Discovering Constraints for Inductive Process Modeling

Ljupčo Todorovski
Faculty of Administration
University of Ljubljana
SI-1000 Ljubljana, Slovenia

Will Bridewell
Center for Biomedical Informatics
Research, Stanford University
Stanford, CA 94305 USA

Pat Langley
Institute for the Study of
Learning and Expertise
Palo Alto, CA 94306 USA

Abstract

Scientists use two forms of knowledge in the construction of explanatory models: generalized entities and processes that relate them; and constraints that specify acceptable combinations of these components. Previous research on inductive process modeling, which constructs models from knowledge and time-series data, has relied on handcrafted constraints. In this paper, we report an approach to discovering such constraints from a set of models that have been ranked according to their error on observations. Our approach adapts inductive techniques for supervised learning to identify process combinations that characterize accurate models. We evaluate the method's ability to reconstruct known constraints and to generalize well to other modeling tasks in the same domain. Experiments with synthetic data indicate that the approach can successfully reconstruct known modeling constraints. Another study using natural data suggests that transferring constraints acquired from one modeling scenario to another within the same domain considerably reduces the amount of search for candidate model structures while retaining the most accurate ones.

Introduction

Constructing explanatory process models of complex systems is a cognitively challenging task that occupies a substantive portion of scientific work. To carry out this task, human experts combine general, theoretical principles from scientific fields with specific assumptions about an observed system. Over the past decade, cognitive systems researchers have developed *inductive process modeling*, an automated approach that combines knowledge with data to build explanatory models (Langley *et al.* 2002). Two forms of knowledge play important roles in both human and machine approaches to scientific process modeling. Generic processes capture knowledge about which entities in a complex system interact with each other and which mathematical equations account for the dynamics of their interactions. Although generic processes represent plausible components for models, many of the potential combinations are scientifically implausible. Constraints on potential combinations of processes, such as knowing that two processes are mutually exclusive, limit the space of acceptable model structures (Bridewell & Langley 2010).

Early research on the problem of inductive process modeling (e.g., Todorovski *et al.* 2005) assumed the availability of manually encoded constraints. However, the task of handcrafting such constraints is tedious and challenging, as it involves eliciting and representing tacit knowledge. Although scientists routinely and easily identify implausible model structures, they are usually unable to state formally the properties of plausible structures. We address this issue with a computational system that discovers constraints from example models that are ranked according to their ability to account for data. Initial research on constraint discovery (Bridewell & Todorovski 2007; 2010) treated the problem as a supervised learning task. Using an arbitrary threshold value on the model error, the most accurate model structures were labeled as positive examples, while the others were labeled negative. A supervised learning algorithm then induced classification rules expressed as conjunctions of features that denoted the required presence or absence of processes in a model.

In this paper, we address two limitations of previous approaches to constraint learning. The first is the arbitrary selection of a threshold value for distinguishing between accurate and inaccurate models. The second is the limitation in the expressiveness of the learned constraints, which captured only a restricted kind of knowledge found useful for inductive process modeling. To address these limitations, we have developed a new approach to constraint discovery that incorporates a novel method for evaluating constraints against a ranked set of models, alleviating the need for an arbitrary threshold, and aligns the hypothesis space of the learning method with the types of modeling constraints that reflect scientific knowledge. We hypothesize that this approach can reconstruct known constraints and discover new ones that successfully transfer across modeling tasks, thereby improving the behavior of an inductive process modeling system.

After reviewing previous research on inductive process modeling and the SC-IPM system, which addresses that task, we present our new approach to constraint discovery. We then report an evaluation that uses synthetic data to show the system can reconstruct known constraints from a ranked set of model structures. We also examine the system's be-

havior on natural scientific data to support our claim that learned constraints transfer to similar modeling tasks and thus reduce subsequent search costs. In closing, we discuss related work and directions for further research.

Inductive Process Modeling

As we noted earlier, scientists commonly model complex systems with the express goals of explaining observations and predicting future behavior. Such models often refer to processes that govern system dynamics and to the entities involved in those processes. Equations offer one way to formalize processes as predictive models, but they fail to make contact with the process knowledge shared among scientists. In response, Langley *et al.* (2002) proposed a language for *quantitative process models* that draws on Forbus's (1984) qualitative process theory and that ties processes and entities to their mathematical formulations. This process modeling formalism supports both model simulation through standard techniques for solving systems of differential equations (Cohen & Hindmarsh 1996) and model induction from time series and background knowledge through machine learning (Bridewell *et al.* 2008).

