knowledge, we make use of simple polynomial models as presented in Table 12. The first two modeling alternatives, `F0` and `G0`, are the simplest possible models, i.e., constants. The next two, `F1` and `G1`, are using polynomials of the appropriate system variables. Finally, we used an additional modeling alternative for the g function, `G2`, that replaces the wind direction value (that represents angle) with its sine and cosine transformation, respectively.

Results. Table 13 summarizes the results of the model completion and provides a comparison with the black-box modeling case where no knowledge about model structure was used (`Polynomial`). The best cross-validated performance is gained using the partial model specification provided by the experts in combination with

Table 12. Grammar productions specifying modeling alternatives that can be used to complete the model of water level variation in the Ringkøbing fjord

`F0`	$F \rightarrow const$
`F1`	$F \rightarrow F + const * T_F \mid const * T_F$
	$T_F \rightarrow T_F * V_F \mid V_F$
	$V_F \rightarrow a$
`G0`	$G \rightarrow const$
`G1`	$G \rightarrow G + const * T_F \mid const * T_F$
	$T_G \rightarrow T_G * V_G \mid V_G$
	$V_G \rightarrow W_{vel} \mid W_{dir}$
`G2`	$V_G \rightarrow W_{vel} \mid \sin(W_{dir}) \mid \cos(W_{dir})$

Table 13. The root mean squared errors (RMSE, estimated on both training data and using 10-fold cross-validation) of the four water level variation models induced by LAGRAMGE with (three first rows) and without (last row) using the partial model specification provided by the domain experts. The last column gives number of candidate models (#CMS) considered during the search.

task	training RMSE	cross-validated RMSE	#CMS
`F0 + G0`	0.0848	0.106	1
`F1 + G1`	0.0655	0.0931	378
`F1 + G2`	0.0585	0.0903	2184
`Polynomial`	0.0556	2.389	2801

the `F_1` and `G_2` modeling alternatives for the unspecified parts of the structure. The graph on the left-hand side of Figure 4 shows the simulation of this model compared to the measured water level in the Ringkøbing fjord. We ran a long-term simulation of the model from the initial value of the water level without restarting the simulation process at any measurement point. For the values of all other system variables needed during the simulation, we used the measurements at the nearest time point in the past.

Note that the model follows the general pattern of water level variation. The long-term simulation of the model, however, fails to precisely capture the short-term (hour) changes of the water level. To test the short-term prediction power of the model, we performed two additional simulations, which we restarted with the true measured water level values at every hour and at every day (24 hours). Table 14 presents the results of this analysis. They show that the model is suitable for short-term prediction of the water level in the Ringkøbing fjord.

Since the model induced by LAGRAMGE follows the partial structure specification provided by the human experts, further analysis can be performed. For example, we can compare the influence of the gate opening (modeled by $f(a)(h_{\text{sea}} - h + h_0)/A$) with the effect of the wind (modeled by $g(W_{\text{vel}}, W_{\text{dir}})$).

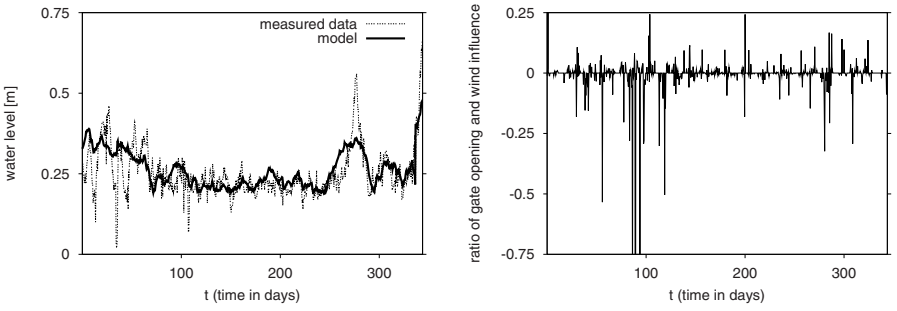

Fig. 4. Simulation of the best water level variation model induced by LAGRAMGE compared to the measured water level (left-hand side) and analysis of the influence of gate opening relative to wind forcing as modeled by LAGRAMGE (right-hand side)

