Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Ljupčo Todorovski

Uporaba predznanja pri modeliranju dinamičnih sistemov z avtomatskim odkrivanjem enačb

Doktorska disertacija

Mentor: prof. dr. Ivan Bratko
Somentor: doc. dr. Sašo Džeroski

Ljubljana, 2002-2003

University of Ljubljana
Faculty for Computer and Information Science

Ljupčo Todorovski

Using domain knowledge for automated modeling of dynamic systems with equation discovery

PhD Thesis

Thesis Supervisor: professor Ivan Bratko
Thesis Co-Supervisor: assistant professor Sašo Džeroski

Ljubljana, 2002-2003

Abstract

The process of establishing an acceptable model of an observed dynamic system from measured data is a challenging task that occupies a major portion of the work of the mathematical modeler. In this thesis, we propose a knowledge-based approach to automated modeling of dynamic systems based on equation discovery methods.

Most work in equation discovery is concerned with assisting the empirical approach to modeling physical systems. Following this approach, the observed system is modeled on a trial-and-error basis to fit observed data. None of the available domain knowledge about the observed system (or a very limited portion thereof) is used in the modeling process. The empirical approach is contrasts with the theoretical approach to modeling, in which the basic physical processes involved in the observed system are first identified. A human expert then uses domain knowledge about the identified processes to write down a proper structure of the model equations.

The equation discovery methods presented in the thesis deal with the problem of integrating the theoretical and empirical approaches to modeling of dynamic systems by integrating different types of theoretical knowledge in the discovery process. Two different types of domain specific modeling knowledge are considered herein. The first concerns basic processes that govern the behavior of systems in the observed domain. The second concerns existing models that are already established in the domain.

In addition, the scope of the existing equation discovery methods is extended toward the discovery of partial differential equations that are capable of modeling both temporal and spatial changes of the state of the observed system.

The newly developed methods are successfully applied to different tasks of modeling real-world systems from artificial and real measurement data in the domains of population dynamics, neurophysiology, classical mechanics, hydrodynamics, and Earth science.

Acknowledgments

I thank my supervisor, Ivan Bratko, for accepting me in his research group and patiently supervising my equation discovery research for almost ten years.

Sašo Džeroski, my co-supervisor and office mate, provided guidance, advice, and countless discussions on every topic that appear in this dissertation. He actually taught me how to become a good researcher. Thank you Sašo.

I am especially grateful to Pat Langley for his encouragement and support. His insightful comments and suggestions strongly influenced the work presented in the thesis.

Thanks to Boris Kompare from Faculty of Civil and Geodetic Engineering in Ljubljana for providing insight into mathematical modeling of population dynamics. Thanks to Christopher Potter, Steven Klooster, and Alicia Torregrosa from NASA-Ames Research Center for providing the CASA model and relevant data and to Sven-Erik Jørgensen from Royal Danish School of Pharmacy in Copenhagen for providing Glumsø data. Pat Langley provided extensive language comments on an earlier version of the manuscript. Špela Vintar proof read the extended abstract in Slovene. Thanks to Bernard Ženko for numerous advices and tips about text formating environment LaTeX, which was used for preparing the manuscript.

I thank the members of the Intelligent Systems Department at the Jožef Stefan Institute for providing stimulative working environment. Special thanks to Nada Lavrač and Mili Bauer, who made their best to make our work enjoyable.

Most of the work presented in the thesis was financially supported by the Slovenian Ministry of education, science and sport. I would also like to acknowledge the financial support of the cInQ (Consortium on discovering knowledge with Inductive Queries) project, funded by the European Commission under contract IST-2000-26469.

Contents

1

Introduction

Scientists and engineers build mathematical models to analyze and better understand the behavior of real-world systems. Establishing an acceptable model for the 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 system variables that are important for modeling, and formulating the model itself. This thesis deals with the automated modeling task, i.e., the task of formulating a model on the basis of observed behavior of the selected system variables. We propose a framework for automated modeling of real-world systems based on equation discovery methods.

Equation discovery is an area of machine learning (Langley, 1995; Mitchell, 1997) that studies methods for computational discovery of quantitative laws, expressed in the form of equations, in collections of measured data. Equation discovery methods are mainly used for automated modeling of real-world systems from measurements and observations. The area of equation discovery is strongly related to the area of system identification (Ljung, 1993). However, most system identification methods work under the assumption that the structure of the model equations is known, i.e., provided by a human expert, and are concerned with determining the proper values of the constant parameters in the model. The focus of equation discovery methods, on the other hand, is on the problem of identifying both an adequate structure of the model equations and appropriate values of the constant parameters.

State of the art equation discovery methods can be used to discover algebraic (Langley et al., 1987; Kokar, 1986; Falkenhainer & Michalski, 1990; Zembowicz & Żytkow, 1992; Washio & Motoda, 1997) or ordinary differential equations (Todorovski, 1993; Džeroski & Todorovski, 1995; Križman, 1998; Todorovski, 1998; Todorovski & Džeroski, 1997). While algebraic equations are mainly used to establish models of static systems that have reached an equilibrium state, ordinary differential equations can be used for modeling the behavior of dynamic systems, i.e., systems that change their state over time (Gershenfeld, 1999). Ordinary differential equations are limited to modeling changes of the observed system along a single (typically temporal) dimension. In order to model changes of the state of the observed system along several dimensions (e.g., spatial and temporal), the extended formalism of partial differential equations can be used (Gershenfeld, 1999; Murray, 1993).

Most of the work in the area of equation discovery is concerned with assisting the empirical approach to modeling physical systems. Following this approach, the observed system is modeled on a trial-and-error basis to fit observed data. The scientist first chooses a structure of the model equations from some general class of structures (such as linear or polynomial) that is believed to be adequate, fits the constant parameters, and checks whether the simulation matches the observed data. If not, the procedure is repeated until an adequate model is found. None of the available domain knowledge about the observed system (or a very limited portion thereof) is used in the modeling process. The empirical (data-driven) approach is in contrast to the knowledge-driven theoretical approach to modeling, where the basic physical processes involved in the observed system are first identified. A human expert then uses domain knowledge about the identified processes to write down a proper structure of the model equations. Finally, the values of the constant parameters of these equations are fitted against the observed data using standard system identification methods (Ljung, 1993).

The empirical approach to modeling is completely data-driven, and the models obtained following this approach do not necessarily reveal the processes that govern the behavior of the observed system. These models are referred to as *black-box* models, as opposed to *white-box* models, i.e., models that reveal the structure of the observed system and the processes that govern its behavior. White-box models are obtained following the theoretical knowledge-driven approach to modeling. While most of the equation discovery methods are data-driven, our goal is to develop knowledge-driven methods capable of integrating domain-specific modeling knowledge into the process of equation discovery. Such methods would allow for integration of theoretical and empirical approaches to automated modeling of real-world systems and establishing white-box models of the observed phenomena.

1.1 Goals

The goal of the thesis is to develop new methods that will extend the scope of equation discovery along the two dimensions presented in Figure 1.1.

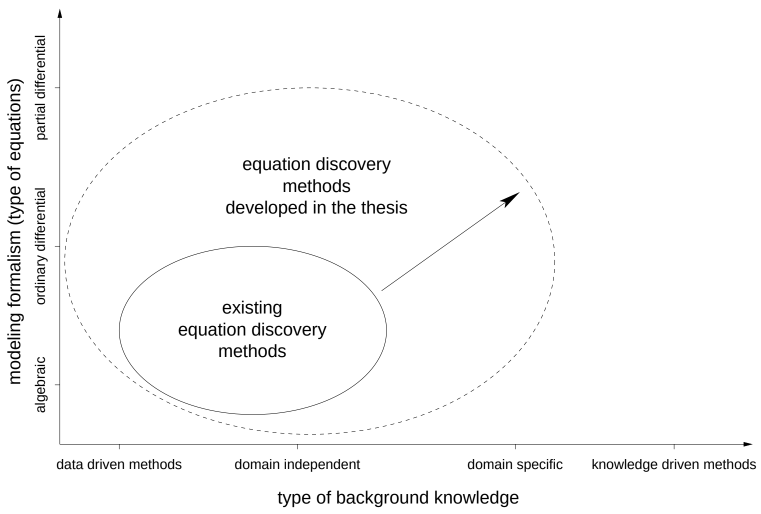


Figure 1.1: Extending the scope of the equation discovery methods.

The first dimension represents the amount and type of knowledge employed in the process of equation discovery. It starts with purely data-driven methods that use no knowledge in the process of equation discovery. The dimension continues with methods that use domain-independent knowledge (e.g., knowledge about measurement units of the system variables), through the methods where domain-specific modeling knowledge is used, toward purely knowledge-driven methods. Existing equation discovery methods range from purely data-driven to such that are capable of integrating limited portions of domain knowledge. The goal here is to extend the amount of the domain knowledge that can be used in the process of equation discovery. The goal is achieved by developing an automated modeling framework based on new equation discovery methods, capable of integrating modeling knowledge from the domain of use, supplied by a human expert. Two different types of domain-specific modeling knowledge are considered in the thesis. The first concerns basic processes that govern the behavior of systems in the observed domain. The second concerns existing models that are already established in the domain. Using modeling knowledge in the process of equation discovery would allow for integration of empirical and theoretical approach to modeling and establishing comprehensible white-box models of the observed real-world systems.

The second dimension represents the complexity of the formalism for representing equation based models. It ranges from algebraic equations, capable of modeling stable states of the observed system that reached its equilibrium, to ordinary and partial differential equations capable of modeling changes of the observed system along a single (temporal) or several (temporal and/or spatial) dimensions, respectively. The goal here is to to extend the class of equations that existing equation discovery methods can handle. This goal is achieved by developing a new equation discovery method capable of discovering partial differential equations that can be used for modeling both temporal and spatial changes of the state of the observed system.

1.2 Original contributions

The thesis contributes to several important aspects of equation discovery methods that improve their applicability to tasks of modeling real-world dynamic systems. First, the scope of existing equation discovery methods is extended to handle partial differential equations. Second, a formalism for encoding modeling knowledge from a specific domain of use is proposed. An equation discovery method is developed that is capable of integrating encoded knowledge in the process of equation discovery. Finally, an equation discovery method is developed capable of starting the discovery process with an existing model of the observed system in order to improve its fit to newly measured data. The contributions stated above will be further elaborated in the following subsections.

1.2.1 Discovery of partial differential equations

Existing equation discovery methods are capable of discovering algebraic and ordinary differential equations. The latter are limited to modeling the change of the observed system state along a single (usually the temporal) dimension. On the other hand, partial differential equations (PDEs) can be used for modeling the change of the observed system state along several dimensions. In the thesis, a method capable of discovering PDEs is presented. The method is based on a transformation of the task of PDEs discovery to the simpler task of discovery of ordinary algebraic equations or ordinary differential equations. In both cases, the transformation allows for the application of existing equation discovery methods to the task of discovery of PDEs. The presented method for PDEs discovery extends the scope of equation discovery methods toward modeling systems whose state changes along several (e.g., temporal and spatial) dimensions.

1.2.2 Integration of domain-specific knowledge in the equation discovery process

Existing equation discovery methods support the empirical approach to modeling where none or a very limited portion of the knowledge from the domain of interest is used. The empirical approach produces black-box models that do not reveal the processes that govern the behavior of the observed system. In the thesis, a formalism for encoding modeling knowledge from a specific domain of interest is presented. The modeling knowledge is organized in the form of a taxonomy of basic processes that govern the behavior of the dynamic systems in the domain at hand. For each process class in the taxonomy, a number of alternative equation model fragments, used by human experts in the domain, are specified. Also, the formalism encodes knowledge about how to combine these models of individual basic processes into a single model of the whole system. A new equation discovery method capable of using the encoded knowledge to constrain the space of candidate equations is presented. This method allows for the integration of the theoretical and empirical approaches to modeling real-world systems. The method can be used for automated building of white-box models that reveal the structure of the observed system and the processes that govern its behavior.

1.2.3 Revision of equation based models

An important type of domain knowledge takes the form of existing model of the observed system, already developed by domain experts. While existing equation discovery methods are not capable of integrating existing models in the process of discovery, a new method is presented that starts the process of equation discovery from a given initial model. The method discovers a revised model that fits the newly measured data better than the initial one. The method follows the minimal revision principle used in earlier theory revision systems. The principle states that among theories (models) with similar accuracy, the ones that are as similar as possible to the initial theory (model) are to be preferred. The usual minimality of change heuristics used in (logical) theory revision are adapted for the case of revising models based on equations.

1.3 Organization of the thesis

The manuscript of the thesis is organized as follows. This chapter has provided an introduction to the material presented in the thesis. It has specified the goals of the thesis and summarized its main contributions.

Chapter 2 gives a brief overview of the previous research that is related to the work presented in the thesis. The overview includes areas of mathematical modeling, system identification, knowledge-driven machine learning, equation discovery, and qualitative reasoning.

The next four chapters present the main contributions of the thesis. Chapter 3 presents two methods for discovering partial differential equations and their empirical evaluation on several tasks of reconstructing known PDE models from the population dynamics and neurophysiology domains. Chapter 4 presents the framework for automated modeling based on integration of domain-specific modeling knowledge in the process of equation discovery. In the chapter, we illustrate the usability of the framework by encoding modeling knowledge from domains of population dynamics, chemical kinetics, classical mechanics as well as domain-independent modeling knowledge based on the measurement units of the observed system variables. We present the results of the empirical evaluation of the modeling framework in Chapter 5. The experiments include applications of the framework to the tasks of modeling dynamic systems from the domains of population dynamics, classical mechanics, and hydrodynamics. The next Chapter 6 presents an equation discovery method capable of revising existing models, based on equations, and its application to the task of revising an environmental model of the net production of carbon by terrestrial plants in the Earth ecosystem using real-world measurements data.

Finally, Chapter 7 concludes the thesis with summary and discussion of directions for future research.

2

Background and related work

In this chapter we give a brief overview of the research work that presents a relevant background or is related to the work presented in the thesis. Note that research on modeling and simulation of dynamic systems is vast and present in many different scientific domains. It is beyond the scope of this thesis (and, to be honest, beyond the abilities of the author) to give a comprehensive overview of all related work. Thus, in this chapter we will concentrate mainly on research closely related to the work presented latter.

2.1 Mathematical modeling and system identification

Scientists formulate mathematical models in order to analyze and better understand the behavior of real-world systems (Gershenfeld, 1999). Mathematical models can integrate potentially vast collections of observations and measurements into a single entity. They can be used for simulation and prediction of the future behavior of the observed system under varying conditions. A very important aspect of mathematical models is that they can reveal the processes that govern the behavior of the observed system or phenomena.

The first milestone in the process of modeling a real-world system is the choice of the modeling formalism. Ordinary differential equations (ODEs) are one of the most widely accepted formalisms for modeling dynamic systems, i.e., systems that change their state over time (Gershenfeld, 1999). ODEs have the limitation of modeling changes of the observed system over one (typically temporal) dimension only. On the other hand, the extended formalism formalism of partial differential equations can be used to model changes of the state of the observed system over several dimensions (e.g., spatial and temporal) (Gershenfeld, 1999; Murray, 1993).

Note that the models based on ordinary and partial differential equations are purely deterministic models. Simulation of deterministic models always produce a unique and exact behavior of the observed system. However, in reality many phenomena are not deterministic. Probabilistic and stochastic models can be used to model the behavior of the observed system as well as the uncertainty of the predicted behavior. Different formalisms can be adapted to represent probabilistic models, stochastic differential equation being one of them (Gershenfeld, 1999).

In the thesis, we deal with deterministic models based on ordinary and partial differential equations. There are two main aspects to the process of establishing of an equation based models of the observed real-world system. First, an appropriate structure has to be determined for the equations involved (the model identification problem). Second, acceptably accurate values for the parameters are to be determined (the parameter estimation problem). Research in the area of system identification focuses on developing methods for solving the parameter estimation problem (Ljung, 1993). Most of the system identification methods make one of the following two assumptions. The first is that the structure of the model is provided by a human expert in the domain of interest. The second assumption is that the structure of the model is chosen from some general well-known class of model structures, such as linear equations, polynomials, or neural networks with different topologies.

The modeling approach that makes the first assumption, that the model identification problem is solved by a human expert, is also known as theoretical approach to modeling. Following this approach, the expert first identifies the processes that govern the behavior of the observed system. Then, using domain-specific knowledge about the identified processes, the expert writes down a proper structure of the model equations. In contrast to the knowledge-driven theoretical approach, the empirical approach adopts a data-driven trial-and-error paradigm. The scientist first chooses a structure of the model equations from some general class of structures (such as linear or polynomial) that is believed to be adequate, fits the constant parameters, and checks whether the simulation match the observed data. If not, the procedure is repeated until an adequate model is found. A very limited portion (if any) of the domain knowledge about the observed system is used in the modeling process. Consequently, the models obtained following the empirical approach do not necessarily reveal the processes that govern the behavior of the observed system. These so-called *black-box* models are typically obtained using the data-driven empirical approaches to modeling. They are in contrast to the *white-box* models that reveal the physical structure of the observed system and the processes that govern its behavior.

2.2 Background knowledge in machine learning

Studies in machine learning have shown that using background expert supplied knowledge from the domain of interest leads to better performance of learned models on novel test examples (Pazzani & Kibler, 1992). This is especially true in complex domains where the space of possible models is huge and the amount of training examples is limited. Although these results are well known, many machine learning methods do not allow for explicit integration of knowledge in the learning process. The expert knowledge is usually involved in the preparation or preprocessing phase, when the set of variables (features) important for modeling of the observed phenomena are chosen, or after the learning process is over and learned model is interpreted.

Notable exception are learning methods developed within the area of inductive logic programming (ILP) (Lavrač & Džeroski, 1994). The use of background knowledge there is explicit and background knowledge is part of the learning task specification. ILP methods deal with induction of first-order logic programs from examples and background knowledge is also represented in first-order logic.

The use of background knowledge is closely related to the notion of inductive bias (Nédellec et al., 1996), which refers to any kind of basis for choosing one generalization over another. Different kinds of inductive bias include language bias (i.e, the definition of the hypothesis space), search bias (i.e., which part of the hypothesis space is searched and how), and validation bias (i.e., acceptance or stopping criterion). Depending on how bias is specified within a learning method, it may be non-declarative (built-in), parametrized, or declarative. Typical methods for inducing decision trees (Quinlan, 1993), are examples of methods with non-declarative language bias, since they explore the fixed hypothesis space of decision trees built using variables from the given data set. Parametrized bias would let the user influence the bias by setting some of its parameters, such as the depth of decision trees. Finally, a declarative language bias lets the user tailor the search space according to background knowledge. Thus, declarative language bias provides a powerful way of integrating background knowledge in the process of learning. Nédellec et al. (1996) provide an overview of declarative bias formalisms used in ILP. Note however that these formalism are developed for the concepts and models expressed in first-order logic and are not directly applicable to the task of building models based on algebraic and differential equations.

Another type of background knowledge is existing theories already available in the domain of interest. Theory revision methods such as EITHER (Ourston & Mooney, 1994) and KBANN (Towell & Shavlik, 1994) start with an existing theory and revise it in order to improve its accuracy on newly acquired training data. Again, revision of theories expressed in first-order logic has been explored also within the area of ILP, as reviewed by Wrobel (1996). However, theory revision research is mainly concerned with the revision of theories expressed in propositional or first-order logic. Therefore, the developed methods are not directly applicable to the task of revising models based on equations.

2.3 Equation discovery

Equation discovery (Langley et al., 1987) is the area of machine learning (Langley, 1995; Mitchell, 1997) that aims at developing methods for computational discovery of quantitative laws, expressed in the form of equations, in collections of measured data. Equation discovery methods are mainly used for automated modeling of real-world systems from measurements and observations. The area of equation discovery is strongly related to the area of system identification (Ljung, 1993), but they differ somehow in their focuses. While the area of system identification emphasizes the parameter estimation aspect of the modeling (i.e., determining the proper values of the constant parameter of a given model structure or a class of model structures), the research in equation discovery focuses on identifying the proper structure of the model equations. Equation discovery methods usually use standard system identification (Ljung, 1993) or non-linear optimization methods (Press et al., 1986) for solving the parameter estimation problem.

Early approaches to equation discovery dealt with rediscovering empirical laws from the history of science. Initial methods were influenced by the methods and approaches used by human scientists. Experiments with early equation discovery systems showed that many apparently complex laws can be discovered by using simple heuristics.

Bacon (Langley et al., 1987) was the pioneer among equation discovery methods. It incorporated a set of data-driven heuristics for detecting regularities (constancies and trends) in data and for formulating hypotheses based on them. Hypotheses in Bacon are proposed at different levels of description. At each level of description, all but two variables are held constant and hypotheses connecting the two changing variables are considered. Using a small set of data-driven heuristics, the method was able to rediscover a number of physical laws including the ideal gas law, the law of gravitation, the law of refraction and Black's specific heat law (Langley et al., 1987). In the process of development from early approaches to the present, the focus of the equation discovery methods has shifted from rediscovering known quantitative laws and models to discovery of new quantitative laws and automated modeling of real-world systems (Langley, 2000).

In the rest of this section we will review the development of equation discovery methods from two perspectives: the amount of background knowledge integrated into the process of equation discovery and the ability to discover different types of equations.

2.3.1 Background knowledge and language bias

As mentioned in the previous section, language bias can be seen as one way to incorporate background knowledge into learning methods. In the case of equation discovery, language bias can be used to constrain the space of possible equations to those that are not in conflict with the existing knowledge about the domain. For example, consider the case where the measured variables of an observed system are not dimensionless, so that some algebraic combinations of the system variables, such as addition or subtraction of mass and energy, are not valid. Beyond this simple example, there are possibilities for more sophisticated inconsistencies of equation structures with background knowledge about the domain of the observed system. Thus, in order to make equation discovery methods applicable to problems from different domains, their language bias should be made declarative, i.e., allow the user to influence or specify the space of candidate equations. In this way, the user can tailor the space of possible equations to the specific domain and modeling task at hand.

However, the language bias of early equation discovery methods is usually non-declarative and often takes the form of pre-defined, reasonably small class of possible equation structures, such as polynomials or trigonometric functions. Bringing this to the extreme, Scheffer (1993) proposes the use of very strong pre-defined class of only 5 candidate equation structures. The bias was implemented in the E* method for discovery of bivariate equations, which relate only two variables. Scheffer (1993) reported the results of experiments based on 155 different cases systematically collected from issues of the Physical Review journal. Using this very strong language bias, E* was able to reconstruct the correct models in about a third of the test cases.

Two other equation discovery methods, EF (Zembowicz & Żytkow, 1992) and LAGRANGE (Todorovski, 1993; Džeroski & Todorovski, 1995), use polynomials as a pre-defined language bias. They let the user influence the space of candidate equations by specifying values of different parameters, such as the maximal degree of the polynomial or maximal number of terms on the right-hand side of the equation (Džeroski & Todorovski, 1995). In addition, EF (Zembowicz & Żytkow, 1992) lets the user specify functions that can be used to introduce new variables that appear in the polynomials. LAGRANGE also allows the use of trigonometric functions for introducing new variables. However, note that these limited ways of specifying the language bias do not allow for integration of the domain-specific modeling knowledge in the process of equation discovery.

One type of (domain-independent) knowledge used by existing equation discovery methods to constrain the space of possible equations is information about measurement units of the observed system variables. The equation discovery method COPER (Kokar, 1986) uses such knowledge and considers equations that properly combine variables and terms with different measurement units. The constraints used in COPER are based on dimensional analysis theory (Giordano et al., 1997). The equation discovery method SDS (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. The relation between domain-independent knowledge used in SDS and the domain-specific knowledge used in the framework presented in the thesis will be discussed in Chapter 4.

Knowledge about measurement units or the measurements scale types thereof is domain independent. Experts from a specific domain of interest can usually provide much more modeling knowledge about the system or domain 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 order to incorporate this knowledge in the process of equation discovery, we should provide the user with more sophisticated declarative bias mechanisms. First steps in this direction were taken in by LAGRAMGE (Todorovski, 1998; Todorovski & Džeroski, 1997), which used the formalism of context-free grammars to specify the space of possible equations. Grammars are general enough to express many different types of domain-specific knowledge. 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 (Todorovski, 1998). In another example, knowledge about the basic processes that govern population dynamics was 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 not very easy for domain experts to express or encode their domain knowledge about in the form of a grammar. Another problem with grammars is that they are usually task specific. That means 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., another lake with a slightly different set of observed variables). Nevertheless, the use of grammars to specify the space of possible equations is crucial for all the methods developed in the thesis. In Chapter 4, we propose a method capable of integrating domain specific modeling knowledge into the process of equation discovery by transformation of the knowledge into grammars. In Chapter 6, we use grammars to integrate existing models from the domain in the process of equation discovery.

2.3.2 Discovery of differential equations

The most important contribution of the equation discovery system LAGRANGE (Todorovski, 1993; Džeroski & Todorovski, 1995) is an extension of the scope of equation discovery to differential equations. LAGRANGE can discover a set of algebraic and/or differential equations involving more than two variables from observational data only. The basic idea of LAGRANGE is to extend the set of system variables with numerically calculated time derivatives of the given variables. Then, an arbitrary equation discovery method can be used to discover algebraic equations in the extended set of variables. However, an algebraic equation in the extended set of variables that includes time derivatives may actually be an ordinary differential equation. A similar approach will be used in Chapter 3 to discover partial differential equations.

This simple approach has a major drawback of introducing large errors by numerical differentiation (Press et al., 1986), which makes LAGRANGE very sensitive to noisy data. The equation discovery method GOLDHORN proposes the use of numerical integration instead of differentiation to avoid calculation of the highest order derivatives (Križman, 1998). The improvement is based on the fact that numerical integration is much more precise and stable than numerical differentiation. The successor of LAGRANGE, LAGRAMGE is also capable of discovering ordinary differential equations using the GOLDHORN method to avoid numerical calculation of the highest order derivatives (Todorovski & Džeroski, 1997; Todorovski, 1998).

2.3.3 LAGRAMGE

The equation discovery method LAGRAMGE is capable of discovering a single ordinary differential equation of the form $\dot{v_d} = E$, where v_d is a user-specified dependent system variable, $\dot{v_d}$ is the time derivative of v_d, and E is an expression that can be derived using a user provided context-free grammar G (Hopcroft & Ullman, 1979). LAGRAMGE can employ exhaustive or heuristic search through the space of equation structures specified by the grammar G.

Each equation structure considered during the search contains one or more generic constant parameters. In order to obtain the equation out of the equation structure, the values of these generic constant parameters are fitted against the measurements of the observed system variables. The quality of the obtained equation is then evaluated using the SSE (sum of squared errors) heuristic function:

$$\mathrm{SSE}(\dot{v_d} = E) = \sum_{i=1}^{m} (v_d(i) - \hat{v_d}(i))^2,$$

where $v_d(i)$ is the measured value of the v_d variable at i-th measurement point, $\hat{v_d}(i)$ is the value of the v_d at the same measurement point i, but obtained with simulating the equation $\dot{v_d} = E$, and m is the number of measurement points. The SSE heuristic function measures the discrepancy between the measured values of the dependent variable and the value obtained with simulating the equation.

An alternative MDL heuristic function can be used in LAGRAMGE that takes into account the complexity of the equation:

$$\mathrm{MDL}(\dot{v_d} = E) = \mathrm{SSE}(\dot{v_d} = E) + \frac{l(E)}{10 \cdot l_{\max}} \cdot \mathrm{SSE}(\dot{v_d} = E_0),$$

where $l(E)$ is the length of the expression on the right-hand side of the equations (expressed in number of terminal symbols), $l_{\max}$ is the maximal length of expression that can be derived by the grammar, and E_0 is the simplest (and first) expression derived by the grammar. The second term in the equation for MDL introduces a penalty for the complexity of the equation. Thus, the MDL heuristic function introduces a preference toward simpler equations in LAGRAMGE.

As output, LAGRAMGE returns the best equation encountered during its heuristic search, according to the SSE or MDL heuristic function. We present further details of the LAGRAMGE algorithm elsewhere (Todorovski & Džeroski, 1997; Todorovski, 1998).

2.4 Qualitative reasoning and compositional modeling

Many artificial intelligence approaches to the task of modeling physical systems fall within the area of qualitative reasoning (QR) (Kuipers, 1994), which deals with the problem of reasoning about physical systems in the presence of incomplete knowledge. The representational formalisms in this framework allow for qualitative descriptions (models) of the mechanisms in the physical world that emphasize the *qualitative* differences and ignore others. The *qualitative* differences are those that are important for the observer/modeler of the system.

As in traditional approaches to mathematical modeling, the first milestone in the qualitative modeling process is the choice of the modeling formalism. Several alternatives have been proposed in the QR literature. The most well known is the one of qualitative differential equations (QDE), associated with the QSIM algorithm for simulating qualitative models (Kuipers, 1994). The QDE formalism lets the modeler to abstract the ordinary differential equations of the model, in order to obtain QDEs.

While QDEs and the QSIM framework were proposed primarily to support the representation and simulation of qualitative models, Forbus (1984) qualitative process theory (QPT) provides a framework for building qualitative models. In this framework, the models of physical systems are organized around the central notion of physical *processes*. A process is specified by the components of the system to which the process applies, internal and external conditions for the activity of the process, constraints on the parameters of its components, and effects of the process on these parameters. The automated modeling framework QPC (Farquhar, 1993) uses (and extends) the QPT representation to organize domain-specific knowledge into three parts: an ontology of objects or components of the system, a library of model fragments that specify models of processes or components of the observed system, and knowledge about how to compose models of the whole system out of the model fragments. QPC uses QDEs and QSIM for representing and simulation qualitative models.

The QPC modeling framework follows the paradigm of compositional modeling (Falkenheiner & Forbus, 1991), an automated approach to building qualitative models from observations in presence of domain-specific modeling knowledge. In the compositional modeling, knowledge is organized as a library of model fragments. Given a modeling task specification (or scenario), compositional modeling methods compose a set of appropriate fragments into a model that is suitable for modeling the observed system. The obtained model is evaluated by qualitative simulation. The compositional modeling approach is mainly applied to the tasks of building qualitative models. For example, Garrett et al. (2004) apply this approach to the task of inducing qualitative models of chemical reaction pathways from noisy measurement data.

Although the concepts introduced within the QR area are also relevant for automated building of quantitative models of real-world systems, this idea has not been widely explored. A notable exception is the PRET reasoning system for automated modeling of dynamic systems (Bradley et al., 2001; Stolle, 1998), which employs two kinds of knowledge. The first is domain-specific knowledge in the form of "conservation rules", such as Kirchoff's law in the domain of electrical circuits, which specifies that the sum of input and output currents at any observed point in the circuit is zero. Similarly, the force balance rule in the mechanics specifies that the sum of forces at any observed coordinate of the mechanical system is zero. These rules are more general than domain knowledge about model fragments used in compositional modeling approaches, and constrain the space of possible models much less. PRET compensates this lack of constraints by using second kind of domain-independent knowledge about models based on ordinary differential equations. An example of such rule specifies that "a model with oscillatory behavior must be second-order". This kind of rule allows for very efficient elimination of inappropriate models by high-level qualitative reasoning. We further discuss the relation of PRET to our framework for automated modeling in Chapter 4.

3

Discovery of partial differential equations

Ordinary differential equations (ODEs) are one of the most widely accepted formalisms for modeling dynamic systems, i.e., systems that change their state over time. Several equation discovery methods allow for the discovery of ODEs (Todorovski, 1993; Džeroski & Todorovski, 1995; Križman, 1998; Todorovski, 1998; Todorovski & Džeroski, 1997). These enable the application of equation discovery to the omnipresent task of modeling real-world dynamic systems. However, note that ODEs can only model changes in the observed system over one dimension, typically time. In order to model changes of the observed system over several dimensions (e.g., spatial and temporal), an extended formalism of partial differential equations (PDEs) should be used. PDEs are one of the most powerful and widely accepted analytical formalisms for modeling biological systems, being used routinely to model physiological transport processes, such as gas exchange mechanisms and fluid flow in arteries, predator-prey behavior, the movement and growth of carcinogenic cells, viral infection in humans, animal coat patterns, fluid-flow in arteries, transmission of the electric signals along and between nerve cells, etc. (Murray, 1993).

In this chapter, we present two methods that are capable of PDE discovery. The approach to discovering PDEs is based on the transformation principle already used for discovering ODEs in the LAGRANGE method described earlier. We consider two different transformation approaches. In the first, we extend the initial set of system variables with their partial derivatives with respect to the given temporal and spatial dimensions of the observed system. The partial derivatives are calculated using numerical differentiation methods (Press et al., 1986). Thus, the original problem of PDE discovery in the original set of system variables is transformed to the problem of discovery of algebraic equations in the extended set of variables, where an arbitrary equation discovery method can be used. The proposed methodology is straightforward, but the transformed problem tends to be much more complex than the original one, especially for systems with many dimensions.

