

Extracting Constraints for Process Modeling

Will Bridewell
Computational Learning
Laboratory, CSLI
Stanford University
Stanford, CA 94305 USA
willb@csli.stanford.edu

Stuart R. Borrett^{*}
Computational Learning
Laboratory, CSLI
Stanford University
Stanford, CA 94305 USA
borretts@uncw.edu

Ljupčo Todorovski[†]
Computational Learning
Laboratory, CSLI
Stanford University
Stanford, CA 94305 USA
ljupco.todorovski@ijs.si

ABSTRACT

In this paper, we introduce an approach for extracting constraints on process model construction. We begin by clarifying the type of knowledge produced by our method and how one may apply it. Next, we review the task of inductive process modeling, which provides the required data. We then introduce a logical formalism and a computational method for acquiring scientific knowledge from candidate process models. Results suggest that the learned constraints make sense ecologically and may provide insight into the nature of the modeled domain. We conclude the paper by discussing related and future work.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning—*Induction, Knowledge Acquisition*

General Terms

Experimentation

Keywords

declarative bias, inductive process modeling

1. INTRODUCTION

Science is about knowledge; indeed, the English word ‘science’ derives from the Latin term for ‘knowledge’. This knowledge takes many forms, two of the most familiar being empirical laws and theoretical principles.

^{*}Currently affiliated with the Department of Biology and Marine Biology at the University of North Carolina Wilmington, Wilmington, NC

[†]Also affiliated with the Faculty of Administration at the University of Ljubljana, Ljubljana, Slovenia

However, in mature disciplines, scientists typically work at the intermediate level of *models* that combine theoretical principles with concrete assumptions to explain particular observations [29]. Models play an especially important role in fields like biology and geology, where one explains phenomena in terms of interactions among unobserved processes.

There is a growing literature on computational methods for constructing scientific process models. Early work on ‘compositional modeling’ [8] provided software environments for the manual creation of such models. More recently, researchers have reported automated or semi-automated systems for inferring process models in fields like medicine [18], ecology [28], and genetics [9]. These approaches combine a set of candidate processes into a coherent model that accounts for observations.

Elsewhere [3], we have argued that processes are an important form of knowledge both because they explain phenomena in terms familiar to scientists and because they limit the search for explanatory models. However, our experience suggests that other structures are equally important in guiding search through the model space. We claim that scientists also utilize *constraints* among processes to make model construction more tractable, and that these constraints constitute another important type of scientific knowledge. One example from ecology is the constraint that, if a model includes a birth process for an organism, it should also include a death process.

Although few traditions formalize process knowledge, scientific papers abound with terms that reveal their authors’ commitment to a process ontology. Unfortunately, the situation is quite different with constraints on process models, which appear to be more implicit. Scientists can easily identify models that violate intuitive constraints, but they are far less able to state them in the formal and general way needed for automated model construction. The need to infer such constraints poses the technical challenge addressed in this paper.