We can describe the task of inductive process modeling in terms of its inputs and outputs:

Given:

- observations for a set of continuous variables as they change over time;
- a set of entities that the model may include;
- generic processes that specify relations among entities;
- constraints that indicate plausible relations among processes and entities;

Find: a specific *process model* that explains the observed data and that predicts unseen data accurately.

Table 1 shows a high-level representation of a library of generic entities, generic processes, and constraints for population dynamics. The library has two types of entities, producers that can grow on their own and grazers that feed on producers. Grazing is modeled with a combination of the *grazing* process and a rate process that specifies the details of the interaction (e.g., *lotka_volterra* or *ivlev*).

Combining generic processes with the specific entities involved in a complex system produces a set of instantiated components for possible inclusion in model structures. Without constraints, the number of these components tends to grow exponentially with the number of entities. For example, in a simple system with two producers and two grazers, each growth process leads to two process instances (one for each producer), and the loss process leads to two instances, while each of the grazing and the seven grazing-rate processes lead to four instances (one for each producer–grazer combination). Ignoring model structures that involve duplicates for any of these processes, there are 2^{38} or over 270 billion combinations.

Obviously this is an intractable number of model structures for one to consider explicitly. To cope with this problem, scientists use domain-specific knowledge to prune a vast number of implausible process combinations. This knowledge lets them concentrate on a few plausible structures and fit their parameters to observations. We can provide systems for inductive process modeling with this ability by providing formal constraints on the model structures. For instance, the HIPM system (Todorovski *et al.* 2005) used a taxonomy of generic processes to encode a context-free grammar that defines a space of plausible model structures. These taxonomies offered an elegant view of the domain theory behind model construction, but they proved unwieldy to encode and difficult to map to instantiated models. In re-

Table 1: A generic library for modeling population dynamics. For simplicity, we omit details about some generic processes. The expression *d[X,t,1]* denotes the first derivative of the variable *X* with respect to time *t*.

library *population_dynamics*

generic entity *grazer*
  variables *c*, *gr*
  parameters *ef*[*0.001, 0.8*], *max_gr*[*0.0001, 1*]

generic entity *producer*   variables *c*

generic process *exponential_growth*
  relates *P*{*producer*}
  parameters *gr*[*0,3*]
  equations *d*[*P.c, t, 1*] = *gr* · *P.c*

generic process *logistic_growth*
generic process *exponential_loss*   relates *G*{*grazer*}

generic process *grazing*
  relates *G*{*grazer*}, *P*{*producer*}
  equations
  d[*G.c, t, 1*] = *G.ef* · *G.gr* · *G.c*
  d[*P.c, t, 1*] = *-1* · *G.gr* · *G.c*

generic process *lotka_volterra*
  relates *G*{*grazer*}, *P*{*producer*}
  equations *G.gr* = *G.max_gr* · *P.c*

generic process *generalized_gause*
generic process *holling_type_1*
generic process *holling_type_3*
generic process *ivlev*
generic process *monod*
generic process *ratio_dependent_2*

constraint *loss_requirement*
  type necessary
  processes *exponential_loss(P)*

constraint *grazing_requirement*
  type necessary
  processes *grazing(G,_)*

constraint *grazing_alternatives*
  type exactly-one
  processes *lotka_volterra(G,P)*, *generalized_gause(G,P)*, *holling_type_1(G,P)*, *holling_type_3(G,P)*, *ivlev(G,P)*, *monod(G,P)*, *ratio_dependent_2(G,P)*

Table 2: Specialization operator for SC-IPM constraints. The first column refers to the initial constraint, the second column enlists its specialization, and the third column describes the specialization in terms of models that are ruled out. Throughout the table, q denotes a qualifier, *gp* a generic process, and *gps* a set of at least two generic processes.

