A FRAMEWORK FOR INTERACTIVE MODELING AND DISCOVERY OF SCIENTIFIC KNOWLEDGE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
MASTER OF SCIENCE

Javier Nicolás Sánchez
May 2002

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science

Hector Garcia-Molina
(Principal Co-Advisor)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Master of Science

Pat Langley
(Principal Co-Advisor)

Preface

Existing methods for computational scientific discovery generate knowledge in established scientific formalisms, but their encoding of domain knowledge often uses different notations. Here we present a process modeling language for stating models and knowledge in terms familiar to scientists. We also describe an interactive environment for knowledge discovery that lets the user construct, edit, and visualize scientific models, use them to make predictions, and revise them to better fit available data. We report experiments in two different domains that show the effectiveness of this approach on the task of model revision and the representational power of the underlying formalism. Finally, we discuss related work on modeling formalisms and model revision, as well as suggesting several topics for additional research.

Acknowledgments

I would like to thank my advisor, Professor Pat Langley, for accepting me in his research group and giving me the possibility of exploring an exciting area of Computer Science. His guidance and and support where fundamental for this work.

I am also grateful to Professor Hector Garcia-Molina, who made it possible to complete this research and obtain the distinction in research.

Thanks also to Ljupco Todorovski for his help with the integration of LaGramge into my system, as well as Stephen Bay and Dan Shapiro for their comments and feedback about it. Finally, I would like to thank Chris Potter, who provided ecosystem models and data for some of the experiments described in this work.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Scientific discovery is often regarded as one of the most prominent facets of creative thought. Thus, it is not generally viewed as an area where computational approaches can be successful. However, despite this common opinion, computational approaches to scientific discovery have progressed steadily over the past 30 years, going from the early stages, where they rediscovered knowledge from the history of science, to the present, where they are helping to find new knowledge in various scientific disciplines.

However, considerable work still needs to be done. One aspect that has recently attracted the attention of researchers concerns the incorporation of domain knowledge into discovery techniques. Domain knowledge can prove useful in several ways, such as supplying constraints to help the discovery algorithms reduce or guide the search and improving the communicability of the resulting discoveries.

Therefore, recently researchers have extended computational discovery techniques to take advantage of domain knowledge in various ways, ranging from starting from existing models to specifying the general form of the equations for a domain. However, these approaches are not as transparent to domain experts as needed to let them encode their knowledge effectively. Furthermore, little has been done to support closer interaction between scientist and computer during the discovery process. However, such interaction can draw in the strength of scientist and discovery algorithm to produce better results by enabling a synergic relationship.

In response, this thesis proposes a new formalism for expressing background knowledge and discovered knowledge that makes close contact with scientists quantitative models, thus providing a coherent and communicable representation throughout the whole discovery process. A second contribution is an integrated framework that supports interaction between domain scientists and discovery algorithms by providing tools that let the scientist formulate, use, analyze, and revise their models, thus fostering a smooth interaction between the two parties.

1.1 Previous work

Research in computational scientific discovery has, throughout its history, addressed several activities, ranging from the induction of taxonomies and laws to the formation of models that describe phenomena in terms of hypothesized mechanisms. Early work in the field focused on descriptive knowledge such as taxonomies, qualitative laws, and numeric equations. This research produced systems like CLUSTER/2 (Michalski & Stepp [17]) and AUTOCLASS (Cheeseman et al. [3]), which deal with the task of taxonomy formation, NGLAUBER (Jones [10]), which searches for qualitative laws, and BACON (Langley [12], Langley et al. [15]), which finds numeric laws. Later, work led to systems like DALTON (Langley et al. [15] and GELL-MANN (Żytkow [25]), which create structural models, and systems like MECHEM(Valdés-Pérez [24]) and ASTRA (Kokabas and Langley [11]), which construct process models that explain phenomena in terms of hypothesized mechanisms that involve change over time.

As the field has evolved, its focus has shifted from replicating discoveries from the history of disciplines like mathematics, physics, chemistry, and biology towards the discovery of new scientific knowledge. As Langley [13] has noted, many of these efforts have led to refereed publications in the relevant scientific literature demonstrating the potential of computer-aided research. Experience with these systems has also revealed the importance of domain knowledge in constraining search and in producing knowledge that is communicable to scientists.

Current discovery systems incorporate domain knowledge in several ways, ranging from the use of existing models to the representation of results using formalisms and abstractions that are familiar to the domain scientists. For example, Chown and Dietterich [4] report a technique that improves the accuracy of an ecosystem model by revising its parameters. Thus, they incorporate knowledge about the domain in the form of an existing model and cast the discovery task as one of finding parameter values that reduce the model's error on available data. Their approach to global optimization involves solving smaller parameter optimization problems. First they recast the model in a loop-free, single assignment programming language. Then, they apply automatic analysis tools to detect control paths in the model that involve a small number of relevant parameters, along with data relevant to those subproblems. Finally, they apply simulated annealing to revise the parameters of all the subproblems found using their corresponding data sets. The result produced is a set of new parameters for a model that is already familiar to domain scientists.

Saito et al. [19] presents a related approach that revises an existing quantitative model cast as a set of numerical equations. Their technique revises individual equations by recasting them as a neural network with an associated error function, which is then trained applying a second order gradient descent algorithm (Saito and Nakano [20]) to find parameters that better fit the data. The result is then cast as a revised equation embedded in the original model, thus providing knowledge in a form that is familiar to domain scientists.

Todorovski and Džeroski [22] propose an alternative approach to revising existing quantitative models that uses context-free grammars. They recast the model into a grammar which is the input accepted by their system LaGramge, which also incorporates domain knowledge by letting the user specify alternative productions; these provide a declarative bias that the system uses when searching the space of possible revisions. The resulting model is expressed as a differential or instantaneous equation that takes a form familiar to scientists. More recently Todorovski and Džeroski [23] have proposed a closely related formalism for encoding domain knowledge that is based on first-order logic. This new approach encodes background knowledge in a manner that is less arcane for domain scientists but that can be easily translated into the grammars used by LaGramge. This new system also supports context-dependent grammars, which provide more expressive power to encode domain specific constraints.

A fourth approach appears in Bannai et al. [1], who propose an extension to the functional language ML, which they call VML. Their approach to computational discovery provides tools that aim to facilitate the trial-an-error cycle for knowledge discovery. Models in their formalism are constructed by composing views, which they define as a generalized form of attributes. These views are encoded as named functions and can be manipulated, composed, and redesigned through an iterative process. This interactive framework lets the scientist play a central role by iteratively constructing and refining the model.

Easley and Bradley [6] have proposed an approach that casts the discovery task in terms of gray-box system identification for nonlinear dynamic systems. Their approach encodes background knowledge as "generalized physical networks", which take the form of generalized equations that describe idealized two-terminal components. Furthermore, they also incorporate qualitative knowledge that can be used to constrain the search space. Using those two types of inputs, their system carries out a search through the space of systems of ordinary differential equations consistent with them. The resulting model is cast as a system of ordinary differential equations, which are easily understood by scientists.

Another approach is presented by Valdés-Pérez [24], whose MECHEM system formulates reaction pathways in catalytic chemistry. The system accepts as input the reactants and products for a chemical reaction, along with experimental phenomena to be explored, background knowledge about the domain of catalytic chemistry, and user specified constraints that are used in the generation of the pathways. The system then invokes a constrained exhaustive search and returns the shortest reaction pathways expressing them in a notation familiar to chemists.

Finally, another approach uses logic to express domain knowledge about a model and uses inductive logic programming to make revisions. For example, Mahidadia and Compton [16] present an integrated environment for the development and revision of qualitative causal models. Their system provides a graphical interface for model presentation and construction that makes close contact with qualitative models in neuroendocrinology thus allowing easy encoding of existing domain knowledge. The program transforms this representation into Horn-clause logic, revises the model using a logic-based learner, and translates the results back into the graphical notation. Thus, it presents the revised model in the same formalism as the initial one, making it accessible to the domain scientists.

1.2 Limitations of Previous Work

Having introduced the most recent work in the area, we will now proceed to review the limitations that exist in that work in order to motivate the goals of this thesis.

Chown and Dietterich [4] have shown successful results in automatically revising a model with many parameters. However, they must express the model in a single-assignment and loop-free programming language that makes little contact with the formalisms used by scientists. Furthermore, their approach only supports revision of model parameters and not more profound revisions such as altering the form of equations. Finally, their automated methods provide no way for a domain expert to influence the revision process, which we hold to be important.

We find a different approach in Saito et al. [19], whose method takes as input a quantitative model expressed as numerical equations and thus makes close contact with domain scientists formalisms. Furthermore, their technique revises not only parameters but also the structure of individual equations, thus supporting a broader range of revisions. The discovery system's output is communicable, since it is also cast as a set of equations. However, their approach requires the user to specify, for each revision, a different error function to guide the neural algorithm's search. Although their approach makes contact with scientific formalisms by encoding the initial and revised models in established notations, their mechanism for directing the revision process requires significant effort, as well as knowledge unavailable to most domain scientists.