Table 14. The RMSE and correlation coefficient (r) for the short-term (one hour and one day) prediction of the water level in the Ringkøbing fjord compared to the RMSE and r^2 of the simulation over the whole observation period

prediction/simulation period	RMSE	r
one hour	0.0168	0.976
one day	0.0425	0.845
whole observation period	0.0585	0.659

The graph on the right-hand side of Figure 4 shows the ratio of the gate opening and the wind influences on the water level change in the Ringkøbing fjord. The low magnitude of the ratio shows that the influence of the wind prevails over the influence of the gate opening most of the time. The only exceptions occur in the period from 80 to 100 days from the beginning of the measurement, that is, the end of March and beginning of April 1999.

Finally, note that the polynomial model of the water level variation ignores the partial specification, but performs best on the training data. However, the model's small RMSE is due to the obvious overfitting of the training data, since the cross-validated RMSE of the same model (2.389) is much larger than the cross-validated RMSE of the models that follow the partial model specification. This result confirms the importance of integrating available knowledge in the process of model induction.

6 Related Work

The work presented in this chapter is mainly related to other modeling approaches presented in literature. Our methodology is closest in spirit to the compositional modeling (CM) paradigm (Falkenheiner & Forbus, 1991). In our methodology, models of individual processes correspond to model fragments

in CM and combining schemes to combining rules in CM. Both CM and our methodology views model building as a search for an appropriate combination of model fragments. The PRET reasoning system for automated modeling also follows the CM paradigm to modeling, but it employs a slightly different kind of modeling knowledge (Stolle & Bradley, this volume; Easley & Bradley, this volume). The first kind of knowledge used in PRET is domain-specific knowledge in the form of "conservation rules". An example of such a rule in the spring mechanics domain specifies that "the sum of forces at any observed coordinate of the mechanical system is zero". These rules are more general than the domain knowledge about model fragments and their composition used in compositional modeling approaches. Therefore, PRET rules constrain the space of possible models much less. PRET compensates for this lack of constraints by using a second kind of domain-independent knowledge about models of dynamic systems based on ordinary differential equations. An example of such a rule specifies that "a model with oscillatory behavior has to be second-order". This kind of ODE rules allows PRET to efficiently rule out inappropriate models by high-level abstract (qualitative) reasoning. As we have illustrated in (Todorovski, 2003), both kinds of modeling knowledge, used in PRET, can be easily encoded within our formalism. Note, however, that LAGRAMGE is not capable of ruling out inappropriate candidate models based on qualitative reasoning, but rather tries to perform quantitative simulation of the candidate models and find out that they can not fit the measured data well.

Another related study is presented by Garrett et al. (this volume). They apply the compositional modeling approach to the task of inducing models of chemical reaction pathways from noisy measurement data. However, the models they induce are qualitative. Although the concepts introduced within the area of compositional modeling are also relevant for automated building of quantitative models of real-world systems, this idea has not been widely explored.

Our approach is similar to the ECOLOGIC approach (Robertson et al., 1991) in the sense that it allows for representing modeling knowledge and domain-specific knowledge. However, in ECOLOGIC, the user has to select among the alternative models, whereas in our approach observational data is used to select among the alternatives. It is also related to process-based approaches to qualitative physics (Forbus, 1984). We can think of the food-chain or domain-specific part of the knowledge as describing processes qualitatively, whereas the modeling part together with the data introduces the quantitative component. However, the ECOLOGIC approach is limited to modeling systems in the environmental domain, whereas our approach is applicable in a variety of domains. Salles and Bredeweg (2003) presents a framework similar to ECOLOGIC that can be used for building models of population and community dynamics. In contrast to the work presented here, they focus on building qualitative conceptual models that do not require numeric data nor provide precise simulation of system behavior.