The increased complexity is due to the increased number of variables in the extended set, which can significantly enlarge the space of possible equations. In order to constrain the space of possible PDEs, we also investigate an alternative approach, where the problem of discovery of PDEs is first decomposed into a number of ODE discovery problems. The idea here is to take slices of the training data for fixed values of all but dimension and search for ODEs in these slices. The ODE structures that are most frequently discovered in different slices are used to constrain the space of candidate PDEs. The discovery problem in the constrained (and therefore less complex) space can be addressed following the first straightforward approach.

We evaluate these approaches empirically on several tasks of reconstructing known PDE models. These include the well-known FitzHugh-Nagumo model (FitzHugh, 1961; Nagumo et al., 1962) for the conductance of sodium and potassium ions across the cell membrane, which plays an important role in the transfer of signals between nerve cells. The experiments show that the applicability of the first approach is limited to the re-construction of simple PDE models, while the second approach can reconstruct the structure of the FitzHugh-Nagumo model from simulated data.

The chapter is organized as follows. We present the relevant background for partial differential equations in Section 3.1 and then define the problem of PDE discovery in Section 3.2. Section 3.3 presents the straightforward approach to PDE discovery and discusses its limitations and inability to re-construct complex PDE models. Section 3.4 presents the two-stage approach to discovering complex PDEs, along with the results of its empirical evaluation. Section 3.5 summarizes the chapter.

3.1 Partial differential equations

Ordinary differential equations (ODEs) are used to describe the behavior of dynamic systems, i.e., systems whose state changes over time. In ordinary differential equations, time is the only dimension along which change of state is considered. The time change of the variable u is assessed through the (ordinary) derivative of u with respect to time t, defined as:

$$\frac{du}{dt} = \frac{d}{dt}u = \lim_{\Delta t \to 0} \frac{u(t + \Delta t) - u(t)}{\Delta t},$$

where $u(t + \Delta t)$ and $u(t)$ denote the values of variable u at time points $t + \Delta t$ and t, respectively. This is the first-order (time) derivative of u. The second derivative is defined as:

$$\frac{d^2u}{dt^2} = \frac{d}{dt}(\frac{du}{dt}).$$

The symbols $\dot{u}$ and $\ddot{u}$ are also used to denote the first and second order time derivative of u, respectively, i.e., $\dot{u} = du/dt$ and $\ddot{u} = d^2u/dt^2$.

Consider now another variable v that changes its state along two dimensions t (time) and x (one-dimensional space). The change of v along the space dimension is assessed through the *partial* derivative of v with respect to x, which is defined as:

$$\frac{\partial v}{\partial x} = \lim_{\Delta x \to 0} \frac{v(t, x + \Delta x) - v(t, x)}{\Delta x}.$$

Similarly, we can define the partial derivative of v with respect to time t. These two derivatives ($\partial v/\partial t$ and $\partial v/\partial x$) are the two first-order partial derivatives of v. There are three second-order derivatives of v. The one with respect to t and x, which describes the change of v along both dimensions, is defined as:

$$\frac{\partial^2 v}{\partial t \partial x} = \frac{\partial}{\partial t}(\frac{\partial v}{\partial x}) = \frac{\partial}{\partial x}(\frac{\partial v}{\partial t}).$$

The other two second-order derivatives of u are $\partial^2 v/\partial t^2$ and $\partial^2 v/\partial x^2$.

A partial differential equation (PDE) is an equation that involves one or more partial derivatives of a variable with respect to more than one dimension. The order of a PDE is the order of the highest-order derivative that appears in the equation. An example of a first-order PDE is the non-linear first-order wave equation:

$$\frac{\partial u}{\partial t} + u \cdot \frac{\partial u}{\partial x} = 0,$$

and an example of a second-order PDE is the linear second-order wave equation:

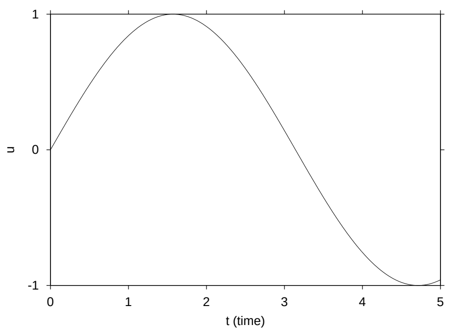
$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0.$$

The latter second-order wave PDE is used to model a vibrating ideal elastic string (e.g., guitar string) fixed at both ends. If the string is distorted at some initial time and then allowed to vibrate, the wave displacement along the string in time will be a solution of this equation. It is also widely used for modeling other physical systems, such as propagation of sound waves in a tube.

Given a model of a dynamic system in the form of one or more ODEs, the behavior of the system can be simulated/derived by solving the these equations. To solve ODEs, an initial state must be provided. A general numerical integration method, such as Runge-Kutta integration (Press et al., 1986; Gershenfeld, 1999), can then be applied.

For PDEs, the situation is more complicated. Boundary conditions, which are more complex than just specifying an initial state, are required. A similar range of choices is available for performing the time integration as for ODEs, while the spatial derivatives are typically handled using either finite difference or finite element methods (Gershenfeld, 1999). In either case, a suitable spatial mesh must be generated, with a finer mesh typically giving a smaller numerical error during simulation, but requiring a larger computational effort. Many PDE problems, including the FitzHugh-Nagumo model considered latter in this chapter, are also non-linear and may be very sensitive to slight changes in initial conditions or display different behavior for slight variations in equation parameters. Where experimental systems display such complex behavior, it can be very difficult to determine the appropriate form of the equations and may require lengthy and painstaking observational work in the laboratory, as was the case for Hodgkin and Huxley (1952).

ODE wave	PDE wave
$\dfrac{d^2u}{dt^2} = -u$	$\dfrac{\partial^2 u}{\partial t^2} = \dfrac{\partial^2 u}{\partial x^2}$
$u(0) = 0, \dfrac{du}{dt}(0) = 1$	$u(0,x) = e^{-x^2}, \dfrac{\partial u}{\partial t}(0,x) = 0$



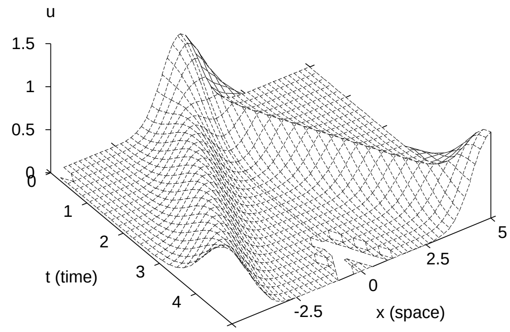


Figure 3.1: Simulation of an ordinary differential wave equation (left-hand side) and partial differential wave equation (right-hand side).

A comparison of two simple wave models, one of them based on ODEs and the other based on PDEs, is given Figure 3.1. The left-hand side of the figure presents an ODE that can be used to model the time change of the position of a pendulum. The initial condition is simple: it specifies the initial position of the pendulum $u(0)$ and its initial velocity $\frac{du}{dt}(0)$. The right-hand side of the figure presents a PDE that can be used for modeling the vibration of an elastic string fixed at both ends. In contrast with the ODE, the simulation of the PDE gives insight into the spatial propagation of a vibration along the string (as the graphs in Figure 3.1 illustrate). However, the initial conditions necessary for the simulation of the PDE model are more complex. They must provide the model of the initial impulse that causes the vibration of the string. For the simulation results presented in Figure 3.1, we used a narrow Gaussian pulse of the form $u(0,x) = e^{-x^2}$ as an initial condition.

3.2 Problem definition

The problem of discovering partial differential equations can be formalized as follows:
Given

- a set of variables of the observed system $\mathcal{V} = \{V_1, V_2, \ldots, V_n\}$, observed or measured along
- a set of dimensions $\mathcal{D} = \{D_1, D_2, \ldots, D_d\}$
- where the table of m measurements of the system variables takes the form:

$$\begin{array}{cccc|cccc}
D_1 & D_2 & \ldots & D_d & V_1 & V_2 & \ldots & V_n \\
\hline
d_{1,1} & d_{2,1} & \ldots & D_{d,1} & V_{1,1} & V_{2,1} & \ldots & V_{n,1} \\
d_{1,2} & d_{2,2} & \ldots & D_{d,2} & V_{1,2} & V_{2,2} & \ldots & V_{n,2} \\
\vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\
d_{1,m} & d_{2,m} & \ldots & D_{d,m} & V_{1,m} & V_{2,m} & \ldots & V_{n,m}
\end{array}$$

find a (set of) PDE equation(s) that minimizes the discrepancy between the measured values of the system variables and their values obtained by simulation of the discovered equation(s).

3.3 A straightforward approach to PDE discovery

Our first approach to PDE discovery mirrors the one taken in LAGRANGE (Todorovski, 1993; Džeroski & Todorovski, 1995), which transforms the task of ODE discovery to the task of discovering algebraic equations. The transformation is done by introducing (numerically calculated) time derivatives of the observed system variables as new variables. In a similar manner, we transform the task of PDE discovery to the task of discovering algebraic equations by introducing numerically calculated partial derivatives of the system variables as new variables. In this section, we give a detailed description of the PDED-1 algorithm that implements the straightforward approach. Then, we illustrate its use on two tasks that involve re-constructing PDE based wave models. In conclusion, we will discuss the problems and limitations of the straightforward approach.

3.3.1 The PDED–1 algorithm

Table 3.1 presents the PDED-1 algorithm that implements the straightforward approach to PDE discovery. The algorithm takes as input measurements of the variables of the observed system $\mathcal{V}$ along the dimensions from $\mathcal{D}$, as well as the highest order o of the partial derivative that can appear in the discovered PDEs.

Table 3.1: A straightforward algorithm for discovery of partial differential equations.

procedure PARTIALDERIVATIVES($\mathcal{V}$, $\mathcal{D}$, o)

1 $\mathcal{P} =$

2 **for** $o_c = 1$ **to** o **do**

3 **foreach** multi-set $\mathcal{D}_c \subset \mathcal{D}$ such that $|\mathcal{D}_c| = o_c$ **do**

4 **foreach** $V \in \mathcal{V}$ **do**

5 $\mathcal{P} = \mathcal{P} \cup \{$partial derivative of V with respect to dimensions in $\mathcal{D}_c\}$

6 **endfor**

7 **endfor**

8 **endfor**

9 **return** $\mathcal{P}$

endprocedure

procedure PDED-1($\mathcal{V}$, $\mathcal{D}$, o)

10 $\mathcal{P} =$ PARTIALDERIVATIVES($\mathcal{V}$, $\mathcal{D}$, o)

11 **foreach** $V \in \mathcal{V}$ **do**

12 LAGRAMGE ($\mathcal{V} \cup \mathcal{P}$, $\frac{\partial^o u}{\partial t^o}$, G)

13 **endfor**

endprocedure

In the first step, PDED-1 uses a numerical method (Press et al., 1986) to calculate the partial derivatives of the system variables from $\mathcal{V}$ with respect to the dimensions in $\mathcal{D}$ up to the user specified maximum order of o. This task is implemented in the PARTIALDERIVATIVES procedure (lines 1–9 in Table 3.1). Each partial derivative of a given variable V is calculated using the following procedure. First a fifth degree multivariate polynomial is fitted through the values of V, where the dimensions from $\mathcal{D}$ are used as polynomial variables. The fitted polynomial is then analytically differentiated with respect to the appropriate dimensions. Finally, the derivative of the polynomial is

Table 3.2: The grammar used by LAGRAMGE for discovery of PDEs. It specifies an arbitrary polynomial of the observed variables (denoted by a single non-terminal symbol `Variable`).

```
Polynomial -> Term | Term + Polynomial
Term -> const | const * Product
Product -> Variable | Variable * Product
```

evaluated for the appropriate values of the dimensions to obtain the numerical derivative of V. The numerical calculation of a partial derivative is performed in line 5 of the PDED-1 algorithm.

The first two loops (in lines 2 and 3) enumerate all possible combinations of dimensions up to the maximal order of o. Multi-sets, which allow duplicate elements, with cardinality at most o are used for this purpose. For example, consider a system with two variables $\mathcal{V} = \{u, v\}$ measured along two dimensions of time and space $\mathcal{D} = \{t, x\}$. In case the user specifies $o_{max} = 2$, five multi-sets will be enumerated. Two of these, $\{t\}$ and $\{x\}$, have cardinality 1 ($o_c = 1$) and lead to calculation of the four first-order partial derivatives: $\partial u/\partial t$, $\partial v/\partial t$, $\partial u/\partial x$, and $\partial v/\partial x$. Another three, $\{t, t\}$, $\{t, x\}$, and $\{x, x\}$, with cardinality 2 ($o_c = 2$) lead to the calculation of the six second-order partial derivatives: $\partial^2 u/\partial t^2$, $\partial^2 v/\partial t^2$, $\partial^2 u/\partial t\partial x$, $\partial^2 v/\partial t\partial x$, $\partial^2 u/\partial x^2$, and $\partial^2 v/\partial x^2$.

Once PDED-1 has calculated the set of partial derivatives $\mathcal{P}$, an existing equation discovery method, capable of discovering algebraic equations can be applied to the extended set of system variables $\mathcal{V} \cup \mathcal{P}$. Our current implementation employs LAGRAMGE for this purpose. Recall that LAGRAMGE requires two parameters to be specified, a dependent variable V_d and a grammar G, to discovers equations of the form $V_d = E$, where E is an expression that can be derived using the grammar G. The experiments presented here use the highest-order partial derivatives of the system variables with respect to the time dimension (i.e., $\frac{\partial^o V}{\partial t^o}$) as dependent variables. LAGRAMGE is then employed once for each dependent variable. All the experiments with PDE discovery use the grammar presented in Table 3.2 that specifies an arbitrary polynomial of the extended set of system variables, i.e., the set that includes initial system variables and their partial derivatives.

Reconsider the previous example, in which $\mathcal{V} = \{u, v\}$. LAGRAMGE will be employed two times, first with $\frac{\partial^2 u}{\partial t^2}$ as a dependent variable and then with a $\frac{\partial^2 v}{\partial t^2}$ as a dependent variable on the left-hand side of the equation. In both cases, the right-hand side will be a multivariate polynomial of the observed variables and their partial derivatives.

3.3.2 Experimental evaluation of PDED–1

We evaluated the PDED-1 algorithm on different tasks of re-construction of known PDE based models. In each experiment, we selected a known model and simulated it to obtain training data. Then PDED-1 was applied to the training data with an appropriate setting of the o parameter. We measured the success of PDED-1 in terms of whether the structure of the original model equation is among the ten best equation structures returned.

We obtain the structure of an equation by rewriting it in a canonical form such that the left-hand side contains only the highest-order time derivative and abstracting the constant parameters in the left-hand side to generic constants. Thus, the structure of the equation $\partial u/\partial t + 0.657043 \cdot u \cdot \partial u/\partial x$ is $\partial u/\partial t = -c_1 \cdot u\ \partial u/\partial x$. We do not explore here a logical semantics for this generalization, except to note that two equations will be said to have the same structure if there is a trivial rewrite of the abstracted coefficients that makes the structures identical. For example, $\partial u/\partial t = -0.657043 \cdot u\ \partial u/\partial x$ and $\partial u/\partial t = u\ \partial u/\partial x$ have the same structure.

Wave equations

PDED-1 successfully re-constructs the structure of textbook equations, including the non-linear first-order wave equation

$$\frac{\partial u}{\partial t} + u \cdot \frac{\partial u}{\partial x} = 0$$

and the second-order wave equation

$$\frac{\partial^2 u}{\partial t^2} - \frac{\partial^2 u}{\partial x^2} = 0$$

from simulated data. The actual forms of the recovered equations are:

$$\frac{\partial u}{\partial t} = -0.657043 \cdot u \cdot \frac{\partial u}{\partial x}$$

and

$$\frac{\partial^2 u}{\partial t^2} = -1.05818 \cdot 10^{-5} + 1.00098 \cdot \frac{\partial^2 u}{\partial x^2}.$$

The error in the coefficient for the first-order wave equation is high. This is due to the large errors in the numerically calculated partial derivative of u. In general, numerical calculation of partial derivatives introduces large errors, as does the numerical calculation of ordinary derivatives, especially if the measurements are sparse (taken on a coarse mesh).

Predator-Prey Model

We next tested PDED-1 on the slightly more complex task of reconstructing a population dynamics model. The predator-prey model describes situations such as the population of rabbits and foxes on an island, where foxes prey on rabbits and rabbits have an unlimited supply of food. Variable u is the dimensionless population of the prey, v is the dimensionless population of the predator. This model allows for spatial variations so that the predators have to move to catch the prey, and the prey can move to evade the predator:

$$\begin{aligned}\frac{\partial u}{\partial t} &= u(1-u-v) + 0.1 \cdot \frac{\partial^2 u}{\partial x^2}\\ \frac{\partial v}{\partial t} &= v(u-0.02) + 0.1 \cdot \frac{\partial^2 v}{\partial x^2}.\end{aligned}$$

The training data set was generated with a simple simulation method for PDEs, using numerical approximations of the partial derivatives. The simulation step size in the t-direction is 10^{-5} and the step size in the x-direction is 0.5. The small time step is needed for the stability of the numerical approximation. The numerical solutions for u and v were then saved at 201 values of x equally spaced between -50 and 50, and 35 values of t, equally spaced between 0 and 34, giving a total of $201 \cdot 35 = 7035$ data points. No external noise was added to the numerical solution. However, the training data set is not completely noise free, due to the numerical error of the simulation method and saving the simulation results at a coarser mesh than the one used for simulation.

In the experiments with PDED-1, both heuristic functions, SSE and MDL, were used in combination with beam search (width 25) through the space of multivariate polynomial equations. In both cases, none of the 25 best equations found by LAGRAMGE corresponded to the structure of the original predator-prey equations. Exhaustive search could not be used for this task, due to a vast search space that contains $7.5 \cdot 10^{11}$ equation structures. Thus, PDED-1 fails to recover the correct structure of the predator-prey equations. To better understand why, let us consider more closely the difficulty of the PDE discovery task.

3.3.3 Problems and limitations of PDED–1

The PDE discovery problem becomes more difficult if: (1) we have sparser measurements (coarser mesh), (2) higher-order derivatives are involved, and (3) the degree (of non-linearity) of terms in the equations is higher.

The first two items are related to numerical differentiation errors, while items 2 and 3 are related to the number of possible models considered during the search. The coarser the mesh and the higher the derivatives order, the larger the errors. The higher the derivatives order and the degree of terms, the greater the number of possible equations.

The first difficulty is due to the large error introduced by numerical differentiation. Using finer measurement mesh (i.e., having dense measurements along all dimensions) can help. However, it is well known that the numerical differentiation methods are unstable: increasing the resolution of the mesh above some limit can increase the numerical error (Press et al., 1986).

Two difficulties arise when we have a large space of possible equations. First, it takes a long time to search this space, and thus only non-exhaustive heuristic search strategies can be applied. Second, it is more difficult to select the appropriate equation structure. Given the same data, the more models we consider, the more likely we are to find models that fit the data by chance rather than true regularities. The first difficulty is addressed, and can be partly overcome, using non-exhaustive strategies (like beam search). The second difficulty can be overcome by introducing a stronger language bias, that constrains the space of possible equations.

In the case of the predator-prey model, the size of the space of polynomial equations that contains the target equations is of the order 10^{11}. We can identify this as the main reason for the failure of our straightforward approach. Greedy (beam) search considers only a fraction of the space of all possible equations, but misses the original equation structure. We therefore need to constrain the space of possible models and equations.

As we show in Chapter 4, modeling knowledge from the domain of interest can be used to constrain the space of possible equations. In contrast to the knowledge-driven approach presented there, in the rest of this chapter we explore a data-driven approach to constraining the space of equations.

3.4 A two-level approach to PDE discovery

Our second approach to PDE discovery relies on a transformation into the simpler problem of ODE discovery, that implements involves two stages. In the first, the algorithm learns how to constrain the space of possible PDEs by decomposing the problem of PDE discovery into a number of ODE discovery problems. The idea is to take slices of the training data for fixed values of all but the time dimension and search for ODEs in each slice. The structures of the ODEs that are most frequently (re)discovered in different slices are used to constrain the space of candidate PDEs.

In the second stage, the PDE discovery problem in the constrained (and therefore much less complex) space is solved following the straightforward approach presented in the previous section. This section gives a detailed description of the PADLES algorithm that implements the two-stage approach, along with with the results of the empirical evaluation on two reconstruction tasks.

3.4.1 The PADLES algorithm

Table 3.3 presents the PADLES algorithm, which that implements the two-stage approach to PDE discovery. The algorithm takes as input the measurements of the variables of the observed system $\mathcal{V}$, along the dimensions from $\mathcal{D}$, as well as the highest order o of the partial derivative that can appear in the discovered PDEs.

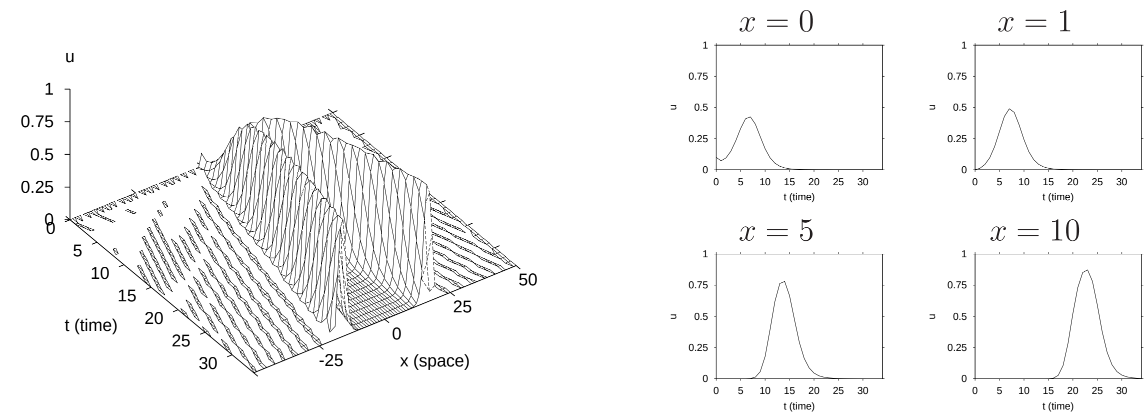


Figure 3.2: Simulation data of the predator-prey model presented in Section 3.3 (left-hand side) and slices of the data for four different fixed values of the space dimension x (right-hand side).

In the first stage (lines 1-9), the problem of PDE discovery is decomposed into a number of ODE discovery problems. Each problem is concerned with finding an ODE for a slice of the original data, where the values of all but one dimension (time) are fixed. The slicing of the data (implemented in lines 1-3 of the algorithm in Table 3.3) is illustrated by an example in Figure 3.2. The graph on the left-hand side represents the changes of the value of the system variable u along dimensions t and x. The graphs on the right-hand side represent four slices of the data, which are obtained by fixing the value of the x dimension to the values 0, 1, 5, and 10, respectively.

In each of the slices, the system variable is changing along a time dimension. Therefore, each slice of the data can be modeled using an ODE. Note that the ODE models the individual slices of the data are expected to be similar (in structure) to the PDE model of the whole data, with the exception of the terms that involve the partial derivatives with respect to the spatial dimensions. When the values of all space dimensions are fixed, the values of the partial derivatives with respect to these dimensions diminish, since they assess the change of the system variable along these dimensions. Therefore, a good approximate structure of the ODE model in each slice can be obtained by disregarding the partial derivatives with respect to the space dimensions. The "slicing" strategy presented here is similar to the "levels of description" heuristic used in equation discovery system BACON, where all but two variables are held constant at each level, and allows to discover equations relating arbitrary number of variables, using simple heuristics about bi-variate relations (i.e., relations involving two variables only) (Langley et al., 1987).

If we follow this reasoning in the opposite direction, we can say that ODE models of individual slices of data can give us a good starting point for the structure of the PDE model of the whole data. Consider again the first equation of the predator-prey model from Section 3.3. The ODE structure most frequently discovered in the different slices of the data is $\partial v/\partial t = c_1 \cdot uv + c_2 \cdot v$, which is equivalent to the structure of the PDE if we disregard the partial derivative $\frac{\partial^2 u}{\partial x^2}$.

Table 3.3: A two-stage algorithm for the discovery of partial differential equations.

	procedure PADLES($\mathcal{V}$, $\mathcal{D}$, o)
1	**foreach** tuple of values D_x of $\mathcal{D} - \{t\}$ **do**
2	$\mathcal{V}_x = \{$measurements M, such that DIMENSIONVALUES$(M, \mathcal{D} - \{t\}) = D_x\}$
3	**endfor**
4	$\mathcal{P}$ = PARTIALDERIVATIVES($\mathcal{V}$, $\mathcal{D}$, o)
5	**foreach** variable $V \in \mathcal{V}$ **do**
6	**foreach** $\mathcal{V}_x$ **do**
7	$\mathcal{E}_{V,x}$ = the set of 20 best equations from LAGRAMGE($\mathcal{V}_x \cup \{t\}$, dV/dt, G)
8	**endfor**
9	Let $\mathcal{S}_V$ be the set of most-frequent equation structures in $\cup_x \mathcal{E}_{V,x}$
10	Build grammar G_V based on the structures in $\mathcal{S}_V$
11	LAGRAMGE($\mathcal{V} \cup \mathcal{P}$, $\partial V/\partial t$, G_V)
12	**endfor**
	endprocedure

The above reasoning lets us use the ODEs discovered in the individual slices of data to constrain the space of possible PDEs. The constraints are induced for each system variable in $\mathcal{V}$ separately (line 5). First, ODEs are discovered in each data slice $\mathcal{V}_x$ (lines 6-9). Again, LAGRAMGE method is applied to this task. The structures of the 20 best ODEs (the union of the ten best according to the SSE heuristic and ten best according to the MDL heuristic) are kept for each data slice (line 7). Then all the structures found in the different data slices are merged and their frequencies are calculated. Only the most frequent ones are used to determine the possible PDE structures that are to be considered in the second stage of the algorithm (line 9). In particular, PADLES keeps only equation structures with frequencies within the interval $[f_{\max}/2, f_{\max}]$, where $f_{\max}$ is the frequency of the most frequent equation structure.

Table 3.4: An example grammar used to extend the most frequent ODE structures, discovered in the first stage, to PDE structures for the first equation of the predator-prey model, presented in Section 3.3.

`PDE -> ODE + const * PD	ODE`
`PD ->` $\partial u/\partial x$ | $\partial^2 u/\partial x \partial t$ | $\partial^2 u/\partial x^2$
`PD ->` $\partial v/\partial x$ | $\partial^2 v/\partial x \partial t$ | $\partial^2 v/\partial x^2$
`ODE -> const * u * v + const * v`
`ODE -> const * u * v`
$\vdots$

Finally, a context-free grammar is built that extends the most frequently discovered ODE structures into appropriate PDE structures. The grammar has three nonterminal symbols. The productions for the `ODE` nonterminal enumerate the most frequent ODE structures, the productions for `PD` enumerate the partial derivatives with respect to at least one dimension other than time, and the start symbol `PDE` is used to extend the ODE structures to appropriate PDE structures by linearly combining the partial derivative with the ODE structure. Table 3.4 presents an example of a context-free grammar used to discover the second predator-prey equation.

Note that the grammar defines a limited class of PDEs, where the partial derivatives with respect to variables other than time are linearly coupled with the rest of the equation terms. Although this is a serious restriction on the class of PDEs that can be discovered by the method, this restricted class still includes relevant models of biological systems (Murray, 1993). However, note that this limitation can be easily lifted by adding alternative productions to the grammar.

The second stage of PADLES algorithm involves LAGRAMGE to search through the constrained space of PDE structures (line 11 in Table 3.3) and find the ones that fit training data best. Note that LAGRAMGE is applied to the extended data set consisting of system variables and their partial derivatives, which are calculated numerically using the PARTIALDERIVATIVES procedure from Table 3.1.

3.4.2 Experimental evaluation of PADLES

We evaluated the PADLES algorithm on two tasks that involved reconstructing known PDE based models of biological systems. The first is the predator-prey model that PDED-1 algorithm could not reconstruct. The second is the FitzHugh-Nagumo model of signal transmission between nerve cells.

Predator-prey model

PADLES successfully reconstructed the structure of the predator-prey model from simulated data. Analysis of the results for each of the two equations revealed two interesting points.

First equation (for $\partial u/\partial t$). In the first stage experiments, 64 different ODE structures were discovered in the slices of original data. After filtering out the infrequent ones, seven ODE structures remained, which the context-free combined into a total of 49 possible PDE structures. In the second stage, LAGRAMGE successfully reconstructed the original structure of the first predator-prey equation (for $\partial u/\partial t$) out of these 49 possibilities.

Second equation (for $\partial v/\partial t$). In the first stage, PADLES found 96 different ODE structures and kept 19 most frequent ones. The grammar then combined these 19 ODE structures into a total of 113 possible PDE structures. Again, LAGRAMGE successfully reconstructed the original structure of the second predator-prey equation (for $\partial v/\partial t$).

The comparison of the number of possible PDE structures considered in the process of equation discovery explains why PADLES succeeds and PDED-1 does not. Using the first stage experiments, the number of possible PDE structures was reduced by a factor of over

10^9. This reduction of the complexity of the search space makes the reconstruction task feasible. Having successfully reconstructed a moderately complex PDE model using the PADLES, we now turn to the task of re-constructing more complex and practically relevant PDE model.

FitzHugh-Nagumo model

The research on modeling electric signaling or firing of individual nerve or neurons is particularly common in the field of neural communication modeling. Hodgkin and Huxley (1952) were awarded a Nobel prize for their seminal work on establishing mathematical model of neuron firing and propagation of impulses along the nerve axon. FitzHugh (1961) and Nagumo et al. (1962) independently derived simplified versions of the Hodgkin-Huxley equations, which retain the most important biological features. The form of the FitzHugh-Nagumo equations is

$$\begin{aligned}\frac{\partial v}{\partial t} &= \frac{\partial^2 v}{\partial x^2} + v(v-a)(1-v) - w \\ \frac{\partial w}{\partial t} &= b(v - dw),\end{aligned}$$

where a, b, and d, are constant parameters, and v and w are functions of time t and distance x. For a given initial condition (e.g., a narrow Gaussian pulse), this system might display any one of three types of behavior: simple decay; a single traveling wave solution; or multiple traveling wave solutions, determined by the values of the three parameters. These might correspond, respectively, to a nerve stimulus being inadequate to initiate axon firing; a nerve stimulus being sufficient to initiate a single nerve impulse; and repeated nerve firing such as occurs in the sinus node in the heart.

Three behavior traces (data sets) were generated using numerical simulation of the model. The first data set was generated using the values for the constant parameters of $a = -0.02$, $b = 0.005$, and $d = 3$. The other two data sets were generated using two different initial conditions and the values for the constant parameters of $a = 0.02$, $b = 0.005$, and $d = 3$. The equations were simulated using the same method as in the predator-prey experiments. The numerical solutions for v and w were then saved at 201 values of x

Table 3.5: Number of all and frequent ODE structures discovered in the first stage and number of PDE structures defined by the appropriate context-free grammars for the first (left-hand side) and the second FitzHugh-Nagumo equation (right-hand side).

	First equation			Second equation		
	1	2	3	1	2	3
All ODE structures	328	291	182	334	277	247
Frequent ODE structures	9	19	6	8	8	8
PDE structures	63	113	42	56	56	56

Table 3.6: Ranks of the original PDE structure among the ten best equations discovered by PADLES in the experiments with three different data sets and two different heuristics used by LAGRAMGE.

	First equation			Second equation		
	1	2	3	1	2	3
MDL	*5	*5	*2	1	1	1
SSE	6	*5	1	*	*	*

equally spaced between -100 and 100, and 13 values of t, equally spaced between 0 and 120, giving a total of $201 \cdot 13 = 2613$ data points in each data set. No external noise was added to the numerical solution.

Tables 3.5 and 3.6 sumarize the analysis of the results of the experiments with each of these three data sets.

First equation (for $\partial v/\partial t$). The summary of the first stage in Table 3.5 shows the number of possible PDE structures is reduced to 63, 113, and 42 for the first, second, and third data set, respectively.

A summary of the second stage is presented in Table 3.6, which shows the rank of the original PDE equation structure among the ten best equations discovered by PADLES. The *N means that the true PDE structure was not among the ten best as evaluated by the respective error heuristic (MDL or SSE), but a structure with one missing term (and otherwise identical to the original one) had rank N. The true PDE structure of the first equation is discovered in the experiments with the first and the third data set using the SSE heuristic. In the experiment with the second data set, a PDE with a similar but simpler equation structure was found.

Second equation (for $\partial w/\partial t$). The summary of results in the right-hand side of Table 3.5 shows that, after the first stage, the number of PDE structures was reduced to 56 for all the data sets. Out of these 56 possible structures, the original structure of the second FitzHugh-Nagumo equation was recovered in the experiments with all three data sets, using the MDL heuristic.