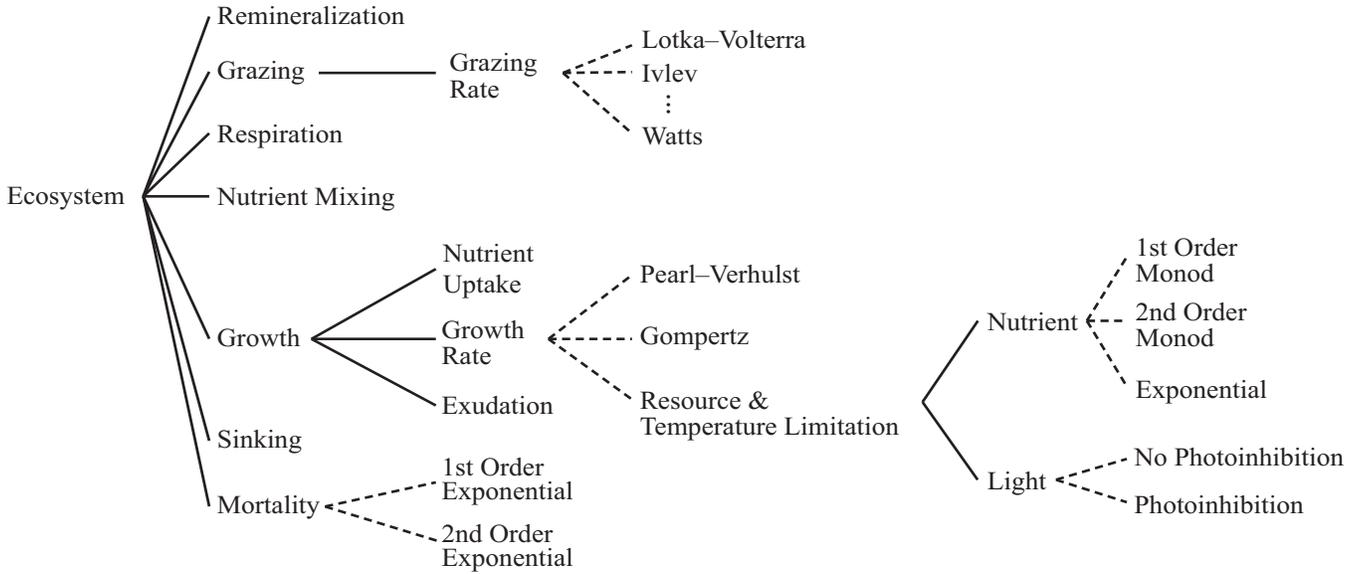


Figure 1: This figure shows the structure of a generic process library for aquatic ecosystems. The nodes of the hierarchy are generic processes. A solid line indicates that the parent process requires the presence of the subprocesses. A dotted line indicates that the children form a mutually exclusive choice point. We show a reduced set of the fourteen alternative forms of the grazing rate process.

Our approach builds on previous work in inductive process modeling [16], which carries out search through a space of quantitative process models to explain time-series data. This activity produces many candidate models, some that match the observations well and some that fare poorly. We use these models to generate positive and negative training cases, which we pass to a system that induces constraints among processes to distinguish between successful and unsuccessful models. We then present the inferred constraints to domain scientists for acceptance or rejection.

In the remaining pages, we report this approach in more detail. We start by reviewing our earlier research on inductive process modeling and how constraints on the model space could aid it. Next, we introduce a logical formalism for specifying constraints among processes, along with an algorithm for inducing these constraints from successful and unsuccessful models. After this, we report results on modeling aquatic ecosystems and present an ecologist’s (Stuart Borrett) analysis of the constraints that the system infers. We conclude by discussing related research and directions for future work.

2. INDUCTIVE PROCESS MODELING

The approach described in this paper works with the background knowledge and models relevant to HIPM [28], an inductive process modeler. This program takes as input observed trajectories of key variables, a list of entities operating in the modeled system, and domain knowledge in the form of a hierarchy of process templates that defines plausible model structures. As out-

put, HIPM produces *quantitative process models* whose simulated behavior fits the input trajectories.

The generated models comprise entities and the processes that relate them. Both classes of objects may contain constants that take on fixed numerical values, but only the entities have variables. One can define the system variables either with an equation or, in the case of exogenous terms, with a time series. Unlike entities, processes may contain Boolean conditions that control their activity and equations, either algebraic or differential, that define their effects. When multiple processes influence a single variable, HIPM assembles their equation elements according to that variable’s combining scheme (e.g., addition, multiplication). This procedure converts the process model into a system of differential equations that one can solve numerically. HIPM uses CVODE [5] for this purpose, which lets one compare simulated trajectories with available measurements.

To learn quantitative process models, HIPM requires a library of process templates, or *generic processes*, each of which resembles its instantiated counterpart with three notable exceptions. First, instead of entities, generic processes relate entity roles that must be filled by objects having a specific type. For instance, the grazing process requires an entity that is a primary producer (e.g., a plant) and one that is an animal. Second, the constants in the library have acceptable ranges that constrain them to plausible values. And third, the processes contain subprocesses and have types which let one impose constraints on candidate model structures.

