Inducing Hierarchical Process Models in Dynamic Domains

Ljupčo Todorovski,[1,2] **Will Bridewell,**[1] **Oren Shiran,**[1] and **Pat Langley**[1]
[1]Computational Learning Laboratory, CSLI, Stanford University, Stanford, USA
[2]Department of Knowledge Technologies, Jožef Stefan Institute, Ljubljana, Slovenia
ljupco.todorovski@ijs.si, {willb@csli, oren@sccm,langley@csli}.stanford.edu

Abstract

Research on inductive process modeling combines background knowledge with time-series data to construct explanatory models, but previous work has placed few constraints on search through the model space. We present an extended formalism that organizes process knowledge in a hierarchical manner, and we describe HIPM, a system that carries out constrained search for hierarchical process models. We report experiments that suggest this approach produces more accurate and plausible models with less effort. We conclude by discussing related research and directions for future work.

Introduction

Scientists spend much of their time constructing *models* that explain observed phenomena in terms of established theoretical concepts. This claim holds especially for researchers in integrative fields like Earth science, who attempt to understand the behavior of complex systems as interactions among their parts. Although research on computational scientific discovery has a long history within AI (Langley *et al.* 1987), there has been relatively little effort on techniques for inferring such explanatory models.

Recently, some researchers (Langley *et al.* 2003; Todorovski 2003) have argued that processes play a central role in many scientific accounts, and that the induction of quantitative process models is an important task for the field. Their initial systems combine domain knowledge with time-series data to construct accurate and meaningful differential equation models in a variety of environmental, biological, and engineering domains. However, analyses suggest that their methods' operation would benefit from additional constraints about the space of plausible models, both to lower variance and to reduce computation time.

We extend these earlier efforts by incorporating the notion of hierarchical processes that characterize an observed system's behavior at distinct levels. Providing a model induction method with such hierarchical knowledge lets it carry out search through an AND/OR space rather than an OR space, thus reducing the number of candidate models considered and ensuring that these models will make sense to domain scientists. This approach requires slightly greater effort to encode background knowledge, but we anticipate it will be offset by more efficient search, more plausible models, and better fits to data.

In the next section, we review the previous work on inductive process modeling, motivate the need for a hierarchical representation, and present an expanded framework that incorporates this idea. After this, we describe HIPM, a system that searches a space of hierarchical process models constrained by background knowledge and observations. We then report experimental studies, with both synthetic and natural data, that evaluate the system's ability to induce accurate and plausible scientific models. In closing, we discuss related work and identify directions for additional research.

Quantitative Process Models

We live in a world where entities participate in activities that instigate change in the participants' states. Any means for modeling change over time must address this fundamental relationship. In equations, variables represent the state of the entities, while mathematical operators encode the interactions. Although this structure provides a powerful method for representing such change, it forfeits information about the higher-level relationships among terms in return for generality of application. Thus, mathematical equations serve better as a base language upon which we can build models of complex dynamic systems.

Previous Work on Process Modeling

Following Forbus (1984), we represent models as a collection of variables and processes. Each process includes at least one differential or algebraic equation that relates the values of participating variables. The major conceptual shift involves seeing the equations within the processes as explicitly grouped terms of a larger system of differential and algebraic equations. Table 1 shows the relationship between the process and the system of equations representations for a predator–prey model. Notice how the processes give meaning to the terms, especially in the case of predation, which shows that the terms in the right-hand side intertwine.

In addition to clarifying a model's meaning, processes provide a way to constrain model induction. Instead of searching through the space of all possible equations, an induction system can piece together only the meaningful terms. One advantage of this approach is that a program can more readily determine the plausibility of equation forms; for example, a model of predator–prey interaction must have a term specifying the growth rate of the prey. Another advantage is that grouped terms stand or fall together. For instance, because predation affects both participating popula-

Table 1: A differential equation model of predator–prey interactions and a corresponding quantitative process model.