Todorovski et al.'s [22] LaGramge incorporates context-free grammars as a powerful mechanism to introduce declarative bias in the search space, but grammars are not easily understood by most scientists. To overcome that problem, they have recently proposed a new formalism for encoding background knowledge (Todorovski

& Džeroski [23]) based on first-order logic. However, even though first-oder logic is probably more understandable to many scientists, it remains some distance from the way in which scientists usually conceptualize their models. Another limitation of their approach is the lack of higher-level concepts that subsume equations which are often found in scientific modeling in order to manage the complexity of large systems.

A different approach is taken by Bannai et al. [1], who have extended a functional language to incorporate the notion of views. They provide tools to facilitate the iterative process of refinement and composition of those views until a satisfactory result is obtained. Although this work provides a flexible framework for iterative model development, it relies on notations that are quite different from those in scientists' models. Therefore, while views provide a good tool for constructing complex models, they do not represent either background knowledge or the discovered models in a formalism that is familiar to scientists.

Easley and Bradley's [6] approach combines qualitative and quantitative knowledge in order to generate models of nonlinear dynamic systems. Their approach shows powerful techniques for constraining search and using domain knowledge to guide the discovery process. However, it does not support interaction with a user and thus does not take advantage of potential synergies between human and computer. Furthermore, while generalized physical networks describe abstract relations that hold in several engineering domains, they are not readily extensible to other domains like biology or physics, so the scope of their technique is limited. Finally, as noted with other approaches, it does not support concepts like processes and intermediate theoretical terms that are often found in scientific models.

Valdés-Pérez's MECHEM [24] is a discovery system designed for the formulation of reaction pathways. He's approach has a notion of processes to express reactions in the pathway and also incorporates spatial constraints when searching for possible models. However, since it was designed for the task of pathway formulation its formalism does not readily provide support to other domains that could benefit from the techniques introduced there. Furthermore, while the system supports interaction with the user by allowing the introduction of constraints to be applied during the search, Valdés-Pérez notes that its use is not straightforward to chemists, an obstacle that must be overcome before it can gain use by scientists.

Finally, we also mentioned research by Mahidadia et al. [16], that successfully addresses the communicability of models by representing background knowledge in ways that are consistent with the formalisms used by scientists. Furthermore, their environment also encourages interaction between the discovery algorithm and the scientist, thus creating a synergic relation. However, a drawback of their approach is that it only supports qualitative causal models, thus making it unsuitable for performing discovery in domains where scientists require quantitative predictions and explanations.

1.3 Statement of the Problem

One goal of this work is to propose a representation that is closely related with scientists' models and that can be used in all stages of the discovery process, from the declaration of background knowledge to model refinement to the representation of discovered knowledge. Furthermore, we aim to provide an integrated framework that supports distinct stages in the discovery of new scientific knowledge by providing tools for construction, visualization, and edition of process models, for utilizing them in predictive simulation, and for constrained revision in response to observations. Consequently, another goal is to construct an environment that explicitly exploits the interaction between scientist and discovery algorithm to improve the discovery process. We will focus our research efforts in the area of quantitative modeling, an area that would benefit from advances in these fronts.

Our survey of the literature should have convinced the reader that there is currently no approach that fulfills the objectives mentioned above. Many researchers have developed systems that discover knowledge in some communicable form, but none have applied this idea consistently to all stages of the discovery process. Furthermore, most previous work has emphasized automation, rather than fostering collaboration between scientist and discovery algorithm. Bannai et al. [1] propose a system that encourages such interaction, but their formalism for encoding knowledge is not communicable to scientists, thus hindering the benefit of the interactive mechanism. Mahidadia et al. [16] provide mechanisms for user-system interaction, but their work deals with qualitative models, whereas we are concerned with quantitative modeling.

To leverage the power of the computational discovery techniques the field needs an integrated approach that provides both a communicable language for the representation of models and tools to build, test, analyze, revise, and discover models. Currently, scientists who want to use discovery algorithms must often learn a completely new formalism to represent their models. Moreover, usually those systems do not provide mechanisms to simulate and analyze models to help the scientist generate hypotheses to be used in their revision. Furthermore, there are few mechanisms that support model revisions in an iterative cycle that lets scientists test these hypotheses and build on the results. Therefore, there is a clear need for an integrated framework that can streamline the task of scientific discovery and help scientists incorporate computational discovery tools in their modeling efforts.

1.4 Thesis Outline

In the remainder of the thesis, we present our progress in this research problem. Chapter 2 introduces a modeling language that is communicable to scientists and well suited for model discovery. Chapter 3 presents an integrated environment that supports construction, visualization, and edition of process models, their use for predictive simulation, and their constrained revision. Chapter 4 discusses the revision of two models from different domains and the results produced by these efforts. In closing, Chapter 5 draws some conclusions and proposes topics for further work.

Chapter 2

The Process Modeling Language

2.1 Motivation

As we have noted, existing computational approaches to scientific discovery fail to provide a formalism for representing background knowledge that is familiar to scientists. Furthermore, while several of the most recent computational discovery systems generate knowledge in terms of numeric equations or other formalisms familiar to scientists, those notations often leave implicit an aspect that plays an important role in many scientific fields – the notion of a *process*. Moreover, no current computational approach to scientific discovery provides a unified formalism that maintains a consistent representation of knowledge during all the stages of the modeling and discovery process.

To satisfy these objectives, we designed a modeling language that supports quantitative equations, as do many other approaches, but that also organizes them into explicit processes that relate variables. Each process expresses a causal relation between its input and output variables through its component equations, which can be differential (for a process that involves change over time) or static (for one that involves instantaneous effects). Furthermore, a process may include conditions, stated as threshold tests on its input variables, describing the preconditions that must be satisfied for the process to be active.

Consequently, a model in our formalism consists of a set of processes that link observable input variables with observable output variables, possibly through unobservable theoretical terms. Many examples in the scientific literature testify that scientists often think in this manner. For instance, processes like fluid flow, boiling, and motion often occur in physical models. Similarly, processes such as growth, decay, and consumption play an important role in biology. Further evidence that modeling in terms of processes is natural in science comes from Cartier et al. [2]. In their study of the nature and structure of scientific models, they note that models incorporate empirical or theoretical terms and the processes in which they participate. Furthermore, they emphasize that models can be used to explain and predict phenomena, something that our formalism also supplies.

Another field where process models have arisen is qualitative physics. In particular, our modeling formalism borrows many ideas from Qualitative Process Theory (Forbus [7]), which also includes processes that relate variables and that can include preconditions. However, Qualitative Process Theory casts these relations in qualitative form, while ours take the form of numerical equations.

The previous exposition should convince the reader that our representation makes close contact with common conceptualizations of scientific models. As we will see the formalism not only encodes scientific knowledge but also serves as the declarative representation for our discovery methods. Discoveries made by the system are cast in the same process modeling language, making its results highly communicable to scientists and providing a single formalism that supports all the stages of the modeling and discovery process.

2.2 The Structure of Process Models

A model in the language consists of variables and processes that relate those variables. Among the variables that appear in a model, we distinguish a special type of *observable* variable that must be specially declared and that serve as input for making predictions. Furthermore, all variables have an associated type that is declared in the model, although a default numeric type is assumed if no type is provided. Finally, the model can contain one or more processes that include one or more static or differential equations, along with an optional set of preconditions stated as threshold tests that determine when the process and its associated equations are active.

2.2.1 Variables and Variable Types

Variables in our modeling language are quantitative in that they can only take numeric values. Additionally, each variable is associated with a type that describes the category of magnitude the variable represents, such as temperature, mass, or population. Types are organized into a taxonomy stored as a tree, with the root always having the type *number*. The user can customize this type hierarchy for the domain at hand; the language does not impose a fixed taxonomy on the user. Types are defined in a special file that contains a succession of statements with the form:

<*type_name*> `subtypeof` <*type_name*>`;`

We introduced the type system into the language to provide constraints on the revision algorithm. Chapter 4 illustrates their role on the selection of processes during revision.

Besides being associated with a type, a variable in the system can also belong to a special group of variables called observable variables, a group that denotes the input to the model. Therefore, declaring a variable as observable indicates that if the model is going to be simulated then data will be provided to initialize that variable. The following regular expressions describe how variables are declared in our process models:

`variables` <*variable_name*> [`{`<*type_name*>`}`]
[`,`<*variable_name*> [`{`<*type_name*>`}`]]* `;`
`observable` <*variable_name*> [`,`<*variable_name*>]* `;`

Notice that every variable that appears in the model must be declared in the variables clause, even if they are observable, since this tells the simulator the variables that occur in the model and their type.

2.2.2 Processes

As previously noted, a process groups equations that relate its input and output variables. Each equation specifies a dependent variable in terms of a mathematical formula involving zero or more influencing variables. This makes explicit the causal dependencies between processes by simple inspection of the equations, thus making the model structure more understandable. Furthermore, we explicitly forbid the use of "local variables" that occur only in the equations of a single process, since this would hide a causal relation.

Table 2.1 shows the notation for declaring a process in the modeling language. To declare a process, the user must give it a name that does not occur elsewhere in the model and declare its conditions and equations. As can be observed, the conditions statement is optional; its absence indicates that the process is always active.