The work on revising models presented here is related to two other lines of work. In the first, Saito et al. (this volume) address the same task of revising models based on equations. Their approach transforms a part of the model into

a neural network, retrains the neural network on available data, and transforms the trained network back into an equation-based model. They obtained revised models with a considerably smaller error rate than the original one, but gained a slightly lower accuracy improvement than our method did. A limitation of their approach is that it requires some hand-crafting to encode the equations as a neural network. The authors state that "the need to translate the existing CASA model into a declarative form that our discovery system can manipulate" is a challenge to their approach.

The approach of transforming equation-based models to neural networks and using these for refinement is similar in spirit to the KBANN approach proposed in (Towell & Shavlik, 1994). There, an initial theory based on classification rules is first encoded as neural network. Then, the topology of the network is refined and the network is re-trained with the newly observed data. Finally, the network is transformed back into rules. However, the application of KBANN is limited to theories and models expressed as classification rules. In other related work, Whigham and Recknagel (2001) consider the task of revising an existing model for predicting chlorophyll-a by using measured data. They use a genetic algorithm to calibrate the equation parameters. They also use a grammar-based genetic programming approach to revise the structure of two subparts of the initial model, one at a time. A most general grammar that can derive an arbitrary expression using the specified arithmetic operators and functions was used for each subpart. Unlike the work presented here, Whigham and Recknagel (2001) do not present a general framework for the revision of equation-based models, although their approach is similar to ours in that they use grammars to specify possible revisions.

Different aspects of the work presented in this chapter has been already published in other articles and papers. First, Todorovski and Džeroski (1997) introduce the grammar-based equation discovery method LAGRAMGE. The paper illustrates the use of grammars for integrating different aspects of domain knowledge in the process of equation discovery — measurement units and process-based knowledge being among others. The successful application of LAGRAMGE to modeling phytoplankton growth in Lake Glumsø is the topic of the paper by Todorovski et al. (1998), which also identifies the difficulty of encoding knowledge into grammars as a main drawback of LAGRAMGE. The work presented by Todorovski and Džeroski (2001b) introduces a new higher-level formalism for encoding process-based knowledge about population dynamics. These ideas lead also to a separate line of work starting with (Langley et al., 2002). Finally, Todorovski and Džeroski (2001a) present a grammar-based methodology for revising equation-based models.

7 Conclusion

In the chapter, we presented a methodology for integrating various aspects of knowledge specific to the domain of interest in the process of inducing equation-based models from data. The methodology is based on idea of transforming the

domain-specific knowledge to a language bias for induction that specifies the set of candidate hypotheses. Context free grammars are used as a formalism for specifying the language bias. We show how three different types of domain-specific knowledge can be easily transformed to grammars. We demonstrated the utility of our approach by performing experiments using LAGRAMGE, an induction method that can take into account declarative language bias encoded in a form of a grammar. The results of the experiments show that models induced using knowledge make sense to domain scientists and more importantly, scientists can easily understand and interpret the induced models. Furthermore, the models reveal important non-trivial relations between observed entities that are difficult to infer using black-box models. A comparison of models induced with and without using domain-specific knowledge in the last experiment shows that using knowledge can considerably reduce overfitting and increase model prediction performance on test cases unseen in the induction phase.

The immediate direction of further work is establishing libraries of encoded knowledge in different domains. These libraries will be built in cooperation with domain experts that have expertise in modeling real-world systems from measured data. Establishing such libraries will make the developed methods usable by domain experts that collect data about real-world systems, but are not experienced mathematical modelers. First steps toward establishing a library for modeling of aquatic ecosystems, based on recent developments in the domain, have been already made (Atanasova and Kompare 2003; personal communication). Furthermore, the same team of experts work on a library for establishing models of equipment used for waste water treatment. In both cases, the libraries will be used for automated modeling based on collections of measurements.

The automated modeling approach based on transformation to grammars is limited to modeling tasks where the domain expert is capable to provide processes that are expected to be important for modeling the observed system. However, there are many real-world tasks, where experts are not able to specify the list of processes. In these cases, a two level search procedure should be developed that is capable of discovering the processes that influence the behavior of the observed system. At the higher level, the search will look for the optimal set of processes. For each set of processes, the proposed modeling framework will be used at the lower level to find the model, based on the particular set of processes, that fits the measured data best.