$\frac{d}{dt}prey = 2.5 * prey + 0.3 * 0.1 * prey * pred$

$\frac{d}{dt}pred = -1.2 * pred - 0.1 * prey * pred$

process prey_growth
  equations $d[prey, t, 1] = 2.5 * prey$
process pred_loss
  equations $d[pred, t, 1] = -1.2 * pred$
process predation
  equations $d[prey, t, 1] = -0.1 * prey * pred$
$d[pred, t, 1] = 0.3 * 0.1 * prey * pred$

tions, we need not consider any model of predation wherein the predator kills, but does not eat, the prey. Of course, we do not always know the variable interactions or the system parameters, so we would like to search the space of plausible process models to identify such relationships. To facilitate this task, an induction system should employ background knowledge cast as *generic processes*.

These generic processes resemble their instantiated counterparts, except that they contain placeholders for variables, which have type constraints, and parameters, which have upper and lower bounds. Table 2 shows a set of generic processes related to the specific processes from Table 1. Notice that the type restrictions in the predation process will keep an induction system from suggesting implausible models in which the prey devours the predator. This constraint illustrates the power of the process model representation, which can avoid such missteps.

The IPM procedure (Langley *et al.* 2003) for inducing process models searches concurrently through the space of model structures and parameter instantiations. This approach has produced interesting and useful models of battery charge and discharge cycles, aquatic ecosystems, and fjord hydrodynamics. The process modeling formalism not only organizes models of dynamic systems in a manner related to our beliefs about causation, but provides a powerful means for capturing domain knowledge that constrains the construction of scientific models.

Limitations of the Previous Formalism

Although the previous approach to inducing process models has proven successful on a variety of modeling tasks, two assumptions about how to combine processes into a model limit its usefulness. The first suggests that one can combine any set of generic processes to produce a valid model structure. This assumption leads to an underconstrained model space containing many candidates that violate domain experts' expectations. The second portrays all process influences as additive, which is unrealistic in some domains. We can illustrate the problems raised by these assumptions with an example from population dynamics.

Consider an aquatic ecosystem in which a single plankton species depends on two inorganic nutrients—nitrate and phosphorus. A human expert would expect a well-formed

Table 2: Generic processes for predator–prey interaction. Both *predator* and *prey* are types of *population*.

generic process exponential_growth
  variables $P\{population\}$
  parameters $g[0, inf]$
  equations $d[P, t, 1] = g * P$
generic process exponential_loss
  variables $P\{population\}$
  parameters $g[0, inf]$
  equations $d[P, t, 1] = -1 * g * P$
generic process predation_h1
  variables $P1\{prey\}$, $P2\{predator\}$
  parameters $r[0, inf]$, $e[0, inf]$
  equations $d[P1, t, 1] = -1 * r * P1 * P2$
$d[P2, t, 1] = e * r * P1 * P2$

model to include a process for plankton growth, whereas IPM would consider models that omit it. The expert would also treat some processes as mutually exclusive, such as exponential and nutrient-limited plankton growth, whereas the system would consider models that include both processes. Finally, the expert would understand that the effects of nitrate and phosphorus limits on plankton growth should be multiplied, whereas IPM would assume they are added. As a result, the program would search a much larger space of model structures than necessary and might return candidates that seem ill formed to the human modeler.

Hierarchical Process Models

To overcome these limitations, we designed an extended formalism that supports hierarchical process models like the one in Table 3. This model specifies that two high-level processes influence the change in plankton concentration. The first, *limited_growth*, states that the concentrations of two nutrients limit plankton growth, whereas *exponential_loss* characterizes the loss of plankton. To indicate the specific limitations the nutrients impose on plankton growth, the model incorporates two subprocesses having type *holling_type_1*, one for each nutrient.

The model in Table 3 is organized not as a simple set of processes, but as a hierarchy that reflects domain-specific rules about models in population dynamics. One such rule, applied at the top level, specifies that change in plankton concentration results from the processes of both growth and loss. The next level has another rule that specifies the type of each limiting factor. For example, any model structure that includes *limited_growth(phyto, {nitrate, phosphate})* but lacks one of the *holling_type_1* processes would be deemed incomplete.