In summary, PADLES successfully recovers the structure of a complex and practically important PDE model from simulated data.

3.4.3 A final remark

Note that the context-free grammar used in the second phase of the two-level approach (see Table 3.4) defines a limited class of PDEs, where the partial derivatives with respect to variables other than time are linearly coupled with the rest of the equation terms. The latter are defined upon the ODE structure, most frequently discovered in the first phase of the two-level approach. Note however, that we could use polynomials of the system variables for the rest of equation terms and thus avoid the first phase of the algorithm. Such a grammar would defined a more complex space of equations then the one obtained with the first phase learning, but still far less complex than the space of polynomials of variables and their partial derivatives considered by the straightforward approach. The experiments show that the use of such an intermediate grammar allows for successful reconstruction of the predator-prey model, but still fails to reconstruct the first equation of the FitzHugh-Nagumo model.

3.5 Summary

In this chapter, we described two methods for PDE discovery. The first is a straightforward extension of existing equation discovery methods to the task of PDE discovery and is capable of reconstructing simple textbook PDEs. However, due to the complexity of the space of potential PDEs, it cannot reconstruct more complex models. The second method uses a data-driven approach to constrain the space of possible PDEs by decomposing the task of PDE discovery to a number of simpler ODE discovery tasks. The second method can reconstruct an important class of PDE based mathematical models of neurophysiology systems that have very wide-ranging applications (Murray, 1993).

While the use of PDE models is common in the physical sciences and engineering, our motivation has predominantly been the modeling of biological systems. The need for quantitative models of biological processes is growing rapidly, and we expect it to play a significant role in establishing the kind of mathematical understanding sought from enterprises like the Human Physiome Project (Bassingthwaighte, 2002 Web page update). We believe that an automated model discovery method of the form proposed here will greatly assist the analysis of data expected to result from the project.

Further work is needed before methods for the discovery of partial differential equations can be useful to domain experts. In the short term, further experiments with more models and with truly observational data are necessary. The simulated data used in our experiments contain some error, but this have a different nature then the measurement errors found in real measurement and observational data. We need to establish that PADLES works robustly under both conditions.

The work presented in this chapter has focused on data-driven approach to equation discovery, in which different models are constructed on a trial-and-error basis, and the selection is made based on their fit to data. This contrasts with a knowledge-driven approach, in which the basic processes involved in the modeled system are first identified. In the next chapter, we will shift our focus from data-driven to knowledge-driven approaches to the discovery of equation-based models.

4

Domain-specific modeling knowledge for equation discovery

Most state-of-the-art equation discovery methods follow the empirical data-driven approach to modeling, in which very little (if any) domain-specific knowledge is used to guide the modeling process. One type of knowledge used by some of the existing equation discovery methods involves measurement units of variables of the observed system (Washio & Motoda, 1997). However, domain experts can provide much more knowledge about the domain at hand than merely enumerating the measurement units of the observed system variables.

Many textbooks on mathematical modeling give a comprehensive overview of relevant knowledge about a specific domain, e.g., biology (Murray, 1993) or biochemistry (Voit, 2000). In order to incorporate this knowledge in the process of equation discovery, it must be appropriately encoded. The encoded knowledge can be then used to constrain the space of equations considered during equation discovery.

The discovery method LAGRAMGE (Todorovski, 1998) relies on the formalism of context-free grammars to specify the space of possible equations. Grammars are general enough to express many different types of domain-specific knowledge. For example, knowledge about the measurement units has been used to build a grammar for modeling the pole on cart mechanical system (Todorovski, 1998). In another example, knowledge about the basic processes that govern population dynamics was used to automatically model phytoplankton growth in Lake Glumsø in Denmark from a sparse and noisy set of real-world measurements (Todorovski et al., 1998). However, it is difficult for domain experts to express or encode their modeling knowledge about a domain in the form of a grammar. Another limitation of grammars is that they are typically task dependent. For example, the grammar used for modeling phytoplankton growth in Lake Glumsø, can not be reused for similar tasks with different sets of system variables.

This chapter presents a more flexible formalism for encoding domain knowledge. The formalism organizes knowledge in a taxonomy of process classes, each of which represents an important class of basic processes that influence behavior in the domain. For each process class, a number of alternative equation models, usually used by modeling experts in the domain, can be specified. Knowledge also encodes how to combine the models of individual basic processes into a single model of the whole system. We illustrate the use of the formalism by encoding knowledge from three domains: population dynamics, biochemical kinetics, and spring mechanics. In addition, we also encode domain independent knowledge about measurement units of the observed system variables within the formalism.

The resulting knowledge is independent of the particular modeling task and allows automated modeling of an arbitrary system in the target domain. In order to use the knowledge for automated modeling of a particular observed system, we require a modeling task specification that states the types of the system variables along with the process classes that are expected to influence the system behavior. Whereas the domain-specific knowledge should be provided by a modeling expert with extensive experience, the task specification can be provided by a user who is familiar with the domain but who does not have much modeling expertise.

In order to integrate this knowledge into the equation discovery methods, we again invoke the transformation principle. Given the modeling task specification, the encoded knowledge is used to build a grammar that specifies the space of possible models for the observed system. Nonterminal symbols in the grammar denote process classes, while the alternative productions for each nonterminal symbol specify possible expressions for modeling the corresponding process class. The starting symbol of the grammar combines the expressions for individual processes into candidate models of the whole system. The LAGRAMGE method is then used to search through the space of candidate models of the observed system and find the one that fits the measured data best. The grammar parse tree used to derive this best model indicate the processes that govern the behavior of the observed system.

The chapter is organized as follows. We introduce the formalism for encoding domain-specific modeling knowledge in Section 4.1 and illustrate its use on the example from the population dynamics domain. Section 4.3 presents further examples of encoding modeling knowledge in several other domains, as well as of encoding domain-independent modeling knowledge. We present the method for transforming the encoded knowledge into grammars for equation discovery and the necessary improvements of LAGRAMGE in Section 4.2. Finally, Section 4.4 summarizes the chapter and discusses related research.

4.1 Encoding of domain-specific modeling knowledge

Our new formalism for encoding domain-specific modeling knowledge organizes the content in three parts. The first contains knowledge about what types of variables occur in systems from the domain of interest. The second part contains models of typical processes that govern the behavior of systems in the domain. The third part encodes knowledge about how to combine models of individual processes into a single model of the entire system. We will illustrate the use of the formalism on the example of population dynamics modeling. Thus, we start with an introduction to the basics of population dynamics modeling. We will finish the section with examples of several population modeling tasks specifications.

4.1.1 Population dynamics modeling

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 the same area. More specifically, we consider modeling the dynamics of populations, especially how their density changes through time (Murray, 1993).

For example, consider a simple model based on two populations, foxes and rabbits. The latter graze on grass and the foxes are carnivores that hunt rabbits. We assume that rabbits

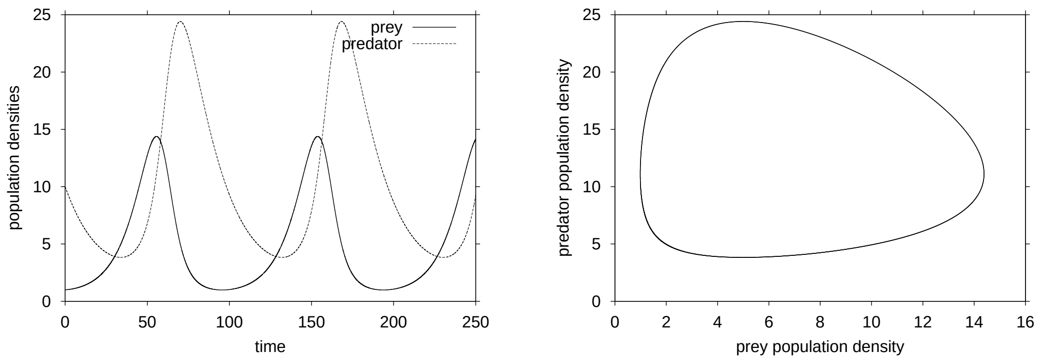


Figure 4.1: Trajectories in time space (left) and phase space (right) for the simulation of a simple Volterra-Lotka model with the values of the constant parameters set to $a = 0.1$, $b = 0.009$, $c = 0.01$, and $d = 0.05$, and initial population densities $N(0) = 1$ and $P(0) = 10$.

are the only food of foxes, an unlimited supply of grass is available to the rabbits, and there are no seasonal changes. Under these assumptions, if the rabbit population is large, the fox population grows rapidly. However, this causes many rabbits to be eaten, thus diminishing the rabbit population to the point where the food for foxes is not sufficient. Consequently, the fox population decreases, which causes faster growth of the rabbit population.

Population dynamics models are based on the seminal research work by Volterra (1926) and Lotka (1920), in which they proposed a simple model of predator-prey interaction between two species (Murray, 1993):

$$\begin{aligned} \dot{N} &= aN - bNP \\ \dot{P} &= cNP - dP, \end{aligned}$$

where N is the prey (rabbit) population density and P is the predator (fox) population density. Figure 4.1 shows the simulation of the Volterra-Lotka model with values of the constant parameters set to $a = 0.1$, $b = 0.009$, $c = 0.01$, $d = 0.05$ and initial population densities $N(0) = 1$, $P(0) = 10$. The trajectories reflect the oscillatory change of population densities described above.

The simple Volterra-Lotka model has the following structure. The first term aN in the first equation models the prey population growth in the absence of predation. The model used makes the assumption that the growth is unlimited and exponential. The effect of predation on the prey population growth is modeled by the second term $-bNP$ in the first equation. The assumption here is that the predation rate is proportional to the prey and predator populations. The predation contribution to the predator population growth is modeled by the same term (cNP) in the second equation. Finally, in the absence of predator-prey interactions, the predator decay is exponential (term $-dP$).

The insight into the structure of the Volterra-Lotka model and the function of the individual terms in the equations is important, because it provides considerable knowledge in the domain of population dynamics. It is this kind of knowledge that we intend to formalize and use for automated modeling with equation discovery. We will formalize this population dynamics knowledge in the following sections.

4.1.2 Taxonomy of variable types

The first part of the domain knowledge is the taxonomy of variable types that can be used in the models. Table 4.1 presents an example of such a taxonomy for the population dynamics domain.

Table 4.1: A taxonomy of variable types that are can be used in population dynamics models.

```
type Concentration is nonnegative_real_number
type Population is Concentration
type Inorganic is Concentration
```

The generic variable type in the population dynamics domain is `concentration`, since we are interested in the change of concentrations of different populations inhabiting the observed environment. The definition of concentration specifies that this type must be a nonnegative real number. The concentration type has two sub-types. The first, `population`, denotes a concentration of an organic species, such as foxes and rabbits in the example from the previous section. The second type, `inorganic`, denotes the concentration of an inorganic nutrient that can be consumed by organic species. Note that inheritance rule applies to the taxonomy, so that the population and inorganic types are also known to be nonnegative real numbers.

4.1.3 Taxonomy of process classes

The most important part of the modeling knowledge is the taxonomy of process classes. Each process class represents a class of basic processes that govern or influence the behavior of dynamic systems in the domain of interest. Table 4.2 presents an example of a taxonomy of process classes for the population dynamics domain.

Table 4.2: A taxonomy of process classes, each representing a class of processes that influence the behavior of the population dynamics systems.

```
process class Growth(Population p)

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

process class Decay(Population p)

process class Exponential_decay is Decay
   expression const(decay_rate,0,1,Inf) * p

process class Feeds_on(Population p, Concentration c)
   condition p ≠ c

process class Unsaturated_feeds_on(Population p, Concentration c) is Feeds_on
   expression p * c

process class Saturated_feeds_on(Population p, Concentration c) is Feeds_on
   expression p * c / (c + const(saturation_rate,0,1,Inf))
```

The taxonomy consists of three generic process classes. The first, `Growth`, represents the processes of a single species' growth in when the influence of any (predator-prey) interaction with other populations in the observed environment is neglected. Similarly, the second process class `Decay` represents the processes of a single population's decay. Finally, the `Feeds_on` process class refers to processes of predator-prey interaction between two populations (in cases where the variable `c` represents a population) or grazing of a population on an inorganic nutrient (in cases where `c` denotes an inorganic nutrient).

Each of the Growth and Decay classes has a single subclass that specifies an (unlimited) exponential growth (or decay) of the population. On the other hand, the Feeds_on process class has two subclasses, each specifying an alternative model of consumption. The first consumption model, specified by Unsaturated_feeds_on, corresponds to unlimited consumption. This model assumes that the predation capacity of the predator population is unlimited. However, this assumption is often unrealistic, as in many cases the predators do have a 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 some limit. An alternative class that corresponds to cases of saturated consumption is specified by Saturated_feeds_on.[1]

The definition of each process class consist of three parts, each specifying one aspect of the processes in the class and/or the models thereof. We will present each part in more detail below.

Types of variables involved. The first part of the definition specifies what types of variables that can influence and be influenced by processes in the class. Recall that variable types are defined in the taxonomy. For example, each process in the Growth process class involves a single population p. Furthermore, the processes in the Feeds_on class involve one population variable p and one variable c of type concentration, which can be either a population or an inorganic nutrient.

The declarations of variable types are inherited through the taxonomy of process classes. For example, processes in the Exponential_growth class inherit from the parent class Growth the fact that they involve a single variable of type population.

Conditions on variables involved. The second part of the process class definition specifies additional constraints on the variables involved in the processes. The condition $p \neq c$ in the Feeds_on process class forbids cannibalism within a single species, so that the population cannot predate on itself.

[1]Note that there are also other possible models of saturated consumption, which we discuss in Section 4.3.

Declaration of process models. The final part of the process class definition specifies the equation template that is used by domain experts to model processes in the class. This template includes variables involved in the process, and generic constant parameters. The values of the generic constant parameters can be latter fitted against measurements of the system. In the equation template, symbol `const(name, lower_bound, initial, upper_bound)` is used to specify a generic constant parameter. The symbol specifies the name of the constant parameter, along with its lower bound, default value, and upper bound. For example, consider the equation template used to model `Exponential_growth` processes. This involves a single nonnegative (note that a lower bound of 0 as well as infinite upper bound are specified) constant parameter that represents the growth rate with the default value of 1. Note that the default value of the constant parameter is used as its initial value when fiting the model parameters against measured data.

Note that each process model, encoded within our formalism, should be defined as an expression template that evaluates to a single real-valued number. Note however, that this does not mean that the process model influence a single system variable. As we present in the following section, a single process can influence more than one system variable, as specified by the appropriate combining schemes.

Note furthermore that specifying a single process model for a class does not imply that each process class have a single model template. The taxonomy of process classes is defined in such a way that it specifies that the process model can be used for modeling processes in the current class as well as processes from the more general (ancestor) classes in the taxonomy. For example, the taxonomy from Table 4.2 specifies that either saturated or unsaturated model template can be used for modeling `Feeds_on` processes.

4.1.4 Schemes for combining models of individual processes

Our modeling formalism also specifies schemes that are used to combine the models of individual processes into a model of the whole system. Table 4.3 presents two such combining schemes for combining processes in population dynamics.

Table 4.3: Combining schemes specify how to combine the models of individual population dynamics processes into a model of the entire system.

```
combining scheme Population_dynamics(Inorganic i)
  i̇ = - ∑p const(_,0,1,Inf) * Feeds_on(p, i)

combining scheme Population_dynamics(Population p)
  ṗ = + Growth(p) - Decay(p)
    + ∑food const(_,0,1,Inf) * Feeds_on(p, food)
    - ∑predator const(_,0,1,Inf) * Feeds_on(predator, p)
```

The first combining scheme specifies how to build the equation that models the time change of an inorganic nutrient i from the individual process models. The time derivative i̇ of i is negative sum of all expressions used to model those Feeds_on interactions in which an arbitrary population p consumes the inorganic nutrient i. Note that the Feeds_on(p, i) symbol is used to denote the process model for the Feeds_on process class. The $\sum$ aggregation function is used to sum up the models of all such proceses; note that the scope of p in the sum is defined by the declaration of the Feeds_on process class. Potentially, the scope of p is the set of all population type variables.

The second combining scheme specifies how to combine process models into equations for modeling the change of a population p. The first line specifies that the time derivative of p increases with the population growth Growth(p) and decreases with its decay Decay(p). In contrast to the case of inorganic nutrient, where all Feeds_on processes negatively influence the change, Feeds_on processes can positively or negatively influence the change of p, depending on its role in the interaction. The processes that involve p as a consumer or predator positively influence the change of p, while the processes where p is involved as a prey negatively influence the change of p. Again, influences of these processes are summed up, as shown in the last two lines in Table 4.3.

The use of aggregation functions is beneficial when the processes are not present in the observed system. In such cases, the use of the $\sum$ function specifies that the value of these corresponding terms equals zero. Similarly, the use of the $\prod$ (product) aggregation function in a term would specify that the value of the term equals one, when the corresponding processes are not present.

4.1.5 Specification of the modeling task

Our modeling knowledge, presented above, is general in the sense that it supports modeling of an arbitrary system that involves population dynamics. In order to use the knowledge for modeling of a particular system, a specification of the system must be provided. The specification includes a list of system variables and their associated types, along with a list of processes (and their classes) that govern the dynamics of the observed system.

Table 4.4: A task specification used for modeling the Volterra-Lotka system of a single predator-prey interaction between two populations.

```
variable Population rabbit, fox

process Growth(rabbit) rabbit_growth
process Decay(fox) decay_fox
process Unsaturated_feeds_on(fox, rabbit) fox_rabbit_predator_prey
```

Table 4.4 gives an example of a modeling task specification for the Volterra-Lotka predator-prey system. This includes the types of the two system variables, representing the populations of rabbits and foxes and the three processes of rabbit population growth, fox population decay, and predator-prey interaction between foxes and rabbits.

An automated modeling system can use the encoded modeling knowledge to transform the task specification into a model as described below. First, the combining scheme for a population type variable from Table 4.3 is applied to the first system variable to obtain the equation for the temporal change of the rabbit population:

$$\dot{\mathtt{rabbit}} = \mathtt{Growth}(\mathtt{rabbit}) - 0 + 0 - \mathtt{const}(\_, 0, 1, \mathtt{Inf}) * \mathtt{Unsaturated\_feeds\_on}(\mathtt{fox}, \mathtt{rabbit}),$$

which combines the two processes (`rabbit_growth` and `fox_rabbit_predator_prey`) that involve the rabbit population. Note the `0` terms in the middle, which is due to the absence of `Decay(rabbit)` and `Feeds_on` processes where `rabbit` is the predator or consumer. Similarly, the same combining scheme, applied to the second system variable, generates the equation for the temporal change of the fox population

$$\dot{\mathtt{fox}} = 0 - \mathtt{Decay}(\mathtt{fox}) + \mathtt{const}(\_, 0, 1, \mathtt{Inf}) * \mathtt{Unsaturated\_feeds\_on}(\mathtt{fox}, \mathtt{rabbit}) - 0.$$

This equation combines the two processes of fox_decay and fox_rabbit_predator_prey. To obtain the final model, each process class instance in these equations must be replaced with the appropriate expressions specified in the process taxonomy from Table 4.2. For example, consider the Growth(rabbit) instance of the Growth class in the equation for changes in the rabbit population. Querying the taxonomy for all possible expressions used to model Growth(rabbit), we obtain a single expression const(growth_rate, 0,1,Inf) * rabbit, which specifies an unlimited exponential growth. Finally, performing two more similar queries to the taxonomy, for the Decay(fox) and Unsaturated_feeds_on(fox, rabbit) process class instances, we obtain the following model (with generic constant parameters):

$$\begin{aligned} \dot{\mathtt{rabbit}} &= \mathtt{const}(\mathtt{growth\_rate}, 0, 1, \mathtt{Inf}) * \mathtt{rabbit} - \mathtt{const}(\_, 0, 1, \mathtt{Inf}) * \mathtt{fox} * \mathtt{rabbit} \\ \dot{\mathtt{fox}} &= -\mathtt{const}(\_, 0, 1, \mathtt{Inf}) * \mathtt{fox} + \mathtt{const}(\_, 0, 1, \mathtt{Inf}) * \mathtt{fox} * \mathtt{rabbit}. \end{aligned}$$

The model has exactly the same structure as the Volterra-Lotka model, that we presented in the beginning of this section. This model structure is the only one that can be obtained given the task specification from Table 4.4. However, this is not always true, as the formalized knowledge allows for task specifications at different levels of detail, some leading to a number of different model structures. The less detailed the specification, the larger the number of possible structures. In the example above, we specified that the predation rate of foxes on rabbits is unlimited. An alternative specification of the fox_rabbit_predator_prey process that does not specify whether the predator rate is saturated or unsaturated, i.e., the specification process Feeds_on(fox, rabbit) fox_rabbit_predator_prey, would leave both modeling alternatives possible, thus giving two possible model structures.

In the second modeling task specification example presented in Table 4.5, we observe an aquatic ecosystem that involves an inorganic nutrient, phytoplankton and zooplankton (Crispi & Mosetti, 1993). Here the first variable (nut) has inorganic type, while the other two variables (phyto and zoo) are populations. The phyto_decay and zoo_decay processes specify that the populations of phytoplankton and zooplankton tend to decrease in absence of any interactions with the environment and other species. The two Feeds_on processes

Table 4.5: A task specification used for modeling a simple aquatic ecosystem that consists of two consumption interactions between three populations of inorganic nutrient, phytoplankton, and zooplankton.

```
variable Inorganic nut
variable Population phyto, zoo

process Decay(phyto) phyto_decay
process Feeds_on(phyto, nut) phyto_nut_consumption

process Decay(zoo) zoo_decay
process Feeds_on(zoo, phyto) zoo_phyto_predator_prey
```

specify that phytoplankton consumes inorganic nutrient and that zooplankton consumes phytoplankton. The type of consumption (saturated or unsaturated) is not specified.

The specification leads to four possible model structures, with different types of consumption between phytoplankton and nutrient, and between zooplankton and phytoplankton, which can be passed to an automated modeling method. Given the observed values of the system variables over time, the automated modeling method must choose the model that fits the observed data best. The following section presents an example of such an automated modeling framework based on the equation discovery method Lagramge.

4.2 Using the modeling knowledge for equation discovery in Lagramge

We schematize the integration of domain-specific modeling knowledge into the process of equation discovery in Figure 4.2. The integration relies on the transformation principle. The domain-specific knowledge is transformed into a grammar based on the declarations of the system variables and processes. The resulting grammar specifies the space of candidate models for the observed system. The equation discovery method Lagramge can be then used to search through the space of candidate models and find the one that fits the measured data best. The grammar derived from the modeling knowledge is not necessarily context-free as in Lagramge, so we must improve Lagramge to allow the use

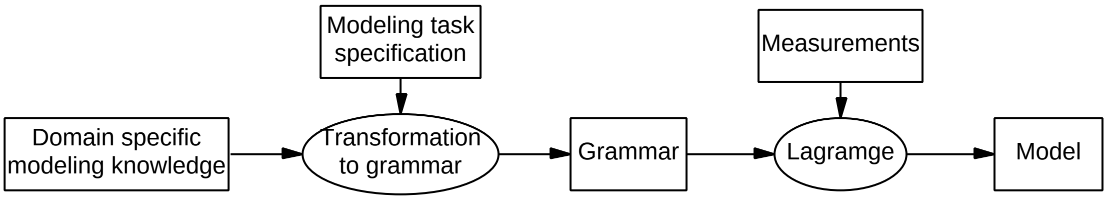


Figure 4.2: An automated modeling framework based on the integration of domain-specific modeling knowledge in the process of equation discovery.

of context-dependent constraints in the grammar. This section presents the algorithm for transforming the modeling knowledge into a grammar, as well as the improvements to LAGRAMGE that allow its use for equation discovery.

4.2.1 Transforming the modeling knowledge into a grammar

The algorithm for transforming the encoded knowledge into grammar takes two arguments as input. The first is the library of modeling knowledge for the domain at hand, and the second is the modeling task specification for the types of the variables and the classes of the processes.

The transformation of the modeling knowledge proceeds in a top-down manner. It starts with the starting symbol and assigns productions to it, then proceeds with other nonterminal symbols. The starting symbol of the grammar corresponds to the combining schemes in the encoded knowledge, which is used to combine the models of individual processes into a single model of the entire system. Other nonterminal symbols in the grammar correspond to process classes. Alternative productions for each nonterminal symbol specify alternative models of the corresponding process class.

For example, consider the aquatic ecosystem example from Table 4.5 in Section 4.1.5. The start symbol uses the combining schemes from Table 4.3 to compose a model of the whole ecosystem as follows:

```
Start ->
  time_deriv(nut) = - const[_:0:1:]  * Feeds_on_phyto_nut;
  time_deriv(phyto) = 0 - Decay_phyto + const[_:0:1:]  * Feeds_on_phyto_nut
    - const[_:0:1:]  * Feeds_on_zoo_phyto;
  time_deriv(zoo) = 0 - Decay_zoo + const[_:0:1:]  * Feeds_on_zoo_phyto.
```

The right-hand side of the production builds three equations, one for each variable of the observed ecosystem. The first equation is created by summing the effects of all the consumption processes that involve inorganic nutrient `nut` as food. Only one such process is specified for the system (process `Feeds_on(phyto, nut)` in Table 4.5). The second equation is formed by summing the growth processes for the `phyto` population (the leading `0` means that no such process is specified), the `phyto` decay processes (i.e., `Decay(phyto)`), the consumption processes where `phyto` is involved as consumer (i.e., `Feeds_on(phyto, nut)`), and the predator-prey processes where `phyto` has the role of prey (i.e., `Feeds_on(zoo, phyto)`). The third equation, for the change of the `zoo` concentration, is built in a similar manner.

Table 4.6: The grammar specifying the candidate models for modeling the simple aquatic ecosystem presented in Table 4.5.

```
Start ->
  time_deriv(nut) = - const[_:0:1:]  * Feeds_on_phyto_nut;
  time_deriv(phyto) = 0 - Decay_phyto + const[_:0:1:]  * Feeds_on_phyto_nut
    - const[_:0:1:]  * Feeds_on_zoo_phyto;
  time_deriv(zoo) = 0 - Decay_zoo + const[_:0:1:]  * Feeds_on_zoo_phyto

Feeds_on_phyto_nut -> Unsaturated_feeds_on_phyto_nut
Feeds_on_phyto_nut -> Saturated_feeds_on_phyto_nut
Unsaturated_feeds_on_phyto_nut -> phyto * nut
Saturated_phyto_nut -> phyto * nut / (nut + const[saturation_rate:0:1:])

Decay_phyto -> Exponential_decay_phyto
Exponential_decay_phyto -> const[decay_rate:0:1:]  * phyto

Feeds_on_zoo_phyto -> Unsaturated_feeds_on_zoo_phyto
Feeds_on_zoo_phyto -> Saturated_feeds_on_zoo_phyto
Unsaturated_feeds_on_zoo_phyto -> zoo * phyto
Saturated_zoo_phyto -> zoo * phyto / (phyto + const[saturation_rate:0:1:])

Decay_zoo -> Exponential_decay_zoo
Exponential_decay_zoo -> const[decay_rate:0:1:]  * zoo
```

Note that each of the process instances on the right-hand side (i.e., `Feeds_on_phyto_nut`, `Decay_phyto`, etc.) are nonterminal symbols. The productions for each of these nonterminal symbols are based on the definition of the corresponding class in the taxonomy of process classes from Table 4.2. Table 4.6. presents the complete grammar with all the nonterminal symbols and their productions.

For example, consider the two productions for the `Feeds_on_phyto_nut` nonterminal symbol. Each corresponds to one of the subclasses of the `Feeds_on` class. Furthermore, we obtain the production for `Unsaturated_feeds_on_phyto_nut` by instantiating the general expression for the `Unsaturated_feeds_on` class with the appropriate values of the process class parameters. By instantiating the same expression with `zoo` and `phyto` parameter values, we obtain the production for `Unsaturated_feeds_on_zoo_phyto`. Similarly, by following the `Decay` part of the process classes taxonomy, we obtain the productions for `Decay_phyto` and `Decay_zoo` symbols. Note that the terminal symbols `nut`, `phyto`, and `zoo` represent the system variables, while the terminal symbols of the form `const[name:lower:init:upper]` denote constant parameters. This symbol includes the name of the constant parameter, the lower and upper bound of its value, and its default initial value `init`.

Strictly speaking, the grammar in Table 4.6 is not context-free. The production for the starting symbol generates two `Feeds_on_phyto_nut` symbols, one in the first equation and another in the second. In a context-free grammar, these two nonterminal symbols can generate two different expressions. In population dynamics models, however, these two expressions must be the same. The use of context-dependent constraints can overcome this limitation of context-free grammars.

Note that the grammar in Table 4.6 contains redundancy due to the similar productions for the same process class, since they all have the same form and only differ due to the different process variables. This redundancy can be avoided using the Prolog notation of definite clause grammars, which allows the use of parameters in nonterminal symbols (Bratko, 2001). Using this notation, the four `Feeds_on` productions can be replaced by two definite clauses of the form `Feeds_on(P, I) -> Unsaturated_feeds_on(P, I)` and `Feeds_on(P, I) -> Saturated_feeds_on(P, I)`. The use of definite clause grammars is not supported in our current implementation of LAGRAMGE, but it seems a plausible direction for further development.

4.2.2 Necessary improvements to LAGRAMGE

In order to use grammars like the one from Table 4.6 for equation discovery, we developed the equation discovery system LAGRAMGE 2.0, an improved version of LAGRAMGE

Table 4.7: The top-level of the LAGRAMGE 2.0 algorithm.

procedure `Lagramge-2.0`($\mathcal{V}$, G, b)
  M_0 = simplest parse tree in G
  M_0.quality = `Fit_model`(M_0, $\mathcal{V}$)
  $Q = \{T_0\}$
  repeat
    Q_r = {refinements of parse trees in Q}
    foreach parse tree $M \in Q_r$ **do**
      M.quality = `Fit_model`(M, $\mathcal{V}$)
    endfor
    Q = {best b trees from $Q \cup Q_r$ according to $\mathcal{H}$}
  until Q unchanged during the last iteration
  print Q

(Todorovski, 1998). These incorporated improvements along three fronts. First, the use of context-dependent constraints in the grammar must be supported. Second, a system of simultaneous equations must be discovered at once instead of discovering an equation for each system variable separately. Third, the constraints on the lower and upper bound of the values of the constant parameters must be considered.

Table 4.7 presents the top-level algorithm of LAGRAMGE 2.0. The algorithm takes as input the measurements of the system variables $\mathcal{V}$, a context-dependent grammar G specifying the space of possible models, and a parameter b specifying the beam width used in the beam search procedure. The output of LAGRAMGE 2.0 consists of the b best models according to some evaluation criterion.

The search space of LAGRAMGE is ordered using two refinement operators on parse trees, as described by Todorovski (1998). The first is used for efficient enumeration of all the parse trees up to a maximum tree depth $d_{\max}$. This refinement operator is used for exhaustive search through the space of possible parse trees. The second refinement operator is used for heuristic beam search through the space of parse trees. Table 4.7 outlines the beam search procedure of LAGRAMGE 2.0. For a description of the exhaustive search procedure and the corresponding refinement operator, see Todorovski and Džeroski (1997) and Todorovski (1998).

In each iteration of the beam search procedure, LAGRAMGE computes the refinements of the parse trees in the current beam Q and collects them in Q_r (line 5). Each parse tree in Q_r generates a model of an observed system with generic constant parameters. LAGRAMGE fits the values of the constant parameters against the measurements of the system variables $\mathcal{V}$ using the `Fit_model` procedure (line 7). The fitting procedure minimizes the discrepancy between the measured values of the system variables and the values obtained from simulating the model. This discrepancy equals the sum of squared errors (SSE). LAGRAMGE measures the quality of the model as the SSE of the model with the optimized values of the constant parameters. At the end of each iteration, only the best b trees are kept in the beam Q. LAGRAMGE proceeds with the beam search until the beam elements remain unchanged.

In the next three subsections, we present the three necessary improvements of the original version of LAGRAMGE.

Context-dependent constraints

The context-dependent aspects of the grammar from Table 4.6, described in the previous section, are implemented in LAGRAMGE 2.0 in form of constraints. An arbitrary number of these constraints can be attached to each production in the grammar. Each constraint specifies that two or more nonterminal symbols on the right-hand side of the production must generate the same expression. Examples of productions with context-dependent constraints appear in Table 4.8.

Table 4.8: Examples of grammar productions with context-dependent constraints.