Table 1: A model structure generated by HIPM.

entities
 P: phytoplankton, Z: zooplankton, N: nitrate,
 F: iron, E: environment, D: detritus

ecosystem(P, Z, E, D, [N,F])
 growth(P, E, D, [N, F])
 pearl_verhulst(P, E, [N, F])
 exudation(P)
 nutrient_uptake(P, N)
 nutrient_uptake(P, F)
 mortality_exp(P, E, D)
 mortality_exp(Z, E, D)
 grazing(P, Z, E, D)
 ivlev(P, Z)
 remineralization(D, N)
 remineralization(D, F)
 respiration(Z)
 sinking(P)
 sinking(D)
 nutrient_mixing(E, N)
 nutrient_mixing(E, F)

Figure 1 illustrates a hierarchical generic process library for aquatic ecosystems. The left most node, *Ecosystem*, is the root of the tree and comprises processes for remineralization, grazing, and so on. The grazing subprocess includes a single subprocess with type *Grazing Rate*, which expands to one of *Lotka-Volterra*, *Ivlev*, or any of twelve others. In the figure, the dotted lines represent mutually exclusive branches in the tree and the solid lines are required ones. HIPM supports optional processes, but this library specifies only required ones. When combined with entities representing two nutrients, a primary producer, a grazer, a detrital pool, and the general environment, this library leads to 1,120 unique model structures. To illustrate how the structural space grows, specifying *Remineralization* as optional would double the number of candidate models.

To induce models, HIPM performs two types of search—one symbolic, the other numeric. The symbolic search systematically generates model structures, such as the one in Table 1, that conform to the hierarchical constraints. By default, HIPM performs an exhaustive search, but when optional processes exist in the library, the program can apply a beam search that refines acceptable structures by adding a single instantiated option at a time. After generation, each candidate structure passes through a parameter estimation routine, which uses the data of observed variables to determine the values of the model constants. Since most models considered by HIPM are nonlinear, this step is non-trivial and requires a significant amount of computation (generally over 99% of the total computation time).

Table 2: The logical predicates for describing a generic process library.

process(generic_process)
 process_type(process_type)
 process_is_a(generic_process, process_type)
 entity(generic_entity)

The program currently uses the least-squares algorithm developed by Bunch et al. [4] coupled with random restarts in the parameter space. The learning system can optimize the sum of squared error or the relative mean squared error, which provides normalization for variables of different scales. In the next section, we describe how to extract generalizable domain knowledge from the candidate models evaluated during search.

3. EXTRACTING KNOWLEDGE FROM CANDIDATE MODELS

Here we introduce a knowledge extraction method that learns high-level constraints by inducing classifiers for candidate structures. The program’s input consists of three main elements: a description of the relevant process library, descriptions of each of the models, and background predicates for interpreting and relating the descriptions. As output, the program produces Horn clauses where the consequent asserts that a matching structure is either accurate or inaccurate. Given these rules, one can present the output to domain experts for validation. After describing the logical formalism, we discuss the learning task in greater detail.

3.1 Logical Representation

Table 2 shows the predicates associated with the library, which define the relations *process*, *process.type*, *process.is.a*, and *entity*. Using these predicates, one can describe most, but not all, of the structural information in the library. For instance, the formalism says nothing about the roles that generic entities play in the processes. However, each instantiated process produced by HIPM contains only those entities that match the associated role information captured in the library. Similarly, we leave out information about the AND branches in the hierarchy, since every model will respect these constraints as well.

In addition to the library predicates, we created those shown in Table 3 for instantiated process models. These relations differ from those for generic processes in that they must connect the processes to individual models, and they must associate the entities with process instantiations. To address these requirements, we label every model and every process with a unique identifier using the *model* and *process.instance* predicates, respec-

Table 3: The logical predicates for describing an instantiated model.

<code>model_instance(model_id)</code>
<code>process_instance_id(process_id)</code>
<code>entity_instance(entity_instance)</code>
<code>instance_of(entity_instance, generic_entity)</code>
<code>process_instance(model_id, process_id, generic_process)</code>
<code>parameter(model_id, process_id, entity_instance)</code>
<code>performance(model_id, metric, numeric_value)</code>

tively. Just as the *process_instance* predicate associates an instantiated process with a particular model, the *parameter* predicate associates an entity instance with a process within a model.