In the extended formalism, generic processes encode the knowledge about model structure. Table 4 presents a simplified hierarchy of generic processes for modeling population dynamics. The top node of the hierarchy, *primary_concentration_change*, relates one primary-producer population *P* to a set of nutrients *N*, where the optional qualifier *<0 to inf>* gives the lower and upper bounds of this set's cardinality. The third line of the first process states that changes in

Table 3: A hierarchical process model of an aquatic ecosystem wherein phytoplankton ($phyto$) depends on the nutrients $nitrate$ and $phos$.

process primary_concentration_change($phyto$, $\{nitrate, phos\}$)
  process limited_growth($phyto$, $\{nitrate, phos\}$)
    equations $d[phyto.conc, t, 1] = 2.3 * phyto.conc * phyto.limit\_rate$
    $d[nitrate.conc, t, 1] = -1.2 * phyto.conc * phyto.limit\_rate * 2.3$
    $d[phos.conc, t, 1] = -0.9 * phyto.conc * phyto.limit\_rate * 2.3$
  process holling_type_1a($nitrate$, $phyto$)
    equations $phyto.limit\_rate = nitrate.conc$
  process holling_type_1b($phos$, $phyto$)
    equations $phyto.limit\_rate = phos.conc$
  process exponential_loss($phyto$)
    equations $d[phyto.conc, t, 1] = -1.2 * phyto.conc$

primary concentration result from processes of growth and loss, with the latter being optional. Thus, a plausible model of primary-producer change includes exactly one growth process and at most one loss process. Similarly, lower in the hierarchy, the process *limited_growth* indicates that exactly one *limiting_factor* must exist for each specified nutrient.

This structural hierarchy interleaves with a process type, or is-a, hierarchy. For example, the process identifier *growth* refers to a generic process type that includes two alternatives: *exponential_growth* and *limited_growth*. The type hierarchy can specify mutually exclusive modeling alternatives. For example, an induction system can use either the *exponential_growth* or *limited_growth* process to model the primary producer's growth, but not both.

Thus, the interleaved structure–type hierarchy of generic processes facilitates placing two kinds of constraints on the space of candidate models. Using process types, we can specify mutually exclusive modeling alternatives. Using a subprocess hierarchy, we can define a correct model structure in terms of the minimal set of necessary component submodels. This organization contrasts with IPM's "flat" collection of generic processes, which could be combined arbitrarily into candidate model structures.

Finally, note that Table 4 also replaces variables with entities that group properties of the observed organisms and nutrients. Entities can have two kinds of properties: variables, which can change over time (such as the concentration *conc*), and parameters, which describe constant aspects (e.g., *loss_rate*) of an entity. Process influences on entity variables can be combined using different aggregation functions. For example, influences on concentration variables are added, whereas influences on limitation rates are multiplied.

Inducing Hierarchical Process Models

One previous method for inducing process models, IPM, employs constrained exhaustive search through the space of candidate model structures (Langley *et al.* 2003). That system operates in three stages, the first of which creates

Table 4: A simple hierarchy of generic processes for modeling population dynamics.

entity primary_producer
  variables $conc\{add\}$, $limit\_rate\{multiply\}$
  parameters $max\_growth[0, inf]$, $loss\_rate[0, inf]$
entity nutrient
  variables $conc\{add\}$
  parameters $toCratio[0, inf]$
process primary_concentration_change
  relates $P\{primary\_producer\}$, $N\{nutrient\}$ <0 to inf>
  processes $growth(P, N)$, $[loss(P)]$
process exponential_growth$\{growth\}$
  relates $P\{primary\_producer\}$
  equations $d[P.conc, t, 1] = P.max\_growth * P.conc$
process limited_growth$\{growth\}$
  relates $P\{primary\_producer\}$, $N\{nutrient\}$ <0 to inf>
  processes limiting_factor(N, P)
  equations $d[N.conc, t, 1] = -1 * N.toCratio * P.conc * P.limit\_rate * P.max\_growth$
  $d[P.conc, t, 1] = P.conc * P.limit\_rate * P.max\_growth$
process holling_type_1$\{limiting_factor\}$
  relates $N\{nutrient\}$, $P\{primary\_producer\}$
  equations $P.limit\_rate = N.conc$
process exponential_loss$\{loss\}$
  relates $P\{primary\_producer\}$
  equations $d[P.conc, t, 1] = -1 * P.loss\_rate * P.conc$