Table 2.1: Regular expression for declaring a process

```
process <process_name>;
  [conditions <boolean_expression> [, <boolean_expression>]* ;]
   equations <equation>; [<equation>;]*
```

The language also incorporates the concept of an *intrinsic property*, which maps from nominal values to numerical ones. We find several examples in science of properties that take this form, such as gravitational mass, index of refraction, and density. Each case involves a mapping from nominal values (e.g., the planet name or substance name) to numerical values associated with them. Intrinsic properties in our language take the form of pseudo-processes that encode the mapping of values. Although they work in a very similar way, they have a special notation that the regular expression in Table 2.2 describes.

Table 2.2: Regular expression for declaring an intrinsic property

```
intrinsic <property_name>;
  variables <variable_name>;
  definition <variable_name> == <nominal_value> -> <numeric_value>;
             [ <variable_name> == <nominal_value> -> <numeric_value>; ]*
```

2.3 Examples of Process Models

The use of the modeling language can be illustrated with some example models. Here we present two examples, one for a dynamic system that involves differential equations and one for a static model that involves only instantaneous equations.

2.3.1 A Dynamic Model

The first example, taken from thermodynamics, models water transitions between its solid, liquid, and gaseous phases. In particular, this model deals with the phase changes as they occur when heat is added at a constant rate to a mass of ice and how it undergoes phase changes to liquid water and then to steam. Let us consider what happens in a system of this type.

1. The ice warms, that is, the temperature of the system increases until it reaches 0 degrees Celsius. The change in temperature is governed by the equation

$$\frac{\partial temp}{\partial t} = \frac{heat}{0.00206 \times ice\_mass} \quad . \tag{2.1}$$

2. Once the ice reaches 0 degrees Celsius, the temperature stops increasing while the ice melts changing from ice to liquid water. Two equations are used in this case, one to model the change in the mass of ice and the other to model the change in the mass of water

$$\frac{\partial ice\_mass}{\partial t} = -\frac{(18 \times heat)}{6.02} \tag{2.2}$$

$$\frac{\partial water\_mass}{\partial t} = \frac{(18 \times heat)}{6.02} \quad . \tag{2.3}$$

3. Once all the ice has melted, the temperature starts to increase again until the liquid water reaches 100 degrees Celsius. The change in temperature is governed by the equation

$$\frac{\partial temp}{\partial t} = \frac{heat}{0.004184 \times water\_mass} \quad . \tag{2.4}$$

4. When the liquid water has reached 100 degrees Celsius, the temperature stops increasing and the liquid water transforms into steam. The transference of mass between liquid water and steam is modeled with the equations

$$\frac{\partial water\_mass}{\partial t} = -\frac{(18 \times heat)}{40.7} \tag{2.5}$$

$$\frac{\partial steam\_mass}{\partial t} = \frac{(18 \times heat)}{40.7} \quad . \tag{2.6}$$

5. After all the liquid water has become steam, it starts to increase its temperature. That phenomena is described with the equation

$$\frac{\partial temp}{\partial t} = \frac{heat}{0.00202 \times steam\_mass} \quad . \tag{2.7}$$

One can find similar descriptions explaining the behavior of such systems in introductory textbooks in thermodynamics. Translating this description into one of our process models is straightforward, since our formalism was designed with them in mind. To perform the translation, we must first identify the processes that appear in the model. From the behavioral description, we can conclude that this system involves five processes: ice warming, ice melting, liquid water warming, liquid water boiling, and finally steam warming. The resulting model appears in Table 2.3, which shows the processes have conditions that reflect when they should be active to generate the behavior described previously. The equations associated with those processes describe the change over time of the variables in the model as described in the previous text. The notational convention for equations is similar to that used in mathematical software, though we should clarify the notational convention `d[temp,t,1]`, which represents the first order derivative of `temp` with respect to `t` (time). Also, notice that processes can involve more than one equation as is exemplified here by ice_melting and water_boiling which describe the transference of mass between phases of water using two differential equations.

Table 2.3: A process model of mass and temperature change in an ice-water-steam system

```
model WaterPhaseChange;
variables temp, heat, ice_mass, water_mass, steam_mass;
observable temp, heat, ice_mass, water_mass, steam_mass;

process ice_warming;
  conditions ice_mass > 0, temp < 0;
  equations
    d[temp,t,1] = heat / (0.00206 * ice_mass);

process ice_melting;
  conditions ice_mass > 0, temp == 0;
  equations
    d[ice_mass,t,1]   = -(18 * heat) / 6.02;
    d[water_mass,t,1] =  (18 * heat) / 6.02;

process water_warming;
  conditions ice_mass == 0, water_mass > 0, temp >= 0, temp < 100;
  equations
    d[temp,t,1] = heat / (0.004184 * water_mass);

process water_boiling;
  conditions water_mass > 0, temp >= 100;
  equations
    d[steam_mass,t,1] =  (18 * heat) / 40.7;
    d[water_mass,t,1] = -(18 * heat) / 40.7;

process steam_warming;
  conditions water_mass == 0, steam_mass > 0, temp >= 100;
  equations
    d[temp,t,1] = heat / (0.00202 * steam_mass);
```

Table 2.4: Equations used in the NPPc portion of the CASA ecosystem model

$$NPPc = E \times IPAR$$

$$E = 0.56 \times T1 \times T2 \times W$$

$$T1 = 0.8 + 0.02 \times topt - 0.0005 \times topt^2$$

$$T2 = 1.18/[1 + e^{0.2\times(topt-tempc-10)}] \times [1 + e^{0.3\times(-(topt-tempc)-10)}]$$

$$W = \begin{cases} 0.5 & (if \quad PET = 0) \\ 0.5 + 0.5 \times eet/PET & (if \quad PET \neq 0) \end{cases}$$

$$PET = \begin{cases} 0 & (if \quad tempc \leq 0) \\ 1.6 \times (10 \times tempc/ahi)^A \times pet\_tw\_m & (otherwise) \end{cases}$$

$$A = 0.000000675 \times ahi^3 - 0.0000771 \times ahi^2 + 0.01792 \times ahi + 0.49239$$

$$IPAR = 0.5 \times FPAR\_FAS \times monthly\_solar \times days\_per\_month \times 0.0864$$

$$FPAR\_FAS = \left[\left(\frac{1000+fas\_ndvi}{1000-fas\_ndvi}\right) - 1.08\right] / srdiff\,(umd\_veg)$$

$$srdiff\,(umd\_veg) = \begin{cases} 3.06 & (if umd\_veg = A \ or \ F) \\ 4.35 & (if umd\_veg = B \ or \ C) \\ 4.05 & (if umd\_veg = D \ or \ G \ or \ H \ or \ I \ or \ K) \\ 5.09 & (if umd\_veg = E \ or \ J) \end{cases}$$

2.3.2 A Static Model

Our second example comes from Earth Science and corresponds to a portion of CASA, a model developed by Potter and Klooster [18] at NASA Ames Research Center. CASA accounts for the global production and absorption of biogenic trace gases in the Earth atmosphere, as well as predicting changes in the geographic pattern of major vegetation types on the land. For this example, we chose a part of the model that predicts the monthly net production of carbon, or NPPc, at a given location in the globe. Table 2.4 shows the equations that occur in the NPPc portion of the CASA ecosystem model, which as can be seen only contains instantaneous equations.

The translation of the NPPc model into our formalism resulted in the code presented in Tables 2.5 and 2.6. As can be seen, we treated each equation as a process, which made the translation straightforward. Equations with conditions were easily mapped into processes with conditions and the mapping from nominal values to discrete values denoted *srdiff* (*umd_veg*) was translated as an intrinsic property.

2.4 Concluding Comments

In this chapter we presented a process modeling language that we maintain maps well into the way in which scientists usually structure their models. Furthermore, we claim that casting models in terms of processes with associated equations is desirable, since it lets scientists express their background knowledge directly without the expense of recasting the representation in some other terms. To support our claims, we have not only provided arguments based on previous work, but also we have shown two examples from two quite different domains that demonstrate the straightforward task of mapping existing quantitative models into our process modeling language.

Despite its broad applicability, we should clarify that our formalism is not universal; it is well suited only for domains in which quantitative models are appropriate. However, many branches of science and engineering make extensive use of quantitative modeling, thus giving our formalism a broad range of application. We have successfully applied it to problems from thermodynamics, population dynamics, Earth science, and physics. Moreover, other areas that can be approached with our modeling language include biology, engineering, and economics as well as many others where quantitative models are regularly used.

In summary, we have presented a new formalism for modeling that is closely related to the conceptualizations often found in scientific and engineering domains, in particular ones that make use of quantitative modeling with numerical equations. Although our formalism is not suited to model every type of problem, there are many domains that make use of quantitative modeling, thus providing fertile ground for the application of our framework.