Constraint	Specialization	Specialization description
necessary *gp*	add q	do not include an instance of *gp* involving an entity instance of q
always-together *gps*	add q	include instances of a proper subset of *gps* involving an entity instance of q
	add *gp*, $gp \notin gps$	include instances of *gps* and do not include an instance of *gp* do not include instances of *gps* and include an instance of *gp*
at-most-one *gps*	remove q	include two or more instances of *gps* involving an entity instance of q
	add *gp*, $gp \notin gps$	include an instance of *gps* and an instance of *gp*
	change to exactly-one	do not include any instance of *gps*

sponse, SC-IPM (Bridewell & Langley 2010) explicitly represents modular constraints that capture most of the semantics of HIPM's grammar.

SC-IPM supports four types of constraints: *necessary*, *always-together*, *at-most-one*, and *exactly-one*. *Necessary* constraints assert that at least one instance of each listed generic process must appear in a model. The population-dynamics library in Table 1 includes two *necessary* constraints. The first, *loss_requirement*, specifies that a model must include an instance of the *exponential_loss* generic process. Moreover, the qualifier G used in the specification of the generic process, ensures that a model includes an instance of the loss process for each grazer entity. In addition, the second constraint, *grazing_requirement*, asserts that a model must include a grazing process for each grazer. *Always-together* constraints are similar, but declare that a model must instantiate all or none of the specified processes. This relationship is useful in population-dynamics models that include nutrient-limited growth, which must appear with a corresponding nutrient absorption process.

Two other types of constraints let users assert that generic processes are mutually exclusive. The *at-most-one* constraint specifies two or more generic processes, at most one of which can appear in a model at a time. The other variant of mutual exclusion, *exactly-one*, combines the declarative features of *at-most-one* and *necessary* to define a mutually exclusive group of generic processes, exactly one instance of which must appear in the model. The final constraint in the population dynamics library, *grazing_alternatives*, states that, for each pair of grazer and producer entities, *exactly one* of the seven enlisted grazing-rate processes must appear in a valid model.

The three constraints in the population dynamics library rule out over 99% of component combinations that arise in the case of two producers and two grazers. Together they reduce the initial space of more than 270 billion candidate combinations to roughly 345 thousand. Although this is still far more than a scientist would consider, additional constraints (e.g., an *exactly-one* constraint for the growth processes) reduce the space even further.

Discovering Constraints

Approaches to inductive process modeling adopt the general paradigm of heuristic search through the space of candidate solutions. In this framework, nodes in the search space correspond to candidate solutions, in particular, process models. A heuristic function maps the nodes to the quality of the corresponding solutions in terms of model error on the observed data. Although the search algorithm usually reports only the few best solutions, its trace can reveal the entire set of candidates considered during the search, along with their corresponding scores. These traces can act as training sets for learning constraints. More formally, we can state the task of discovering constraints as:

Given: a set of models and their errors on observational data;

Find: a set of constraints that rule out inaccurate models and that generalize well to other modeling tasks.

When solving this task, we define a search space in which the nodes are candidate constraints. The corresponding heuristic function should capture the notion of accurate constraints that *rule out* inaccurate models. In the remainder of this section, we describe the organization of the search space and the heuristic function for evaluating constraints.

Organizing the Search

We structure the search space of candidate constraints according to generality. The generality (or specificity) of a constraint is related to the size of the set of models that it matches (or rules out). The most general constraint is an empty one that matches all models. At the other extreme, the most specific constraint is the one that rules out all models. The partial, general-to-specific ordering of constraints corresponds to the inclusion ordering of the corresponding sets of matched models. In Table 2, we define a specialization operator that can be used to generate constraints in the general-to-specific order. At each step, the operator generates the least-general specializations of a given constraint (De Raedt 2011).

We illustrate the use of the operator on a modeling task in population dynamics (Table 1) with two producer