`E -> A + B + A, B + A { A.1 == A.2; A.1 == A.3; }`
`E -> A + B + A, B + A { A.1 == A.2; B.1 == B.2; }`

In the first production, the two constraints `A.1 == A.2` and `A.1 == A.3` specify that all three occurrences of the symbol `A` (referred to as `A.1`, `A.2`, and `A.3`) on the right-hand side of the production should generate the same sub-expression. For example, the expression `a1 + b1 + a1, b2 - a2` cannot be generated using the first production, because the first and third occurrence of the `A` symbol generate different expressions (i.e., `A.1 -> a1` is different from `A.3 -> a2`). On the other hand, the expression `a1 + b1 + a1, b2 - a1` can be generated using the first production. However, the latter expression cannot be derived using the second production due to the second constraint `B.1 == B.2`. Finally, the expression `a1 + b1 + a1, b1 - a1` can be derived using both productions. Thus, the production for the `Start` nonterminal symbol from Table 4.6 must include the constraints `Feeds_on_phyto_nut.1 == Feeds_on_phyto_nut.2` and `Feeds_on_zoo_phyto.1 == Feeds_on_zoo_phyto.2`.

LAGRAMGE 2.0 improves the original LAGRAMGE refinement operators to take into account such context-dependent constraints. The new refinement operators generate only parse trees that satisfy the constraints.

Simultaneous equations

The LAGRAMGE method was not capable of discovering models that consist of simultaneous equations. This task was performed by discovering an equation for each dependent variable in turn. The inability to discover simultaneous equations prevents the application of context-dependent constraints on the expressions in two different equations.

LAGRAMGE 2.0 improves on the original procedure that evaluates a parse tree on the given measurements of the system variables $\mathcal{V}$. The new `Fit_model` procedure can evaluate an entire model that consists of one or several simultaneous equations at once.

As before, the heuristic functions SSE and MDL functions are used in LAGRAMGE 2.0 to estimate the quality of the equation based models. The SSE of a complete model that contains of simultaneous equations is calculated as a sum of SSE of the individual equations:

$$SSE(M) = \sum_{v_d \in \mathcal{V}} SSE(M.v_d),$$

where $M.v_d$ represents a single model equation for the system variable v_d. The MDL heuristic is calculated in the same way as for a single equation, except that the complexity of the entire model is taken into account. Again, the complexity of the model equals the sum of complexities of individual model equations.

Note that summing the SSE of individual system variables can be problematic in cases when system variables have different scales. In such cases, a weighted sum should be applied, although we do not address the issue in our current implementation. We assume that the measurements have been normalized in such a way that all the system variables have comparable scales.

Lower and upper bounds on the values of the constant parameters

Finally, the downhill simplex and Levenberq-Marquart algorithms (Press et al., 1986) that LAGRAMGE uses to infer the values of the constant parameters, do not let the user specify lower and upper bounds on them. Thus, in LAGRAMGE 2.0 we replaced these algorithms with a non-linear optimization algorithm, proposed by Bunch et al. (1993), that can take into account such bounds on parameter values. Following our formalism for encoding domain-specific knowledge, the default value of a constant parameter specified there is used as its inital value for the parameter optimization method.

Finally, non-linear optimization methods suffer from the problem of getting stuck in a local optimum that is the nearest to the initial values of the parameters to be optimized (Press et al., 1986). This makes them sensitive to the choice of the initial parameter values. A technique that is usually used to increase the robustness of such methods is restarting them with different randomly chosen combinations of initial values. This multi-start technique increases the likelihood of finding a true local, or even global optimum, so LAGRAMGE 2.0 incorporates this method, with the number of restarts being specified by the user.

4.2.3 Implementation

The algorithm for transforming the domain-specific knowledge and modeling task specifications into grammars as well as LAGRAMGE are implemented in the C programming language using the Gnu C Compiler. Both algorithms were developed, tested, and run on an Intel platform working under the RedHat Linux operating system. All the programs are available for download from `http://www-ai.ijs.si/~ljupco/ed/`.

4.3 Examples of encoded modeling knowledge

The described formalism is general enough to represent knowledge from different domains. We illustrate this by encoding modeling knowledge from the domains of population dynamics, chemical kinetics, and engineering. Furthermore, we show that the formalism allows for encoding of domain-independent knowledge, such as knowledge about the measurement scales used for measuring the system variables.

4.3.1 Population dynamics

The simple Volterra-Lotka model we used for illustrative purposes in the previous section makes several assumptions about the population dynamics processes involved. These assumptions are not always realistic and should be relaxed in order to allow practical applications of these models for modeling real-world environments. The simple Volterra-Lotka model can be schematized as:

$$\begin{aligned}\dot{N} &= \mathit{growth\_rate}(N) - \mathit{feeds\_on}(P, N) \\ \dot{P} &= \mathit{feeds\_on}(P, N) - \mathit{decay\_rate}(P).\end{aligned}$$

The first assumption made in the simple Volterra-Lotka model is that the growth rate of the prey population in the absence of predation is proportional to its density, so that $\mathit{growth\_rate}(N) = aN$. However, this means that the growth of the population is exponential and unlimited, which is unrealistic in many cases. Natural environments sometimes have carrying capacity for the population that limits the population density. For example, this can be a limited supply of grass that rabbits graze on. In such cases, one can use the alternative logistic growth model (Murray, 1993):

$$\mathit{growth\_rate}(N) = aN(1 - \frac{N}{K}),$$

where K is a constant that determines the carrying capacity of the environment.

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. In analogy with growth, this means that the predation growth is exponential and unlimited.

Table 4.9: An extension of the process class taxonomy for the population dynamics domain from Table 4.2 with a new process class that represents logistic growth processes.

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

Table 4.10: A re-definition of the Feeds_on process class for the population dynamics domain from Table 4.2 in order to take into account the saturation of the predation (or consumption) rate.

```
process class Feeds_on(Population p, Concentration c)
   condition p ≠ c
   expression p * Saturation(c)
```

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 some limit. Several different terms can be used to model the predator saturation response to the increase of prey density (Murray, 1993):

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

where A is the limit value of the predation capacity saturation and B is the constant that determines the saturation rate.

Relaxing the assumptions made in the Volterra-Lotka model in these ways, we can build different, more complex and more realistic models of predator-prey population dynamics. The modeling knowledge about different models of population growth and predator rate saturation can be easily encoded within the formalism from the previous section.

For example, we can extend the `Growth` process class with the new subclass that defines the logistic growth model as Table 4.9 shows.

Furthermore, to take into account saturation we can redefine the process classes of `Feeds_on` and `Predator_prey` as in Table 4.10.

Note that the new definitions of the new process class refers to the function `Saturation`. In order to specify different alternatives for modeling saturation, we can introduce a taxon-

Table 4.11: Taxonomy of function classes specifying different expressions for modeling the saturation of predation rate in population dynamics.

```
function class Saturation(Concentration c)

function class No_saturation is Saturation
  expression c

function class Saturation_type_1 is Saturation
  expression c / (c + const(saturation_rate,0,1,Inf))

function class Saturation_type_1 is Saturation
  expression c * c / (c * c + const(saturation_rate,0,1,Inf))

function class Saturation_type_3 is Saturation
  expression 1 - exp(-const(saturation_rate,0,1,Inf) * c)
```

Table 4.12: A taxonomy of variable types that can appear in population dynamics models with multi-species interactions.

```
type Concentration is nonnegative_real_number
type Concentrations is set(Concentration)

type Population is Concentration
type Populations is set(Populations)

type Inorganic is Concentration
```

omy of function classes. The definitions in this new taxonomy are the same as definitions of process classes. We distinguish between functions and processes because the former do not represent processes from the domain; rather they are used to specify alternative subexpressions that commonly appear in models of individual processes.

Table 4.11 presents the taxonomy of function classes for specifying different saturation responses. The unsaturated response is specified by the first `No_saturation` function class. The last three function classes specify the three alternative saturation responses presented above.

The modeling knowledge presented so far allows for modeling interactions between only two species. However, in nature we can observe interactions between many species.

Table 4.13: The definition of the multi-species `Feeds_on` process class specifying the dependence of population `p` on one or more food sources `cs` at the same time.

```
process class Feeds_on(Population p, Concentrations cs)
   condition p ∉ cs
   expression p * ∏_{c∈cs} Saturation(c)
```

For example, we can have three populations of foxes, rabbits, and pheasants, where foxes can feed on both rabbits and pheasants as alternative food sources. This multi-species interaction can be still represented within our formalism by two two-species interactions of `Feeds_on(fox, rabbit)` and `Feeds_on(fox, pheasant)`. The two predation process happen in parallel and are combined additively, which means that there are two `Feeds_on` terms in the differential equation for the (predator) population of foxes. In contrast to this kind of interactions that represent dependence on several alternative food sources, a population may depend on consumption of several food sources *at the same time*. For example, some species of phytoplankton in aquatic ecosystems need both phosphorus and nitrogen as inorganic nutrients at the same time to achieve optimal growth. In our formalism, this would be represented as `Feeds_on(phyto, {nitro, phosp})`. Thus, in order to encode models of this kind of multi-species interactions, we must extend our formalism to handle sets of variables. This can be declared using set types, as in Table 4.12. The two new set types of `Concentrations` and `Populations` denote non-empty sets of variables of `Concentration` and `Population` type, respectively.

Now we can define a more general `Feeds_on` class that can be used to model dependence on one or more nutrients and/or prey populations at the same time, as presented in Table 4.13. Note that the second process class argument, `cs`, represents a set of food sources, on which population `p` depends. Like the original definition in Table 4.2, the condition `p ∉ cs` is used to specify that the population cannot predate on itself. The expression first multiplicatively combines the predator saturation terms on different food sources and finally multiplies the obtained product with `p`. Thus, the expression used to model the `Feeds_on(phyto, {nitro, phosp})` process is `phyto * Saturation(nitro) * Saturation(phosp)`.

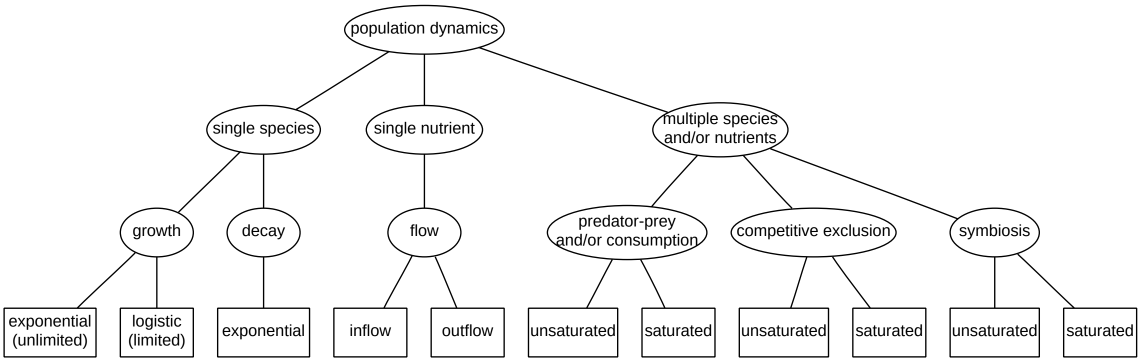


Figure 4.3: An improved taxonomy of classes of processes used for modeling population dynamics.

Another extension of the population dynamics library involves different kinds of interactions between species. Until now, we have concentrated on one type of predator-prey (or consumption) interaction, where the growth rate of one (prey) population decreases while the other (predator) increases. There are also two other important types of interactions between two species (Murray, 1993). If the growth rate of both populations decreases, then we observe the process of competitive exclusion. This kind of interaction appears when two species compete for the same limited food source or inhibit each other's growth in some other way. The other kind of interaction, symbiosis or mutuality, involves enhancements of the growth rates of both populations. Symbiosis usually plays a crucial role in the survival of such species.

Taking into account these other types of interactions, we can enhance our basic taxonomy of process classes to obtain the one that is presented graphically in Figure 4.3. The complete formalized library of modeling knowledge about process classes in this enhanced taxonomy appears in Appendix A.

Finally, we should note that the library of modeling knowledge for population dynamics presented here can be also used in other areas of mathematical biology. For example, modeling the dynamics of infectious diseases (Capasso, 1993) is an important area where the structure of mathematical models is identical to the models of population dynamics.

4.3.2 Biochemical kinetics

Biochemical reactions take place continually in the metabolic processes of all living organisms. Biochemical kinetics studies the rates of biochemical reactions and the dynamic

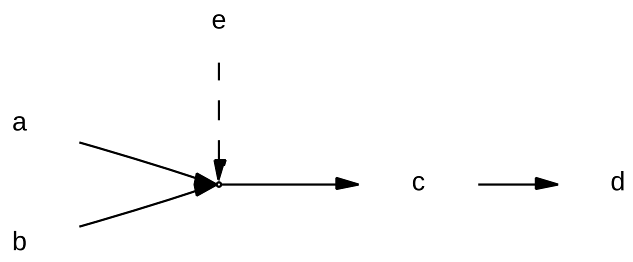


Figure 4.4: An example metabolic pathway map representing a network of two chemical reactions involving four chemical substances and an enzyme.

change of the concentration of various reactants (proteins and enzymes) involved in a particular metabolic process (Voit, 2000). The dynamic change of the reactants' concentrations is modeled using ordinary and partial differential equations.

The metabolic process is usually presented graphically as a network of chemical reactions, that is referred to as a *metabolic pathway map*. Figure 4.4 presents an example of a simple metabolic pathway map that consists of two chemical reactions (Voit, 2000). The first chemical reaction is activated by enzyme `e`, takes two substrate substances `a` and `b` at input, and produces a single substance `c`. The second chemical reaction does not involve activating enzymes and transforms `c` into `d`.

There are several methods for transforming a metabolic pathway into ordinary differential equations for modeling the change of the concentrations of `a`, `b`, `c` and `d`. Here we consider the S-system method, presented by Voit (2000). In the S-system approach, the differential equations for the metabolic pathway from Figure 4.4 would be formulated as:

$$\begin{aligned}
\dot{\mathtt{a}} &= -\beta_{\mathtt{a}} \cdot \mathtt{e} \cdot \mathtt{a}^{\gamma_{\mathtt{a}}} \cdot \mathtt{b}^{\gamma_{\mathtt{b}}} \\
\dot{\mathtt{b}} &= -\beta_{\mathtt{b}} \cdot \mathtt{e} \cdot \mathtt{a}^{\gamma_{\mathtt{a}}} \cdot \mathtt{b}^{\gamma_{\mathtt{b}}} \\
\dot{\mathtt{c}} &= \alpha_{\mathtt{c}} \cdot \mathtt{e} \cdot \mathtt{a}^{\gamma_{\mathtt{a}}} \cdot \mathtt{b}^{\gamma_{\mathtt{b}}} - \beta_{\mathtt{c}} \cdot \mathtt{c}^{\gamma_{\mathtt{c}}} \\
\dot{\mathtt{d}} &= \alpha_{\mathtt{d}} \cdot \mathtt{c}^{\gamma_{\mathtt{c}}},
\end{aligned}$$

where $\alpha_{\mathtt{c}}$, $\beta_{\mathtt{a}}$, and $\beta_{\mathtt{b}}$ are constant parameters representing the rates of the chemical reaction, while $\mathtt{c}_{\mathtt{a}}$ and $\mathtt{c}_{\mathtt{b}}$ are constant parameters representing the kinetic orders of the reaction with respect to the chemical substances `a` and `b`, respectively. Following the S-

Table 4.14: Library of modeling knowledge for the biochemical kinetics domain.

```
type Concentration is nonnegative_real_number

type Substance is Concentration
type Substances is set(Substance)

type Enzyme is Concentration
type Enzymes is set(Enzyme)

process class Reaction(Enzymes es, Substances ins, Substances prods)
   condition ins ∩ prods = ∅
   expression ∏_{e∈es} e * ∏_{i∈ins} pow(i, const(_, 0, 1, Inf))

combining scheme Biochemical_kinetics(Substance s)
   ṡ = ∑_{s∈prods} const(_, 0, 1, Inf) * Reaction(es, ins, prods))
     - ∑_{s∈ins} const(_, 0, 1, Inf) * Reaction(es, ins, prods))
```

system method, each chemical reaction is represented by a single term that appears in the equations for all the substances that are involved in the reaction. In our example, there are two such terms, i.e., $\mathtt{e} \cdot \mathtt{a}^{\gamma_a} \cdot \mathtt{b}^{\gamma_b}$ and $\mathtt{c}^{\gamma_c}$, corresponding to the first and second chemical reaction, respectively.

Following the rules of the S-system method, we can formalize the library of modeling knowledge for the domain of biochemical kinetics, as presented in Table 4.14. The single process class `Reaction` represents biochemical reactions, Each of which is activated by a set of enzymes `es`, takes substrates `ins` at input, and generates a set of products `prods`. The expression used to model a chemical reaction builds a term that positively influences the change of products concentrations and negatively influences the change of inputs concentrations. The last two facts are encoded in the combining scheme. The example metabolic pathway map from Figure 4.4 can be then formalized using the task specification from Table 4.15.

Encoding knowledge about the kinetics of biochemical metabolic pathways opens an important potential application area of automated modeling. The need for quantitative models of biological processes is growing rapidly, and we expect it to play a significant role

Table 4.15: Task specification used for modeling the metabolic pathway from Figure 4.4.

```
variable Substance a, b, c, d
variable Enzyme e

process Reaction({e}, {a, b}, {c}) reaction_1
process Reaction({}, {c}, {d}) reaction_2
```

in establishing the kind of mathematical understanding sought from enterprises like the Human Physiome Project (Bassingthwaighte, 2002 Web page update). We believe that an automated model discovery method of the form proposed here will greatly assist the analysis of the large quantities of data expected to be available as a result of the project.

4.3.3 Spring mechanics domain

Although the targeted application areas of the modeling framework developed in this thesis are environmental and biochemical dynamic systems, the formalism can be also used to encode knowledge in engineering domains. Here we present a formalization of knowledge about spring mechanics, used in the PRET reasoning system for automated modeling of dynamic systems (Bradley et al., 2001; Stolle, 1998).

PRET uses domain knowledge about engineering domains that is encoded in the form of domain rules. For example, its mechanical knowledge consists of a single force balance rule and a single variable type representing a point coordinate in the observed system. The force balance rule applies to every point coordinate in the observed system and specifies that the sum of all forces that apply to a point coordinate is zero. Bradley et al. (2001) emphasize that the rule is not strictly domain-specific. In the electrical domain, the same rule is expressed by Kirchoff's law, which states that the sum of currents at an observed point in an electrical circuit is zero. Thus, the modeling knowledge used in PRET is not domain-specific but rather general knowledge used for modeling dynamic systems in various engineering domains.

We will illustrate the use PRET's of domain knowledge on the spring and masses dynamic system from Bradley et al. (2001), as depicted in Figure 4.5. Table 4.16 presents

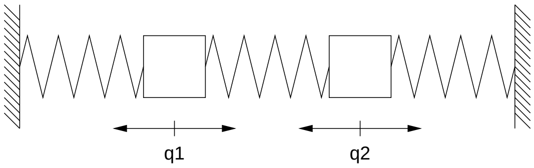


Figure 4.5: Springs and masses dynamic system.

Table 4.16: The domain-specific knowledge used in Pret for modeling springs and masses system from Figure 4.5.

```
(point-sum <force> 0)

(state variables <q1> <q2>)
(point coordinates <q1> <q2>)
(hypotheses
   (<force> (* m1 <q̈1>)
   (<force> (* m2 <q̈2>)
   (<force> (* k1 <q1>))
   (<force> (* k2 (- <q1> <q2>))
   (<force> (* k3 <q2>)))
```

the knowledge used for modeling the spring and masses system. The first row in the table represents the domain-specific knowledge, in particular the force balance rule. The next two rows specify the observed system variables and their types. Finally, the last five rows encode knowledge that is task specific to the observed system. They specify forces that potentially govern the dynamics of the springs and masses.

An analysis of Pret's knowledge for modeling springs and masses reveals several important differences between our approaches. Our knowledge representation formalism follows the compositional modeling paradigm (Kuipers, 1994; Falkenheiner & Forbus, 1991), in which the model fragments of individual processes (or components) are combined into a model of the entire system. In contrast, the hypotheses in Pret do not necessarily correspond to the processes or components in the domain. Rather, they represent quantities that obey a balance rule in the domain. In the mechanics, these quantities are forces that obey the force balance law, whereas in the electrical domain they correspond to currents that follow the Kirchoff's law.

Table 4.17: Representation of the modeling knowledge used in PRET for modeling the springs and masses system.

```
type Coordinate is real

function class Single_spring_force(Coordinate q)
   expression const(k,0,1,Inf) * q

function class Two_springs_force(Coordinate q1, Coordinate q2)
   condition q1 ≠ q2
   expression const(k,0,1,Inf) * (q1 - q2)

combining scheme Spring_mechanics(Coordinate q)
   q̈ = - ∑ Single_spring_force(q) - ∑q2 Two_spring_force(q, q2)
      + ∑q1 Two_spring_force(q1, q)
```

Despite this fundamental difference between the approaches, the knowledge used in PRET for modeling the springs and masses system can be encoded in our formalism, as shown in Table 4.17. There are several important differences to the knowledge representation from Table 4.16. First, although the combining scheme used in our formalization is based on the force balance rule, it is written in a form that makes it an explicit form of the model.[2] On the other hand, the balance force rule in PRET is represented in its implicit form $\sum \texttt{Forces} = 0$.

Second, the two knowledge representations differ in terms of the spaces of possible models they specify. The combining scheme states that the models of the mass-spring systems must be second-order, i.e., they must include the second-order derivatives of the system variables. This kind of knowledge is encoded within PRET in the form of general ordinary differential equations (ODE) rules and not as a domain-specific knowledge. An ODE rule in PRET specifies that models of systems with oscillatory behavior must be second-order. Thus, PRET will also consider the zero (algebraic equations) and first-order models of the mas-spring system, and using the qualitative reasoning with the above

[2] The implicit form of a model of a dynamic system with a single variable x is $f(x, \dot{x}) = 0$, as opposed to the explicit form $\dot{x} = g(x)$. Note that most existing methods for simulating and fitting the parameters of a dynamic system model operate on the explicit form only.

mentioned general ODE rule PRET will rule out these models. Furthermore, PRET will consider models with a single model equation. Again, reasoning with general ODE rules encoded in the knowledge base will discard these models. On the other hand, the grammar obtained by transforming the knowledge from Table 4.17 will produce only models that include equations for both coordinates of the system.

The third difference in our knowledge representations lies in the generality of the encoded knowledge. Although PRET's knowledge could be extended could be extended to incorporate a third mass body in the system by adding additional `<hypothesis>` declaration, our knowledge can be used without any additional specifications except for declaration of a new system variable.

4.3.4 Dimensional analysis

Giordano et al. (1997) review dimensional analysis as a method for mathematical modeling that helps determine the relationships between measured system variables on the basis of the their dimensions or measurement units. The approach rests on the assumption that the system variables have dimensions and that the form of the equations relating them does not change with the measurement units.

The key theorem of dimensional analysis is the *Buckingham Pi theorem*, which provides a method for grouping the system variables into dimensionless terms. The benefit of grouping is that it reduces the number of independent variables, since there are fewer dimensionless terms than system variables. The early equation discovery method COPER (Kokar, 1986) employed dimensional analysis and the Buckingham Pi theorem to constrain the space of possible equations.

The application of dimensional analysis methods (including the version used by COPER) is limited to the cases where the dimensions of all system variables are known. Washio and Motoda (1998) have extended the original Buckingham Pi theorem to the cases where only information about the type of the measurement scale used to measure the system variables is available. The measurement scale provides information about the nature of the measured quantity. Three types of measurement scales are usually acknowledged: interval, ratio, and absolute scale.

Table 4.18: A taxonomy of measurement scale types used for measuring the system variables.

```
type Absolute is real
type Absolutes is set(Absolute)

type Ratio is real
type Ratios is set(Ratio)

type Interval is real
type Intervals is set(Interval)
type Intervals_covering is set(Intervals)
```

Interval Scale can be used for quantities that possess magnitude, i.e., one value can be judged greater than, less than, or equal to another. In addition, the units of measurement are the same across the entire measurement scale regardless of where the unit falls. Temperature is an example of an interval quantity, as the difference between 100 degrees and 99 degrees is the same as the difference between 40 degrees and 39 degrees. Interval quantities do not necessarily have an absolute zero point, e.g., a temperature of zero degrees does not indicate that there is no temperature.

Ratio Scale is more specific than interval scale, as it must have an absolute (invariant) zero point. Distance or mass are both examples of ratio scale quantities, where the distance or mass of zero indicates no distance or mass.

Absolute Scale is more specific than ratio scale, as it must have an absolute (invariant) unit, i.e., the distance between two consecutive tics on the measurement scale. Both counts and dimensionless quantities are examples of absolute scale quantities.

The extended Buckingham theorem, proposed by Washio and Motoda (1998), specifies how original system variables with known measurement scale types can be combined into dimensionless terms, referred to as "regimes". The taxonomies of function classes defined in Table 4.19, Table 4.20, and Table 4.21 formalize the rules of the extended Buckingham theorem for building "regimes" out of system variables of ratio, interval, and a mixture of ratio and interval scale types, respectively.

Table 4.19: A taxonomy of function classes that transforms variables of ratio scale-types into dimensionless regimes.

```
function class Ratio_regime(Ratios rvs) is Regime

function class Ratio_regime_1() is Ratio_regime
  expression ∏_{r∈rvs} pow(fabs(r), const(a,-Inf,1,Inf))

function class Ratio_regime_2() is Ratio_regime
  expression ∑_{r∈rvs} const(a,-Inf,1,Inf) * log(fabs(r))
```

Table 4.20: A taxonomy of function classes that transforms variables of interval scale-types into dimensionless regimes.

```
function class Linear_combination(Intervals ivs)
  expression const(c,-Inf,1,Inf) + ∑_{i∈ivs} const(b,-Inf,1,Inf) * fabs(i)


function class Interval_regime(Intervals_covering ivsc) is Regime

function class Interval_regime_1() is Interval_regime
  expression ∏_{ivs∈ivsc} pow(Linear_combination(ivs), const(a,-Inf,1,Inf))

function class Interval_regime_2(Intervals ivs) is Interval_regime
  condition ∧_{ivs1∈ivsc} ivs1 ∩ ivs = ∅
  expression ∑_{ivs1∈ivsc} const(a,-Inf,1,Inf) * log(Linear_combination(ivs1))
    + Linear_combination(ivs)
```

The rules of the extended Buckingham theorem define a huge space of possible "regimes" even for a small number of system variables, especially if many of them are of interval type. The space can quickly become intractable. To overcome this problem, Washio and Motoda (1997) propose a set of data-driven heuristics that can prune the space of possible regimes to the ones that are used to model the observed system. Their pruning heuristics makes the space of possible regimes tractable for large sets of variables, which lets their equation discovery method, SDS, to reconstruct complex models of electrical circuits from noisy data.

However, the knowledge encoded in the taxonomy of function classes do not encode the heuristics for pruning the space of possible regimes. The encoding of the data-driven

Table 4.21: A taxonomy of functions that transforms variables of mixed (ratio and interval) scale-types into dimensionless regimes.

```
function class Mixed_regime(Ratios rvs, Intervals_covering ivsc) is Regime

function class Mixed_regime_1() is Mixed_regime
  expression Ratio_regime_1(rvs) * Interval_regime_1(ivsc)

function class Mixed_regime_2(Intervals ivs) is Mixed_regime
  expression Ratio_regime_2(rvs) + Interval_regime_2(ivsc, ivs)
```

heuristics would require additional constructs that would allow for conditions based on the characteristics of the data.

Despite the limitation mentioned above, the encoded domain-independent knowledge based on the extended Buckingham theorem constitutes an important step toward the integration of dimensional analysis with other aspects of domain-specific modeling knowledge. In order to make the encoded knowledge operational, we have to encode the SDS pruning heuristics within the function classes. Furthermore, we also have to define a combining scheme that will combine the regimes into model equations.

4.4 Summary

In this chapter, we have presented an approach to automated modeling of dynamic systems that allows for the representation and use of domain-specific modeling knowledge. The approach follows the compositional modeling paradigm, where the fragment models representing individual processes are combined into models of the entire system (Kuipers, 1994; Falkenheiner & Forbus, 1991). The knowledge is organized in a taxonomy of process classes, each representing an important class of processes in the observed domain. The knowledge includes also states how to combine models of individual processes into a model of the entire system.

This high-level knowledge representation can be automatically transformed to the operational form of grammars that specify the space of candidate models of the observed system. Equation discovery method Lagramge can be then used to search through the space of candidate models and find the one that fits the measured data best. The grammars generated using the presented approach are context-dependent and generate complete models consisting of simultaneous (differential) equations. The equation discovery method Lagramge 2.0 was developed that can deal with context-dependent grammars as well as simultaneous equations.

While our approach to automated modeling follows the compositional modeling paradigm (Falkenheiner & Forbus, 1991), the Pret reasoning system for automated modeling employs different kinds of modeling knowledge (Bradley et al., 2001; Stolle, 1998). 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 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 Section 4.3.3, both kinds of modeling knowledge, used in Pret, can be easily encoded within our formalism. Note however, 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 it can not fit the measured data well.

Another related study is presented by Garrett et al. (2004). 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 himself 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.

An immediate direction of further (and partly ongoing) work is establishing libraries of encoded knowledge in different domains and applying the framework on real-world problems. 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 measurement data.

A serious limitation of our automated modeling framework is the assumption that the domain expert can specify the set of processes that govern the behavior of the observed system. In many real-world situations, the domain expert will not be able to specify the list of processes, but can only provide the specification of system variables. In such cases, the task of the automated modeling method is to find the set of processes that govern the behavior of the observed system as well as the particular modeling alternative for each of the processes in the set. Thus, the task specification will only provide the list of the observed system variables.

The information about possible classes of processes from the taxonomy along with the set of system variables and their types, can be used to enumerate all possible processes that can appear in the observed system. These are equivalent to all possible instantiations of the process and function classes in the library of domain-specific knowledge. As we have illustrated in the springs and masses example, grammar productions can be used to enumerate all subsets of the set of all possible processes. However, this approach is impractical, since the number of possible process and function class instantiations explodes with the number of system variables.

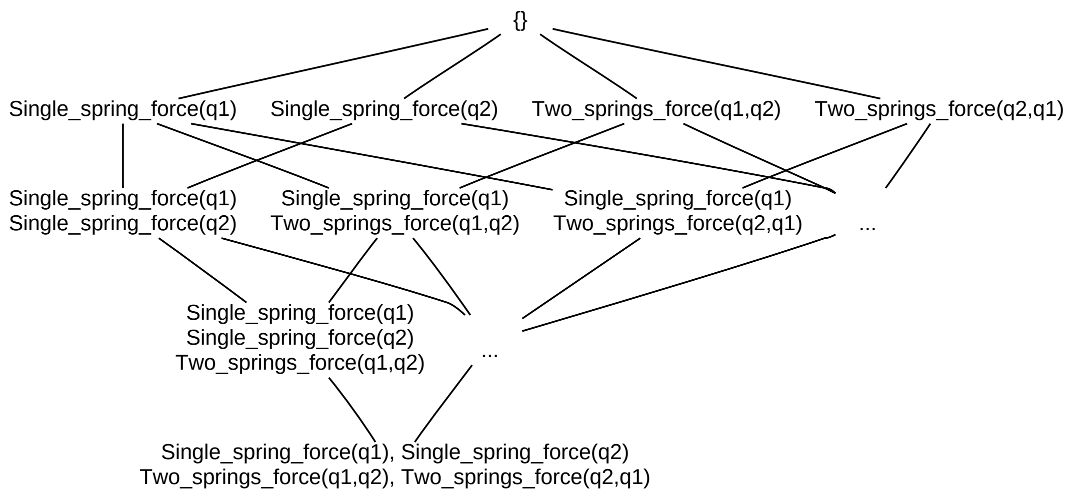


Figure 4.6: The search space of all possible subsets of processes that can govern the behavior of the springs and masses system from Figure 4.5.

One way to overcome this limitation is to apply a two-level search method through the space of possible models. At the higher level, search through the space of subsets of possible processes should be performed (for an illustration of the possible ordering of the search space for modeling the springs and masses system see Figure 4.6). At each node in the search space, the particular set of processes can be used to specify the space of candidate models in the form of a grammar. Then, the search on the lower level is performed to find the model, based on the particular set of processes, that fits the measured data best. The search at the lower level will find the set of processes and the model that fits the measured data best. The development of the two-level search procedure is another direction for further development of the automated modeling framework.

In the next chapter, we will empirically evaluate the proposed framework for automated modeling on several tasks of modeling dynamic systems from synthetic and real-world measurement data.

5

Experimental evaluation and examples of use

In this chapter, we empirically evaluate our modeling framework on tasks of modeling dynamic systems from the domains of population dynamics, spring mechanics, and hydrodynamics. The chosen tasks illustrate the usability and flexibility of the proposed framework and evaluate its performance.

In the first series of experiments on synthetic data, we examine the ability of our framework to reconstruct known models from noisy simulation traces. The goal of these experiments is to evaluate the inductive performance and noise robustness of the framework. The experimental results show that both context-dependent constraints and bounds on the constant parameters greatly improve the efficiency, inductive performance, and noise robustness of Lagramge.