Although the predicates from Tables 2 and 3 suffice to describe the structure both of generic process libraries and of models, our experience has shown that they lead to knowledge that is difficult to interpret. To increase comprehensibility, we introduced higher level predicates that define properties of the models in terms of their components. Presented in Table 4, these predicates combine the lower level ones to explicitly infer that a model includes an instance of a particular generic process, that it includes a process having a specific type, and so forth. For example, the predicate *includes_process*(*m1*, *ivlev*) asserts the conjunction *model_instance*(*m1*) \wedge *process*(*ivlev*) \wedge *process_instance*(*m1*, *_, ivlev*) where the underscore is an anonymous variable in Prolog. The definition of the explicit negation of this predicate, *does_not_include_process*(*model_id*, *generic_process*) looks similar except that *process_instance* is enveloped by *not*(*.*).

Whereas most of the predicates describe model structure, the *performance* relation serves a different purpose. During knowledge extraction, the system uses this value to determine the threshold for labeling models as either accurate or inaccurate. In particular, we assign models to one of these two classes with the rules *accurate_model*(*model_id*) $:-$ *performance*(*model_id*, *r2*, *X*) \wedge (*X* \geq *threshold*) and *inaccurate_model*(*model_id*) $:-$ *performance*(*model_id*, *r2*, *X*) \wedge (*X* $<$ *threshold*). By explicitly associating the performance measures with a model, we can alter the threshold more easily and explore the sensitivity of the knowledge produced to the threshold value.

As HIPM generates models, it produces various measures of fitness for each one by comparing simulation results with the training data. In the context of this paper, we measure performance with the coefficient of determination (r^2), which falls in the [0,1] interval and

Table 4: The domain-general predicates for model interpretation. Each predicate also has an explicitly defined negation.

<code>includes_process(model_id, generic_process)</code>
<code>includes_process_type(model_id, process_type)</code>
<code>includes_entity_instance(model_id, entity_instance)</code>
<code>includes_entity(model_id, generic_entity)</code>
<code>includes_process_entity_instance(model_id, generic_process, entity_instance)</code>
<code>includes_process_entity(model_id, generic_process, generic_entity)</code>
<code>includes_process_type_entity_instance(model_id, process_type, entity_instance)</code>
<code>includes_process_type_entity(model_id, process_type, generic_entity)</code>

indicates how well the shapes of the simulated behaviors match those of the observed trajectories.

3.2 Generalization

For knowledge extraction, we require examples labeled as either accurate or inaccurate. To generate these examples, we run an inductive process modeler and store the logical representation of every model considered during search along with its r^2 score. We then run a program that identifies candidate thresholds for labeling. Intuitively, we select thresholds by viewing the models ranked by their accuracy and drawing a line between distinct clusters. We automate this procedure with software that ranks the models according to their r^2 values and divides them into ten bins. The program then selects the point of maximal gradient in each bin. If two such points are close to each other, we keep only one as a candidate threshold. Figure 2 displays the results of this procedure on a set of models learned in the course of the experiments described in the next section.

After generating the examples, we can use a standard implementation of inductive logic programming [17] to extract domain knowledge. For instance, the system could learn the Horn clause, *accurate_model*(*M*) $:-$ *includes_process*(*M*, *photoinhibition*) and report the number of positive and negative examples covered by this rule. From the perspective of learning constraints, one can prune the model space by only considering accurate models or by filtering out inaccurate ones. As we mentioned in Section 2, HIPM spends roughly 99% of its time fitting the parameters of each candidate structure, so pruning models based on their structure alone substantially reduces runtime. Of equal importance, these rules can identify those process combinations that best explain observed system dynamics.

4. LEARNING CONSTRAINTS FOR AQUATIC ECOSYSTEMS

We applied the approach described in Section 3 to an ecological domain. Specifically, we considered the Ross Sea ecosystem, which includes the phytoplankton species *Phaeocystis antarctica* as the dominant primary producer, some of its required nutrients, and the zooplankton that graze on it. Next we describe the experimental methodology, which includes details of the data and the learning system. Afterward, we present an ecological analysis of some of the extracted knowledge.