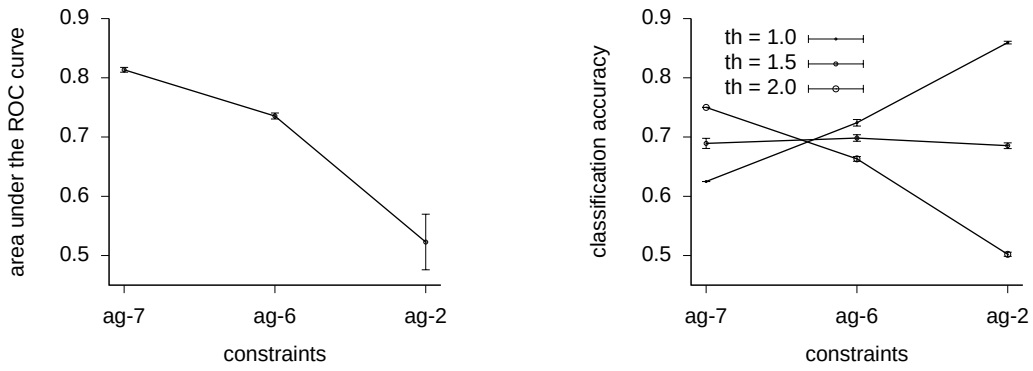

Figure 1: Comparison of two metrics for evaluating constraint accuracy: area under the ROC curve (left) and classification accuracy (right) using three thresholds (th) for distinguishing between accurate and inaccurate models. *ag-7* denotes the alternative grazing constraint with seven mutually exclusive processes used to generate evaluation data, while *ag-6* and *ag-2* are two more general constraints with six and two mutually exclusive grazing processes, respectively.

and two grazer entities. The constraint *necessary exponential_loss* requires models that include a loss process for one or both grazer entities. By adding a qualifier, we obtain its specialization *necessary exponential_loss(G)*, which rules out the models with a single loss process for only one of the grazer entities. Furthermore, consider the constraint *always-together exponential_growth, exponential_loss*, which matches models with either (A) one or more *exponential_growth* and one or more *exponential_loss* processes or (B) none of them. By adding *logistic_growth* to its list of generic processes, we obtain a specialization that rules out the models of class (A) that *do not* include a *logistic_growth* process and the models of class (B) that *do* include a *logistic_growth* process. Finally, consider the constraint *at-most-one exponential_growth, logistic_growth*, which matches models with one exponential growth and no logistic growth processes, one logistic growth and no exponential growth process, or none of them. When changing its type to *exactly-one exponential_growth, logistic_growth*, we rule out all models from the third class.

Our system for discovering constraints carries out beam search through the structured space of candidates. In the first iteration, the search procedure considers the most general constraints; at each consecutive iteration, least-general specialization of the current constraints are considered. Constraints that are too specific, matching less than a user specified number of models, are removed from the search space. The search stops when the beam stagnates: when the accuracies of the constraints considered in the most recent iteration are too low to displace those in the current beam.

Evaluating Constraints

The heuristic function's measure of constraint quality relates to the accuracy of the models that a constraint matches. One way to evaluate constraint quality would rely on standard classification accuracy used in supervised learning, defined as the ratio of the number of correctly classified examples to the number of all examples. When learning constraints, a correctly classified example takes two forms: (1) models that fit observed data well and are matched by the constraint and (2) models that fit the data poorly and are ruled out by the constraint. This approach implies that someone has set a threshold that distinguishes models with a good fit to the data from those with a poor fit.

However, the selection of an appropriate threshold is problematic. Consider the constraint *grazing-alternatives* from Table 1 that specifies the set of seven mutually exclusive grazing processes. Let us compare the classification accuracy of this constraint with the accuracies of two more general constraints: one that includes a set of six mutually exclusive grazing-rate processes and one that includes only two of them. We evaluate these constraints on a synthetic data set of 1000 models used and described in the section that follows: the models and their scores are generated in a way that guarantees that the most accurate models are those that match the constraints from Table 1. Thus, one would expect that the original constraint will rank as better than the other two candidates.

The graph on the right-hand side of Figure 1 shows that the classification accuracy measure fails to meet our expectations. The threshold value of 1.0 leads to classification accuracy estimates that rank the constraint including two mutually exclusive processes as best. The threshold value of 1.5 does not provide a clear ranking. Only the threshold value of 2.0 leads to the expected ranking of the three constraints. Importantly, this example illustrates that classification accuracy is sensitive to the threshold value used to distinguish between inaccurate and accurate models.