In the second series of experiments, we study the ability of the framework to induce models of dynamic systems from real-world measurement data. The experiments show that Lagramge can build accurate and comprehensible models of population dynamics from real-world measurements. Our framework compares favorably with the equation discovery method GoldHorn (Križman, 1998) and with previous version of Lagramge in terms of flexibility and performance on the tasks of modeling algae growth in the Lagoon of Venice (Coffaro et al., 1993) and Lake Glumsø (Jørgensen et al., 1986). The last experiment, on modeling the water level in Ringkøbing fjord, shows the applicability of the framework to the task of completing partially specified models.

5.1 Reconstructing known models from synthetic data

In the first series of experiments, we applied our framework to several tasks of reconstructing known models from synthetic data. This experimental setup is beneficial since it allows a comparison of the discovered model with the original one and an evaluation of the reconstruction success. This is the first evaluation criterion used in the studies.

In each experiment, we first choose a model of a dynamic system and simulate it from ten randomly chosen initial states to obtain ten simulation traces. These are used for model induction experiments with LAGRAMGE. In order to evaluate error of induced models on test data unseen during the discovery process, we use 10-fold cross-validation procedure. In each iteration of this procedure, LAGRAMGE induces a model from nine out of ten simulation traces, the induced model is in turn tested for consistency with the remaining tenth trace. We measure the discrepancy between simulations of the original and induced model using root mean squared error (RMSE) measure. The cross-validated RMSE estimate is the second evaluation criterion. For simulating the original and induced model, we use an adaptive-step method (Press et al., 1986), as implemented in the Octave[1] programming language for numerical computations.

The third evaluation criterion is the complexity of the space of models considered in the process of automated modeling.

In addition to testing the ability to reconstruct models from noise-free synthetic data, we also tested the noise robustness of the proposed framework. To each trace, we added artificially generated random Gaussian noise was added at five relative noise levels of 1%, 2%, 5%, 10%, and 20%. A relative noise level of $l\%$ means that we replaced the original value x with the noisy value of $x \cdot (1 + l \cdot r/100)$, where r is a normally distributed random variable with mean zero and standard deviation one.

In the experiments with synthetic data, we aimed at evaluating the performance gain due to the improvements to the earlier version of LAGRAMGE. We evaluated separately context-dependent constraints and bounds on the values of the constant parameters. In order to do this, we induced models with four different versions of a grammar. The first

[1] http://www.octave.org/

is a fully constrained grammar (labeled `L2`), the second is a grammar without context-dependent constraints (labeled `L2-cdc`), the third is a grammar without constant parameters' bounds (labeled `L2-cpb`), and the last is a grammar without any constraints (labeled `L1`, since it is equivalent to using the earlier version of LAGRAMGE).

5.1.1 First experiment in the population dynamics domain

We performed the first experiment on the task of reconstructing models of a simple aquatic ecosystem specified in Table 4.5. This specification along with the population dynamics knowledge from Appendix A can be transformed to a grammar, which generates sixteen candidate models of the ecosystem. Each takes a form of three differential equations (for `nut`, `phyto`, and `zoo`) with generic constant parameters. We initialized the constant parameters with random values uniformly distributed over the $[0, 2]$ interval and simulated each model from ten initial states randomly chosen from the $[0.5, 1.5]$ interval. Each simulation trace included 100 time steps of 0.1 time units.

We used these traces (with artificially added noise) as training data for reconstructing each of the sixteen models of the simple aquatic ecosystem in turn. Figure 5.1 and Table 5.1 summarize the results of the reconstruction experiments.

The graph in Figure 5.1 shows the improvement in noise robustness due to both context-dependent constraints and bounds on the values of the constant parameters. The `L2` line shows that the fully constrained grammar is capable of reconstructing 90% of the models (14 out of 16) from noise-free data and 50% of the models from data with 20% relative noise.

Comparing the `L2` line with the `L2-cdc` and `L2-cpb` lines shows that both context-dependent constraints and bounds on the values of the constant parameters are almost equally beneficial for the noise-robustness of LAGRAMGE. The grammar without context-dependent constraints (`L2-cdc`) can reconstruct only 50% of the models from noise-free data. Note that this equals the rate gained on data with 20% relative noise using fully-constrained grammar. Similarly, the successful reconstruction rate significantly drops, if a grammar without bounds on the values of the constant parameters is used (`L2-cpb`). The corresponding line on the graph shows that bounds on the values of constant parameters becomes more important for successful reconstruction at higher noise levels.

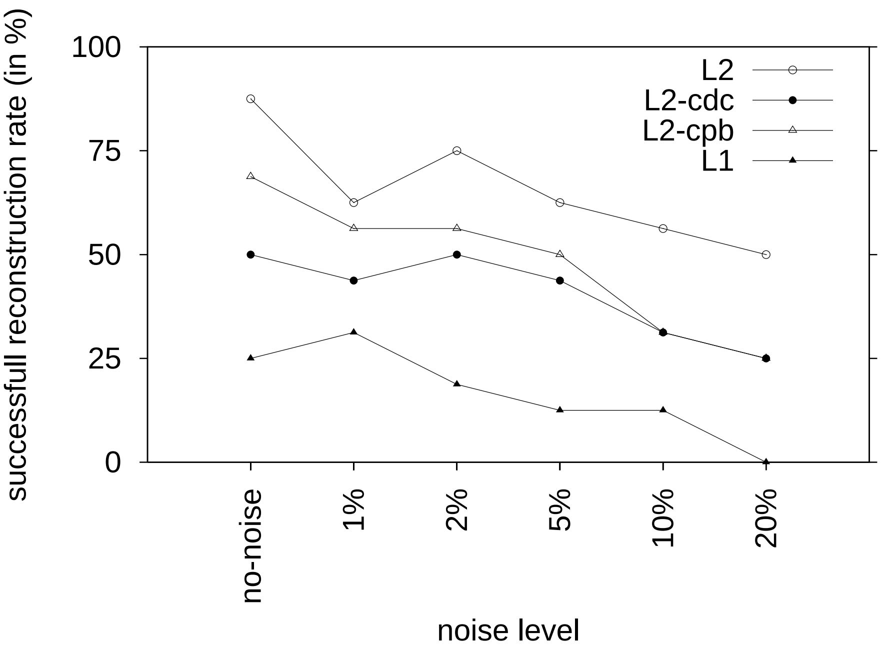


Figure 5.1: The percentage models that were successfully reconstructed by LAGRAMGE using four different versions of the grammar. Legend: `L2` = fully constrained grammar, `L2-cdc` = grammar without context-dependent constraints, `L2-cpb` = grammar without bounds on the values of the constant parameters, and `L1` = fully unconstrained grammar (which is equivalent to the previous version of LAGRAMGE).

The successful reconstruction rate of previous version of LAGRAMGE is much lower than the one of the improved one at all noise levels. Previous version of LAGRAMGE could not reconstruct any model from data with 20% relative noise.

Furthermore, the comparison of the four lines on the graph in Figure 5.1 shows that the performance improvements gained with the two kinds of constraints almost sum up. That means that the two kinds of constraints are orthogonal, i.e., each improves a different aspect of LAGRAMGE's induction performance.

Finally, context-dependent constraints reduce the complexity of the space of models considered by LAGRAMGE. While the context-dependent grammars generates 16 candidate models, the context-free grammar (i.e., the one without context-dependent constraints) generates 256 models. Thus, context-dependent constraints reduce the search space by a factor of 16. This reduction is reflected in reduced run times of LAGRAMGE. The average

Table 5.1: The average cross-validated root squared mean errors (RMSE) of the sixteen models reconstructed by LAGRAMGE using four different versions of the grammar (see the legend from Figure 5.1). For noisy data, errors on the left-hand side are measured with respect to noisy data, while errors on the right-hand side are measured with respect to noise-free data.

noise level	`L2`		`L2-cdc`		`L2-cpb`			`L1`	
no-noise	0.00643		0.00364		0.00404			0.00379	
1%	0.00880	0.00797	0.00755	0.00660	0.00775	0.00689		0.0119	0.0109
2%	0.0139	0.0110	0.0131	0.00986	0.0132	0.0101		0.0131	0.00986
5%	0.0319	0.0214	0.0310	0.0201	0.0310	0.0201		0.0309	0.0199
10%	0.0293	0.0290	0.0278	0.0276	0.0279	0.0278	(*1)	0.118	0.0274
20%	0.0549	0.0523	0.0629	0.0502	0.0548	0.0530	(*5)	0.0647	0.0540

time[2] of a single run with the context-dependent grammars was about 40 seconds, which is more than 30 times less time than the average of 1500 seconds needed for a single run with the context-free versions of the grammar.

In sum, integrating domain-specific knowledge in the process of automated modeling reduces the space of candidate models and greatly improves the reconstruction ability, noise robustness, and efficiency of LAGRAMGE.

Table 5.1 shows the average cross-validated RMSE of the sixteen models discovered by LAGRAMGE. For models induced from noisy data, we measured RMSE with respect to both noisy (on the left-hand side of each column) and noise free data (on the right-hand side). Note that, in most cases, these errors are comparable, the general trend is that induced models fit the noise-free data better than the noisy data, which confirms the noise robustness of LAGRAMGE.

Models discovered with unconstrained grammars (i.e., `L2-cdc`, `L2-cpb`, and `L1`) are on average more accurate than the models discovered with fully constrained grammar. Grammars without context-dependent constraints allow for models with inappropriate structure, where two or more different expressions are used for the same process in different equations. Grammars without bounds on the values of the constant parameters allow for models with inappropriate (unrealistic) values of the constant parameters (e.g., negative growth or sat-

[2]All CPU run times are measured on an Intel platform with two 2GHz Pentium IV processors.

uration rate). By analyzing the models induced using unconstrained grammars, we found that some can fit the data better than the original models. This is why the average RMSE with unconstrained grammars are lower than the ones with the fully constrained grammar. However, note that inappropriate structure or constant parameters can lead to problematic models that can not be simulated.[3] The (*N) symbols beside the last two figures from Table 5.1 denote that N out of 16 reconstructed models could not be simulated. Thus, the corresponding error is not an average RMSE of all 16 models, but only of those that were successfully simulated.

The models we tried to reconstruct in the first experiment are not necessarily realistic, since the values of the constant parameters in the models were chosen randomly. It is known that some combinations of parameter values can lead to unstable or unrealistic behaviors (i.e, a concentration of a species explodes beyond any reasonable limits (Murray, 1993)). Such unstable behaviors can make the reconstruction task much more difficult or even impossible, since they cause instability of the procedure that fits constant parameters. To avoid such difficulties, in the next two studies, we will focus on tasks of reconstructing realistic population dynamic models, which have stable and realistic behaviors.

5.1.2 Reconstructing two simple population dynamics models

In the second experiment, we address two tasks of reconstructing realistic models of a simple aquatic ecosystem. The considered ecosystem is similar to the one used in the first experiment, as it involves three system variables of inorganic nutrient, phytoplankton, and zooplankton and six processes, as shown in the first column of Table 5.2.

For the two models used in the experiments, we have to first decide on the modeling alternative for each involved process. Second and third column of Table 5.2 show the choice of modeling alternatives for the first and second model, respectively. The only difference lies in the expression used to model the `Feeds_on(phyto, {nut})` consumption process. In the first model the consumption of nutrient by phytoplankton is unsaturated, while in the second it is saturated.

---

[3]In population dynamics, the usual reason for the inability to simulate a model is "division by zero" singularities encountered due to a negative value of the saturation rate constant parameter.

Table 5.2: A task specification used for modeling a simple aquatic ecosystem that involves two consumption interactions between three system variables. The last two columns specify modeling alternatives used for individual processes in each model.

Process	First model	Second model
`Flow(nut)`	`2.0`	`2.0`
`Growth(phyto)`	$0.1 \cdot (1 - \texttt{phyto}/0.7)$	$0.1 \cdot (1 - \texttt{phyto}/0.7)$
`Decay(phyto)`	$0.2 \cdot \texttt{phyto}$	$0.2 \cdot \texttt{phyto}$
`Feeds_on(phyto, {nut})`	$\texttt{phyto} \cdot \texttt{nut}$	$\texttt{phyto} \cdot \texttt{nut}/(\texttt{nut} + 0.5)$
`Decay(zoo)`	$0.1 \cdot \texttt{zoo}$	$0.1 \cdot \texttt{zoo}$
`Feeds_on(zoo, {phyto})`	$\texttt{zoo} \cdot \texttt{phyto}/(\texttt{phyto} + 0.5)$	$\texttt{zoo} \cdot \texttt{phyto}/(\texttt{phyto} + 0.5)$

Finally, expressions for the individual processes are combined into models of the entire ecosystem as

$$\begin{aligned}
\dot{\texttt{nut}} &= \texttt{Flow(nut)} - 1.0 \cdot \texttt{Feeds_on(phyto}, \{\texttt{nut}\}) \\
\dot{\texttt{phyto}} &= \texttt{Growth(phyto)} - \texttt{Decay(phyto)} + 0.7 \cdot \texttt{Feeds_on(phyto}, \{\texttt{nut}\}) \\
&\quad -0.5 \cdot \texttt{Feeds_on(zoo}, \{\texttt{phyto}\}) \\
\dot{\texttt{zoo}} &= -\texttt{Decay(zoo)} + 0.25 \cdot \texttt{Feeds_on(zoo}, \{\texttt{phyto}\}).
\end{aligned}$$

Figure 5.2 presents the simulation traces of the first and the second model. We generated the simulation traces that were used in the experiments by simulating the models using ten randomly chosen initial states chosen using a random variable uniformly distributed over the $[0.5, 1.5]$ interval. Each simulation trace included 100 time steps of 0.5 time units.

Table 5.3 summarizes the results of the second experiment. They show that context-dependent constraints reduce the space of candidate models by a factor of 16. While the context-dependent grammar generates 64 models, the context-free grammar specifies the space of 1024 models. Context-dependent constraints reduce the average run time of Lagramge by a factor of 20. While the average time for a single run with the context-dependent grammar was about five minutes, a single run with the context-free grammar took about two hours.

Lagramge successfully reconstructed both models from the data with up to 5% relative noise using the fully constrained grammar. From data with 10% relative noise, Lagramge failed to reconstruct the first model; the discovered model suggests exponen-

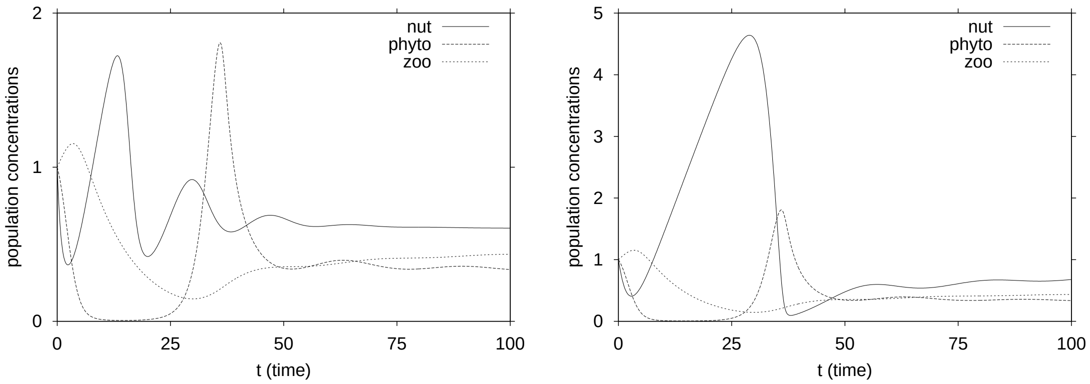


Figure 5.2: Simulation of two first (left-hand side) and second (right-hand side) model of the simple aquatic ecosystem for the initial values of the system variable set to $\texttt{nut}(0) = \texttt{phyto}(0) = \texttt{zoo}(0) = 1.0$.

Table 5.3: Number of candidate models considered during the search, number of successfully reconstructed models (SR), and the average cross-validated RMSE of the models reconstructed by LAGRAMGE using four different grammar versions.

	L2		L2-cdc		L2-cpb		L1	
number of candidate models	64		1024		64		1024	
noise level	SR	RMSE	SR	RMSE	SR	RMSE	SR	RMSE
no-noise	2	0.0714	1	0.0855	2	0.250	0	0.0557
1%	2	0.0601	1	0.0823	1	0.0601	1	0.0905
2%	2	0.0726	2	0.190	1	0.0726	0	0.142
5%	2	0.221	0	0.106	1	0.121	0	0.103
10%	1	0.345	0	0.308	1	0.503	0	0.438
20%	1	0.514	0	0.549	0	0.425	0	0.509

tial phytoplankton growth as opposed to the logistic growth in the original model. From data with 20% relative nose, LAGRAMGE failed to reconstruct the second model; while the original model has zooplankton predation of type 1, the reconstructed model suggests saturation of type 3.

The fully constrained grammar clearly outperforms unconstrained grammars both in terms of cross-validated RMSE and in terms of successful reconstruction rate. Comparing the `L2-cdc` and `L2-cpb` columns in Table 5.3 shows that context-dependent constraints are more important for successful reconstruction than bounds on the values of the constant parameters. Note that previous version of LAGRAMGE failed to reconstruct both models at almost all noise levels.

Table 5.4: A task specification used for modeling a complex aquatic ecosystem consisting of five interactions between five system variables. The second column specifies the models of the individual processes used to obtain the simulated data for the experiments.

Variables	
`variable Inorganic nitro, phosp`	
`variable Population phytoa, phytob, zoo`	
Process	Model
`Flow(nitro)`	0.2
`Flow(phosp)`	0.1
`Growth(phytoa)`	$0.1 \cdot \mathtt{phytoa} \cdot (1 - \mathtt{phytoa}/2.0)$
`Feeds_on(phytoa, {nitro,phosp})`	$\mathtt{phytoa} \cdot \mathtt{nitro} \cdot \mathtt{phosp}/(\mathtt{phosp} + 0.6)$
`Decay(phytob)`	$0.1 \cdot \mathtt{phytob}$
`Feeds_on(phytob, {nitro})`	$\mathtt{phytob} \cdot \mathtt{nitro} \cdot \mathtt{nitro}/(\mathtt{nitro} \cdot \mathtt{nitro} + 0.4)$
`Feeds_on(phytob, {phosp})`	$\mathtt{phytob} \cdot \mathtt{phosp}$
`Decay(zoo)`	$0.5 \cdot \mathtt{zoo}$
`Feeds_on(zoo, {phytoa})`	$\mathtt{zoo} \cdot \mathtt{phytoa}/(\mathtt{phytoa} + 1.7)$
`Interaction({phytob,zoo})`	$\mathtt{zoo} \cdot \mathtt{phytob}$

5.1.3 Reconstructing a complex population dynamics model

In the third and last population dynamics experiment, we selected a fairly complex model of an aquatic ecosystem involving two inorganic nutrients and three organic populations, as shown in Table 5.4. The table specifies the modeling alternatives chosen for individual processes in the model, which represents an open aquatic ecosystem with constant inflows of nitrogen and phosphorus from the environment. The system involves two phytoplankton species, `phytoa` and `phytob`. While the first has a limited growth and needs *both* inorganic nutrients for survival and optimal growth, the second, `phytob`, consumes nitrogen and phosphorus based nutrients (not necessarily both at the same time) and decays exponentially. The zooplankton population also decays and consumes `phytoa`. Finally, there is a symbiotic interaction between the zooplankton and the `phytob` phytoplankton. The models of the individual processes are combined into a model of the entire ecosystem as

$$
\begin{aligned}
\dot{\mathtt{nut}} &= \mathtt{Flow(nitro)} - 1.2 \cdot \mathtt{Feeds\_on(phytoa,\{nitro,phosp\})} \\
&\quad -0.4 \cdot \mathtt{Feeds\_on(phytob,\{nitro\})} \\
\dot{\mathtt{phosp}} &= \mathtt{Flow(phosp)} - 0.7 \cdot \mathtt{Feeds\_on(phytoa,\{nitro,phosp\})} \\
&\quad -1.2 \cdot \mathtt{Feeds\_on(phytob,\{phosp\})}
\end{aligned}
$$

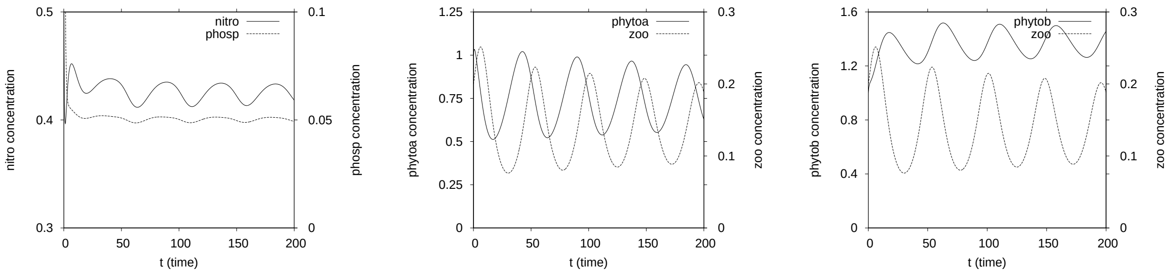


Figure 5.3: Simulation of the complex aquatic ecosystem model for the initial values of the system variable set to $\mathtt{nitro}(0) = \mathtt{phosp}(0) = 0.5$, $\mathtt{phytoa}(0) = \mathtt{phytob}(0) = 1.0$, and $\mathtt{zoo}(0) = 0.2$.

$$
\begin{aligned}
\dot{\mathtt{phytoa}} &= \mathtt{Growth}(\mathtt{phytoa}) + 0.7 \cdot \mathtt{Feeds\_on}(\mathtt{phytoa}, \{\mathtt{nitro}, \mathtt{phosp}\}) \\
&\quad -1.5 \cdot \mathtt{Feeds\_on}(\mathtt{zoo}, \{\mathtt{phytoa}\}) \\
\dot{\mathtt{phytob}} &= -\mathtt{Decay}(\mathtt{phytob}) + 0.2 \cdot \mathtt{Feeds\_on}(\mathtt{phytob}, \{\mathtt{nitro}\}) \\
&\quad +0.2 \cdot \mathtt{Feeds\_on}(\mathtt{phytob}, \{\mathtt{phosp}\}) + 0.2 \cdot \mathtt{Interaction}(\{\mathtt{phytob}, \mathtt{zoo}\}) \\
\dot{\mathtt{zoo}} &= -\mathtt{Decay}(\mathtt{zoo}) + 1.2 \cdot \mathtt{Feeds\_on}(\mathtt{zoo}, \{\mathtt{phytoa}\}) \\
&\quad +0.1 \cdot \mathtt{Interaction}(\{\mathtt{phytob}, \mathtt{zoo}\}).
\end{aligned}
$$

Figure 5.3 presents the behavior of the ecosystem. We generated the simulation traces that were used in the experiments by simulating the models from ten randomly chosen initial states. The initial values of the system variables were chosen using a random variable uniformly distributed over the $[0, 1]$ interval. Each simulation trace included 100 time steps of 0.1 time units.

We ran experiments with LAGRAMGE using a grammar built from the task specification and the modeling knowledge library from Appendix A. The grammar generates 262,144 candidate models. The space of candidate models explodes further if we use a context-free grammar, which generates more than 10^{11} models. Thus, context-dependent constraints reduce the space of candidate models by a factor of over 500,000. Due to the vast search space, we decided to use a beam search strategy with beam width parameter set to 25. The use of beam search reduced the average number of models considered by LAGRAMGE to 1200 for the context-dependent grammar (a single run took an hour on average), and to 5000 for the context-free grammar (average time of a single run was about five hours).

The results of the experiments show that LAGRAMGE fails to reconstruct the model even from noise free data using unconstrained grammars. In contrast, the completely constrained grammar successfully reconstructed the original model from noisy data with relative noise level up to 5%. LAGRAMGE successfully reconstructed eight out of the ten processes of the original model from the 10% noise data. Two processes that were not reconstructed are `Feeds_on(phytoa, {nitro,phosp})` (saturated instead of unsaturated consumption was reconstructed) and `Feeds_on(zoo, {phytoa})` (saturation of type 3 instead of type 1). LAGRAMGE also failed to successfully reconstruct these two processes from the 20% relative noise data. In addition, LAGRAMGE reconstructed the wrong type of symbiotic interaction between zooplankton and `phytob` phytoplankton.

5.1.4 Reconstructing a model of the mass-spring system

In our last experiment with synthetic data, we illustrate the capabilities of our framework to integrate various types of domain knowledge in the process of automated modeling. We use PRET's kind of domain knowledge in attempt to reconstruct the model of the springs and masses system from Figure 4.5 (Bradley et al., 2001):

$$\begin{aligned}
\ddot{\mathtt{q1}} &= -0.1 \cdot \mathtt{q1} - 0.2 \cdot (\mathtt{q1} - \mathtt{q2}) \\
\ddot{\mathtt{q2}} &= 0.2 \cdot (\mathtt{q1} - \mathtt{q2}) - 0.3 \cdot \mathtt{q2}.
\end{aligned}$$

Figure 5.4 presents the behavior of the system in time and phase space. The simulation traces used in the experiments were generated from ten randomly generated initial states (random values uniformly distributed over the $[-0.5, 0.5]$ interval). The length of each simulation trace was 100 time steps of 0.1 time units.

We performed experiments LAGRAMGE using the grammar built from the task specification and modeling knowledge presented in Table 4.17. Note that in this experiment the task specification includes only the list of system variables, i.e., `variable Coordinate q1, q2`. In this case, the grammar enumerates all possible combinations of processes and/or functions that can appear in the model. In the case of mass-spring model with two coordinates, there are four possible functions, i.e., `Single_spring_force(q1)`, `Single_spring_force(q2)`, `Two_springs_force(q1,q2)`, and `Two_springs_force(q2,q1)`, that lead to 16

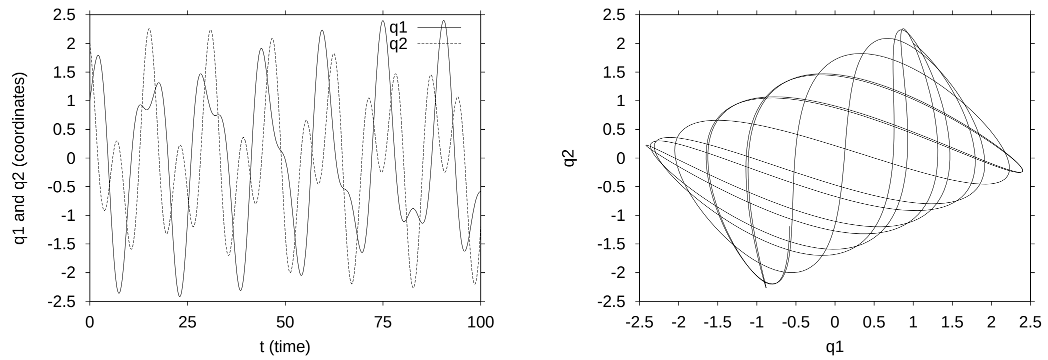


Figure 5.4: Simulation of the springs and masses model with initial values of the system variables set to $\mathtt{q1}(0) = 1$ and $\mathtt{q2}(0) = 2$: (left-hand side) time space and (right-hand side) phase space trajectories.

possible subsets of functions or 16 candidate model structures.[4] The average time of a single run was 4 seconds.

Lagramge successfully reconstructed the original model using the constrained and unconstrained grammar from data at all different noise levels. However, note that the use of context-dependent constraints does not influence the search space in this case, since there is a single modeling alternative for each of the function classes.

5.2 Modeling from real-world measurements

In the second series of experiments, we applied our framework to three modeling tasks that involved real-world data. Two tasks from the population dynamics domain have been already addressed by existing equation discovery methods. The third task is from the hydrodynamics domain. The evaluation criteria used in these experiments are root mean squared error (RMSE) and comprehensibility of the discovered models as evaluated by domain experts.

5.2.1 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 m. It is heavily influenced by anthropogenic inflow of nutrients – 7 [mio kg/year] of

[4] This approach of enumerating all the subsets of possible process and/or function instances with grammar productions cannot be applied in general, since it does not scale well for a large number of candidate class instances. We discuss this issue further in Section 4.4 as a direction for further research.

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. Four sets of measured data were available (Coffaro et al., 1993) for modeling the growth of algae in the Lagoon. The data were 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 $P0_4$ (all in [μg/l]), dissolved oxygen DO (in percentage of saturation), temperature T ([degrees C]), and algal biomass B (dry weight in [g/m^2]).

In previous experiments with automated modeling of algal growth in the Lagoon of Venice with equation discovery, the GoldHorn method (Križman, 1998) was used (Kompare & Džeroski, 1995). Since GoldHorn could not find an accurate model based on the set of measured system variables, two additional variables were calculated and added to the set of system variables. These are the 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%), do not fit the data perfectly, but it 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 algae growth in the Lagoon of Venice from Table 5.5 specifies the types of the observed system variables and the processes that are important for the biomass (algae) growth 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 to search for the model that would reveal them.

Table 5.5: A task specification used for modeling biomass growth in the Lagoon of Venice.

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

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

The experiments with LAGRAMGE were performed using a grammar automatically built from the task specification and the library of modeling knowledge from Appendix A. The grammar generates 6248 candidate model structures. Due to the high measurement errors in the data, we used the MDL heuristic function to avoid overfitting. Among the 6248 candidate models, the model with the minimal value of the MDL heuristic function on the Location 0 data was:

$$\begin{aligned}\dot{\texttt{biomass}} &= 6.17 \cdot 10^{-5} \cdot \texttt{biomass} \cdot (1 - \frac{\texttt{biomass}}{1.80}) \\ &+3.01 \cdot 10^{-4} \cdot \texttt{biomass} \cdot \texttt{DO} \cdot \frac{\texttt{NO3}}{\texttt{NO3} + 6.28} - 0.0319 \cdot \texttt{biomass}.\end{aligned}$$

The model for Location 0 tells us that the limiting factors for the biomass growth in the lagoon are dissolved oxygen (`DO`) and nitrogen in nitrate (`NO3`).

Furthermore, LAGRAMGE discovered another model from the Location 2 data:

$$\begin{aligned}\dot{\texttt{biomass}} &= 4.79 \cdot 10^{-5} \cdot \texttt{biomass} \cdot (1 - \frac{\texttt{biomass}}{0.844}) \\ &+0.406 \cdot \texttt{biomass} \cdot (1 - e^{-0.216 \cdot \texttt{temp}}) \cdot (1 - e^{-0.413 \cdot \texttt{DO}}) \cdot \frac{\texttt{NH3}}{\texttt{NH3} + 10} \\ &-0.0343 \cdot \texttt{biomass}.\end{aligned}$$

This model tells us 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 the 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 on the other two locations (1 and 3), LAGRAMGE did not find an accurate model of the biomass growth. Note that these results still

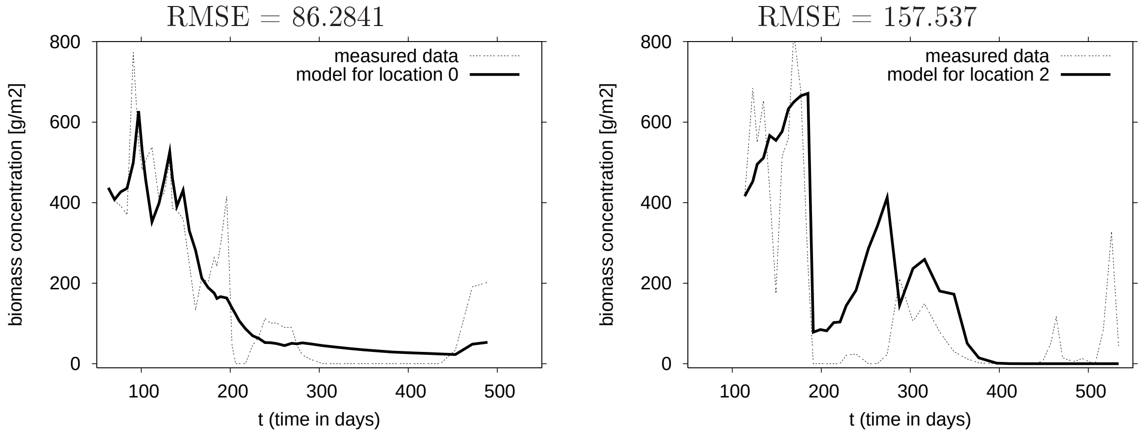


Figure 5.5: Simulations of the two models of the biomass growth in the Lagoon of Venice, discovered by LAGRAMGE, compared to the measured biomass concentration (left-hand side: Location 0, right-hand side: Location 2).

compare favorably with results obtained by GOLDHORN, which discovered an acceptable model for Location 0 only.

Figure 5.5 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, e 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 an 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 important limiting factors for the biomass growth in the Lagoon of Venice.