4.1 Experiments

To generate the model structures for analysis, we gave HIPM two separate data sets from the Ross Sea ecosystem, which is in the Southern Ocean, and the generic process library summarized in Figure 1. In addition we instantiated entities for phytoplankton, zooplankton, nitrate, iron, and the physical environment, which includes properties for light and temperature. The data cover phytoplankton blooms that occurred during the austral summer seasons of 1996–1997 and 1997–1998. We used average daily values taken from satellite measurements for three exogenous variables: photosynthetically usable radiation (light), temperature, and ice coverage all at the surface level. We also used data from three cruises that recorded the amount of chlorophyll *a*, which is assumed to be mostly *P. antarctica*, and nitrate. Unlike the satellite measurements, these data were smoothed and interpolated to give average daily values of the two system variables.

Given these five trajectories, we ran HIPM in its exhaustive search mode. The library contained no optional processes, and we required each model to use all of the five instantiated entities. In addition, we allowed 32 random restarts for the parameter estimation routine and had it minimize the relative mean squared error. HIPM produced 1,120 models for each of the two data sets. Figure 2 shows the distribution of the r^2 values for the 1996–1997 data; the corresponding figure for 1997–1998 looks similar.

We enlisted the Aleph [25] system for inductive logic programming to extract knowledge from the models. As input, we provided the generic process library encoded as described in Section 3.1. We also provided both collections of models produced by HIPM labeled according to the three highest r^2 thresholds. This procedure yielded six total example sets for knowledge discovery. We then applied Aleph to each of these sets using the program’s default parameters, with a few exceptions. Specifically, we increased the *noise* parameter to 10, which lets a rule cover up to 10 negative examples; we increased the *minpos* parameter to 2 so that the program would not add ground facts to the theory; and we increased the *nodes* parameter to 10,000, which enables a greater search space complexity. As output,

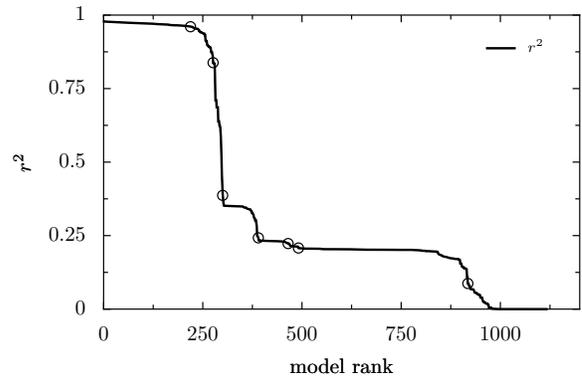


Figure 2: This figure shows the distribution of models with respect to r^2 along with proposed thresholds, as circles, for class assignment. HIPM induced these models for the 1996–1997 Ross Sea data.

the program reported relational theories that classified models as accurate or inaccurate. We reinterpret the individual clauses as general characteristics of the two classes that impose constraints on the structures.

4.2 Resulting Rules and Analysis

Our experiments produced a total of 28 unique rules of which 7 were both well supported by the examples and judged scientifically interesting by an ecologist familiar with the Ross Sea (i.e., Stuart Borrett). In this section, we list these rules along with their positive and negative coverage and describe their ecological relevance.

Rule 1: A model that includes a second order exponential mortality process for phytoplankton will be inaccurate.¹ (positive:560, negative: 0)

Steele and Henderson proposed the second order exponential mortality process [26] as a formulation to capture both the natural mortality of a population as well as predation on that population that their ecosystem models did not explicitly include [7]. Rule 1 suggests that this formulation is inappropriate for the phytoplankton in our system, which is logical because we are explicitly modeling its grazing mortality.

Rule 2: A model that includes the Lotka–Volterra grazing formulation will be inaccurate. (positive: 80, negative: 0)

¹We give a natural language interpretation of the rules that corresponds to the induced Horn clauses. With very little orientation, the ecologist understood the Horn clause formalism. As an example, the original form of Rule 1 was `inaccurate(M) :- includes_process_entity(M, death_exp2, phytoplankton)`.