To address this problem, we propose an alternative measure of constraint accuracy, based on receiver operator characteristic (ROC) analysis (Fawcett 2006). In supervised learning, ROC analysis is often used to evaluate learned classifiers that rank examples according to their likelihood of belonging to a given class. In such a setting, one must choose a likelihood threshold in order to classify examples. ROC analysis is then used as a tool for selecting an appropriate decision threshold. When evaluating constraints, rather than ranking the predictions we instead rank the training models

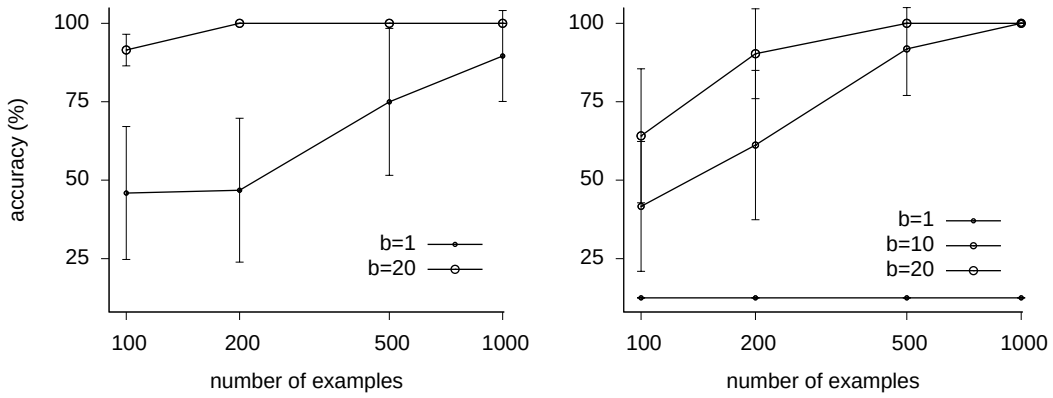

Figure 2: Results on reconstructing constraints in two synthetic domains with different numbers of examples and beam widths (b). The first domain (left) includes seven alternative grazing-rate processes, whereas the second (right) includes 13 processes.

based on their fits to the data. Thus, we can obtain an ROC curve by altering the threshold value that distinguishes between models which fit the data well and those which do not. In turn, we can empirically estimate the area under the ROC curve (AUC) and use this quantity as an estimate of a given constraint's accuracy.

The graph on the left-hand side of Figure 1 reports the AUC estimates for the three constraints described above. Note that these estimates produce the correct constraint ranking, where the constraint used to generate the validation examples is ranked best with an AUC value above 0.8. Based on these results, we hypothesize that the area under the ROC curve can be used as a reliable measure to evaluate constraint accuracy without needing an explicit threshold value that separates accurate and inaccurate models. In the next section, we report results from two empirical tests of this conjecture.

Empirical Evaluation

To evaluate our approach to constraint discovery, we carried out two empirical studies. The first of these tested the method's ability to reconstruct known constraints from synthetic data. The second study examined how well the approach discovers new constraints that improve the behavior of inductive process modeling when transferred to new but related modeling tasks.

Reconstructing Constraints

In the first study, we carried out controlled experiments with synthetic data produced from the population dynamics domain using the initial set of constraints presented in Table 1. We first generated a random sample of 20,000 models that involved two producers and two grazers. The sample was stratified with respect to the constraints being matched: one eighth of the models matched all three constraints, three eighths violated only one of the first, second, and third constraints, three eighths violated two constraints, and one eighth violated all three constraints. We initially set the score for each model to the number of violated constraints. We randomized ranking of models with the same value by adding to each score a number sampled from the [0,1) interval.

From the 20,000 models, we selected a single stratified sample of 5,000 models for testing. From the remaining 15,000 models, we took training samples of sizes 100, 200, 500, and 1,000, generating 10 training sets for each size. To address the complexity of the domain, we generated another set of training and testing sets for a variant of population dynamics with 13 alternative grazing-rate processes. In each condition, we examined whether the original constraints were successfully reconstructed, first by checking their validity on the test data and then by syntactically comparing them to the initial constraints.

Figure 2 summarizes the experimental results for the validity of discovered constraints on test cases. For the more complex domain (on the right), the system needs more examples (500) to successfully reconstruct the initial constraints than it does for the simpler domain (on the left), where it needs only 200 cases. Note that more search can compensate for the lack of examples. Increasing the beam width in both domains decreases the number of examples necessary for complete reconstruction. In the more complex domain, the system requires 1,000 examples for successful reconstruction with a beam width of 10, whereas it needs only 500 with a beam width of 20. A comparison of the reconstructed constraints that had 100% accuracy on the test cases revealed a perfect syntactic match with the constraints utilized for generation.