5.2.2 Modeling phytoplankton growth in Lake Glumsø

Lake Glumsø (Jørgensen et al., 1986) is situated in a sub-glacial valley in Denmark. It is shallow with average depth of about 2 [m] and its surface area is 266,000 [m^2]. For several years, it was receiving mechanically-biologically treated waste water from a community

Table 5.6: A specification of the modeling the phytoplankton growth in Lake Glumsø task.

```
variable Inorganic temp, nitro, phosp
variable Population phyto, zoo

process Decay(phyto) phyto_decay
process Feeds_on(phyto, *) phyto_grazing
process Feeds_on(zoo, phyto) zoo_grazing
```

with 3,000 inhabitants and a surrounding area which was mainly agricultural with almost no industry. The high nitrogen and phosphorus concentration in the treated waste water has caused hypereutrophication. The lake contained no submerged vegetation, probably due to the low transparency of the water and the oxygen deficit at the bottom.

Domain experts considered concentrations of phytoplankton (`phyto`), zooplankton (`zoo`), soluble nitrogen (`nitro`), soluble phosphorus (`phosp`), and with the water temperature (`temp`) relevant for modeling the phytoplankton growth. These variables were measured at 14 distinct time points over a period of two months. The amount of measured data itself was far too small for modeling, so additional processing was applied to obtain a suitable data set (Kompare, 1995). First, dotted graphs of the measurements were plotted and given to three human experts to draw a curve that, in their own opinion, described the dynamic behavior of the observed system variable between the measured points. A properly plotted expert curve can be regarded as an additional source of reliable data. Curves drawn by the human experts were then smoothed with Besier splines. Finally, three data sets were obtained by sampling the splines derived from each of the three human experts' approximations at regular time intervals with time step $h = 0.03215$ day. The data set provided by the first expert was used for the experiments.

The task of modeling phytoplankton growth in the Lake Glumsø from Table 5.6 specifies the types of the observed system variables and the processes that are important for the growth. Note that the specification of the `phyto_grazing` process leaves the nutrient parameter of the `Feeds_on` class unspecified (denoted using the symbol `*`). This is because experts did not know the limiting factors for the biomass growth, and therefore we let Lagramge search for the model that would identify them.

The experiments with LAGRAMGE were performed using a grammar automatically built form the task specification and the library of modeling knowledge from Appendix A. The grammar generates 496 candidate model structures. We used the MDL heuristic function to avoid overfitting. Among the 496 candidate models, the model with the minimal value of the MDL heuristic function on the measured data was:

$$\dot{\texttt{phyto}} = 0.553 \cdot \texttt{temp} \cdot \frac{\texttt{phosp}}{0.0264 + \texttt{phosp}} - 4.35 \cdot \texttt{phyto} - 8.67 \cdot \texttt{phyto} \cdot \texttt{zoo}.$$

The structure of the discovered equation tells us that phosphorus is a limiting factor for phytoplankton growth in the lake and that the growth is temperature dependent.

Note that the same model was already discovered by LAGRAMGE (Todorovski et al., 1998) using a hand-crafted grammar based on the human expert knowledge about modeling population dynamics. However, there is an important difference between the experiment performed here and the one performed with the previous version of LAGRAMGE. In the previous study, we were not able to specify bounds on the values of the constant parameters, so the output of LAGRAMGE was manually post-processed in order to filter out the equations with invalid values of the constant parameters (e.g., negative growth or saturation rate). In the experiment with LAGRAMGE 2.0 there is no need for this additional step, since the knowledge about the valid values of the constant parameters was encoded within the domain knowledge used for equation discovery.

5.2.3 Modeling the water level variation in Ringkøbing fjord

In the last series of experiments, we illustrate that the proposed formalism allows for partial model specification. In such a case, human expert specifies only some parts of the model structure and leaves others unspecified or partly specified. Our framework can be used then to determine both the structure and parameters 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.[5] Wind forcing causes large short term variation of the water level (h)

[5]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

Table 5.7: Formalization of the partially specified model of the water level variation in the Ringkøbing fjord.

```
function class Salt_water_drive(Opening a, Level h_sea, Level h, Surface A)
  expression (h_sea - h + const[h_0:-5:0.1:5])

function class Fresh_water_flow(Flow Q_f, Surface A)
  expression Q_f / A

combining scheme Water_level_change(Level h)
  time_deriv(h) = (F(a) / A) * Salt_water_drive(a, h_sea, h, A) +
    Fresh_water_flow(Q_f, A) + G(W_vel, W_dir)
```

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:

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

The water level response to the wind forcing, dependent on both wind speed (variable W_{vel}, measured in [m/s]) and direction (W_{dir}, measured in degrees), is modeled by an unknown function g. Apart from wind forcing, the water level is dominated by the fresh water supply (Q_f, measured in [m^3/s]). 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 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 observed variable.

In order to apply our framework to the task of model completion, we first encode the partial specification within our formalism. The formalization of the partial model

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 5.8: Formalization of the modeling alternatives for the unspecified parts of the model of the water level variation in the Ringkøbing fjord.

```
function class F(Opening a)

function class F_0 is F
  expression const[_:-5000:0.1:5000]

function class F_1 is F
  expression Polynomial({a}, const[_:-5:0.1:5], 5)
```
```
function class G(Velocity W_vel, Direction W_dir)

function class G_0 is G
  expression const[_:-5000:0.1:5000]

function class G_1 is G
  expression Polynomial({W_vel, W_dir}, const[_:-5:0.1:5], 5)

function class G_2 is G
  expression Polynomial({W_vel, sin(W_dir), cos(W_dir)}, const[_:-5:0.1:5], 5)
```

specification from Table 5.7 follows the partial model formula proposed by the domain experts. The formula is decomposed into two building blocks following the explanation of the partial model specification.

In the second step, we formalize the modeling alternatives for each of the unspecified parts of the model, i.e., the f and g functions. In the experiments, we use simple constant and polynomial models due to the lack of additional domain knowledge. The modeling alternatives used in the experiments are presented in Table 5.8. The first modeling alternatives `F_0` and `G_0` for f and g are the simplest possible models, i.e., constants within the interval $[-5000, 5000]$ with the initial values of 0.1. Next two alternatives (`F_1` and `G_1`) are polynomials of the appropriate system variables with constant parameters within the interval $[-5, 5]$ (and initial values of 0.1). The maximal degree of the polynomials is five. Finally, we used one additional modeling alternative for the g function (`G_2`) that replaces the wind direction value (that represents angle) with the sine and cosine transformation thereof in the polynomial.

Table 5.9: 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. Last column gives number of candidate model structures (#CMS) considered during the search.

task specification	training RMSE	cross-validated RMSE	#CMS
`F_0` + `G_0`	0.0848	0.106	1
`F_1` + `G_1`	0.0655	0.0931	378
`F_1` + `G_2`	0.0585	0.0903	2184
polynomial	0.0556	2.389	2801

The data about the observed variables is collected by hourly measurements of all the observed variables within the period from 1st of January to 10th of December 1999. We used the task specification presented in Tables 5.7 and 5.8 to induce a model from the measurements with LAGRAMGE. We examined three experimental conditions. In the first, we used `F_0` and `G_0` modeling alternatives, in the second we used `F_1` and `G_1` modeling alternatives, and in the third condition we used `F_1` and `G_2` modeling alternatives. In order to evaluate the benefit of using partial model specification, we looked at one additional condition in which no knowledge was used. In this last condition, we used polynomial model of the water level change in the fjord. We used 10-fold cross-validation for estimating the RMSE of the induced models.

Table 5.9 summarizes the results of the experiments. The best cross-validated performance is gained using the partial model specification provided by the experts in combination with `F_1` and `G_2` modeling alternatives for the unspecified parts of the structure. LAGRAMGE proposed the following models for f and g:

$$\begin{aligned}
f(a) &= 5 + 5\cdot a + 5\cdot a^2 + 5\cdot a^3 - 1.01\cdot a^4 \\
g(W_{\text{vel}}, W_{\text{dir}}) &= -0.00137 - 0.0106\cdot\cos W_{\text{dir}} + 0.218\cdot\cos W_{\text{dir}}\cdot\sin W_{\text{dir}} \\
&\quad +0.0106\cdot W_{\text{vel}}\cdot\cos W_{\text{dir}}\cdot\sin W_{\text{dir}} - 0.0128\cdot W_{\text{vel}}^2\cdot\cos W_{\text{dir}}\cdot\sin W_{\text{dir}} \\
&\quad -0.000428\cdot W_{\text{vel}}^3\cdot\cos W_{\text{dir}}\cdot\sin W_{\text{dir}}.
\end{aligned}$$

The graph on the left-hand side of Figure 5.6 shows the simulation of this model compared to the measured water level in the Ringkøbing fjord. We ran long-term simulation of the

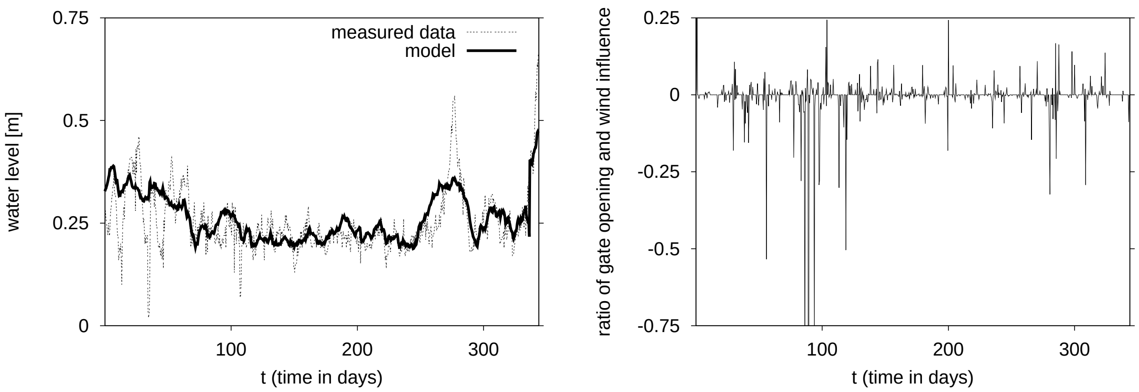


Figure 5.6: Simulation of the water level variation model induced by LAGRAMGE compared to the measured water level (left-hand side) and ratio of the gate opening and the wind influences on the water level change in the Ringkøbing fjord as modeled by LAGRAMGE.

Table 5.10: 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 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

model from the initial value of the water level without restarting the simulation process at any 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.

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 in the fjord. 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 5.10 presents the results of this analysis. They show that 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 of the model 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 $g(W_{\text{vel}}, W_{\text{dir}})$). The graph on the right-hand side of Figure 5.6 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.[6]

The polynomial model of the water level variation that ignores the partial specification of the model performs best on the training data. However, the model's small RMSE is due to the overfitting of the training data, since the cross-validated RMSE of this model (2.389) is much larger than the cross-validated RMSE of the models that follow the partial structure specification.

In sum, the Ringkøbing fjord experiments show the capability of our framework to address modeling tasks in which human experts can partially specify the model structure and leave some of its parts unspecified.

5.3 Summary

In this chapter, we have presented an empirical evaluation of the automated modeling framework on several tasks that involve modeling dynamic systems from synthetic data and real-world measurements.

The results of the experiments with synthetic data show that context-dependent constraints can considerably reduce the space of candidate models considered in the process of automated modeling. Thus, introducing such constraints in LAGRAMGE improves its efficiency on automated modeling tasks. Furthermore, the results show that both context-dependent constraints and bounds on the values of the constant parameters improve the noise robustness of LAGRAMGE. First, all the models discovered by LAGRAMGE using fully constrained grammars can be properly simulated and generate stable behaviors. Second, all the models provide clear and comprehensible interpretation and explanation of the

[6]Note again that we could not obtain expert comments on these results.

system behavior from a biological point of view. And third, fully constrained grammars reconstruct the original model structure much more often than unconstrained grammars. Finally, the experiments with the mass-spring model show that LAGRAMGE can successfully use PRET's kind of knowledge for reconstructing models from noisy data.

The results of the experiments with measurement data show that our framework is capable of building comprehensible dynamic systems' models from real-world data. Our framework performs better than existing equation discovery methods on the tasks of modeling algae growth in Lagoon of Venice and Lake Glumsø in terms of performance, flexibility, and comprehensibility of the discovered models. The final experiment on modeling water level variation in Ringkøbing fjord illustrates the capability of our framework to address modeling tasks, in which a human expert partially specifies the model structure and leaves other parts unspecified. LAGRAMGE can then be applied to complete the partially specified model, i.e., to induce the missing parts from data.

The experimental evaluation shows that our framework integrates several aspects of domain-specific knowledge from variety of the domains in the process of automated modeling. Still, there is an important aspect of domain knowledge that is outside its scope – existing models already established in the domain of interest. Their integration in the process of automated modeling is the topic of the next chapter.

6

Revision of equation based models

Another type of domain-specific knowledge that is neglected by most equation discovery methods are the existing models already established in the domain. Rather than starting the search with an existing, 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 research is mainly concerned with the revision of models expressed in propositional or first-order logic. Therefore, the methods are not directly applicable to the task of revising models based on equations.

In this chapter, we propose a flexible, grammar-based, equation discovery method for revision of 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 the encoded 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 the equation discovery method LAGRAMGE to search through the space of possible revisions and find the one that fits the data better than the initial model.

Theory revision methods follow the minimal revision principle: among theories of similar goodness of fit to the data, ones that are closer to the original theory are preferred. In order to incorporate this principle in our method, we modify the MDL heuristic function used in LAGRAMGE that introduces preference toward simpler equations. The MDL heuristic takes into account complexity of an equation along with its goodness of fit to the data. We replace the complexity of an equation-based model in the MDL heuristic with the distance of the model from the initial one. For measuring this distance, we use a standard measure of distance between tree-structured terms (Shasha & Zhang, 1997).

We evaluate this method for revising quantitative models on a portion of the CASA model that concerns the net production of carbon by terrestrial plants in the Earth ecosystem (Potter & Klooster, 1997). Experimental results show that the method can find revisions that considerably reduce the error of the initial CASA model on the available data.

The chapter is organized as follows. Section 6.1 defines the problem of revising equation based models. Section 6.2 describes the transformation of the given initial model into a grammar. We describe the process of adding modeling alternatives to the initial grammar in Section 6.3. Section 6.4 describes the minimality of change principle for revising equation-based models. We present the experimental methodology used for evaluating of the approach as well as the experimental results in Section 6.5. Finally, Section 6.6 summarizes the chapter and discusses related research.

6.1 Problem definition

The standard problem of theory revision can be defined as follows: **Given** an imperfect domain theory in the form of classification rules and a set of classified examples, **find** an approximately minimal revision of the domain theory that correctly classifies the examples.

A representative method that addresses this problem is EITHER (Ourston & Mooney, 1994), which refines propositional Horn-clause theories using a suite of abductive, deductive, and inductive techniques. Deduction is used to identify the problems with the domain theory, while abduction and induction are used to correct them. The problem of theory revision has received considerable attention in the field of inductive logic programming (Lavrač & Džeroski, 1994), where a number of approaches have been developed for revising theories in the form of first-order Horn clauses. For an overview, we refer the reader to Wrobel (1996).

By analogy with theory revision, the problem of revising equation-based models can be defined as follows: **Given**

- an imperfect existing model M_I of the observed system expressed in the form of equations and
- a set of observations or measurements of the system variables,

find a revised model M_R that

- minimizes the discrepancy between the observed values of the system variables and the values obtained with simulating the model, and
- differs from the initial model M_I as little as possible.

Although this definition is very similar to the one for theory revision, the possible changes or revisions to the two types of models are quite different. As theories are typically logical theories in theory revision settings, the changes typically include the addition and deletion of entire rules (propositional or first-order Horn clauses), as well as the addition and deletion of conditions in individual rules. Latter in this chapter, we propose a framework for specifying plausible changes to equation-based models.

6.2 Transforming the initial model into a grammar

In a typical setting for revising an equation-based model, we would only have observational data and the model, i.e., equations developed by scientists to explain a particular phenomenon. A grammar that would explain how this model was actually derived and provide options for alternative models is typically not available. This is especially true for simpler models.

Table 6.1: The CASA-NPPc model consists of a portion of the CASA model defining *NPPc* variable.

$$
\begin{aligned}
&NPPc = \max(0, E \cdot IPAR)\\
&\quad E = 0.389 \cdot T1 \cdot T2 \cdot W\\
&\quad\quad T1 = 0.8 + 0.02 \cdot topt - 0.0005 \cdot topt^2\\
&\quad\quad T2 = 1.1814/((1 + \exp(0.2 \cdot (TDIFF - 10))) \cdot (1 + \exp(0.3 \cdot (-TDIFF - 10))))\\
&\quad\quad\quad TDIFF = topt - tempc\\
&\quad\quad W = 0.5 + 0.5 \cdot eet/PET\\
&\quad\quad\quad PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^A \cdot pet\_tw\_m\\
&\quad\quad\quad\quad A = 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239\\
&\quad IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5\\
&\quad\quad FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95)\\
&\quad\quad\quad SR\_FAS = (1 + fas\_ndvi/1000)/(1 - fas\_ndvi/1000)\\
&\quad\quad SOL\_CONV = 0.0864 \cdot days\_per\_month
\end{aligned}
$$

However, when the model equations are complex, the model is rarely written as a single equation defining the target variables. More often it is written as a set of equations defining the target variable, which also contains equations that define intermediate unobserved variables. The latter define meaningful concepts in the domain of interest. Often, alternative equations defining an intermediate variable would be possible and the modeling scientist would choose one of these. These alternatives would rarely (if ever) be documented in the model itself, but might be mentioned in a scientific article describing the derived model and the modeling process.

Table 6.1 presents an example of such a complex equation based model, CASA-NPPc, which is one portion of CASA, an earth-science model of the global production and absorption of biogenic trace gases in the Earth's atmosphere. The model, which is described in detail in Section 6.5.1, defines the *NPPc* variable (the net primary production of carbon) in terms of other variables, such as *topt* and *tempc*. Lower case variable names are used to denote observable variables (with the exception of the dependent variable *NPPc*). The remaining variables are unobservable and must be computed from others using their defining equations. The tree-structured graph in Figure 6.1 depicts the dependencies between the observed and unobserved variables in the CASA-NPPc model. Square nodes in the graph denote observable variables, while oval nodes denote unobservable intermediate variables.

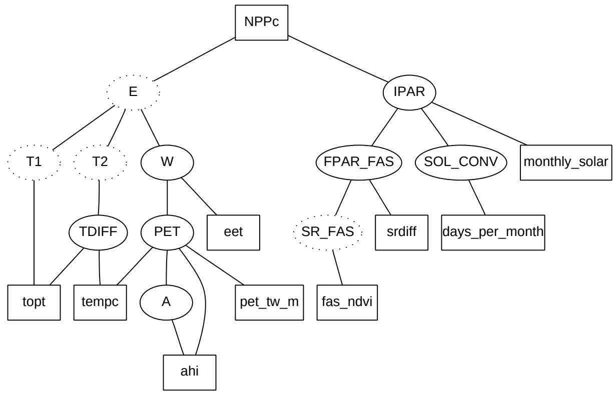


Figure 6.1: The dependencies between observed and unobserved variables in the CASA-NPPc model from Table 6.1.

Table 6.2: A grammar derived from the CASA-NPPc model in Table 6.1. The grammar generates the original CASA-NPPc model only.

`NPPc ->`	`max(0, E * IPAR)`
`E ->`	`0.389 * T1 * T2 * W`
`T1 ->`	`0.8 + 0.02 * topt - 0.0005 * topt * topt`
`T2 ->`	`1.1814 / ((1 + exp(0.2 * (TDIFF-10))) * (1 + exp(0.3 * (-TDIFF-10))))`
`TDIFF ->`	`topt - tempc`
`W ->`	`0.5 + 0.5 * eet / max(PET, 0)`
`PET ->`	`1.6 * pow(10 * max(tempc, 0) / ahi, A) * pet_tw_m`
`A ->`	`0.000000675*ahi*ahi*ahi - 0.0000771*ahi*ahi + 0.01792*ahi + 0.49239`
`IPAR ->`	`FPAR_FAS * solar * SOL_CONV * 0.5`
`FPAR_FAS ->`	`min((SR_FAS - 1.08) / srdiff, 0.95)`
`SR_FAS ->`	`(1 + fas_ndvi / 1000) / (1 - fas_ndvi / 1000)`
`SOL_CONV ->`	`0.0864 * days_per_month`

A set of equations defining a target variable through some intermediate variables can easily be turned into a grammar like the one presented in Table 6.2. The starting symbol of this grammar represents the dependent variable *NPPc*, the nonterminal symbols represent the intermediate variables, and the terminal symbols denote the observed variables and the model's constant parameters. Each nonterminal symbol in the grammar has a single production that generates the model equation used to calculate the respective intermediate variable. Therefore, the grammar in Table 6.2 generates a single model that is equivalent to the one from Table 6.1.

Note, however, that grammar in Table 6.2 lets us specify an arbitrary number of alternative models for each intermediate variable by providing additional productions for the nonterminal symbols in the grammar. These additional productions would specify alternative modeling choices, only one of which will eventually be chosen for the final (revised) model. Observational data could be then used to select among combinations of such choices, with a grammar-based equation discovery system like LAGRAMGE.

6.3 Extending the initial grammar with alternative productions

Note that when alternative productions are specified for an intermediate variable, there are no restrictions (at least in principle) on these productions. For example, they can introduce new intermediate variables and productions defining them. They can also specify arbitrary functional forms. However, they must eventually derive (in the context of the entire grammar) valid sub-expressions involving the set of terminal symbols that represent observed variables.

A very common alternative production would replace the particular constant parameter value on the right-hand side of an existing production with a generic unspecified constant parameter, allowing the equation discovery system to refit them to the given data. The change can be achieved by replacing a terminal symbol that denotes a fixed value constant parameter with the generic symbol `const` that allows for an arbitrary value of the constant parameter. In our experiments with the CASA-NPPc 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:v:2 · 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 $v + 100\% \cdot v = 2 \cdot v$. The default value of the constant parameter is the same as its initial value, in that it is set to v.

Slightly more complex alternative productions would allow for replacing a particular polynomial on the right-hand side of a production with an arbitrary polynomial of the same (intermediate) variable(s). An example of such alternative productions for the nonterminal symbol `T1` from the grammar in Table 6.2 is given in Table 6.3. These productions can be used to generate an arbitrary polynomial of the system variable *topt*.

Table 6.3: Two alternative productions that allow an arbitrary polynomial of the observed variable *topt* to be used for calculating the value of *T1*.

`T1 ->`	`const`
`T1 ->`	`const + (T1) * topt`

This grammar-based framework lets human experts to point out which parts of the model they are completely confident in. These parts should be left intact in the revision process, i.e., no alternative productions should be specified for the corresponding nonterminal symbols. For example, the Earth science experts who built the CASA model pointed out what they considered its "weak" parts. The dotted nodes in the graph in Figure 6.1 the intermediate variables for which they lacked confidence in the associated equations. These are the variables for which alternative productions should be added to the initial grammar.

Another source of alternative productions can be the domain-specific modeling knowledge, encoded with the formalism presented in Chapter 4, although we do not explore this possibility in our experiments.

6.4 The minimality of change principle

While the approach presented above takes into account the initial model, it allows for a completely different model to be derived, depending on which alternative productions are provided for the intermediate variables. It is here that the minimal revision/change principle comes into play: among theories of similar quality (fit to the data), ones that are closer to the original theory are preferred.

The crucial concept that is necessary in order to implement the minimality of change principle is the measure of change or distance between the revised model and the initial model. Since parse trees are used in Lagramge to represent models, we use a measure of distance between tree-structured terms as a measure of distance between models. Thus, our distance measure assesses syntactic structural distance, i.e., the amount of change in the structure of the model's equations.

A common approach to computing distances between strings or tree structured terms is the *editing* approach, leading to *edit distance measure*. Following the editing approach, a set of basic edit operations is first defined. The edit operations available for editing trees are relabeling (changing the label), deleting and inserting a node in the tree. Costs are assigned to these operations, depending on the labels of the nodes involved. The problem of computing the distance between two tree structured terms $\mathcal{T}_1$ and $\mathcal{T}_2$ is then transformed into the problem of finding a minimal cost sequence of basic editing operations that transforms a tree $\mathcal{T}_1$ into a tree $\mathcal{T}_2$.

This problem is $\mathcal{NP}$-complete for the case of unordered tree structures, i.e., structures where the left-to-right order of the children of a node is unimportant. In our case, we are dealing with *ordered* parse trees, since the left-to-right order of the children is important and determined by the production applied to the nonterminal. Thus, the distance between ordered tree structures can be efficiently computed. An overview of algorithms that can be used for computing an edit distance between ordered tree structures is given by Shasha and Zhang (1997). The computation of distances between parse trees can be even more efficient, as illustrated be Richter (1997).

For the purpose of calculating the edit distance between equation based models (or more precisely their parse trees), we use the algorithm proposed in (Richter, 1997). The costs of the basic edit operations are: 1 for deleting a node, 1 for inserting a node, and 1 for relabeling a node, if the label is actually changed or 0 otherwise. Note that for nonterminal symbols which denote constant parameters, the actual value of the constant parameter is considered to be a label.

Measuring the edit distance between two parse trees is illustrated on the example in Figure 6.2. The first tree (on the left-hand side) is generated using a single `T1` production from the initial grammar in Table 6.2. It derives a second degree polynomial of the *topt* variable used to calculate *T1* in the original CASA-NPPc model from Table 6.1. The second parse tree (on the right-hand side) is generated by the alternative productions for `T1` from Table 6.3. It generates a fourth degree polynomial of *topt*. The connections between the nodes of the first and second parse tree in Figure 6.2 represent the minimal cost sequence of editing operations that are needed to transform the first parse tree into the second. First,

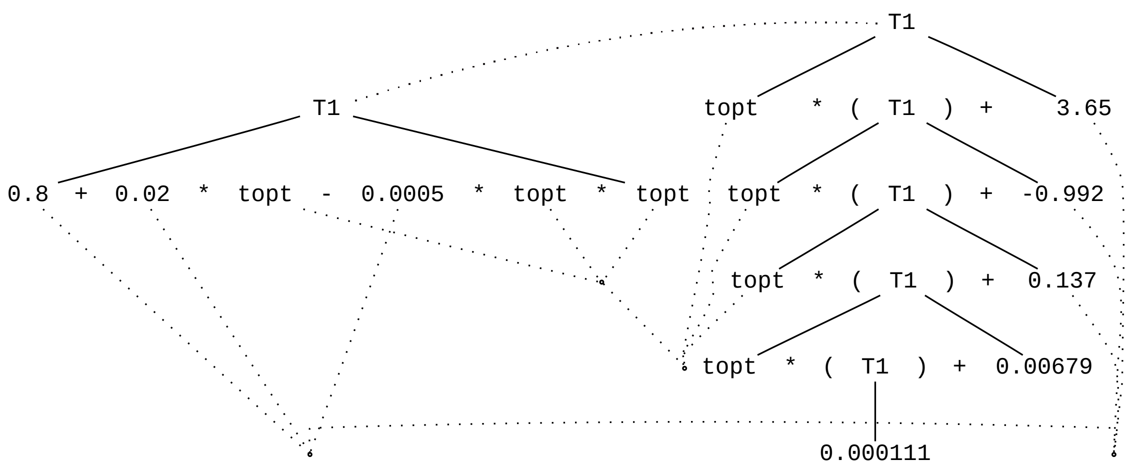


Figure 6.2: Calculating the edit distance between two parse trees representing two polynomials derived by the grammars in Table 6.1 (left-hand side) and Table 6.3 (right-hand side).

the nodes in the first parse tree that are not connected to any node in the second tree have to be removed. Since there is only one such node in the first parse tree the total cost of the remove operations is 1. Second, the nodes in the second tree that are not connected to any of the nodes in the first one have to be inserted. There are 19 such nodes, so the total cost of the insert operations is 19. Finally, the connections between nodes represent the relabeling operations. Note that there are three connections where relabeling is actually necessary (i.e., the labels of the connected nodes are different). Thus, the total cost of the relabel operations is 3. Summing up the cost of the remove, insert and relabel operations gives us the edit distance of 23. Therefore, the distance between the parse trees on the left-hand side and the right-hand side of Figure 6.2 is 23.

Once we have defined a distance measure between models, we can incorporate it into LAGRAMGE by modifying the MDL heuristic that introduces preference toward simpler equations. This heuristic, which takes into account both the complexity of an equation and its goodness of fit to the data (Todorovski, 1998; Todorovski & Džeroski, 1997), can be stated as

$$\mathrm{MDL}(M) = \mathrm{SSE}(M) + \frac{l(M)}{10 \cdot l_{\max}} \cdot \mathrm{SSE}(M_0),$$

where $\mathrm{SSE}(M)$ is the sum of squared errors of the current model on the training data, $\mathrm{SSE}(M_0)$ is the error of the simplest model, $l(M)$ is the length of the current model M

(in number of terminal symbols) and $l_{\max}$ the length of the most complex equation in the search space. Since the LAGRAMGE search space consists of parse trees with limited depth, the maximal length $l_{\max}$ can be easily computed in advance. Roughly speaking, the second part of the MDL heuristic function of LAGRAMGE adds a penalty for equation complexity to the sum of squared errors.

By analogy to the MDL heuristic, we can define the MC (minimality of change) heuristic function as

$$\mathrm{MC}(M) = \mathrm{SSE}(M) + \frac{\mathrm{distance}(M, M_0)}{C} \cdot \mathrm{SSE}(M_0),$$

where $\mathrm{distance}(M, M_0)$ is the distance between the current model M and the initial model M_0. Note that the maximal distance is not available as in the case of maximal length for MDL, so we introduce a user-defined parameter C. This can be used to trade off between the current model's goodness of fit and its distance from the initial model. Large values of C will diminish the "change penalty" term of the MC heuristic, leading to a preference toward accurate models that are not necessarily similar to the initial one. On the other hand, small values of C increase the "change penalty" term, leading to a preference toward models that are similar to the initial one.

6.5 Experimental evaluation

We applied the method for revising of equation-based models to the task of revising the part of the CASA model (Potter & Klooster, 1997). In this section, we briefly review the initial CASA-NPPc model, describe the methodology of the revision experiments, and present the experimental results.

6.5.1 The CASA earth-science model

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, as well as predicting changes in the geographic patterns of major vegetation types (e.g., grasslands, forest, tundra, and desert) on the land.

Table 6.4: Observed system variables and unobserved intermediate variables used in the NPPc portion of the CASA model.

Observed system variables
NPPc is the net production of carbon by terrestrial plants at a site.
topt is the average temperature for the month at which *fas_ndvi* takes on its maximum value at a site.
tempc is the average temperature at a site for a given month.
eet is the estimated evapotranspiration (water loss due to evaporation and transpiration) at a site.
pet_tw_m is a component of potential evapotranspiration that takes into account the latitude, time of year, and days in the month.
ahi is an annual heat index that takes the time of year into account.
fas_ndvi is the relative greenness as measured from space.
monthly_solar is the average radiation incoming for a given month at a site.

Unobserved intermediate variables
E is the photosynthetic efficiency at a site after factoring various sources of stress.
T1 is a temperature stress factor ($0 < T1 < 1$) for cold weather.
T2 is a temperature stress factor ($0 < T2 < 1$), nearly Gaussian in form but falling off more quickly at higher temperatures.
W is a water stress factor ($0.5 < W < 1$).
PET is the potential evapotranspiration (water loss due to evaporation and transpiration given an unlimited water supply) at a site.
A is a polynomial function of the annual heat index at a site.
IPAR is the energy intercepted from the sun after factoring in the time of year and days in the month.
FPAR_FAS is the fraction of energy intercepted from the sun that is absorbed photo-synthetically after factoring in vegetation type.
SOL_CONV is 0.0864 times the number of days in each month.

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. The model is based on difference equations that represent the terrestrial carbon cycle, as well as processes that mineralize nitrogen and control vegetation type. These equations describe relations among quantitative variables and lead to changes in the modeled outputs over time. CASA 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 lies on the model's "fringes" and that does not involve any difference equations. Table 6.4 describes the variables that occur in this submodel, in which the dependent variable, NPPc, represents the net production of carbon by terrestrial plants. As Table 6.1 indicates, the model predicts this quantity as the product of two unobservable variables, the photosynthetic efficiency, E, at a site and the solar energy intercepted, *IPAR*, at that site.

Photosynthetic efficiency is in turn calculated as the product of the maximum efficiency (0.389) and three stress factors that reduce this efficiency. One stress term, *T2*, takes into account the difference between the optimum temperature, *topt*, and actual temperature, *tempc*, for a site. The second 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 photo-synthetically 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 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 the equations in Table 6.1.

6.5.2 Experimental methodology

The training data set used in the experiments of CASA-NPPc model revision consists of 303 data points, each of which contains measurements of the observed system variables for a distinct location on the Earth.