Rule 3: A model that includes the generalized Lotka–Volterra grazing formulation will be inaccurate. (positive: 76, negative: 8)

The Lotka–Volterra grazing formulation assumes a linear increase in the predator’s consumption rate as prey density increases, which means that a predator’s capacity to consume additional prey is never limited. Holling [13] termed this relationship between grazing rate and prey density a Type I functional response. The generalized Lotka–Volterra form expands the original alternative with an exponent that controls whether the Type I functional response is concave or convex. With this context in mind, Rules 2 and 3 suggest that models with an asymptotic functional response better explain the observed Ross Sea dynamics. This finding coheres with biological evidence that *P. antarctica*, the dominant component of the phytoplankton community, is difficult for grazers to consume [11, 24].

Rule 4: A model that includes the Hassell–Varley I grazing formulation will be inaccurate. (positive: 74, negative: 4)

Rule 5: A model that includes the Hassell–Varley II grazing formulation will be inaccurate. (positive: 76, negative: 8)

The role of predator interference with their own ability to graze prey is a long running debate in the ecological literature [12, 15, 1]. Seven of the fourteen alternative grazing processes in the ecosystem library explicitly model predator interference and include the two Hassell–Varley type formulations as presented by Jost and Ellner [15]. Rules 4 and 5 state that these two grazing alternatives are inappropriate for the Ross Sea data. However, we did not find rules that exclude the remaining five formulations with predator interference.

Rule 6: A model that does not include the first or second order Monod formulation of iron limitation of phytoplankton growth will be inaccurate. (positive: 448, negative: 0)

The generic process library contains three alternatives for limiting phytoplankton growth via nutrients. Rule 6 strongly implies that the accurate models include an iron limitation of phytoplankton growth expressed as a type of Monod function, effectively excluding the third formulation. Although we lack a clear biological reason to expect one formulation over another in this case, there is substantial empirical evidence to support the hypothesis that iron is the nutrient resource that most limits phytoplankton growth in the Ross Sea [19, 22].

Rule 7: A model that does not include the second order exponential mortality process (for either the phytoplankton or zooplankton entities) and does include a Monod type growth limitation by iron will be accurate. (positive: 74, negative: 10)

Rule 7 differs from the first six in that it states conditions that tend to make a model accurate rather than inaccurate. The first conjunct excludes the second order mortality process from the model for either phytoplankton or zooplankton. Thus, this assertion generalizes the constraint from Rule 1 and the same biological reasoning applies for its application to the phytoplankton. The extension to zooplankton implies that the first order natural mortality process in conjunction with the zooplankton respiration process sufficiently accounts for all necessary zooplankton loss. However, this finding appears to generalize only when the basic Monod process limits phytoplankton growth by iron. As we described in Rule 6, iron limitation of phytoplankton growth is biologically reasonable.

Interestingly, four of the rules we discovered exclude one of the fourteen grazing formulations. This finding is surprising because our previous experience in this domain and other research on this ecosystem [27] suggest that grazing should be relatively unimportant to the overall system dynamics. We intend to explore this result further, and we suspect that the excluded process alternatives generate more grazing activity than the observations can allow.

The preceding analyses are thick with domain relevant language, but they primarily indicate that the selected rules make sense from an ecological standpoint. Importantly, we make no claims regarding the verity of the extracted knowledge, only that it is supported both by the results of an intensive modeling exercise and by appeal to related findings about the modeled organisms and ecosystem. Nevertheless, the approach that we present revealed unexpected characteristics of the models that warrant further examination. Moreover, we have shown elsewhere [2] that the extracted rules make plausible constraints on future modeling problems that reduce the search space by an order of magnitude.

5. DISCUSSION

Since we apply a learning algorithm to analyze the output of an induction system, our work falls within the scope of metalearning [10]. However, most existing metalearning research takes as its goal, the ability to predict how classification algorithms will perform so that one can select an appropriate learning system or assemble an optimal ensemble of classification algorithms. Our goal differs from this in that we endeavor to extract information about the application domain that may serve

as a novel discovery or as a new search constraint. That is, we would like a learning system to develop a richer understanding of the problems that it solves, whereas the metalearning research that we described could give the system a better understanding of its capabilities.