instantiated processes by matching the types of the problem variables with the placeholders in each generic process, then generating all possible assignments. The second stage comprises searching through a space of limited cardinality subsets of these process instances, where each subset corresponds to a candidate model structure. Using these structures, IPM invokes a nonlinear optimization method to fit the parameters, producing a model score that reflects the discrepancy between observed and simulated data. Finally, IPM outputs the parameterized model with the lowest score.

Exhaustively generating all possible subsets of process instances can be prohibitive, even for a relatively small number of variables. In a two-fold response, the HIPM algorithm uses heuristic beam search and knowledge-guided refinement operators to navigate the model space more effectively. The system takes as input a hierarchy of generic processes, a set of entities with associated variables, a set of observed trajectories $\mathcal{T}$ of the entities' variables, and a parameter b that specifies the beam width.

On each beam-search iteration, HIPM refines the current selection of models by one step, adding the nonredundant structures back to the beam. In the first iteration, the system generates all permitted models that exclude any optional processes. For instance, when given one *primary_producer*, two *nutrients*, and the generic process library from Table 4, HIPM would produce two models, one containing *exponential_growth* and the other containing *limited_growth* with associated *limiting_factor* subprocesses for both nutrients. In subsequent iterations, the refinement operator would add a

single optional process to the current model structure, which may require the addition of multiple processes, depending on the background knowledge. In the above example, HIPM would expand the two initial models to contain an instantiation of *exponential_loss*.

Once HIPM refines the model structures in the beam, it fits their parameters to the training data. Past work (Langley *et al.* 2003) searched through the parameter space using a gradient descent method that evaluated each set of candidate parameters using model simulation over the full time span of $\mathcal{T}$. HIPM extends the previous approach by incorporating *teacher forcing* (Williams & Zipser 1989), which finds parameters that best predict observations $\mathcal{T}_{i+1}$ solely from those at $\mathcal{T}_i$. In both the teacher forcing and full simulation stages of parameter fitting, it employs a nonlinear least-squares algorithm (Bunch *et al.* 1993) that carries out second-order gradient descent. To avoid entrapment in local minima, HIPM does eight restarts with different initial parameter values, one based on the results from teacher forcing. Finally, the procedure selects the best set of parameters for each model structure, based on the sum of squared errors.

In the last step of each refinement iteration, HIPM retains the best b models as determined by their sum of squared errors. The search ends when a further iteration fails to alter the beam contents. At this point, the program returns the parameterized model with the best error score.

Experimental Evaluation

We conjecture that using hierarchical knowledge for process model induction will improve the efficiency of search, increase the plausibility of induced models, and improve the models' predictive performance. We evaluated these hypotheses empirically by applying HIPM to three modeling tasks related to population dynamics. In each experiment, we compared the plausibility and predictive performance of the models induced using hierarchal generic processes to those induced using flat structures. To ensure a fair comparison, we used HIPM in both conditions rather than the older IPM, which uses a different parameter estimation routine. To mimic the latter's behavior, we removed higher-level processes from the inputs, providing only low-level ones.

We measured the predictive performance of an induced model in terms of the discrepancy between the values of observed system variables and the values obtained by simulating the model. To this end, we used the average of the correlation coefficient (r^2) over all the observable variables, which summarizes the amount of variance explained. In addition, we recorded the number of candidate model structures considered during model induction as a measure of the system's search efficiency.

Studies with Synthetic Data

In the first experiment, we used a hypothetical model of an aquatic ecosystem to generate five simulation traces based on five different initial conditions. The model included four entities and nine processes. We sampled each simulated trajectory at 100 equidistant time points with a time step of 1.5. To make the data more realistic, we introduced noise by replacing each simulated value $\mathcal{T}_i$ with $\mathcal{T}_i \cdot (1+0.05 \cdot x)$, where x is a Gaussian random variable with mean 0 and variance 1. To evaluate the generalization performance of the models on novel test data, we used five-fold cross validation. Each training set comprised four of the complete simulation traces described above, whereas each test set consisted of the noise-free version of the fifth trace.