Table 2.5: Modeling language representation of the NPPc model

```
model npp;

variables topt, tempc, monthly_solar, fas_ndvi, eet, ahi, pet_tw_m,
          umd_veg, days_per_month, FPAR_FAS, W, IPAR, E, T2, T1,
          A, PET, NPPc;

observable topt, tempc, monthly_solar, fas_ndvi, eet, ahi, pet_tw_m,
            days_per_month, umd_veg;

process CarbonProd;
  equations
    NPPc = E * IPAR;

process PhotoEfficiency;
  equations
    E = 0.56 * T1 * T2 * W;

process TempStress1;
  equations
    T1 = 0.8 + 0.02 * topt - 0.0005 * (topt ^ 2);

process TempStress2;
  equations
    T2 = 1.18 / ((1 + (e ^ (0.2 * (topt - tempc - 10)))) *
         (1 + (e ^ (0.3 * (-(topt - tempc) - 10)))));

process WSNoEvapoTrans;
  conditions PET==0;
  equations
    W = 0.5;

process WaterStress;
conditions PET!=0;
  equations
    W = 0.5 + 0.5 * (eet / PET);
```

Table 2.6: Modeling language representation of the NPPc model (continued)

```
process NoEvapoTrans;
  conditions tempc<=0;
  equations
    PET = 0;

process EvapoTrans;
  conditions tempc>0;
  equations
    PET = 1.6 * ((10 * tempc / ahi) ^ A) * pet_tw_m;

process HeatAdj;
  equations
    A = (0.000000675 * ahi ^ 3) - (0.0000771 * ahi ^ 2) +
        (0.01792 * ahi) + 0.49239;

process StressComb;
  equations
    IPAR = 0.5 * FPAR_FAS * monthly_solar * 0.0864 * days_per_month;

process AbsRadiation;
  equations
    FPAR_FAS = (((1000 + fas_ndvi) / (1000 - fas_ndvi)) - 1.08) / srdiff;

intrinsic srdiff;
  variables umd_veg;
  definition
    umd_veg == ``A'' -> 3.06;
    umd_veg == ``B'' -> 4.35;
    umd_veg == ``C'' -> 4.35;
    umd_veg == ``D'' -> 4.05;
    umd_veg == ``E'' -> 5.09;
    umd_veg == ``F'' -> 3.06;
    umd_veg == ``G'' -> 4.05;
    umd_veg == ``H'' -> 4.05;
    umd_veg == ``I'' -> 4.05;
    umd_veg == ``J'' -> 5.09;
    umd_veg == ``K'' -> 4.05;
```

Chapter 3

The Integrated Environment

One goal of this thesis is to provide an integrated framework that supports the construction, simulation, analysis, and discovery of scientific models. We have implemented such environment that supports both graphical and textual input of process models as well as visualization, simulation, analysis, and discovery of those models. In this chapter, we describe the main features of this integrated environment, called IKD (Interactive Knowledge and Discovery), focusing on three particular areas, visualization, simulation, and analysis of models.

3.1 Visualization of Models

The environment lets users visualize process models as causal graphs that it derives from the equations that compose each process. It displays causality graphically by laying out the processes from left to right and signaling the directionality of effects by arrows. An arrow from a process to a variable indicates that the process influences the variable, whereas an arrow from a variable to a process indicates that the variable is an input to the process. The conventions earlier described are enough to provide a simple mechanism for visualizing the causal flow that emerges from a model.

Variables and processes are distinguished graphically by representing them with different shapes. Processes take the form of rectangular boxes, while variables are shown as ovals. Intrinsic properties are also shown as ovals to denote that they declare a new variable named after the intrinsic property, but they can be distinguished from variables because they have different colors. Figure 3.1 shows the graphical output that is produced by the systems for the NPPc model discussed in Section 2.3.2. Notice that the intrinsic property *srdiff* appears in a different gray tone than other variables in the model, which corresponds to the different color used in its display.

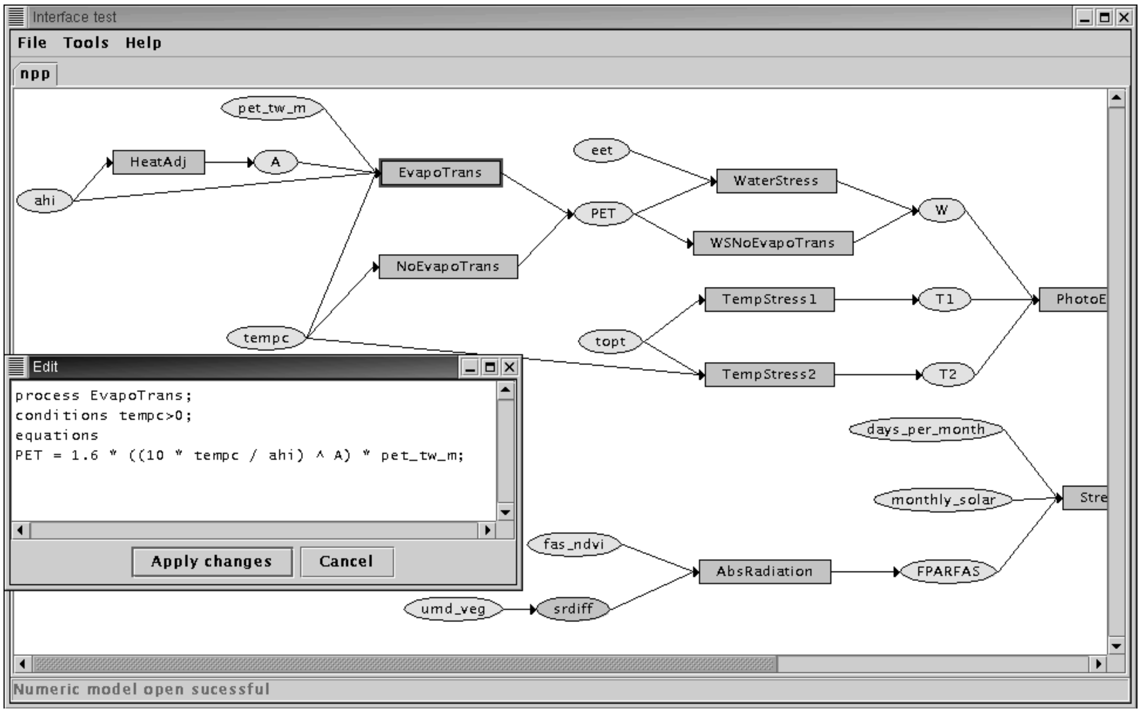

Figure 3.1: A snapshot of the NPPc model being visualized in IKD

Finally, the graphical interface goes beyond the display of models to allow graphical construction and modification of process models, as well as changing the layout of the elements in the output to satisfy the visual needs of the user. Figure 3.1 shows an example of the interface to inspect and modify an existing model where a dialog box appears in the lower left corner. This box shows the code that defines the selected process, in this case is the process EvapoTrans. Modifications to the code for the process are reflected in the model once they are committed, thus allowing interactive construction and refinement of models.

3.2 Simulation and Prediction

Scientists discover models not only to understand phenomena but also to use them for prediction and simulation. Moreover, while refining a model, the ability to analyze the performance of a model often gives insight when the model does not work as desired. For this reason, IKD includes a module for numerical simulation and analysis that operates directly on models expressed in our language. This lets the environment use models to make predictions and to evaluate their ability to fit observations.

Simulation of models in IKD involves translating the model from its encoding in the modeling language into a C program. The translation is done automatically, without user intervention, thus freeing the scientist from error-prone implementation details. Moreover, models can constantly change when the scientist is in the process of construction or revision. Thus, the ability to alter only the high-level representation of the model and ignore implementation details aids productivity considerably.

The modeling language supports two types of models: static models, which consist only of instantaneous equations, and dynamic models, which include both instantaneous and differential equations. IKD treats each type separately when performing the conversion, since they have different requirements, thus generating specially tailored code for each type. For static models, the system generates a program that relies on the standard mathematical library from the C language, since this type of simulation needs nothing more. On the other hand, for dynamic models with differential equations, the environment generates code that invokes CVODE (Cohen & Hindmarsh [5]), a well-known package for solving systems of ordinary differential equations. The code generated is completely integrated with that library, thus taking advantage of its proven functionality and optimizations.

Since conditions in processes dynamically change the form of the equations in the model, simulation must be robust with respect to them. The simulator's architecture solves this problem by allowing dynamical adaptation of the equations based on whether the conditions in processes are met.

Simulation of models that contain only instantaneous equations proceeds by reading values for the input variables from a stream of data and computing the values for the variables in the model for each set of inputs. However, simulation of models including differential equations is different, since they involve solving an initial value problem; thus, only the initial values of the input variables are read from the data file and simulation predicts the trajectory over time for each variable in the model.

Finally, results of the simulation can be visualized in the environment, which presents to the user a table containing the predicted values of the variables in the model as well as the inputs used to compute them.

3.3 Analysis of Model Behavior

The integrated environment provides two mechanisms to assist the user in analyzing how well a model is working. One of those mechanisms lets him observe which processes are active at any point in the data stream (in the case of a static model) or in time (in the case of a dynamic model). The second mechanism lets him compare predictions produced for each variable in the model with their observable values.

The mechanism that shows which processes are active relies on a coloring scheme which assigns a bright color to processes whose conditions are met. Figure 3.2 shows part of the NPPc model from Section 2.5 and the pattern of active and inactive processes. A darker color in the picture indicates that the process is active, as with TempStress1 and TempStress2, which have no conditions, while the process EvapoTrans is not active, since it is conditioned on the value of *tempc* , which is below the specified threshold in this case.