Other research shares a similar goal to our own. For instance, McCreath and Sharma [20] used inductive logic programming to learn meta-level information about the background knowledge used by systems for inductive logic programming. Their goal was to specify, or constrain, the ways in which one can integrate a predicate into the clauses thereby restricting the search space. Notably, their program produced syntactic constraints that were independent from any specific domain. In contrast, our approach induces semantic constraints that are interpretable within the context of the domain. Coming from a different field, research on learning control rules for planning [14] also reflects our goal. That is, the developed algorithms analyze the output of a planning system to improve its future performance. One important distinction is that such systems extract knowledge from individual operators and the context of their application instead of analyzing the complete structure of the plan.

These related endeavors suggest that our approach to knowledge extraction has relevance beyond inductive process modeling. For example, one could adapt the presented system to inductive logic programming by treating each considered clause as an individual model and identifying structural characteristics of those that have high predictive accuracy. To generalize, we believe that our system can extract knowledge from any application that searches a space of model structures as long as it evaluates and stores each one considered.

Beyond incorporating our work into other artificial intelligence tools, there is substantial room for future research. First, our colleagues in ecology would like to know the commonalities among the most accurate models. To meet this request, we need either an inductive logic programming system that generalizes from positive examples only [21] or a method for mining frequent patterns in relational data [6]. Second, we should investigate the effects of iteratively extracting constraints and using them to limit induction. We expect that we will need a method for creating and updating soft constraints to avoid excessive rigidity in the inductive learner. Third, since process models also contain numeric parameters, we would like to modify our approach to identify meaningful bounds on their values.

In addition to the above, we note Pazzani and Kibler's [23] research that shows how using domain knowledge to bias search can both reduce overfitting and improve predictive accuracy. With this finding in mind, we should determine whether automatically induced knowledge can

lead to the same benefits. Due to the problem of overfitting, we should also compare our current system with one that judges a model's accuracy on separate test data. We expect that the form and quality of the constraints will change but are uncertain about the degree.

In this paper, we showed how to extract domain knowledge from the models considered during induction. One can use this knowledge to characterize the domain, to constrain future modeling tasks [2], and to uncover novel relationships. We have also shown that this approach can lead to interesting and plausible generalizations in the context of inductive process modeling, and we have suggested its applicability to other areas of artificial intelligence. We believe that this research can strengthen the field as a whole by enabling intelligent systems both to improve themselves over time within a problem domain and eventually to transfer knowledge across related domains.

6. ACKNOWLEDGMENTS

This research was supported by Grant No. IIS-0326059 from the National Science Foundation. We thank Kevin Arrigo and Gert van Dijken for the use of the Ross Sea data and their expertise in this ecosystem. We also thank Pat Langley and Tolga Könik for discussions relevant to the paper.

7. REFERENCES

- [1] R. Arditi and L. R. Ginzburg. Coupling in predator-prey dynamics — ratio-dependence. *Journal of Theoretical Biology*, 139:311–326, 1989.
- [2] W. Bridewell and L. Todorovski. Learning declarative bias. *Proceedings of the Seventeenth International Conference on Inductive Logic Programming*, to appear, 2007.
- [3] W. Bridewell, J. Sánchez, P. Langley, and D. Billman. An interactive environment for the modeling and discovery of scientific knowledge. *International Journal of Human-Computer Studies*, 64:1099–1114, 2006.
- [4] D. S. Bunch, D. M. Gay, and R. E. Welsch. Algorithm 717: Subroutines for maximum likelihood and quasi-likelihood estimation of parameters in nonlinear regression models. *ACM Transactions on Mathematical Software*, 19:109–130, 1993.
- [5] S. Cohen and A. Hindmarsh. CVODE, a stiff/nonstiff ODE solver in C. *Computers in Physics*, 10:138–143, 1996.
- [6] L. Dehaspe and H. Toivonen. Discovery of relational association rules. In *Relational Data Mining*, pages 189–212. Springer, Berlin, Germany, 2001.