Table 5: Predictive performance (in terms of correlation coefficient, r^2) of the models induced by HIPM on synthetic training (TRAIN) data with 5% Gaussian noise and noise-free test data using hierarchical (H) and flat (F) generic processes and two beam widths (BW). The #CMS column presents the number of candidate structures considered.

BW	LIB	#CMS	TRAIN r^2	CV-TEST r^2
4	H	57.0±1.6	0.98±0.01	0.99±0.01
	F	416.8±18.4	0.99±0.01	0.93±0.16
8	H	99.0±2.3	0.99±0.00	0.99±0.02
	F	797.8±27.5	0.98±0.01	0.91±0.16

Table 5 summarizes the results of this study. The first two rows present the cross-validation scores for the hierarchical and flat conditions with a beam width of four. Comparing efficiency measures shows that the hierarchical scheme greatly reduced the number of model structures considered during search, with the reduction factor being greater than seven. Furthermore, cross-validation estimates suggest that flat models tend to overfit—they perform slightly better than hierarchical models on the training trajectories, but they perform worse on the test data.

The results of running HIPM with a larger beam width, shown in the last two rows of the table, confirm that the hierarchical scheme reduces the number of model structures considered and avoids overfitting. As before, hierarchical models outperform flat ones on the test data. Inspecting the structure of the induced models reveals a possible explanation for this difference. Although all hierarchical models have plausible structures, none of the induced flat models are valid from a population dynamics view. Furthermore, HIPM reconstructed all the processes from the original model structure in six out of ten runs and, in the remaining four, the program induced a model that differed from the original by only a single process.

Studies with Natural Data

In the second experiment, we applied HIPM to the task of modeling population dynamics in the Ross Sea (Arrigo, Worthen, & Robinson 2003), where concentrations of the primary producer *phytoplankton* and the nutrient *nitrate* have been measured for 188 consecutive days, along with *light* levels. Ecology experts who provided measurement data specified that two unobserved entities, *zooplankton* and *residue*, are important for modeling phytoplankton growth.

We again used five-fold cross validation for evaluation. Lacking multiple data sets, we randomly selected time points from the observed trajectories using the measured values to create the equal-sized subsets and ensured that all training sets had the same initial conditions. We ran HIPM on those data, varying the structure of the back-

Table 6: Predictive performance of the models induced by HIPM on aquatic ecosystem data from the Ross Sea.

BW	LIB	#CMS	SSE	r^2
4	H	68	1577	0.68
	F	492	2126	0.53
8	H	117	1564	0.68
	F	845	1787	0.62
32	H	447	1113	0.84
	F	3572	1633	0.68

ground knowledge (hierarchical or flat) and the beam width (4, 8, or 32). The system overlaid the learned trajectories onto the test data to calculate the error scores.

Table 6 presents the results of this experiment, which complement the findings of the synthetic data study. Flat models perform worse than hierarchical ones for all beam widths, and error decreases with beam width. The most accurate hierarchical model induced from the entire training set with beam width of 32 has the same structure as the best known model of the Ross Sea aquatic ecosystem (Asgharbeygi *et al.* in press).[1] Figure 1 compares the observed and simulated trajectories produced by this model for *phytoplankton* and *nitrate* concentrations.

Finally, we considered an ecosystem composed of the microscopic species *aurelia* and *nasutum*, in which the latter preys on the former (Jost & Ellner 2000). The time-series data included 51 measurements of the two species' concentrations at twelve-hour intervals, yielding five cycles of population increase and decline. Here we used the same parameter settings for HIPM as in the previous experiment and carried out a similar five-fold cross validation.