The user can also compare a model's prediction for variables against those observed in a stream of data. Again a coloring scheme combined with filling patterns signals differences in values between the model and the target of the comparison. Figure 3.2 shows that the W variable is being highlighted with a different fill pattern. Several types of comparison tests can be used that establish thresholds which are monitored during simulation and can be observed.

Finally, to allow navigation through the data stream (in the case of a static simulation) or through time (in the case of a dynamic simulation) the interface incorporates a slider. Moving the slider lets the user jump freely in the data stream or in the

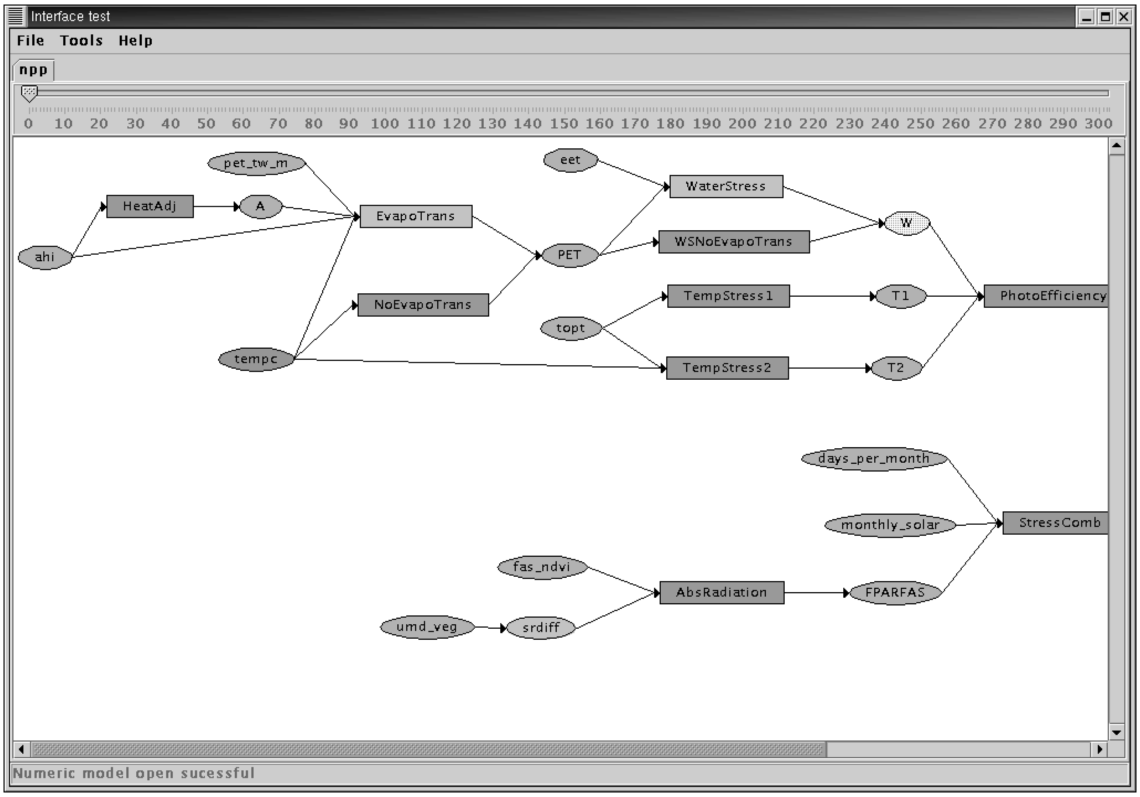

Figure 3.2: A snapshot of the NPPc model simulation being analyzed in IKD

history of the predictions. In Figure 3.2, the slider indicates that we are visualizing the results of the predictions made by the model in the first row of input data, since the NPPc model is a static model.

3.4 Model Revision

An important stage in the process of knowledge discovery is the revision of models. The IKD environment supports this task by giving the user the ability to provide constraints on model revisions. Those constraints define a restricted search space that the system then explores to find the revisions that fit the data better.

Our implementation of model revision relies on the concept of a *generic process*, a special type of process that encodes background knowledge for use in discovery.

A generic process differs from a specific process (the type of process previously described) in that it does not commit to specific variables or parameter values. Hence, a generic process is the general description from which a complete family of specific processes can be derived depending on the particular values of its constants, thus providing a way to encode declarative bias on the space of equations. Otherwise, generic processes are like specific processes in that they can have conditions and instantaneous and/or differential equations.

We claim that generic processes are powerful tools for encoding background knowledge and that they can be easily extracted from existing knowledge. Models usually contain parameterized equations which map directly into the notion of generalized equations that are part of a generic process declaration. Furthermore, many domains use generalized equations to model different phenomena. For instance, in population dynamics the literature mentions processes for growth, death, and predator-prey relations between species; these are modeled using generalized equations whose parameters depend on the species involved, as well as are characteristics of the environment being modeled.

An IKD user can revise a model by specifying candidate generic processes that can take the place of selected specific processes. The revision mechanism relies on the user specifying which parts of the model he/she thinks are problematic, thus obtaining constraints that restrict the search for revised models. The user introduces these constraints by selecting a process and then selecting alternative generic processes drawn from a library of generic processes that can take its place. Since libraries of generic processes tend to grow with time, the system helps the user by listing as options only the generic processes that are consistent with the process being replaced. Two processes are consistent if they have the same number of input and output variables and the variables have the same types, or if any is a subtype of the other.

Once the user has identified all processes that should be considered during model revision and the generic processes that should be tried for them, IKD applies a transformation to the process model that produces a context-free grammar. The resulting grammar encodes the possible revisions of the model in a format suitable for input to LaGramge (Todorovski & Džeroski [21]), an equation discovery system which uses grammars for declarative bias of the search space of equations. Furthermore, the generated grammar includes enough information and structure to enforce the revisions specified by the user at the level of processes, since LaGramge only works at the level of equations, as well as information to reconstruct a process model from it.

The resulting grammar is then fed into LaGramge, which searches the space of model structures derived from the revision criteria and attempts to optimize the parameters in the generic processes to fit the data. The result produced by LaGramge, which is internally cast as a parse tree, is then converted by IKD into the encoded process model. The result is a new process model that lowers error for the available data, or the original model if no revision could improve it.

Notice that the transformations applied to the process model are transparent to the user of the system, who does not have to know how to encode a process model into a grammar or how to specify on it the desired revisions. Finally, note that this automatic transformation mechanism could instead use other revision algorithms, since the modeling language was designed with generality in mind to allow its easy translation into other formalisms.

Chapter 4

Experimental Results

In this chapter we examine two illustrative problems on which we tested our framework. We provide one example that relies on synthetic data from a population dynamics domain and another that uses natural data to revise an existing Earth science model.

4.1 A Predator-Prey Model

Our first experiment involves a two-species predator-prey model taken from the literature on population dynamics. The particular example is a Holling's predator-prey model of type II, in which the activity of predators' searching for food is modeled with a constant rate (Holling [8], [9]). The model is expressed with the following system of equations

$$\frac{\partial prey}{\partial t} = prey\_growth\_rate - consumption\_rate$$
$$\frac{\partial predator}{\partial t} = 0.5 \times consumption\_rate - predator\_decay$$

where

$$
\begin{aligned}
prey\_growth\_rate &= 2 \times prey \times \left(1 - \frac{prey}{1.4}\right) \\
predator\_decay &= -0.1 \times predator \\
consumption\_rate &= \frac{pred \times prey}{(1 + 0.1 \times prey)}
\end{aligned}
$$

The parameters used for the model are: constant search rate of 1.0, reproduction rate for predators of 0.5, time of handling of prey of 0.1, prey capacity of the environment equal to 1.4, rate of prey population increase equal to 2, and decay rate of predators equal to 0.1. Its representation in the process modeling language appears in Table 4.1. We declared four processes: pred_prey_consumption, which models consumption of preys and reproduction of predators using the theoretical term consumption_rate; prey_logistic_growth, which models a logistic growth process for prey; predator_exponential_decay, which models the natural death of predators; and prey_consumption_rate which computes the theoretical term consumption_rate based on Holling's "disc equation" of functional response. Is important to note that IKD assumes that processes have an additive effect on variables; thus this model produces the same system of differential equations as previously described.

To evaluate IKD's ability to revise models in this domain, we used the model in Table 4.1 to generate data for 200 time steps, using initial values of pred=2.5 and prey=1.5. We also introduced a small amount of Gaussian noise by replacing each true value x with $x * (1 + r * 0.05)$, where r was sampled from a Gaussian distribution with mean 0 and standard deviation 1 (which produces noise relative to the size of the actual value). We then changed the process model in two different ways and ran the revision algorithm using the perturbed models to verify if IKD could identify the correct version and discover the original model.