The quality of the revised models is assessed through the discrepancy between the predicted and observed values of the dependent variable: the smaller the discrepancy, the better the model. The discrepancy is measured using standard root mean squared error (RMSE) measure, calculated as $\sqrt{\sum_{i=1}^{303}(NPPc_i - N\hat{P}Pc_i)^2/303}$, where $NPPc_i$ and $N\hat{P}Pc_i$ are the observed and the predicted value of $NPPc$, respectively. The RMSE of the initial model on the training data is 517.665.

In order to estimate the error of the revised models on test data unseen during the process of revision, we applied a 30 fold cross-validation methodology. Following this methodology, the data set of 303 examples is randomly partitioned into 30 partitions, with approximately the same number of (ten) examples in each of them. In each iteration of the cross-validation procedure, twenty-nine out of thirty partitions are used as training data for revision of the initial model and the revised model is then used to predict the values of the dependent variable $NPPc$ on the remaining partition, unseen during the revision phase. By repeating this process thirty times, once for each partition, we obtain 303 predictions of the $NPPc$ value for all the data points in the training set.

6.5.3 A grammar for the revision of the CASA-NPPc model

As described in Section 6.2, the given CASA-NPPc model was first transformed into the initial grammar presented in Table 6.2. In addition, alternative predictions were added to this initial grammar for the four intermediate equations for which experts were not confident. (dotted nodes in the graph in Figure 6.1). Each of these alternative productions specifies one or more possible revisions of the initial CASA-NPPc model. Table 6.5 presents the complete list of alternative productions added to the initial grammar.

Alternative productions for E

`Ec-100` allows a 100% relative change of the constant parameter (with the initial value of 0.389) in the equation for the intermediate variable E.

`Es-exp` allows for a replacement of the product $T1 \cdot T2 \cdot W$ from the product from the initial E equation with an expression that allows for arbitrary exponents

Table 6.5: Alternative productions added to the initial grammar from Table 6.2. Each of them specifies one or more revisions of the initial CASA-NPPc model.

<table>
<tr><td><code>Ec-100: E -></code></td><td><code>const[_:0:0.389:0.778] * T1 * T2 * W</code></td></tr>
<tr><td><code>Es-exp: E -></code></td><td><code>const[_:0:0.389:0.778] * pow(T1, const[_:0:1:])</code>
<code>* pow(T2, const[_:0:1:]) * pow(W, const[_:0:1:])</code></td></tr>
<tr><td><code>T1c-100: T1 -></code></td><td><code>const[_:0:0.8:1.6] + const[_:0:0.02:0.04] * topt</code>
<code>- const[_:0:0.0005:0.001] * topt * topt</code></td></tr>
<tr><td><code>T1s-poly: T1 -></code></td><td><code>const | const + (T1) * topt</code></td></tr>
<tr><td><code>T2c-100: T2 -></code></td><td><code>const[_:0:1.1814:2.3628] / ((1 +</code>
<code>exp(const[_:0:0.2:0.4] * (TDIFF - const[_:0:10:20])))</code>
<code>* (1 + exp(const[_:0:0.3:0.6] *</code>
<code>(-TDIFF - const[_:0:10:20]))))</code></td></tr>
<tr><td><code>T2s-poly: T2 -></code></td><td><code>const | const + (T2) * TDIFF</code></td></tr>
<tr><td><code>SR_FASc-25: SR_FAS -></code></td><td><code>(1 + fas_ndvi / const[_:750:1000:1250])</code>
<code>/ (1 - fas_ndvi / const[_:750:1000:1250])</code></td></tr>
</table>

on the three participating terms (i.e., a product of the form $T1^{c_1} \cdot T2^{c_2} \cdot W^{c_3}$). The initial values of the exponents are set to 1, in which case the product is equivalent to that in the initial E equation.

Alternative productions for T1

`T1c-100` allows for a 100% relative change of the initial values of the constant parameters in the *T1* equation.

`T1s-poly` provides for a replacement of the initial second degree polynomial for *T1* with an arbitrary degree polynomial of the variable *topt*. LAGRAMGE's parameters were set to allow polynomials with the maximum degree of five.

Alternative productions for T2

`T2c-100` enables 100% relative change of the initial values of the constant parameters in the equation for `T2`.

`T2s-poly` supports replacement of the initial equation for *T2* with an arbitrary degree polynomial of the variable *TDIFF*. Again, the maximum degree of the polynomial was limited to five.

Alternative productions for SR_FAS

`SR_FASc-25` lets a 25% relative change of the initial values of the constant parameters in the *SR_FAS* equation. We used 25% here to avoid values of the constant parameters below 750, which would cause singularity (division by zero) problems in the equation for *SR_FAS*.

Note, however, that an arbitrary combination of these alternative productions can be added to the initial grammar. If all the alternative productions are added at the same time, then LAGRAMGE will find the most beneficial combination of revisions, i.e., the one that leads to the best revision of the initial model. In this case, LAGRAMGE considers 384 possible revisions of the original CASA-NPPc model.

6.5.4 Experimental results

Table 6.6 summarizes the results of the experiments with the different modeling (revision) alternatives, discussed above.

When we allow only a single of the seven presented alternatives (the first seven rows of Table 6.6), revising the value of the parameters in the equation for *SR_FAS* gives the largest error reduction over the initial CASA-NPPc model. The initial values of the parameters (both are equal to 1000) define an almost linear dependence of *SR_FAS* on the observed system variable *srdiff*. The revised values of the constant parameters were equal to the lower bound of 750, which increase the non-linearity of the dependence. However, lower values of the parameters in the *SR_FAS* equation would cause singularity (division by zero) problems, due to the range of the *srdiff* variable. In terms of consistency of the revision with Earth science knowledge, we should note that the Earth scientists' confidence in the range of the *srdiff* variable was low due to the limited terrestrial coverage of the *NPPc* measurements. Therefore, the theoretically based argument for high initial values of the constant parameters in the *SR_FAS* equation is not very strong.

The `T1s-poly` revision replaces the original second-degree polynomial for calculating *T1* with a fifth degree polynomial. The structural revision `T2s-poly` replaced the complex initial equation structure for calculating *T2* with a fourth degree polynomial. While the

Table 6.6: The root squared mean error (RMSE) of the revised model, the percentage of relative error reduction (RER) of the revised model when compared to the RMSE of the initial CASA-NPPc model (with RMSE of 517.665) and distance (DIST) of the revised model from the initial one. The RMSE was estimated both on training data (training - the left-hand side of the table) and using 30-fold cross-validation (CV - the right-hand side of the table).

alternative	training			CV		
production(s)	RMSE	RER (%)	DIST	RMSE	RER (%)	DIST
`Ec-100`	458.626	11.40	1	459.212	11.29	1.0
`Es-exp`	442.763	14.47	16	447.456	13.56	16.0
`T1c-100`	458.301	11.47	3	460.352	11.07	3.0
`T1s-poly`	450.265	13.02	46	455.819	11.95	46.0
`T2c-100`	457.048	11.71	3	457.926	11.54	3.0
`T2s-poly`	450.972	12.88	71	463.757	10.41	75.8
`SR_FASc-25`	441.419	14.73	2	441.419	14.73	2.0
All combined	411.627	**20.48**	60	421.758	**18.53**	62.6

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 plausible.

The most interesting structural revision was the one for the equation

$$E = 0.610 \cdot T1^{2.83} \cdot T2^{0.638} \cdot W^{0}$$

The proposed value of 0 for the exponent of the water stress factor W suggests it is not important for predicting the photosynthetic efficiency E. Earth scientists proposed that this influence is already being captured by the satellite measurements of the relative greenness, *fas_ndvi*, and this was unnecessary in the E equation.

The last row of Table 6.6 presents the results of the experiments with search for the optimal combination of all the alternative revisions. As expected, the best combination leads to the maximum relative error reduction of more than 20% on the training data and 18.5% when cross-validated. The combination of `Es-exp`, `T1s-poly`, `T2c-100`, and `SR_FASc-25` productions led to the best revised model, as shown in Table 6.7.

Table 6.7: The revised CASA-NPPc model obtained by allowing an arbitrary combination of modeling alternatives from Table 6.5. The parts of the models that are not revised are printed in gray.

$$
\begin{aligned}
&NPPc = \max(0, E \cdot IPAR)\\
&\quad E = 0.312 \cdot T1^{1.36} \cdot T2^{0.728} \cdot W^{0}\\
&\qquad T1 = 3.65 - 0.992 \cdot topt + 0.137 \cdot topt^2 - 0.00679 \cdot topt^3 + 0.000111 \cdot topt^4\\
&\qquad T2 = 0.818/((1 + \exp(0.0521 \cdot (TDIFF - 10))) \cdot (1 + \exp(0 \cdot (-TDIFF - 10))))\\
&\qquad\quad TDIFF = topt - tempc\\
&\qquad W = 0.5 + 0.5 \cdot eet/PET\\
&\qquad\quad PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^A \cdot pet\_tw\_m\\
&\qquad\qquad A = 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239\\
&\quad IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5\\
&\qquad FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95)\\
&\qquad\quad SR\_FAS = (1 + fas\_ndvi/750)/(1 - fas\_ndvi/750)\\
&\qquad SOL\_CONV = 0.0864 \cdot days\_per\_month
\end{aligned}
$$

After the initial experiments with the revision of the CASA-NPPc model presented here, we found out that the Earth scientists who developed the CASA model corrected the value of the constant parameter in the E equation from 0.389 to 0.56 independently of our experiments. This change reduces the RMSE of the initial CASA-NPPc model on the training data from 517.665 to 465.213. After re-running the revision experiments with the new initial CASA-NPPc model, we obtained the results, as presented in Table 6.8.

The revisions of the new corrected initial CASA-NPPc model led to smaller relative reduction of the RMSE. We obtained the maximum error reduction of almost 11% on the training data and 9% when cross-validated when an optimal combination of modeling alternatives was sought. Table 6.7 presents the best revised model that was obtained using the combination of `Es-exp`, `T1c-100`, `T2s-poly`, and `SR_FASc-25` productions. Note that the sum of the reductions obtained with single alternative productions nearly add up to the error reduction obtained with a combination of them.

Note also that the error of the revision of the corrected model on the training data (414.739) is slightly higher than the error of the best model obtained with revising the original CASA-NPPc model (411.627, see Table 6.6). This is due to the problems with

Table 6.8: The root squared mean error (RMSE) of the revised model, the percentage of relative error reduction (RER) of the revised model when compared to the RMSE of the (corrected) initial CASA-NPPc model (with RMSE of 465.213) and distance (DIST) of the revised model from the initial one. The RMSE was estimated both on training data (training - the left-hand side of the table) and using 30-fold cross-validation (CV - the right-hand side of the table).

alternative	training			CV		
production(s)	RMSE	RER (%)	DIST	RMSE	RER (%)	DIST
`Ec-100`	458.626	1.42	1	460.5	1.01	0.9
`Es-exp`	443.029	4.77	16	443.032	4.77	16.0
`T1c-100`	458.301	1.49	3	460.799	0.95	3.0
`T1s-poly`	450.265	3.21	46	457.37	1.69	45.8
`T2c-100`	457.018	1.76	3	459.633	1.20	3.0
`T2s-poly`	450.972	3.06	71	461.642	0.77	73.4
`SR_FASc-25`	453.157	2.59	2	455.281	2.13	2.0
All combined	414.739	**10.85**	104	423.684	**8.93**	67.4

the convergence of the method for non-linear optimization of the values of the constant parameters. It is well known that these methods can not guarantee convergence toward the global (or real) optimal values, but can get stuck into a local (sub-)optimal values that are closer to the initial values of the constant parameters (Press et al., 1986). This imperfection of the non-linear optimization methods can be partly avoided by using the multi-start procedure presented in Chapter 4 (Section 4.2). However, in these cases re-starting the optimization with 25 randomly generated initial values still did not lead to the optimal values of the constant parameters.

The comparison of the revised models in Table 6.7 and Table 6.9 shows that the revised models are similar. Both suggest that the W (watter stress) segment should be removed from CASA-NPPc model, since it is not important for calculating E. Furthermore, both revised models suggest a lower value (750) for the constant parameter in the SR_FAS equation. On the other hand, the models suggest different revisions of the $T1$ and $T2$ equations.

Table 6.9: The new revised CASA-NPPc model obtained by allowing an arbitrary combination of modeling alternatives from Table 6.5. The parts of the models that are not revised are printed in gray.

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

In another experiment, we explored the influence of the minimality of change principle on the revised models. For this purpose, we used the MC heuristic function (see Section 6.4) with seven different values of the C parameter: 32, 64, 128, 256, 512, 1024, and 2048. Recall from Section 6.4 that the C parameter is used to trade off between goodness of fit of the model and minimality of change with respect to the initial model. Smaller values of C give a higher preference toward models that are similar to the initial one.

Figure 6.3 summarizes the results of this experiment. As expected, the distance of the revised model from the initial model constantly increases with the value of the C parameter. The distance is greatest when SSE heuristic function is used, i.e., the minimality of change principle is neglected. The trend of the relative error reduction, as estimated on training data, is the same: it constantly increases and reaches a maximum with SSE. Thus, the more distant the revised model, the more accurate it is on training data.

The revised model that is most similar to the initial one (i.e., the one found using MC heuristic with $C = 32$) is obtained by revising the values of the constant parameters (`Ec-100` and `SR_FASc-25`) of the initial model, leading to an error reduction of 5.29%. This

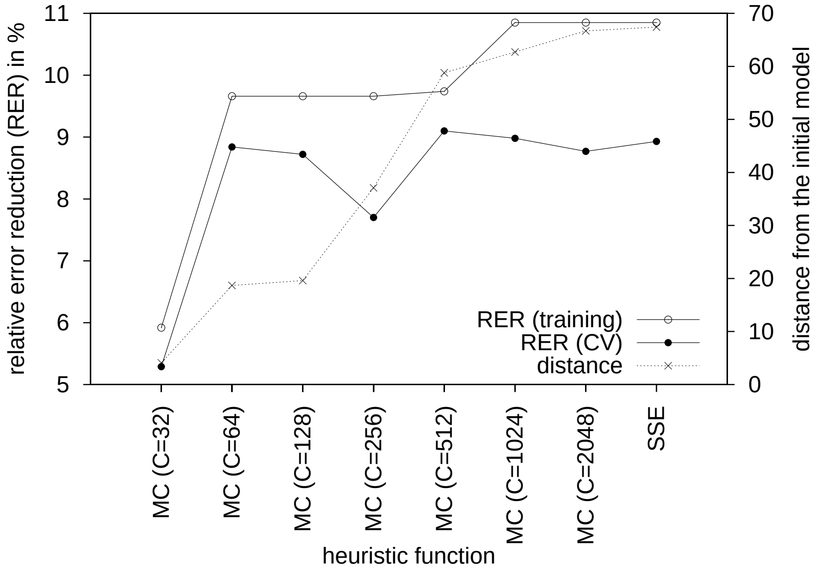


Figure 6.3: Relative error reduction and distance from the initial model for revised models obtained using SSE and MC (minimality of change) heuristic function with different values of the C parameter.

shows that the revisions of the initial equations for E and especially SR_FAS are necessary and important for the error reduction, even if we prefer a minimal change of the initial CASA-NPPc model. The analysis of the second revised model obtained with $C = 64$ gives further support for this claim, as it leads to an error reduction of 8.84% by revising these two equations again, in this case proposing a more complex structural revision (`Es-exp`) of the E equation.

However, the increasing the performance of the revised models on training data can easily lead to overfitting, especially in cases when arbitrary revisions are allowed. Even in these experiments with a limited set of revision alternatives, we can see that the cross-validated error reduction does not constantly increase, and models that are closer to the initial one can perform better on test data. In our experiments, the model obtained using $C = 512$ (error reduction of 9.10%) slightly outperforms the model obtained using the SSE heuristic (error reduction of 8.93%) when cross-validated. The revised model obtained using $C = 512$ leaves the *T2* equation unchanged and has a structure that is otherwise identical to that in Table 6.9, although the values of the constant parameters in the equations are slightly different. This shows that the revision of the *T2* equation is not really important for the reducing the error produced by the initial CASA-NPPc model.

6.6 Summary

In this chapter, we have proposed a flexible grammar-based method for revising equation based models. We use the transformation principle to support the revision of existing models with equation discovery. First, the given existing model is transformed into a grammar that can be used to derive the initial model only, with the nonterminals and their productions reflecting the structure of the initial model. Domain experts then focus the revision process on parts of the model and guide it by providing relevant modeling alternatives that are added to the grammar as alternative productions. In this way, the revision process can be interactive, as is quite often the case when revising theories expressed in logic. The method also incorporates the minimality of change principle in a way that allows a trade off between the revised model's goodness and its similarity to the initial one.

We have applied our approach to the problem of revising a portion of CASA, which models the production of carbon by terrestrial plants in the Earth ecosystem. Experimental study showed that small revisions of both the constant parameters and the equation structure reduce the error of the model considerably (by almost 20%). This improvement is regarded as non-trivial by Earth scientists who developed the CASA model. Furthermore, an experiments with an improved version of the CASA-NPPc model still lead to a revised model that has almost 9% lower error than the initial one. An additional experiment showed the importance of the minimality of change principle from two aspects. First, this heuristic can slightly improve the accuracy of the revised model on test data, the were not used during the revision process. Second, changing the parameter for trading off between goodness of fit and similarity to the initial model can help identify the important revisions that produce largest improvements of the accuracy of the initial model.

The research presented in the chapter is closely related to two other lines of work. In the first, Saito et al. (2001) address the same task of revising models based on equations.

Their approach transforms 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 slightly lower accuracy improvement than did our method. 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 to translate the existing CASA model into a declarative form that our discovery system can manipulate" is a challenge to their approach. Moreover, their method does not incorporate the minimality of change principle.

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 (2000) 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 (2000) 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. However, their grammars are too general to provide much information about the domain at hand, and they do not incorporate minimality of change ideas in their approach. This can be considered as a weakness, since genetic programming methods tend to produce large expressions without a simplicity bias.

Our approach to revising the CASA model employs three kinds of domain knowledge. Domain experts provided the initial CASA model, and they pointed out uncertain parts of the model that should be considered for revisions. The third kind of knowledge concerns the alternative models to be used for the revision. In addition, the domain experts provided feedback about the importance and comprehensibility of each of the proposed revisions. However, there is still room for incorporating additional expert knowledge in the revision process. In particular, note that the measure of distance between models is purely syntactic, in sense that it calculates the minimal number of elementary edit operations (i.e., deleting, inserting and relabeling a node) necessary to transform the initial model into the revised one. By assigning different costs of the elementary edit operations for different label nodes, we can introduce additional expert knowledge about the amount of change introduced by the revisions. This can be seen as a way to introduce semantics in the distance measure and constitutes an interesting direction for further work. However, domain experts can also propose the distance measure that is purely semantic and based on domain-specific knowledge about possible models. Such modified distance measures could be easily incorporated within the MC heuristic measure used in LAGRAMGE.

7

Conclusion

In the thesis we developed and empirically evaluated several methods that extend the scope of equation discovery along two dimensions. The first dimension involves the formalism for representing equation based models and the second concerns the amount and type of domain knowledge integrated within the equation discovery process.

In Chapter 3, we presented a new method that extends the scope of existing discovery methods to handle partial differential equations. These present an important class of equations, since they are capable of modeling the change of the observed system along more than one dimension, as opposed to ordinary differential equations, which can only model changes in one (usually the temporal) dimension.

In Chapter 4, we proposed a framework for automated modeling of real-world systems consisting of a knowledge representation formalism and an improved version of the equation discovery method LAGRANGE that can take into account the encoded knowledge. The knowledge is organized around the basic processes in the domain in a way that is accessible to domain experts. The formalism and the method are general enough to integrate a great variety of domain-specific modeling knowledge into the equation discovery process. Furthermore, in Chapter 6, we presented a method for revision of existing equation based models. The methods described in these two chapters extend the capability of existing equation discovery methods to incorporate different kinds of knowledge, ranging from existing models to knowledge about basic processes in the domain of interest.

We applied the newly developed methods to a number of tasks to illustrate their capabilities. These tasks involve modeling from real-world measurements and synthetic data in the domains of population dynamics, neurophysiology, classical mechanics, hydrodynamics, and Earth science. The results of the modeling experiments provide important empirical evidence for both the expressiveness of the knowledge representation formalism and the inductive power of LAGRANGE.

The evaluation of the modeling framework on synthetic data shows the capability of the method to reconstruct structures of fairly complex population dynamics and spring mechanic models from noisy data. The use of domain knowledge improves both the efficiency and the noise robustness of the equation discovery method LAGRAMGE. The experiments with the methods for discovering partial differential equations show that they can reconstruct particularly important neurophysiology model of temporal and spatial propagation of impulses along the nerve axon from noise-free simulation traces.

Finally, the application on the real-world tasks of modeling from measured data confirms the usability of the approach. LAGRAMGE identifies comprehensible white-box models of algae growth in Lake Glumsø and Lagoon of Venice that help experts identify the limiting factors for the algae growth, which causes environmental problems. The results compare favorably with results obtained using the equation discovery method GOLDHORN and the previous version of LAGRAMGE. With the experiments of modeling water level variation in the Ringkøbing fjord, we illustrate the capability of the approach to complete partially specified models. Finally, we used our model revision method to improve the accuracy of CASA, an earth-science model of the global production and absorption of biogenic trace gases in the Earth's atmosphere. The improvement is regarded as non-trivial by Earth scientists who developed the model.

7.1 Original contributions

The thesis contributes to three important aspects of the equation discovery area. The contributions to each of them are further discussed in the following subsections.

7.1.1 Discovery of partial differential equations

A new method capable of discovering partial differential equations is presented. The utility of the method is evaluated on several tasks of reconstructing known models of real-world systems from synthetic data. These include a model from the domain of neurophysiology that represents a wide and particularly important class of models of biological systems.

7.1.2 Integration of domain-specific knowledge in the equation discovery process

A formalism for encoding domain-specific modeling knowledge is presented. The formalism allows for encoding knowledge from a variety of domains, including population dynamics, biochemistry, and classical mechanics, as well as domain-independent knowledge about modeling based on (possibly incomplete) information about measurement units of the observed system variables. An automated modeling framework is presented based on the discovery method capable of incorporating the encoded knowledge in the process of equation discovery. The framework is successfully applied to several tasks of reconstructing population dynamics models from synthetic data as well as completing a partially specified hydrodynamics model from real-world measurement data.

7.1.3 Revision of equation based models

A new method is developed capable of revising models based on equations. The method can follow the minimality of change principle, i.e., among models with similar accuracy, prefer the ones that are as similar as possible to the initial model. The method is successfully used for revising real Earth science model using real-world measurements.

7.2 Limiations and further work

The directions for further work were already discussed in the summary sections of the individual thesis' chapters. Here, we will provide an annotated list of directions for further work. For further details, consult the summary sections of the appropriate chapter.

7.2.1 Further evaluation

Further evaluation of the proposed methods for discovery of partial differential equations should be performed before they acceptable to mathematical modelers. Further experiments with more models and with truly observational data are necessary. The simulated data used in the experiments presented in the thesis contain some errors (due to the numerical error of the method used to simulate the PDEs of the original model), which are of a different nature to the measurement errors found in experimental data. We would need to establish that the method works robustly under both conditions.

7.2.2 Domain knowledge

The immediate direction of further (and partly ongoing) 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 with the process of modeling. 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 measurement data.

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, the two level search procedure should be developed that is capable discovering the processes that influence the behavior of the observed system. At the high level the search will search 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.

Another direction of improvement is integration of other kind of domain knowledge in the process of model revision, that is knowledge about distance between models. Domain experts can either specify a semantic distance measure or specify "semantic based" costs of the elementary edit operations for the editing distance measure. Formalization of this kind of knowledge is an open issue that can be an interesting topic of further research.

7.2.3 Integration

The methods developed within the thesis are in the early development phase. Each method is 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 the method for discovering partial differential equations is not integrated within LAGRAMGE. There is a clear need for proper integration 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[1] 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.

Another direction for further work includes the integration of equation discovery (and the revision of equation-based models) methods within the framework of inductive databases (Imielinski & Mannila, 1996). In a given application domain, an inductive database contains not only data about the domain, but also patterns or models, such as (in our case) equations. Equation discovery operations can then be viewed as inductive queries posed to the inductive database. They can discover models from data only or revise models

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

from the inductive database in light of newly collected data. Allowing constraints on the search space of candidate models and constraints based on the distance of the induced models from a given one are important primitives for building inductive queries posed on an inductive database of equation based models.

Bibliography

Bassingthwaighte, J. B. (Ed.). (2002 (Web page update)). *Web page of the physiome project*. `http://www.physiome.org/`.

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

Bradley, E., Easley, M., & Stolle, R. (2001). Reasoning about nonlinear system identification. *Artificial Intelligence*, *133*, 139–188.

Bratko, I. (2001). *Prolog programming for artificial intelligence*. Addison-Wesley. Third Edition.

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

Capasso, V. (1993). *Mathematical structures of epidemic systems*, vol. 97 of *Lecture Notes in Biomathematics*. Berlin: Springer.

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

Crispi, G., & Mosetti, R. (1993). Adjoint estimation of aquatic ecosystem parameters. *Coenoses*, *8*, 11–14.

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

Falkenhainer, B., & Michalski, R. (1990). Integrating quantitative and qualitative discovery in the abacus system. In Y. Kodratoff and R. Michalski (Eds.), *Machine learning: An artificial intelligence approach*. San Mateo, CA: Morgan Kaufmann.

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

Farquhar, A. (1993). *Automated modeling of physical systems in presence of incomplete knowledge.* Doctoral dissertation, Artificial Intelligence Laboratory, University of Texas at Austin, Austin, TX.

FitzHugh, R. (1961). Impulses and physiological states in theoretical models of nerve membrane. *Biophysical Journal*, *1*, 445–466.

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

Garrett, S., Coghill, G. M., Srinivasan, A., & King, R. D. (2004). Learning qualitative models of physical and biological systems. In S. Džeroski and L. Todorovski (Eds.), *Computational discovery of communicable knowledge.* Berlin: Springer. To appear.

Gershenfeld, N. (1999). *The nature of mathematical modeling.* Cambridge, UK: Cambridge University Press.

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

Hodgkin, A. L., & Huxley, A. F. (1952). A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, *117*, 500–544.

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

Imielinski, T., & Mannila, H. (1996). A database perspective on knowledge discovery. *Communications of the ACM*, *39*, 58–64.

Jørgensen, S. E., Kamp-Nielsen, L., Chirstenen, T., Windolf-Nielsen, J., & Westergaard, B. (1986). Validation of a prognosis based upon a eutrophication model. *Ecological Modelling*, *32*, 165–182.

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

Kompare, B. (1995). *The use of artificial inteligence in ecological modeling.* Doctoral dissertation, Royal Danish School of Pharmacy, Copenhagen, Denmark.

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

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

Kuipers, B. (1994). *Qualitative reasoning: modeling and simulation with incomplete knowledge.* Cambridge, MA: MIT Press.

Langley, P. (1995). *Elements of machine learning.* San Mateo, CA: Morgan Kaufmann.

Langley, P. (2000). The computational support of scientific discovery. *International Journal of Human-Computer Studies*, *53*, 393–410.

Langley, P., Simon, H. A., Bradshaw, G. L., & Żythow, J. M. (1987). *Scientific discovery.* Cambridge, MA: MIT Press.

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

Ljung, L. (1993). Modelling of industrial systems. *Proceedings of Seventh International Symposium on Methodologies for Intelligent Systems* (pp. 338–349). Berlin: Springer.

Lotka, L. (1920). Contribution to the theory of periodic reactions. *Journal of American Chemical Society*, *42*, 1595–1599.

Mitchell, T. M. (1997). *Machine learning.* New York, NY: McGraw Hill.

Murray, J. D. (1993). *Mathematical biology.* Berlin: Springer. Second, Corrected Edition.

Nagumo, A. S., Arimoto, S., & Yoshizawa, S. (1962). An active pulse transmission line simulating nerve axon. *Proceedings of the Institute of Radio Engineers*, *50*, 2061–2071.

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

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

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

Potter, C. S., & Klooster, S. A. (1997). 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.

Press, W. H., Flannery, B. P., Teukolsky, S. A., & Vetterlin, W. T. (1986). *Numerical recipes*. Cambridge, MA: Cambridge University Press.

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

Richter, T. (1997). *A new measure of the distance between ordered trees and its applications* (Technical Report). Department of Computer Science IV, University of Bonn, Bonn, Germany.

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

Saito, K., Langley, P., Grenager, T., Potter, C., Torregrosa, A., & Klooster, S. A. (2001). The computational revision of quantitative scientific models. *Proceedings of the Fourth International Conference on Discovery Science* (pp. 336–349). Berlin: Springer.

Scheffer, C. (1993). Bivariate scientific function finding in a sampled, real-data testbed. *Machine learning*, *12*, 167–183.

Shasha, D., & Zhang, K. (1997). Approximate tree pattern matching. In *Pattern matching algorithms*, 341–371. Oxford University Press.

Stolle, R. (1998). *Integrated multimodal reasoning for modeling of physical systems*. Doctoral dissertation, Department of Computer Science, University of Colorado at Boulder, Boulder, CO.

Todorovski, L. (1993). *Modeliranje dinamičnih sitemov z avtomatskim odkrivanjem zakonitosti.* BSc Thesis, Faculty of computer and information science, University of Ljubljana, Ljubljana, Slovenia. In Slovene.

Todorovski, L. (1998). *Declarative bias in equation discovery.* MSc Thesis, Faculty of computer and information science, University of Ljubljana, Ljubljana, Slovenia. Available for download at `http://www-ai.ijs.si/~ljupco/ed/`.

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

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

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

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

Volterra, V. (1926). Variazionie fluttazioni del numero d'individui in specie animali conviventi (variations and fluctuations of a number of individuals in animal species living together. *Mem. Aced. Lincei*, *2*, 31–113. Translated to english in R. N. Chapman: *Animal Ecology*. New York: McGrawHill 1932, pp. 409–448.

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

Washio, T., & Motoda, H. (1998). Discovery of first principle equations based on scale-type-based and data-driven reasoning. *Knowledge Discovery in Databases: Techniques and Applications*, *10*, 403–411.

Whigham, P. A., & Recknagel, F. (2000). Predicting chlorophyll-a in freshwater lakes by hybridising process-based models and genetic algorithms. *Book of Abstracts of the Second International Conference on Applications of Machine Learning to Ecological Modeling.* Adelaide University.

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

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

Appendix A

Complete library of modeling knowledge for population dynamics

```
type Concentration is real
type Concentrations is set(Concentration)

type Population is Concentration
type Populations is set(Population)

type Inorganic is Concentration



function class Saturation(Concentration c)

function class No_saturation() is Saturation
   expression c

function class Saturation_type_1() is Saturation
   expression c / (c + const(saturation_rate,0,1,Inf))

function class Saturation_type_2() is Saturation
   expression c * c / (c * c + const(saturation_rate,0,1,Inf))

function class Saturation_type_3() is Saturation
   expression 1 - exp(-const(saturation_rate,0,1,Inf) * c)
```

```
process class Growth(Population p)

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

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


process class Decay(Population p)

process class Exponential_decay() is Decay
  expression const(decay_rate,0,1,Inf) * p


process class Flow(Concentration c)

process class Constant_inflow() is Flow
  expression const(inflow_rate,0,1,Inf)

process class Constant_outflow() is Flow
  expression -const(outflow_rate,0,1,Inf)


process class Feeds_on(Population p, Concentrations cs)
  condition p not in cs
  expression p * product({c}, c in cs, Saturation(c))


process class Interaction(Populations ps)
  condition cardinality(ps) at least 2

process class Competitive_exclusion() is Interaction
  expression -product({p}, p in ps, Saturation(p))

process class Symbiosis() is Interaction
  expression product({p}, p in ps, Saturation(p))
```

```
combining scheme Population_dynamics(Inorganic i)
  time_deriv(i) = + sum({}, true, Flow(i))
    - sum({p, food}, i in food, const(_,0,1,Inf) * Feeds_on(p, food))

combining scheme Population_dynamics(Population p)
  time_deriv(p) = + sum({}, true, Growth(p))
    + sum({}, true, Flow(p))
    + sum({food}, true, const(_,0,1,Inf) * Feeds_on(p, food))
    - sum({}, true, Decay(p))
    - sum({p1, food}, p in food, const(_,0,1,Inf) * Feeds_on(p1, food))
    + sum({ps}, p in ps, const(_,0,1,Inf) * Interaction(ps))
```

Dodatek B

Uporaba predznanja pri modeliranju dinamičnih sistemov z avtomatskim odkrivanjem enačb

Razširjeni povzetek

Strokovnjaki gradijo matematične modele zaradi analize in boljšega razumevanja obnašanja realnih sistemov (Gershenfeld, 1999). Gradnja matematičnega modela za opazovani sistem je zelo zahtevna naloga, ki zajema opazovanje in meritve obnašanja sistema pod razčnimi pogoji, izbor spremenljivk, ki so pomembne za modeliranje sistema, ter gradnjo matematične formulacije (oziroma enačb) modela. V disertaciji se ukvarjamo z nalogo avtomatskega modeliranja realnih sistemov ali natančneje z nalogo gradnje matematičnega modela iz opazovanega obnašanja merjenih spremenljivk sistema.