Table 7 presents the results of this study, which shows that flat models have lower errors and higher r^2 scores than the hierarchical ones for all beam widths. When using the hierarchical library, a beam width of 32 suffices to allow an exhaustive search of the space of unique model structures. Oddly, with a beam width of 8, HIPM appears to examine an extra model. This anomaly can be attributed to the program's lack of a long-term memory. That is, multiple paths in the search space may lead to the same model, but unless a matching structure exists in the beam, HIPM has no way to remember it has been seen.

Inspection of the induced models, which we lack the space to present, revealed that none of the flat structures give a plausible explanation of the predator–prey interaction, in that they omit key processes or include mutually exclusive ones. We conjecture that the generic process library overly constrains the search space so that none of the valid model structures can fit the measurements well. We believe that enriching the library with alternative forms of the basic process types will overcome this problem, but testing this hypothesis is a matter of future work.

[1]IPM found this model, but only when it was provided with a carefully crafted set of generic processes that limited search through the model space and omitted many viable candidates.

Table 7: Predictive performance of the models induced by HIPM on predator–prey data.

BW	LIB	#CMS	SSE	r^2
4	H	18	67157	0.35
	F	120	39771	0.56
8	H	25	67157	0.35
	F	203	37371	0.59
32	H	24	67157	0.35
	F	476	37324	0.58

Concluding Remarks

Our approach to scientific model construction draws ideas from a number of earlier traditions. The most obvious connections are to research on equation discovery in scientific and engineering domains. For example, Żytkow *et al.*'s (1990) FAHRENHEIT specified its search space as a set of candidate functional forms, for which it then fit parameters. More recently, Bradley *et al.*'s (2001) PRET also utilized knowledge to constrain search for quantitative models of dynamical systems, although their framework associated classes of functions with qualitative behaviors like oscillation rather than combining domain-specific elements. Inductive process modeling also has links to work from the qualitative reasoning community on compositional modeling. For instance, Falkenhainer and Forbus (1991) report a method which constructed qualitative models by instantiating and combining model fragments that were directly analogous to our generic processes.

However, none of these earlier systems organized their background knowledge in a hierarchical manner. On this dimension, HIPM comes closest to Todorovski's (2003) LAGRAMGE, which also used hierarchically organized processes to generate differential equation models of time series. A key difference is that its induction procedure transformed the hierarchies into grammars, which precludes ordering the search space by the number of processes involved. The system also required all entities to be directly observable, which our Ross Sea example indicates is unrealistic. Thus, HIPM constitutes a significant conceptual advance over LAGRAMGE, and our experiments provide encouraging evidence of its effectiveness in terms of reduced search, decreased generalization error, and improved plausibility.

Of course, hierarchical knowledge structures have seen wide use in other branches of artificial intelligence, including natural language, planning, and vision. As in our framework, they transform extensive search through an OR space into a more constrained AND/OR search, with corresponding benefits. This idea has also appeared in other areas of machine learning. For example, methods for explanation-based learning (DeJong & Mooney 1986) created conjunctive structures from training cases only if they could be derived from background knowledge, which typically took a hierarchical form. The constraints imposed by process hierarchies also resemble those provided by declarative bias within inductive logic programming (Nédellec *et al.* 1996).