The first perturbation simply changed the rate of predator deaths to a different number. In particular, we tried a model where the original pred_exponential_decay process was replaced by one where the rate of predator deaths was increased to 0.5. To guide revision, we selected a generic version of the exponential decay process to test if the system could identify the correct value for the rate of decay. The particular

Table 4.1: The target process model of two-species predator-prey

```
model predprey_holling;
variables pred{population},prey{population},
          consumption_rate{generic_constant};
observable pred,prey;

process pred_prey_consumption;
equations
  d[prey,t,1] = -consumption_rate;
  d[pred,t,1] = 0.5 * consumption_rate;

process prey_logistic_growth;
equations  d[prey,t,1] = 2 * prey * (1 - prey / 1.4);

process pred_exponential_decay;
equations  d[pred,t,1] = -0.1 * pred;

process prey_consumption_rate;
equations  consumption_rate = pred * prey / (0.1 * prey + 1);
```

generic process, shown in Table 4.2, uses [0,1,10] to encode a generic constant that specifies that the rate of decay can vary between 0 and 10, and that the initial guess for the constant is 1. After running the revision module the system returned a process model where the pred_exponential_decay process was correctly replaced by a revision of it. The revision mechanism detected that the perturbed process was not working well and generated a revised version that comes close to the original equation, containing the parameter $decay\_rate = -0.103055$, whereas the original model has $decay\_rate = -0.1$.

Table 4.2: Generic process for the first experiment of the predator-prey model

```
generic process exponential_decay;
variables P{population};
equations
  d[P,t,1] = - [0,1,10] * P;
```

Table 4.3: Replacement process used for the second experiment of the predator-prey model

```
process prey_exponential_growth;
equations
  d[prey,t,1] = 2 * prey;
```

The second experiment was designed to test the ability of the system to handle a more complex revision that involves changing the nature of a process. This time we perturbed the original model by replacing the logistic growth process for prey by one that assumes exponential growth, as shown in Table 4.3.

We instructed the system to revise the model, in this case giving the generic processes listed in Table 4.4 as alternatives for the growth and death processes. Thus, we let the system determine the exact growth and death processes that appear in the model. The result was a correct process model where the exponential growth process for prey was replaced by a logistic growth process with the equation

$$1.86847 \times prey \times \left(1 - \frac{prey}{2.49097}\right)$$

and the remaining processes where unchanged. Although the discovered logistic growth process is not identical to the original process used to generate the data, whose equation is

$$2 \times prey \times \left(1 - \frac{prey}{1.4}\right) \ ,$$

we should note that this is a very good solution, since two factors need to be considered in analyzing the result. First, the data contains noise, thus, an exact parameter estimation is not feasible, and secondly, this is an example specially difficult due to the oscillatory behavior of the system and the number of degrees of freedom, a characteristic that increases the difficulty of parameter estimation. Nevertheless, even in this difficult case, the system was able to correctly determine the model structure and find a fairly good approximation to the original process.

Table 4.4: Generic processes used for the second experiment of the predator-prey model

```
generic process exponential_decay;
variables P{population};
equations
  d[P,t,1] = - [0,1,10] * P;

generic process exponential_growth;
variables P{population};
equations
  d[P,t,1] =  [0,1,10] * P;

generic process logistic_grwth;
variables P{population};
equations
  d[P,t,1] = [0,1,10] * P (1 - P / [0,1,100]);
```

4.2 An Earth Science Model

Having showed some of the capabilities of the framework in a synthetic environment, we now move to the revision of a model from Earth Science using real data. The model we chose to revise which we already introduced in Table 2.4 is a portion of the CASA model that predicts the net production of carbon. This particular model has already been the subject of a previous publication in computational scientific discovery (Saito et al. [19]), so it constitutes a good problem to be attempted for comparison purposes.

To translate the NPPc model into our formalism, we treat each equation as a process, as already shown in Tables 2.5 and 2.6. Using this encoding of the NPPc model, we revisited all the revisions explored in Saito et al. These included revision of the parameters in the $T2$ and PET equations, the structure of the E equation, and the values of the intrinsic property *srdiff*. To carry out these revisions, we only needed to define three generic processes, one for each equation to be revised, and decide the range of the interval for the intrinsic values. For comparisons purposes, we

Table 4.5: Generic processes used for the NPPc model revision

generic process T2Alt;
variables topt{number},tempc{number},T2{number};
equations
$T2 = [0.0001, 1.1814, 1000]$ /
$((1 + (e\hat{}([-1, 0.2, 1] * (topt - tempc - [-1000, 10, 1000])))) *$
$(1 + (e\hat{}([-1, 0.3, 1] * (-(topt - tempc) - [-1000, 10, 1000])))));$

generic process EvapoTransAlt;
variables tempc{number}, ahi{number}, A{number},
pet_tw_m{number}, PET{number};
conditions $tempc > 0$;
equations
$PET = [0.0001, 1.6, 1000] * (([0.0001, 10, 1000] * tempc/ahi)\hat{}A) * pet\_tw\_m;$

generic process PhotoEfficiencyAlt;
variables T1{number}, T2{number}, W{number}, E{number};
equations
$E = [0, 0.56, 100] * T1\hat{}[0, 1, 100] * T2\hat{}[0, 1, 100] * W\hat{}[0, 1, 100];$

used the same data set as in Saito et al.'s work. The generic processes that encode the desired revisions are shown in Table 4.5.

The first two revisions involved changing constants in the equations for $T2$ and PET. Encoding those rules in the formalism is straightforward, since the alternatives are processes that have the same form for the equations and that change the constants with ranges that the algorithm should consider. The generic processes used to encode these revisions are T2Alt and EvapoTransAlt, shown in Table 4.5.

IKD's revision of $T2$ resulted in the equation

$$T2 = \frac{28.0127}{\left(1 + e^{-0.0058\times(topt - tempc + 160.127)}\right) \times \left(1 + e^{-0.0313\times(-(topt - tempc) - 91.4912)}\right)}$$

which differs from the result reported in Saito et al. where the equation found was

$$T2 = \frac{1.80}{\left(1 + e^{0.05\times(topt - tempc - 10.8)}\right) \times \left(1 + e^{-0.03\times(-(topt - tempc) - 90.33)}\right)} \ .$$

The comparison of the two results leads to the observation that the revised equations agree on the values of the parameters in the second exponential term, while they differ considerably in the remaining parameters. This difference is due to several local minima occurring in the space being explored and how each revision algorithm was able to deal with it. The comparison of the two revisions using the provided data set shows that IKD's result is considerably better than Saito et al.'s published revision, achieving even more reduction in the predictive error of the NPPc model as shown in Table 4.8 shows.

The second revision studied in Saito et al. also involved revising parameters of an equation, which in this case was PET's equation. IKD's result of that revision produced the equation

$$PET = 1.44366 \times (9.4008 \times tempc/ahi)^A \times pet\_tw\_m$$

while Saito et al. reports obtaining

$$PET = 1.56 \times (9.16 \times tempc/ahi)^A \times pet\_tw\_m \ .$$

Again, we are faced with another case of different local minima occurring in the space being explored, however, is interesting to notice that the two revisions agree in producing a result where the two constants involved are smaller than in the current NPPc model. Comparison of the predictive error of the two resulting models shows that IKD's revision again obtains higher reduction error as shown in Table 4.8.

The third revision studied in Saito et al. involved letting the system change the structure of the equation for E, the photosynthetic efficiency. They achieved this by letting the system find values for the exponents of the three variables being multiplied. Since this can be easily encoded in our formalism, we used the same approach, leading to the generic process PhotoEfficiencyAlt shown in Table 4.5. The revised equation returned by the revision module was

$$E = 0.529441 \times T1^0 \times T2^{0.0552} \times W^0$$

while Saito et al.'s result was

$$E = 0.521 \times T1^{0} \times T2^{0.03} \times W^{0} \ .$$

The results produced by both approaches are remarkably similar, in particular both produced a revised version of E's equation that depends only on $T2$. This interesting result, also observed in Saito et al.'s work, has been well received by the Earth scientists, who hypothesize that those terms being discarded by the revision represented influences that are being captured now by improved measurements of the NDVI variable, which where not available when the CASA model was first developed. Comparison of the predictive error of the revisions obtained in both approaches shows again that IKD's revision better predicts the observed data.

The last revision that Saito et al. describes involved revising the 11 values of the intrinsic property *srdiff*, which takes on a different quantitative value for each type of vegetation. To revise intrinsic properties, we need only determine the range of possible values. Using that constraint the system then solves an optimization problem to determine the best numeric parameters for each nominal value. In this case revision yielded the results shown in Table 4.6, which are mostly very similar to the ones obtained previously. The main differences occur on the vegetation types for which less data are available, so that we cannot treat either result as reliable.