Raziskave, opravljene v okviru disertacije sodijo na področje strojnega učenja (Langley, 1995; Mitchell, 1997) ali bolj specifično na področje avtomatskega odkrivanja enačb (*equation discovery*) (Langley et al., 1987). Odkrivanje enačb vključuje razvoj metod za avtomatsko odkrivanje kvantitativnih zakonitosti, izraženih v obliki enačb, iz množice meritev. Metode za odkrivanje enačb uporabljamo predvsem za avtomatsko modeliranje realnih sistemov na osnovi meritev in opazovanj. Področje odkrivanja enačb je zelo povezano s področjem sistemske identifikacije (*system identification*) (Ljung, 1993). Metode za sistemsko identifikacijo slonijo na predpostavki znane strukture modela (ki jo ponavadi poda strokovnjak) in se osredotočijo na problem določanja konstantnih parametrov modela. Za razliko od teh se metode za odkrivanje enačb posvečajo bolj problemu določanja ustrezne strukture modela ter manj problemu določanja ustreznih vrednosti konstantnih parametrov.

Obstoječe metode za odkrivanje enačb lahko odkrivajo algebraične (Langley et al., 1987; Kokar, 1986; Falkenhainer & Michalski, 1990; Zembowicz & Żytkow, 1992; Washio & Motoda, 1997) ali navadne diferencialne enačbe (Todorovski, 1993; Džeroski & Todorovski, 1995; Križman, 1998; Todorovski, 1998; Todorovski & Džeroski, 1997). Algebraične enačbe uporabljamo za modeliranje statičnih sistemov, ki so že dosegli ravnovesno stanje, z navadnimi diferencialnimi enačbami pa lahko modeliramo obnašanje dinamičnih sistemov, tj. sistemov, ki spreminjajo svoje stanje s časom. Za modeliranje sprememb opazovanega sistema v več kot eni dimenziji (npr. prostorski in časovni) hkrati uporabljamo razširjeni formalizem parcialnih diferencialnih enačb.

Večina raziskav na področju odkrivanja enačb uporablja empirični pristop k modeliranju dinamičnih sistemov. Ta pristop sledi paradigmi "poskus in napaka"(*trial-and-error*), kjer preizkušamo različne modele in iščemo takega, ki se dovolj dobro prilega meritvam. Pri empiričnem modeliranju ponavadi ne uporabljamo predznanja s problemskega področja. Za razliko od empiričnega pristopa pri teoretičnem pristopu k modeliranju uporabljamo predznanje s področja uporabe. Strokovnjak s problemskega področja najprej ugotovi, kateri fizični procesi vplivajo na obnašanje sistema, potem pa na osnovi procesov in modelov, ki se običajno uporabljajo za njihovo modeliranje, določi stukturo enačb, iz katerih je sestavljen model. Na koncu uporabimo standarne metode sistemske identifikacije za določanje ustreznih vrednosti konstantnih parametrov modela iz meritev (Ljung, 1993).

V disertaciji se ukvarjamo s problemom združevanja teoretičnega in empiričnega pristopa k modeliranju dinamičnih sistemov z vključevanjem teoretičnega predznanja s področja uporabe v postopek odkrivanja enačb. Razvili smo novo metodo za odkrivanje enačb, ki lahko upošteva predznanje o modeliranju sistemov, kot ga poda strokovnjak s področja uporabe. Upoštevamo dva tipa predznanja. Prvi tip se nanaša na znanje o osnovnih procesih, ki vplivajo na obnašanje sistemov na obravnavanem področju uporabe. Drugi tip predznanja se nanaša na obstoječe modele, ki so že uveljavljeni na področju uporabe.

Nadalje smo razširili doseg potencialne uporabe metod za odkrivanje enačb na področje parcialnih diferencialnih enačb. Slednje lahko uporabimo za modeliranje tako časovnih kot prostorskih sprememb stanja opazovanega sistema.

V tem razširjenem povzetku bomo najprej predstavili izhodišča in podali pregled obstoječih metod za odkrivanje enačb. Nadaljevali bomo s pregledom vsebine disertacije in končali s povzetkom izvirnih prispevkov k znanosti.

B.1 Izhodišča in obstoječe metode

Strokovnjaki gradijo matematične modele zaradi analize in boljšega razumevanja obnašanja realnih sistemov (Gershenfeld, 1999). Matematični modeli ponujajo možnost združevanja potencialno zelo velikih množic opazovanj in meritev v celoto. Uporabljamo jih lahko za simulacijo in napovedovanje bodočega obnašanja opazovanega sistema pod različnimi pogoji. Nenazadnje je zelo pomembna lastnost matematičnih modelov tudi možnost, da razkrijejo procese in pojave, ki vplivajo na obnašanje opazovanega sistema.

Prvi in osnovni korak v postopku modeliranja realnega sistema je izbor matematičnega formalizma za modeliranje. Navadne diferencialne enačbe (NDE) so eden najbolj razširjenih formalizmov za modeliranje dinamičnih sistemov, tj. sistemov, ki spreminjajo stanje s časom. Nekaj obstoječih metod za odkrivanje enačb omogoča odkrivanje NDE (Todorovski, 1993; Džeroski & Todorovski, 1995; Križman, 1998; Todorovski, 1998; Todorovski & Džeroski, 1997). Te metode omogočajo uporabo odkrivanja enačb za avtomatsko modeliranje dinamičnih sistemov, vendar je modeliranje z NDE omejeno na sisteme, ki spreminjajo svoje stanje samo v eni (tipično časovni) dimenziji. Za modeliranje sprememb opazovanega sistema v več kot eni dimenziji hkrati (npr. prostorski in časovni) moramo uporabiti razširjeni formalizem parcialnih diferencialnih enačb.

Naslednji korak v postopku modeliranja je gradnja matematičnega modela opazovanega sistema. Obstajata dva vidika gradnje modela. Prvi vidik je določanje ustrezne strukture enačb, ki tvorijo model (problem določanja strukture). Drugi vidik je določanje vrednosti konstantnih parametrov modela (problem kalibracije modela). Za reševanje problema kalibracije lahko uporabimo eno od številnih metod, razvitih na področjih sistemske identifikacije (Ljung, 1993) in nelinearne optimizacije (Press et al., 1986). Tudi pri odkrivanju enačb uporabljamo te metode za določanje vrednosti konstantnih parametrov. Osrednja tema raziskav na področju odkrivanja enačb je reševanje problema določanja ustrezne strukture modela oziroma enačb, ki tvorijo model.

Pri teoretičnem pristopu k modeliranju problem določanja strukture modela rešuje strokovnjak s področja uporabe. Strokovnjak najprej ugotovi, kateri procesi in pojavi določajo obnašanje opazovanega sistema. Nato z uporabo predznanja o modeliranju sistemov s področja uporabe strokovnjak zgradi ustrezno strukturo modelskih enačb. Za razliko od teoretičnega pristopa, ki temelji na teoretičnem predznanju, je emprirični pristop popolnoma podatkovno voden in sledi paradigmi "poskus in napaka". Strokovnjak najprej predlaga strukturo modela, ki se mu zdi ustrezna, z metodami sistemske identifikacije določi ustrezne vrednosti parametrov modela in preveri, ali se simulacija dovolj dobro prilega meritvam. Če se ne, ponavlja postopek toliko časa, dokler ne najde ustreznega modela. V postopku se tipično uporablja zelo malo (če sploh kaj) predznanja s področja uporabe. Posledica tega je, da modeli, zgrajeni z empiričnim pristopom, ponavadi ne razkrivajo procesov in pojavov, ki določajo obnašanje opazovanega sistema. Zato jih tudi imenujemo modeli črnih škatel (*black-box* models), za razliko od modelov belih oziroma prozornih škatel (*white-box* models) pri katerih je razvidna zgradba opazovanega sistema s procesi in pojavi, ki določajo njegovo obnašanje.

Večina obstoječih metod za odkrivanje enačb uporablja empirični pristop k modeliranju, kjer uporabljamo zelo omejeno količino predznanja s področja uporabe. En tip predznanja, ki ga uporabljajo obstoječe metode za odkrivanje enačb, je znanje o merskih enotah spremenljivk opazovanega sistema. Metoda za odkrivanje enačb COPER (Kokar, 1986) uporablja merske enote za omejevanje prostora preiskanih struktur enačb. COPER obravnava samo tiste enačbe, ki pravilno kombinirajo spremenljivke in izraze z različnimi merskimi enotami. Omejitve se nanašajo na enostavna dejstva o združljivosti merskih enot, kot je recimo to, da ne moremo seštevati ali odštevati izrazov z različnimi merskimi enotami. Metoda za odkrivanje enačb SDS (Washio & Motoda, 1997) nadgrajuje COPER za primere, ko natančne merske enote spremenljivk niso znane. V tem primeru SDS uporablja znanje o tipu merskih lestvic, uporabljenih za merjenje spremenljivk opazovanega sistema, za omejevanje prostora možnih enačb.

Strokovnjaki s področja uporabe pa lahko posredujejo veliko več znanja kot samo naštevanje merskih enot opazovanih spremenljivk. Obstaja veliko učbenikov o matematičnem modeliranju, ki podajajo pregled predznanja o modeliranju sistemov z določenega področja, kot sta na primer biologija (Murray, 1993) ali biokemija (Voit, 2000). Da razpoložljivo znanje vključimo v postopek avtomatskega modeliranja z odkrivanjem enačb, ga moramo najprej formalno zapisati. Formalizirano znanje lahko potem uporabimo za omejevanje prostora enačb, ki ga preiskujejo metode za avtomatsko odkrivanje enačb. Metoda za odkrivanje enačb Lagramge (Todorovski, 1998) uporablja formalizem kontekstno neodvisnih gramatik za določanje prostora enačb. Gramatika je splošen formalizem, ki omogoča formalizacijo različnih tipov predznanja. Tako smo znanje o merskih enotah spremenljivk opazovanega sistema na primer uporabili za uspešno modeliranje dinamičnega sistema dvojnega invertiranega nihala (Todorovski, 1998). V primeru modeliranja rasti planktona v danskem jezeru Glumsø smo gramatiko zgradili na osnovi znanja o osnovnih procesih populacijske dinamike. Lagramge je s pomočjo gramatike odkril zelo natančen model iz zelo šumnih meritev. Vseeno pa je formalizem gramatik dokaj zapleten in pogosto tudi neprimeren za uporabo s strani strokovnjaka s področja modeliranja.

Na širšem področju umetne inteligence je bilo razvitih nekaj pristopov k avtomatskem modeliranju dinamičnih sistemov. Ti pristopi podpirajo vključevanje znanja o modeliranju sistemov s problemskega področja v postopek avtomatskega modeliranja. Pristop sestavljenega modeliranja (*compositional modeling*) uporablja znanje o tipičnih modelskih delcih (*model fragments*) s problemskega področja ter načinih za sestavljanje delcev v model celotnega sistema (Kuipers, 1994). Pristop sestavljenega modeliranja je v okviru naše raziskave samo posredno relevanten, ker je bil uporabljen za gradnjo kvalitativnih in ne numeričnih modelov. Drugi pristop k modeliranju dinamičnih sistemov, implementiran v okviru metode Pret, temelji na kvalitativnem sklepanju (*qualitative reasoning*) (Bradley et al., 2001). Za razliko od našega pristopa Pret gradi modele na osnovi znanja, ki temelji na takoimenovanih "*zakonih o ohranjanju količine*". Primer takega zakona je Kirchoffov zakon, ki določa, da je vsota tokov v opazovanem vozlišču elekričnega vezja enaka nič.

Drug tip znanja, ki ga obstoječe metode za avtomatsko odkrivanje enačb tipično spregledajo, so modeli, ki so že uveljavjeni na področju uporabe. Namesto da bi začeli iskanje z obstoječim modelom, metode za odkrivanje enačb vedno začenjajo iskanje "iz nič". Za razliko od njih metode za revizijo teorij (*theory revision*) (Ourston & Mooney, 1994; Wrobel, 1996) vedno začnejo z obstoječo teorijo in nato hevristično preiskujejo prostor revizij začetne teorije, da bi našli tako, ki se bolj prilega opazovanim oziroma izmerjenim podatkom. A raziskave na področju revizije teorij se tipično nanašajo na logične teorije in ne na numerične modele. Te metode torej niso uporabne za revizijo modelov, ki slonijo na enačbah.

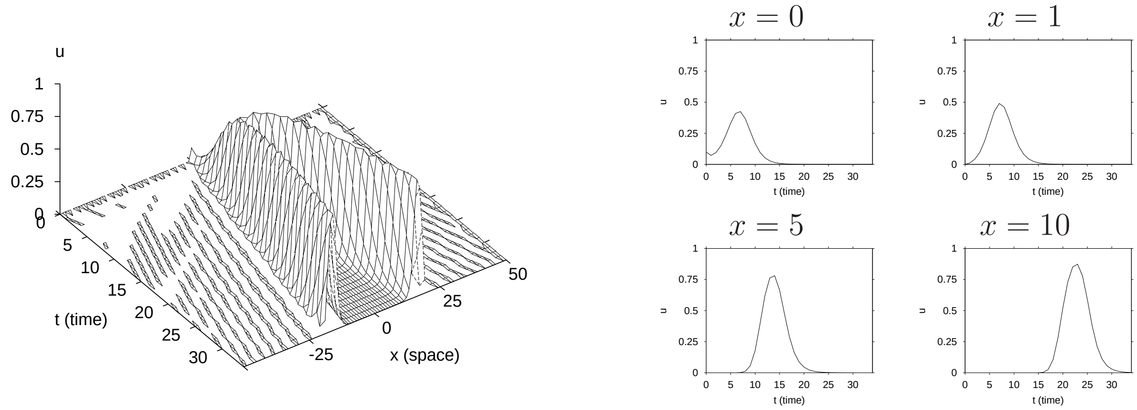
B.2 Pregled vsebine

Disertacija je sestavljena iz sedmih poglavij. Prvo poglavje podaja uvod v disertacijo s poudarkom na zastavljenih ciljih in poglavitnih prispevkih k znanosti. Drugo poglavje se začne z uvodom v področje modeliranja realnih sistemov in nadaljuje s pregledom metod za vključevanje predznanja v postopek avtomatskega učenja. Največji del drugega poglavja je posvečen uvodu v področje avtomatskega odkrivanja enačb in pregledu obstoječih metod. Na koncu poda poglavje še pregled kvalitativnih pristopov k avtomatskemu modeliranju s širšega področja umetne inteligence.

Sledi glavni vsebinski del disertacije, sestavljen iz štirih poglavij, ki predstavlja nove metode in pristope k odkrivanju enačb, razvite v okviru disertacije. Tretje poglavje podaja opis dveh metod za avtomatsko odkrivanje parcialnih diferencialnih enačb ter opis in rezultate njihove empirične evalvacije. Četrto poglavje podaja opis pristopa k avtomatskemu modeliranju realnih sistemov, ki sloni na vključevanju predznanja s področja uporabe v postopek odkrivanja enačb. Pristop k avtomatskemu modeliranju smo empirično ovrednotili na nalogah modeliranja dinamičnih sistemov iz umetnih in realnih podatkov v petem poglavju. Šesto in zadnje poglavje v tem delu predstavi metodo za revizijo obstoječih matematičnih modelov, že uveljavljenih na področju uporabe.

Sedmo poglavje podaja zaključke disertacije. Začne se s pregledom vseh predstavljenih metod, nato pa povzame rezultate njihovega empiričnega vrednotenja. Nadaljuje se s povzetkom izvirnih prispevkov disertacije, konča pa s smernicami za nadaljnje delo.

V nadaljevanju tega razdelka bomo podali bolj podroben pregled vsebine tretjega, četrtega in šestega poglavja disertacije.



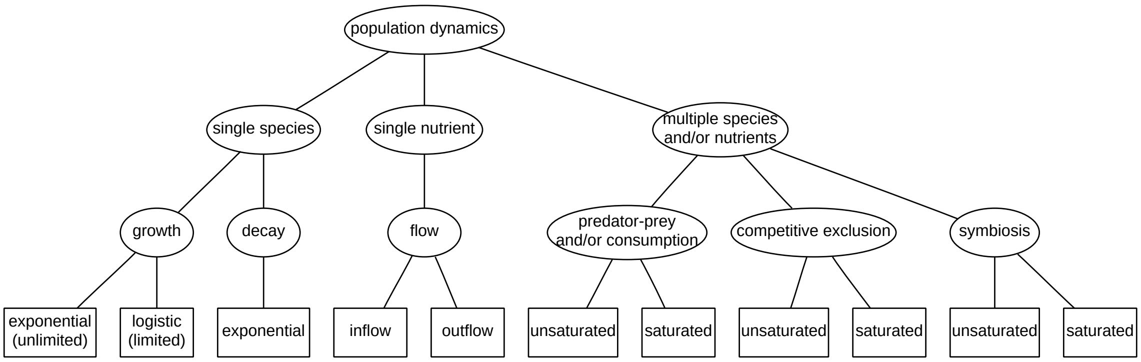
Slika B.1: Podatki, uporabljeni za rekonstrukcijo modela plenilec-plen, predstavljenega v razdelku 3.3 (leva stran slike), ter "rezine"podatkov za štiri različne vrednosti prostorske dimenzije x (desna stran slike).

B.2.1 Odkrivanje parcialnih diferencialnih enačb

Za odkrivanje parcialnih diferencialnih enačb (PDE) uporabimo načelo pretvorbe, ki je bilo že uporabljeno za odkrivanje navadnih diferencialnih enačb (NDE) v metodi LAGRANGE (Todorovski, 1993; Džeroski & Todorovski, 1995).

Preizkusili smo dva različna pristopa. Pri prvem začetno množico opazovanih spremenljivk sistema razširimo z njihovimi parcialnimi odvodi glede na podane časovne in prostorske dimenzije. Pri tem uporabimo numerično metodo za računanje parcialnih odvodov, ki sloni na polinomski interpolaciji (Press et al., 1986). Tako izhodiščni problem odkrivanja PDE pretvorimo v problem odkrivanja algebraičnih enačb, kjer lahko uporabimo katerokoli obstoječo metodo za odkrivanje enačb. Predlagana metoda je zelo enostavna, vendar je novi problem veliko zahtevnejši od začetnega. Razlog za povečano zahtevnost novega problema je število na novo vpeljanih spremenjivk, kar bistveno vpliva na velikost prostora obravnavanih enačb.

Da bi omejili prostor obravnavanih enačb, preizkusimo še drugi pristop, pri katerem problem odkrivanja PDE razstavimo na več problemov odkrivanja NDE. To naredimo tako, da podatke razdelimo na "rezinež nespremenljivo vrednostjo vseh dimenzij razen časovne, kot ponazarja Slika B.1. V vsaki rezini potem odkrivamo NDE. Strukture, ki smo jih



Slika B.2: Taksonomija procesnih razredov za modeliranje sistemov s področja populacijske dinamike.

najpogosteje odkrili v različnih rezinah, uporabimo za omejitev prostora obravnavanih PDE. Problem odkrivanja PDE v omejenem prostoru enačb lahko rešujemo z enostavnim pristopom, opisanim zgoraj.

Uporabnost razvitih metod smo ponazorili s poskusi rekonstrukcije znanih modelov realnih sistemov iz simuliranih podatkov. Prvi model je s področja populacijske dinamike in se nanaša na razmerje dveh populacij, ki živita v istem okolju. Drugi model je Fitzhugh-Nagumo model prenosa električnega signala v živčnih celicah (FitzHugh, 1961; Nagumo et al., 1962). V obeh poskusih je metoda z omejevanjem prostora enačb uspešno rekonstruirala originalne modele iz simuliranih podatkov.

B.2.2 Vključevanje predznanja s področja uporabe v postopek odkrivanja enačb

Predlagani formalizem za zapis znanja o modeliranju sistemov s področja uporabe omogoča organizacijo znanja v obliki taksonomije procesnih razredov. Primer take taksonomije, zgrajen na osnovi predznanja s področja populacijske dinamike (Murray, 1993), je podan na sliki B.2.

Vsako vozlišče predstavlja razred sorodnih osnovnih procesov, ki določajo oziroma bistveno vplivajo na obnašanje sistemov s področja populacijske dinamike. Za vsak procesni razred opredelimo tip spremenljivk, ki vplivajo na ali so pod vplivom procesov iz razreda,

Tabela B.1: Formalni opis razreda procesov populacijske rasti.

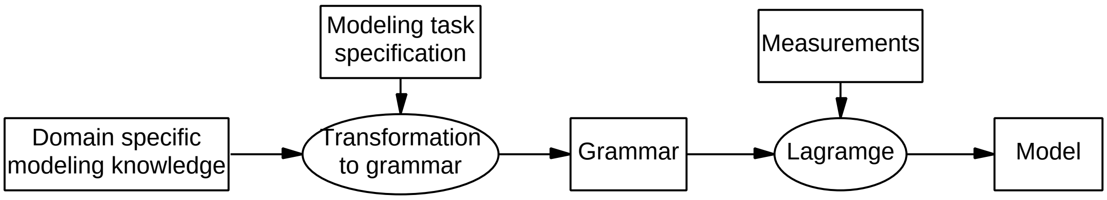
```
process class Growth(Population p)

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

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

ter seznam enačb, ki jih strokovnjaki uporabljajo za modeliranje procesov iz tega razreda. Primer opisa razreda procesov populacijske rasti je podan v Tabeli B.1.

Prvi, najbolj splošni procesni razred `Growth` določa dejstvo, da so procesi populacijske rasti odvisni od in vplivajo na eno samo populacijo, tj. spremenljivko `p` tipa `Population`. Naslednja dva procesna razreda sta podrazreda procesnega razdreda `Growth` ter od njega nasledita spremenjivko `p`. Prvi podrazred `Exponential_growth` določa model neomejene (eksponentne) rasti populacije `p`. Drugi podrazred `Logistic_growth` določa model rasti, ki je omejena s kapaciteto okolja, v katerem populacija živi. Kapaciteto okolja določa konstantni parameter `capac`, ki ga vpeljemo v model z uporabo simbola `const(capac, 0, 1, Inf)`. Ta simbol določa spodnjo (`0`) in zgornjo vrednost (neskončno - `Inf`) konstantnega parametra `capac`, ter njegovo privzeto vrednost (1).

Poleg znanja o osnovnih procesih s področja uporabe in njihovih modelih, formalizem omogoča tudi zapis znanja o tem, kako modele posameznih osnovnih procesov iz različnih razredov sestavimo v modele celotnega sistema. Formalizem za predstavitev znanja je splošen in omogoča predstavitev znanja z različnih področij uporabe. Splošnost formalizma smo pokazali s formalizacijo znanja s področij populacijske dinamike (Murray, 1993), biokemične kinetike (Voit, 2000), klasične mehanike (Bradley et al., 2001) ter problemsko neodvisnega znanja o modeliranju sistemov na osnovi podanih merskih enot opazovanih spremenljivk (Washio & Motoda, 1997).

Pristop k avtomatskemu modeliranju, ki omogoča združevanje tako predstavljenega predznanja v postopek odkrivanja enačb, je predstavljen na sliki B.3. V procesu gradnje



Slika B.3: Pristop k avtomatskemu modeliranju realnih sistemov, ki temelji na združevanju predznanja s področja uporabe v postopek avtomatskega odkrivanja enačb.

modela opazovanega sistema, informacijo o tipih spremenljivk primerjamo z zapisanim predznanjem in ugotovimo, kateri osnovni procesi se lahko pojavijo v modelu. Na osnovi tega in predznanja o modelih posameznih procesov zgradimo gramatiko, ki določa prostor možnih modelov za opazovani sistem. Neterminalni simboli gramatike označujejo razrede osnovnih procesov. Generativna pravila za vsak neterminalni simbol pa določajo prostor možnih modelov za ustrezni razred procesov. Začetni simbol gramatike povezuje modele posameznih procesov v možne modele celotnega sistema. Tako zgrajena gramatika nam omogoča uporabo metode za odkrivanje enačb Lagramge, ki preišče prostor možnih modelov, kot ga določa gramatika, in najde tistega, ki se najbolj prilega meritvam opazovanih spremenljivk. Drevo izpeljave modela, ki ga določa gramatika, lahko uporabimo zato, da ugotovimo, kateri osnovni procesi določajo obnašanje opazovanega sistema.

Zgoraj opisani pristop k modeliranju dinamičnih sistemov smo empirično preizkusili na problemih rekonstrukcije več znanih modelov populacijske dinamike iz simuliranih podatkov z različnimi stopnjami dodanega šuma. Pristop je uspešno rekonstruiral originalne modele. Poskusi so tudi pokazali povečano odpornost na prisotnost šuma v podatkih, ki je posledica vključevanja predznanja v postopek odkrivanja enačb. Uporabnost sistema smo ponazorili tudi z reševanjem problema izpopolnjevanja delno določenega modela dinamike vodne gladine v fjordu Ringkøbing iz realnih meritev opazovanih spremenljivk.

B.2.3 Revizija modelov, ki slonijo na enačbah

Za revizijo obstoječih modelov zopet razvijemo metodo, ki temelji na načelu pretvorbe. Najprej podani obstoječi model pretvorimo v začetno gramatiko, ki lahko izpelje samo

Tabela B.2: Začetni CASA-NPPc model za napovedovanje rastlinske produkcije ogljika na osnovi satelitskih opazovanj in atmosferskih meritev.

$$
\begin{aligned}
&NPPc = \max(0, E \cdot IPAR) \\
&\quad E = 0.389 \cdot T1 \cdot T2 \cdot W \\
&\qquad T1 = 0.8 + 0.02 \cdot topt - 0.0005 \cdot topt^2 \\
&\qquad T2 = 1.1814/((1 + \exp(0.2 \cdot (TDIFF - 10))) \cdot (1 + \exp(0.3 \cdot (-TDIFF - 10)))) \\
&\qquad\quad TDIFF = topt - tempc \\
&\qquad W = 0.5 + 0.5 \cdot eet/PET \\
&\qquad\quad PET = 1.6 \cdot (10 \cdot \max(tempc, 0)/ahi)^A \cdot pet\_tw\_m \\
&\qquad\qquad A = 0.000000675 \cdot ahi^3 - 0.0000771 \cdot ahi^2 + 0.01792 \cdot ahi + 0.49239 \\
&\quad IPAR = FPAR\_FAS \cdot monthly\_solar \cdot SOL\_CONV \cdot 0.5 \\
&\qquad FPAR\_FAS = \min((SR\_FAS - 1.08)/srdiff, 0.95) \\
&\qquad\quad SR\_FAS = (1 + fas\_ndvi/1000)/(1 - fas\_ndvi/1000) \\
&\qquad SOL\_CONV = 0.0864 \cdot days\_per\_month
\end{aligned}
$$

ta začetni model. Gramatiko zgradimo tako, da upošteva strukturo podanega začetnega modela. Tako zgrajeni začetni gramatiki lahko dodamo generativna pravila, ki določajo možne alternativne modele. Te možnosti lahko določi strokovnjak, ali jih določimo na osnovi predznanja s področja uporabe, zapisanega v zgoraj opisanem formalizmu. Razširjena gramatika nam tako določa prostor revizij začetnega modela. Opis tega prostora z gramatiko nam omogoča uporabo metode LAGRAMGE za iskanje tiste revizije začetnega modela, ki se najbolj prilega meritvam.

Metode za revizijo teorij sledijo načelu minimalne spremembe. Ta določa, da med teorijami, ki se (približno) enako prilegajo podanim podatkom, izberemo tisto, ki je najbolj podobna začetni teoriji. Da bi vključili to načelo v metodo LAGRAMGE, smo razširili njeno MDL hevristično funkcijo, ki daje prednost krajšim enačbam pred daljšimi. Funkcija MDL kombinira dolžino enačbe s stopnjo prileganja meritvam. Če zamenjamo dolžino enačbe z razdaljo od začetnega modela, dobimo hevristično funkcijo za ocenjevanje enačb, ki sledi načelu najmanjše spremembe. Za merjenje razdalje med modeli, ki slonijo na enačbah, uporabljamo standardne mere razdalje med strukturiranimi izrazi (Shasha & Zhang, 1997; Richter, 1997).

Uporabnost metode za revizijo modelov, ki slonijo na enačbah, smo ponazorili z revizijo CASA-NPPc modela, ki so ga razvili strokovnjaki s področja okoljskih znanosti (Potter &

Klooster, 1997). Model CASA-NPPc uporabljajo za napovedovanje rastlinske produkcije ogljika iz satelitskih opazovanj in atmosferskih meritev. Gre za relativno zapleten model, ki je sestavljen iz več enačb (glej tabelo B.2). Razvita metoda za revizijo je odkrila model, ki je bolj natančen od začetnega. Še več, z uporabo načela minimalne spremembe smo ugotovili, katere revizije najbolj prispevajo k izboljšani točnosti začetnega modela.

B.3 Izvirni prispevki disertacije

Izvirni prispevki disertacije sodijo na področje avtomatskega odkrivanja enačb. Prispevki izpopolnijo obstoječe metode za odkrivanje enačb v smislu njihove uporabe za modeliranje realnih sistemov. Razširili smo doseg uporabe metod za odkrivanje enačb na področje parcialnih diferencialnih enačb. Razvili smo formalizem za predstavitev znanja o modeliranju sistemov s področja uporabe. Razvili smo metodo za odkrivanje enačb, ki lahko vključi znanje, predstavljeno v zgoraj omenjenem formalizmu, v postopek odkrivanja enačb. Razvili smo tudi metodo za odkrivanje enačb, ki je zmožna revizije modelov, slonečih na enačbah. Omenjeni prispevki so bolj podrobno opisani v naslednjih treh odstavkih.

Odkrivanje parcialnih diferencialnih enačb

Razvili smo metodo za odkrivanje parcialnih diferencialnih enačb (PDE). Metoda temelji na pretvorbi problema odkrivanja PDE na enostavnejši problem odkrivanja algebraičnih ali navadnih diferencialnih enačb. V obeh primerih pretvorba omogoča uporabo obstoječih metod za odkrivanje enačb za odkrivanje PDE. Razvoj take metode razširi doseg metod za odkrivanje enačb na področje avtomatskega modeliranja sistemov, ki spreminjajo svoje stanje v več kot eni dimenziji (npr. časovni in prostorski).

Vključevanje predznanja s področja uporabe v postopek odkrivanja enačb

Razvili smo formalizem za predstavitev znanja o modeliranju sistemov s področja uporabe. Predznanje je organizirano v obliki taksonomije osnovnih procesov, ki vplivajo na obnašanje sistemov na obravnavanem področju uporabe. Za vsak osnovni proces v taksonomiji so podane možne enačbe, ki jih strokovnjaki s področja uporabljajo za modeliranje procesa. Formalizem tudi omogoča predstavitev znanja o tem, kako modele posameznih osnovnih procesov kombiniramo v enotni model celotnega sistema. Razvili smo novo metodo za odkrivanje enačb, ki je sposobna znanje, predstavljeno v zgoraj opisanem formalizmu, vključiti v postopek odkrivanja enačb. Razvoj formalizma in metode omogoča združevanje teoretičnega in empiričnega pristopa k modeliranju realnih sistemov.

Revizija modelov, ki slonijo na enačbah

Razvili smo metodo za odkrivanje enačb, ki lahko začne postopek odkrivanja s podanim obstoječim modelom. Metoda preišče prostor možnih revizij podanega modela in najde tisto, ki se prilega opazovanim oz. izmerjenim podatkom bolje kot začetni model. Metoda omogoča vkjučevanje obstoječih modelov, že uveljavljenih na področju uporabe, v postopek odkrivanja enačb. Načelo minimalnih sprememb (le-ta daje prednost revidiranim modelom, ki so kar se da podobni začetnemu), ki ga uporabljajo metode za revizijo logičnih teorij, smo prilagodili za uporabo na problemu revizije modelov, ki slonijo na enačbah.

Izjava o avtorstvu

Spodaj podpisani Ljupčo Todorovski izjavljam, da sem avtor doktorske disertacije z naslovom *Uporaba predznanja pri modeliranju dinamičnih sistemov z avtomatskim odkrivanjem enačb*, oziroma angleškim naslovom *Using domain knowledge for automated modeling of dynamic systems with equation discovery*. Doktorsko disertacijo sem samostojno izdelal pod mentorstvom prof. dr. Ivana Bratka ter somentorstvom doc. dr. Saša Džeroskega.

V Ljubljani, 7. maja 2003

mag. Ljupčo Todorovski, univ. dipl. inž. rač. in inf.