In terms of computation, the size of the beam directly affects the amount of search as measured by the number of evaluated constraints. With a beam width of one and with 100 examples, the system evaluated an average, over 10 runs, of 324 constraints. Widening the beam to 10 and to 20 led, respectively, to an average of 1,415 constraints ($\sigma = 523$) and 2,983 constraints ($\sigma = 844$). Increasing the number of examples while increasing the time to evaluate each constraint had little effect on the amount of search.

Table 3: Benefits of transferring discovered constraints between the three modeling tasks that involve population dynamics. The *rf* column shows the reduction factor for the space of candidate models, whereas the *bmr-10* and *bmr-50* columns indicate recall of the best 10 and 50 models in the reduced space.

		Improvement	
Transfer target	*rf*	*bmr-10*	*brm-50*
PP1	13.55	40%	24%
PP2	13.55	20%	16%
PP3	13.55	50%	44%

Transfer of Constraints

To determine whether the learned constraints transfer effectively across modeling tasks, we carried out experiments on three additional problems. Each task involved modeling population dynamics from measured concentration trajectories of two species: nasutum, a grazer, and aurelia, a producer (Jost & Ellner 2000). To generate an exhaustive set of model structures for these problems, we ran HIPM (Todorovski *et al.* 2005) on each task using a variant of the library of generic processes from Table 1 that lacked constraints. Unlike SC-IPM, which samples the model space using a constraint solver, HIPM carries out exhaustive search through the space of candidate model structures. This procedure generated 9,402 candidate models that included from one to five processes, along with their error scores on each task. Thus, we obtained three training sets for constraint discovery, which we will refer to as PP1, PP2, and PP3.

After discovering constraints from each of the three training sets, we assessed how well they transferred to the other two tasks using two metrics. The first measure was the factor by which the constraints reduced the search space. We calculated this factor as the ratio of the number of models in the unconstrained space to the number in the constrained space. The second measure was the percentage of the most accurate models in the unconstrained space that were retained in the constrained space.

Interestingly, the system discovered the same set of three constraints from all three training sets. Two of the constraints assert the necessary presence of exponential loss and logistic growth processes in the models. The other is an *at-most-one* constraint that specifies the set of nine mutually exclusive grazing-rate processes. Two of these resemble the constraints handcrafted by the domain scientists. The remaining one asserts the presence of a logistic growth process, which is known to offer a plausible account of growth in population dynamics systems with limited resources.

Table 3 summarizes the transfer results. The discovered constraints reduce the search space by a factor of 13.55 from the initial 9,402 to only 694 candidate model structures. The recall rate of the best models varies from 50% of the ten best models for PP3 to 16% of the 50 best models for PP2. The results support our conjecture that the approach can discover constraints which substantially reduce the search space while retaining many of the most accurate models.

We can compare these results to ones obtained with an earlier approach to learning constraints (Bridewell & Todorovski 2007), which we will refer to as LDB. That system produced higher recall rates that varied between 36% and 100%, while it had lower average reduction factors that varied between 9 and 16. However, the constraints discovered by LDB showed great sensitivity to the training set: for each task, LDB discovered a different set of constraints. While some of these were well known in population dynamics, many others were specific to the particular modeling task. The discovery system presented in this paper appears to be far less sensitive to the training cases it processes.

Related Work

The approach that we have reported builds on two separate lines of research. Our work falls squarely in the paradigm of computational scientific discovery (Shrager & Langley 1990; Džeroski & Todorovski 2007), which aims to acquire knowledge in some established scientific formalism. More specifically, we have built upon earlier work on inductive process modeling (Langley et al., 2002; Bridewell & Langley, 2010), which combines background knowledge with time-series data to construct explanatory models.

We have noted that one drawback of research in this area concerns its reliance on hand-crafted constraints to limit search and eliminate implausible models. An earlier attempt to induce such constraints (Bridewell & Todorovski 2007) suffered in two important ways. First, it relied on an arbitrary threshold to separate accurate models from inaccurate ones. Our new approach instead utilizes ROC curves to learn constraints directly from a ranked list of models without selecting a threshold, leading to more robust behavior. Second, the earlier work was limited to one class of constraints that could be acquired with standard methods from inductive logic programming. In contrast, our approach can induce the entire range of constraints on which inductive process modeling draws, letting them be integrated into generic libraries and transferred across different modeling tasks.