Table 4.6: Initial and revised values for *srdiff*

Vegetation type	A	B	C	D	E	F	G	H	I	J	K
original	3.06	4.35	4.35	4.05	5.09	3.06	4.05	4.05	4.05	5.09	4.05
IKD revision	2.52	4.48	0.01	2.84	3.31	3.27	2.31	0.01	1.71	2.84	1.04
Saito et al. revision	2.57	4.77	2.20	3.99	3.70	3.46	2.34	0.34	2.72	3.46	1.60

Furthermore, since our framework supports multiple simultaneous revisions, something that Saito et al. suggested as future work, we also tested this capability. In particular, we instructed IKD to consider simultaneously all the previously shown

Table 4.7: Initial and revised values for *srdiff* when revising all the equations simultaneously

Vegetation type	A	B	C	D	E	F	G	H	I	J	K
original	3.06	4.35	4.35	4.05	5.09	3.06	4.05	4.05	4.05	5.09	4.05
IKD revision	0.68	0.10	0.10	3.22	0.10	1.52	1.68	0.10	0.10	1.84	1.00

revisions, that is, revising $T2$'s and PET's parameters, the structure of E's equation, and the *srdiff* mappings. To carry out these revisions we used exactly the same generic processes already shown in Table 4.5. The resulting model contained revisions only to PET's equation and to the intrinsic property *srdiff*, leaving both E and $T2$ unchanged, and achieves even more reduction in the error. The revised values for the intrinsic property srdiff appear in Table 4.7, while the revised PET equation is

$$PET = 2.78661 \times (15.7127 \times tempc/ahi)^A \times pet\_tw\_m \ .$$

Comparison of these results with the previous revisions for PET and *srdiff* alone shows noticeable changes from those previously seen. In particular, this time the constants for the PET equation where increased from the values present in the original model (before those constants where revised to lower values), and the values for the intrinsic property consistently shrunk (while previously we saw one type having a revised value larger than the original model). Nevertheless, the predictive error of the revised model is noticeably lower than previous revisions, thus providing interesting results that require detailed analysis from Earth scientists.

Table 4.8 summarizes the key results that we obtained with IKD and that Saito et al. achieved with their approach. For each revision, we report the root mean squared error ($RMSE$), and the value of the correlation coefficient (r), which shows how well the model predictions and the data are correlated, with a value of one denoting a perfect correlation.

As Table 4.8 shows, IKD almost always produced better results than Saito et al., except for the revision of PET where the correlation coefficient is slightly better in

Table 4.8: Comparison of revision results between Saito et al. and IKD

Equation	Saito et al.		IKD	
	$RMSE$	r	$RMSE$	r
$T2$	457.757	0.80325	453.6784	0.80356
PET	464.358	0.79688	460.9788	0.79666
E	443.307	0.81673	439.7965	0.81674
srdiff	432.410	0.82907	424.5990	0.83285
All revisions	N/A	N/A	409.7950	0.84317

the earlier work. Furthermore, being able to revise several equations simultaneously proved useful, since it led to a model with substantially less error than those in which only one equation was revised. In summary, these promising results show the potential of the framework, since not only we obtained more accurate models than the previous approach, but also we accomplished the task with much less effort.

4.3 Closing Remarks

In this chapter, we illustrated IKD's approach to model revision of two different examples. The first one is a model containing differential equations that predict the change over time of the population sizes in a predator-prey situation. We changed processes in that model and let IKD revise it to determine if it could reconstruct the original structure from synthetic data. As the experiments showed, even for a system with oscillatory behavior, the discovery procedure found the correct structure and obtained good approximations to the original values used to generate the training data.

We next tested IKD's revision ability on a model developed at NASA Ames Research Center for the prediction of carbon production at different locations on the globe. Using data provided by Earth scientists, we replicated a number of revisions from a recent publication on the same problem. Furthermore, we also carried out a more ambitious revision that involved multiple changes to the model. Our results showed that we could replicate the same type of revisions as in the earlier work, producing with less effort models that were consistently more accurate.

Finally, all the revisions proceeded by evaluating different alternative generic processes, which we easily constructed from knowledge about the domains. This approach provides a powerful mechanism for model revision that is both flexible and makes contact with the established formalisms in science and engineering.

Chapter 5

Directions for Future Research

This thesis has described a new framework for computational modeling and discovery, as well as an implemented system, called IKD, that instantiates it. IKD offers an integrated approach to the task of knowledge discovery and provides the tools needed to accomplish it successfully. Our framework includes a new formalism for encoding scientific models and domain knowledge that is both highly communicable to scientists and useful for scientific discovery, thus filling a gap in earlier work on discovery systems.

We showed how models can be easily encoded in the formalism while keeping contact with scientists' traditional notions. We applied the framework to the problem of revising two different models, one involving a population dynamics model and another from Earth Science. In the second case, we attempted the same revisions reported in an earlier publication and showed not only that our approach can obtain similar revisions with less effort but that it can carry out more complex revisions as well. Furthermore, the resulting models were generally more accurate than those reported in earlier work.

Topics for further research include expanding the modeling language to incorporate the notion of subsystems that can map to physical entities, thus providing mechanisms to manage models of increasing complexity while retaining high communicability. Another extension would augment the existing hierarchy of variable types to incorporate dimensional information about the variables, thus giving the system enough knowledge to check for invalid models and automatically convert units between processes that use different measures.

Another interesting research topic concerns the incorporation of a hierarchy of processes that would be similar to the current hierarchy of variable types. For example, the process hierarchy could contain a qualitative version of a process like growth, which indicates just the sign of the relation between the variables. Processes lower in the hierarchy would represent more specialized types, such as exponential growth and logistic growth, which are expressed in terms of numerical equations. Such a hierarchy would prove useful in domains where the scientist is constructing a model but where there is not enough knowledge to determine the exact numeric form of equations. In such cases the scientist could initially use a qualitative process and later on, when data allows refinement of the model, the revision mechanism could replace the qualitative version with a quantitative version that expresses a numeric relation between the variables.

Also, although our research has concentrated on quantitative modeling, there is also a potential for applying the framework to qualitative modeling. Furthermore, we believe that a combination of quantitative and qualitative reasoning can provide a more powerful approach that, together with the process hierarchy mentioned above, can provide a framework that lets scientists smoothly transition from qualitative to quantitative models, or that lets them constrain revisions with domain knowledge about qualitative properties of the model.

We also plan to incorporate other discovery algorithms into the system, thus broadening the choices for revision and allowing use of the method that gives the best results for a particular revision task. Since models are stated in the process modeling language, we can build a different transformation for each formalism, keeping the user isolated from the internal representation used to carry out the discovery task.

In addition, we plan to extend the IKD environment to support the induction of process models (Langley, Sánchez, Todorovski and Džeroski [14]), which involves composing specific models from abstract processes to explain quantitative data. Moreover, we believe combining the current revision techniques with process model induction is another interesting area for research, since it would let the environment consider adding and deleting processes in the model, which it cannot do currently.

Finally, we plan to apply our framework to the revision and discovery of other models in different domains and to encourage its adoption by scientists in areas where it can aid their research. This would fulfill a long-term goal of providing scientists with an interactive environment for modeling and discovery that effectively aids their research.

Appendix A

Process Modeling Language Reference

A.1 Model Grammar

ModelFile → *Header Body*

Header → `model` *Name;* `variables` *Variables;* `observable` *Observables;*

Variables → *VariableDecl* | *VariableDecl* , *Variables*

VariableDecl → *VariableName* | *VariableName* {*Type*}

Observables → *VariableName* | *VariableName* , *Observables*

VariableName → *Name* | *Name.VariableName*

Type → *Name*

Body → *IntrinsicProperties Processes*

Processes → *Process Processes* | *Process*

Process → `process` *Name; ProcessDecl*

ProcessDecl → *Conditions* `equations` *Equations*

Conditions → ϵ | `conditions` *ConditionList;*

ConditionList → *RelationalExpr* | *RelationalExpr, ConditionList*

Equations → *Equation* | *Equation Equations*

Equation → *VariableName* = *ArithmeticExpr;* | `d[`*Name*`,t,1]` = *ArithmeticExpr;*

$\mathit{IntrinsicProperties} \rightarrow \mathit{IntrinsicProperty\ IntrinsicProperties} \mid \epsilon$

$\mathit{IntrinsicProperty} \rightarrow \texttt{intrinsic}\ \mathit{Name;\ Assignments}$

$\mathit{Assignments} \rightarrow \mathit{Assignment} \mid \mathit{Assignment\ Assignments}$

$\mathit{Assignment} \rightarrow \mathit{Name}\ \texttt{==}\ \mathit{String} - > \mathit{Value;}$

$\mathit{RelationalExpr} \rightarrow \mathit{VariableName\ RelOperator\ ArithmeticExpr}$

$\mathit{RelOperator} \rightarrow \quad == \quad \mid\ != \ \mid \ > \ \mid \ < \ \mid \ <= \ \mid \ >=$

$\mathit{ArithmeticExpr} \rightarrow \mathit{UnaryArithExpr} \mid \mathit{BynaryArithExpr}$

$\mathit{UnaryArithExpr} \rightarrow \mathit{VariableName} \mid \mathit{Value} \mid (\mathit{ArithmeticExpr}) \mid -\mathit{ArithmeticExpr}$

$\quad \mid \texttt{sin}(\mathit{ArithmeticExpr}) \mid \texttt{cos}(\mathit{ArithmeticExpr}) \mid \texttt{tan}(\mathit{ArithmeticExpr})$

$\quad \mid \texttt{asin}(\mathit{ArithmeticExpr}) \mid \texttt{acos}(\mathit{ArithmeticExpr}) \mid \texttt{atan}(\mathit{ArithmeticExpr})$

$\mathit{BinaryArithExpr} \rightarrow \mathit{ArithmeticExpr\ ArithOperator\ ArithmeticExpr}$

$\mathit{ArithOperator} \rightarrow \texttt{+} \mid \texttt{-} \mid \texttt{/} \mid \texttt{*} \mid \texttt{\^{}}$