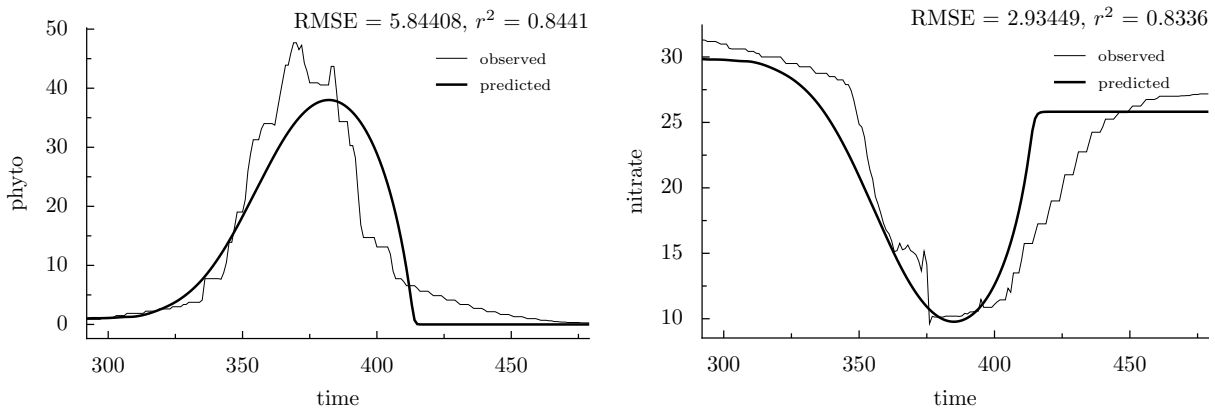

Figure 1: Comparison of observed and simulated trajectories for phytoplankton and nitrate concentrations.

Our hierarchical processes specify relations among entities and, like Horn clauses, organize them into higher-level structures that constrain model construction.

However, inductive process modeling remains a distinct enterprise from other paradigms for reasoning and learning, both in its focus on scientific domains and its concern with continuous time series. The methods we have reported in this paper constitute clear progress over earlier approaches to this problem, and we intend to make use of them in our continuing work on the topic. One promising extension would utilize knowledge about relations among parameters to reduce dimensionality of the parameter space and simplify their estimation. We also hope to make parameter fitting more tractable by decomposing this task into nearly independent subproblems, much as Chown and Dietterich (2000) did in their work on ecosystem modeling. On the structural side, we hope to incorporate more incremental methods for model revision (Todorovski 2003) which, like human scientists, are driven by observational anomalies that their current account cannot explain. Taken together, these extensions should produce an even more robust framework for the induction of scientific process models.

Acknowledgements

This research was supported by NSF Grant Number IIS-0326059. We thank Kevin Arrigo for data from and knowledge about the Ross Sea, along with Kazumi Saito and Nima Asgharbeygi for early work on the predator–prey system.

References

Arrigo, K.; Worthen, D.; and Robinson, D. 2003. A coupled ocean-ecosystem model of the Ross Sea. part 2: Iron regulation of phytoplankton taxonomic variability and primary production. *Journal of Geophysical Research* 108(C7):3231.

Asgharbeygi, N.; Bay, S.; Langley, P.; and Arrigo, K. in press. Inductive revision of quantitative process models. *Ecological Modelling*.

Bradley, E.; Easley, M.; and Stolle, R. 2001. Reasoning about nonlinear system identification. *Artificial Intelligence* 133:139–188.

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

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

DeJong, G. F., and Mooney, R. J. 1986. Explanation-based learning:An alternative view. *Machine Learning*1:145–176.

Falkenhainer, B., and Forbus, K. D. 1991. Compositional modeling: Finding the right model for the job. *Artificial Intelligence* 51:95–143.

Forbus, K. D. 1984. Qualitative process theory. *Artificial Intelligence* 24:85–168.

Jost, C., and Ellner, S. 2000. Testing for predator dependence in predator–prey dynamics: A non-parametric approach. *Proceedings of the Royal Society of London B* 267:1611–1620.

Langley, P.; Simon, H. A.; Bradshaw, G.; and Żytkow, J. M. 1987. *Scientific Discovery*. Cambridge, MA: MIT Press.

Langley, P.; George, D.; Bay, S.; and Saito, K. 2003. Robust induction of process models from time-series data. In *Proceedings of the Twentieth International Conference on Machine Learning*, 432–439. Menlo Park: AAAI Press.

Nédellec, C.; Rouveirol, C.; Adé, H.; Bergadano, F.; and Tausend, B. 1996. Declarative bias in ILP. In de Raedt, L., ed., *Advances in Inductive Logic Programming*. Amsterdam: IOS Press. 82–103.

Todorovski, L. 2003. *Using domain knowledge for automated modeling of dynamic systems with equation discovery*. Ph.D. Dissertation, Faculty of Computer and Information Science, University of Ljubljana, Ljubljana, Slovenia.

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

Żytkow, J. M.; Zhu, J.; and Hussam, A. 1990. Automated discovery in a chemistry laboratory. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 889–894. Boston: AAAI Press.