- [7] A. M. Edwards. Adding detritus to a nutrient–phytoplankton–zooplankton model: A dynamical systems approach. *Journal of Plankton Research*, 23:389–413, 2001.
- [8] B. Falkenhainer and K. D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.
- [9] S. Fröhler and S. Kramer. Logic-based information integration and machine learning for gene regulation prediction. In *Proceedings of the Ninth International Conference on Molecular Systems Biology*, 2006. Available online at <http://www.icmsb06.org/>.
- [10] C. Giraud-Carrier, R. Vilalta, and P. Brazdil. Introduction to the special issue on meta-learning. *Machine Learning*, 54:187–193, 2004.
- [11] K. L. Haberman, R. M. Ross, L. B. Quetin, M. Vernet, G. A. Nevitt, and W. Kozłowski. Grazing by antarctic krill *Euphausia superba* on *Phaeocystis antarctica*: An immunochemical approach. *Marine Ecology Progress Series*, 241:139–149, 2002.
- [12] M. P. Hassell and G. C. Varley. New inductive population model for insect parasites and its bearing on biological control. *Nature*, 223:1133–1137, 1969.
- [13] C. S. Holling. The components of predation as revealed by a study of small mammal predation of the European pine sawfly. *Canadian Entomologist*, 91:293–320, 1959.
- [14] Y. Huang, B. Selman, and H. A. Kautz. Learning declarative control rules for constraint-based planning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 415–422, Stanford, CA, 2000. Morgan Kaufmann.
- [15] C. Jost and S. P. Ellner. Testing for predator dependence in predator–prey dynamics: A non-parametric approach. *Proceedings of the Royal Society of London Series B-Biological Sciences*, 267:1611–1620, 2000.
- [16] P. Langley, J. Sánchez, L. Todorovski, and S. Džeroski. Inducing process models from continuous data. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 347–354, Sydney, 2002. Morgan Kaufmann.
- [17] N. Lavrac and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood, New York, 1994.
- [18] A. Mahidadia and P. Compton. Assisting model-discovery in neuroendocrinology. In *Proceedings of the Fourth International Conference on Discovery Science*, pages 214–227, Washington DC, 2001. Springer.
- [19] J. H. Martin, R. M. Gordon, and S. E. Fitzwater. The case for iron. *Limnology and Oceanography*, 36:1793–1802, 1991.
- [20] E. McCreath and A. Sharma. Extraction of meta-knowledge to restrict the hypothesis space for ILP systems. In *Proceedings of the Eighth Australian Joint Conference on Artificial Intelligence*, pages 75–82, Canberra, Australia, 1995. World Scientific Publishers.
- [21] S. Muggleton. Learning from positive data. In *Proceedings of the Sixth International Workshop on Inductive Logic Programming*, pages 358–376, Stockholm, Sweden, 1996. Springer.
- [22] R. J. Olson, H. M. Sosik, A. M. Chekalyuk, and A. Shalapyonok. Effects of iron enrichment on phytoplankton in the Southern Ocean during late summer: Active fluorescence and flow cytometric analyses. *Deep-Sea Research Part II-Topical Studies in Oceanography*, 47:3181–3200, 2000.
- [23] M. J. Pazzani and D. F. Kibler. The utility of knowledge in inductive learning. *Machine Learning*, 9:57–94, 1992.
- [24] V. Schoemann, S. Becquevort, J. Stefels, W. Rousseau, and C. Lancelot. Phaeocystis blooms in the global ocean and their controlling mechanisms: A review. *Journal of Sea Research*, 53:43–66, 2005.
- [25] A. Srinivasan. *The Aleph Manual*. Computing Laboratory, Oxford University, 2000.
- [26] J. H. Steele and E. W. Henderson. A simple plankton model. *American Naturalist*, 117:676–691, 1981.
- [27] A. Tagliabue and K. R. Arrigo. Anomalously low zooplankton abundance in the Ross Sea: An alternative explanation. *Limnology and Oceanography*, 48:686–699, 2003.
- [28] L. Todorovski, W. Bridewell, O. Shiran, and P. Langley. Inducing hierarchical process models in dynamic domains. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, pages 892–897, Pittsburgh, PA, 2005. AAAI Press.
- [29] J. M. Żytkow. Model construction: Elements of a computational mechanism. In *AISB’99 Symposium on AI and Scientific Creativity*, Edinburgh, Scotland, 1999.