The methodology presented in this chapter is still under development and different methods presented are developed and evaluated independently of the others. For example, the methods do not allow the revision of models based on partial differential equations, although in principle this should not be a problem. Furthermore, the formalism can easily encode domain knowledge about changes of the systems along a spatial dimension, but a method for discovering partial differential equations is not integrated within LAGRAMGE. There is a clear need for proper integration of the methods into a single modeling assistant that would allow establishing new and revising existing models based on algebraic, ordinary, and partial differential equations.

The integrated modeling assistant should be further integrated within standard data analysis and simulation environments[3] that are routinely used by mathematical modelers. Beside improved ease of use, the integration will enable standard techniques for parameter estimation and sensitivity analysis to be used in conjunction with the automated modeling framework to yield a proper scientific assistant.

References

Bendoricchio, G., Coffaro, G., DeMarchi, C.: A trophic model for Ulva Rigida in the Lagoon of Venice. Ecological Modelling 75/76, 485–496 (1994)

Bunch, D.S., Gay, D.M., Welsch, R.E.: Algorithm 717; subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. ACM Transactions on Mathematical Software 19, 109–130 (1993)

Chomsky, N.: Three models for the description of language. IRE Transactions on Information Theory 2, 113–124 (1956)

Coffaro, G., Carrer, G., Bendoricchio, G.: Model for Ulva Rigida growth in the Lagoon of Venice (Technical Report). University of Padova, Padova, Italy. UNESCO MURST Project: Venice Lagoon Ecosystem (1993)

Džeroski, S., Todorovski, L.: Discovering dynamics: from inductive logic programming to machine discovery. Journal of Intelligent Information Systems 4, 89–108 (1995)

Falkenheiner, B., Forbus, K.D.: Compositional modeling: Finding the right model for the job. Artificial Intelligence 51, 95–143 (1991)

Forbus, K.D.: Qulitative process theory. Artificial Intelligence 24, 85–168 (1984)

Giordano, F.R., Weir, M.D., Fox, W.P.: A first course in mathematical modeling. 2nd edn. Brooks/Cole Publishing Company, Pacific Grove, CA (1997)

Hopcroft, J.E., Ullman, J.D.: Introduction to automata theory, languages and computation. Addison-Wesley, Reading, MA (1979)

Kokar, M.M.: Determining arguments of invariant functional descriptions. Machine Learning 4, 403–422 (1986)

Kompare, B., Džeroski, S.: Getting more out of data: automated modelling of algal growth with machine learning. In: Proceedings of the International symposium on coastal ocean space utilisation, Yokohama, Japan, pp. 209–220 (1995)

Križman, V.: Avtomatsko odkrivanje strukture modelov dinamičnih sistemov. Doctoral dissertation, Faculty of computer and information science, University of Ljubljana, Ljubljana, Slovenia. In Slovene (1998)

Langley, P., Sanchez, J., Todorovski, L., Džeroski, S.: Inducing process models from continuous data. In: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 347–354. Morgan Kaufmann, Sidney, Australia (2002)

Langley, P., Simon, H., Bradshaw, G.: Heuristics for empirical discovery. In: Computational models of learning, pp. 21–54. Springer, Heidelberg (1987)

Langley, P., Żytkow, J.: Data-driven approaches to empirical discovery. Artificial Intelligence 40, 283–312 (1989)

Lavrač, N., Džeroski, S.: Inductive logic programming: Techniques and applications. Chichester: Ellis Horwood. (1994), Available for download at `http://www-ai.ijs.si/SasoDzeroski/ILPBook/`

[3] Examples of such systems are MatLab (`http://www.mathworks.com/`), SciLab (`http://www-rocq.inria.fr/scilab/`), and Octave (`http://www.octave.org/`).