$\mathit{Name} \rightarrow \mathit{Letter\ AlphSymbols}$

$\mathit{Letter} \rightarrow \texttt{a} \mid \texttt{A} \mid \texttt{b} \mid \texttt{B} \mid \ldots \mid \texttt{z} \mid \texttt{Z}$

$\mathit{AlphSymbols} \rightarrow \mathit{AlphSymbols\ AlphSymbols} \mid \mathit{Letter} \mid \mathit{Digit} \mid \texttt{\_}$

$\mathit{Value} \rightarrow \mathit{Number.Number} \mid \mathit{Number}$

$\mathit{Number} \rightarrow \mathit{Digit} \mid \mathit{Digit\ Number}$

$\mathit{Digit} \rightarrow \texttt{0} \mid \texttt{1} \mid \ldots \mid \texttt{9}$

A.2 Generic Process Library Grammar

$\mathit{GenericProcessLibraryFile} \rightarrow \mathit{GenericProcesses}$

$\mathit{GenericProcesses} \rightarrow \mathit{GenericProcess\ GenericProcesses} \mid \mathit{GenericProcess}$

$\mathit{GenericProcess} \rightarrow \texttt{generic process}\ \mathit{Name;\ GenericProcessDecl}$

$\mathit{GenericProcessDecl} \rightarrow \texttt{variables}\ \mathrm{VariableDecl}\ \mathit{Conditions}\ \texttt{equations}\ \mathit{Equations}$

$\mathit{VariableDecl} \rightarrow \mathit{VariableName} \mid \mathit{VariableName}\ \{\mathit{Type}\}$

$\mathit{Conditions} \rightarrow \epsilon \mid \texttt{conditions}\ \mathit{ConditionList;}$

$\mathit{ConditionList} \rightarrow \mathit{RelationalExpr} \mid \mathit{RelationalExpr,\ ConditionList}$

$\mathit{Equations} \rightarrow \mathit{Equation} \mid \mathit{Equation\ Equations}$

$\mathit{Equation} \rightarrow \mathit{VariableName} = \mathit{ArithmeticExpr;} \mid \texttt{d[}\mathit{Name}\texttt{,t,1]} = \mathit{ArithmeticExpr;}$

$VariableName \to Name \mid Name.VariableName$

$Type \to Name$

$RelationalExpr \to VariableName\ RelOperator\ ArithmeticExpr$

$RelOperator \to\ == \mid\ != \mid\ > \mid\ < \mid\ <= \mid\ >=$

$ArithmeticExpr \to UnaryArithExpr \mid BynaryArithExpr$

$UnaryArithExpr \to VariableName \mid Constant \mid (ArithmeticExpr) \mid -ArithmeticExpr$
$\mid \texttt{sin}(ArithmeticExpr) \mid \texttt{cos}(ArithmeticExpr) \mid \texttt{tan}(ArithmeticExpr)$
$\mid \texttt{asin}(ArithmeticExpr) \mid \texttt{acos}(ArithmeticExpr) \mid \texttt{atan}(ArithmeticExpr)$

$BinaryArithExpr \to ArithmeticExpr\ ArithOperator\ ArithmeticExpr$

$ArithOperator \to \texttt{+} \mid \texttt{-} \mid \texttt{/} \mid \texttt{*} \mid \texttt{\^{}}$

$Constant \to Value \mid \texttt{[}Value, Value, Value\texttt{]}$

$Name \to Letter\ AlphSymbols$

$Letter \to \texttt{a} \mid \texttt{A} \mid \texttt{b} \mid \texttt{B} \mid \ldots \mid \texttt{z} \mid \texttt{Z}$

$AlphSymbols \to AlphSymbols\ AlphSymbols \mid Letter \mid Digit \mid \texttt{\_}$

$Value \to Number.Number \mid Number$

$Number \to Digit \mid Digit\ Number$

$Digit \to \texttt{0} \mid \texttt{1} \mid \ldots \mid \texttt{9}$

A.3 Types Taxonomy File Grammar

$TypeTaxonomyFile \to TypeList$

$TypeList \to TypeDecl\ TypeList \mid \epsilon$

$TypeDecl \to \texttt{type}\ TypeName\ \texttt{subtypeof}\ TypeName\ ;$

$TypeName \to Letter\ AlphSymbols$

$Letter \to \texttt{a} \mid \texttt{A} \mid \texttt{b} \mid \texttt{B} \mid \ldots \mid \texttt{z} \mid \texttt{Z}$

$AlphSymbols \to AlphSymbols\ AlphSymbols \mid Letter \mid Digit \mid \texttt{\_}$

$Digit \to \texttt{0} \mid \texttt{1} \mid \ldots \mid \texttt{9}$

Bibliography

[1] H. Bannai, Tamada Y., O. Maruyama, and S. Miyano. Vml: A view modeling language for computational knowledge discovery. In *Proceedings of the Fourth International Conference on Discovery Science*. Springer, 2001.

[2] J. Cartier, J. Rudolph, and J. Stewart. The nature and structure of scientific models. The National Center for Improving Student Learning and Achievement in Mathematics and Science, Working Paper, 2001.

[3] P. Cheeseman, D. Freeman, J. Kelly, M. Self, J. Stutz, and W. Taylor. Autoclass: A bayesian classification system. In *Proceedings of the Fifth International Conference on Machine Learning*, pages 54–64. Morgan Kaufmann, 1988.

[4] E. Chown and T.G. Dietterich. A divide and conquer approach to learning from prior knowledge. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 143–150. Morgan Kaufmann, 2000.

[5] S. Cohen and A. Hindmarsh. Cvode, a stiff/nonstiff ode solver in c. In *SciCADE95: Scientific Computing and Differential Equations*. Stanford University, 1995.

[6] M. Easley and E. Bradley. Generalized physical networks for automated model building. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 1047–1053. Morgan Kaufmann, 1999.

[7] K. Forbus. Qualitative process theory. In *Artificial Intelligence*, volume 24, pages 85–168. Elsevier Science Publishers, 1984.

[8] C.S. Holling. The components of predation as revealed by a study of small mammal predation of the european pine sawfly. *Canadian Entomology*, 91:293–320, 1959.

[9] C.S. Holling. Some characteristics of simple types of predation and parasitism. *Canadian Entomology*, 91:385–398, 1959.

[10] R. Jones. Generating predictions to aid the scientific discovery process. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 513–517. Morgan Kaufmann, 1986.

[11] S. Kokabas and P. Langley. Generating process explanations in nuclear astrophysics. In *Proceedings of the ECAI-98 Workshop on Machine Discovery*, pages 4–9, 1998.

[12] P. Langley. Data driven discovery of physical laws. *Cognitive Science*, 5:31–54, 1981.

[13] P. Langley. The computational support of scientific discovery. *International Journal of Human-Computer Studies*, 53:393–410, 2000.

[14] P. Langley, J. Sánchez, L. Todorovski, and S. Džeroski. Inducing process models from continuous data. In *Proceedings of the Eighteenth Conference on Machine Learning*. Morgan Kaufmann, 2002.

[15] P. Langley, H. A. Simon, G. L. Bradshaw, and J. M. Żytkow. *Scientific Discovery: Computational Explorations of the Creative Processes*. MIT Press, 1987.

[16] A. Mahidadia and P. Compton. Assisting model-discovery in neuroendocrinology. In *Proceedings of the Fourth International Conference on Discovery Science*, pages 214–227. Springer, 2001.

[17] R. S. Michalski and R. Stepp. Learning from observation: Conceptual clustering. In R. S. Michalski, J. G. Carbonell, and T. M. Mitchell, editors, *Machine Learning: An Artificial Intelligence Approach*. Morgan Kaufmann, 1983.

[18] C. S. Potter and S. A. Klooster. Interannual variability in soil trace gas (co_2, n_2o,no) fluxes and analysis of controllers on regional to global scales. *Global Biochemical Cycles*, 12:621–635, 1998.

[19] K. Saito, P. Langley, T. Grenager, C. Potter, A. Torregrosa, and S. A. Klooster. Computational revision of quantitative scientific models. In *Proceedings of the Fourth International Conference on Discovery Science*, pages 336–349. Springer, 2001.

[20] K. Saito and R. Nakano. Discovery of nominally conditioned polynomials using neural networks, vector quantizers and decision trees. In *Proceedings of the Third International Conference on Discovery Science*, pages 325–329. Springer, 2000.

[21] L. Todorovski and S. Džeroski. Declarative bias in equation discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 376–384. Morgan Kaufmann, 1997.

[22] L. Todorovski and S. Džeroski. Theory revision in equation discovery. In *Proceedings of the Fourth International Conference on Discovery Science*, pages 389–400. Springer, 2001.

[23] L. Todorovski and S. Džeroski. Using domain knowledge on population dynamics modeling for equation discovery. In *Proceedings of the Twelfth European Conference on Machine Learning*, pages 478–490. Springer, 2001.

[24] R.E. Valdés-Pérez. Machine discovery in chemistry: New results. *Artificial Intelligence*, 74:191–201, 1995.

[25] J. M. Żytkow. Incremental discovery of hidden structure: Applications in theory of elementary particles. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 750–756. AAAI Press, 1996.