Mitchell, T.M.: The need for biases in learning generalizations. In: Readings in machine learning, pp. 184–191. Morgan Kaufmann, San Mateo, CA (1991)

Murray, J.D.: Mathematical biology, 2nd corrected edn. Springer, Heidelberg (1993)

Nédellec, C., Rouveirol, C., Adé, H., Bergadano, F., Tausend, B.: Declarative bias in ILP. In: Raedt, L.D. (ed.) Advances in inductive logic programming, pp. 82–103. IOS Press, Amsterdam, The Netherlands (1996)

Ourston, D., Mooney, R.J.: Theory refinement combining analytical and empirical methods. Artificial Intelligence 66, 273–309 (1994)

Pazzani, M., Kibler, D.: The utility of background knowledge in inductive learning. Machine Learning 9, 57–94 (1992)

Pazzani, M.J., Mani, S., Shankle, W.R.: Acceptance of rules generated by machine learning among medical experts. Methods of Information in Medicine 40, 380–385 (2001)

Potter, C.S., Klooster, S.A.: Global model estimates of carbon and nitrogen storage in litter and soil pools: Response to change in vegetation quality and biomass allocation. Tellus 49B, 1–17 (1997)

Quinlan, J.R.: C4.5: Programs for machine learning. Morgan Kaufmann, San Mateo, CA (1993)

Robertson, D., Bundy, A., Muetzelfield, R., Haggith, M., Uschold, M.: Eco-logic: logic-based approaches to ecological modelling. MIT Press, Cambridge, MA (1991)

Salles, P., Bredeweg, B.: Qualitative reasoning about population and community ecology. AI Magazine 24, 77–90 (2003)

Todorovski, L.: Using domain knowledge for automated modeling of dynamic systems with equation discovery. Doctoral dissertation, Faculty of computer and information science, University of Ljubljana, Slovenia (2003)

Todorovski, L., Džeroski, S.: Declarative bias in equation discovery. In: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 376–384. Morgan Kaufmann, San Mateo, CA (1997)

Todorovski, L., Džeroski, S.: Theory revision in equation discovery. In: Proceedings of the Fourth International Conference on Discovery Science, pp. 389–400. Springer, Heidelberg (2001a)

Todorovski, L., Džeroski, S.: Using domain knowledge on population dynamics modeling for equation discovery. In: Proceedings of the Twelfth European Conference on Machine Learning, pp. 478–490. Springer, Heidelberg (2001b)

Todorovski, L., Džeroski, S., Kompare, B.: Modelling and prediction of phytoplankton growth with equation discovery. Ecological Modelling 113, 71–81 (1998)

Towell, G.G., Shavlik, J.W.: Knowledge-based artificial neural networks. Artificial Intelligence 70, 119–165 (1994)

Utgoff, P.E.: Shift of bias for inductive concept learning. In: Machine learning: An artificial intelligence approach — vol. 2, pp. 107–148. Morgan Kaufmann, San Mateo, CA (1986)

Voit, E.O.: Computational analysis of biochemical systems. Cambridge University Press, Cambridge, UK (2000)

Washio, T., Motoda, H.: Discovering admissible models of complex systems based on scale-types and identity constraints. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, pp. 810–817. Morgan Kaufmann, San Mateo, CA (1997)

Whigham, P.A., Recknagel, F.: Predicting chlorophyll-a in freshwater lakes by hybridising process-based models and genetic algorithms. Ecological Modelling 146, 243–251 (2001)

Williams, R.J., Zipser, D.: A learning algorithm for continually running fully recurrent neural networks. Neural Computation 1, 270–280 (1989)

Wrobel, S.: First order theory refinement. In: Raedt, L.D. (ed.) Advances in inductive logic programming, pp. 14–33. IOS Press, Amsterdam, The Netherlands (1996)

Zembowicz, R., Żytkow, J.M.: Discovery of equations: Experimental evaluation of convergence. In: Proceedings of the Tenth National Conference on Artificial Intelligence, pp. 70–75. Morgan Kaufmann, San Mateo, CA (1992)