Our approach is also closely related to efforts on learning in the context of constraint programming (O'Sullivan 2010). For instance, Bessiere *et al.* (2006) report a version-space technique for learning constraints from solutions and nonsolutions of constraint satisfaction problems. Similarly, Charnley *et al.* (2006) describe an approach to constraint revision for improving performance on constraint satisfaction tasks. These approaches are not directly applicable in the context of inductive process modeling because they assume a crisp distinction between solutions and nonsolutions, but they also carry out search through a space of constraints guided by the results of problem solving in a lower-level space.

Concluding Remarks

In the preceding pages, we have presented an innovative approach to discovering constraints for use during the induction of scientific process models. This constitutes an advance over previous methods in that it introduces a novel technique for evaluating candidate constraints based on ROC analysis that does not depend on an arbitrary threshold be-

tween accurate and inaccurate models. Moreover, the new approach discovers a broader range of constraints used by inductive process modelers while expressing them in a formalism that human scientists should find understandable. In addition, experiments suggest that the approach can reconstruct hand-crafted constraints in a reliable manner and that these constraints transfer well to other modeling tasks, reducing search without hampering the accuracy of the induced models.

This paper has focused on constraints for inductive process modeling that limit the space of possible process combinations. However, generic processes also incorporate constraints on the types of entities they include and on how equation fragments combine into processes. Adapting constraint learning to the latter suggests one way to construct entirely new processes, which must currently be generated manually. In future work, we plan to extend our the approach to acquire these additional kinds of knowledge, which in turn should let us automate even more fully the task of inductive process modeling.

We also anticipate that our approach to constraint learning will prove useful in settings other than inductive process modeling. In principle, it should apply in any context that relies on heuristic search to explore a problem space. Adapting our framework to other tasks will require formalisms that describe the structure of candidate solutions and constraints on plausible structures. With these in hand, the basic approach presented here should easily transfer to other constraint discovery tasks, supporting the improvement of heuristic search in other domains.

Acknowledgements

This research was supported by Grant No. N00014-11-1-0107 from the Office of Naval Research, which is not responsible for its contents. We thank Richard Billington and Matt Bravo for their contributions to the SC-IPM system.

References

Bessiere, C.; Coletta, R.; Koriche, F.; and O'Sullivan, B. 2006. Acquiring Constraint Networks Using a SAT-Based Version Space Algorithm. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 23–24. Boston, MA: AAAI Press.

Bridewell, W., and Langley, P. 2010. Two Kinds of Knowledge in Scientific Discovery. *Topics in Cognitive Science*, 2: 36–52.

Bridewell, W.; Langley, P.; Todorovski, L.; and Džeroski, S. 2008. Inductive Process Modeling. *Machine Learning* 71:1–32.

Bridewell, W., and Todorovski, L. 2007. Learning Declarative Bias. In *Proceedings of the Seventeenth Annual International Conference on Inductive Logic Programming*, 63–77. Corvallis, OR: Springer.

Bridewell, W., and Todorovski, L. 2010. The Induction and Transfer of Declarative Bias for Modeling Scientific Processes. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*, 401–406. Atlanta, GA: AAAI Press.

Charnley, J., Colton, S.; and Miguel, I. 2006. Automatic Generation of Implied Constraints. In *Proceedings of the Seventeenth European Conference on Artificial Intelligence*, 73–77. Riva del Garda, Italy: IOS Press.

Cohen, S., and Hindmarsh, A. 1996. CVODE, A Stiff/Nonstiff ODE Solver in C. *Computers in Physics*, 10:138–143.

De Raedt, L. 2011. Logic of Generality. In Sammut, C., and Webb. G. I. eds. *Encyclopedia of Machine Learning*, 624–631. New York, NY: Springer.

Džeroski, S., and Todorovski, L. eds. 2007. *Computational Discovery of Scientific Knowledge: Introduction, Techniques, and Applications in Environmental and Life Sciences*. Berlin, Heidelberg, Germany: Springer-Verlag.

Fawcett, T. 2006. An Introduction to ROC Analysis. *Pattern Recognition Letters* 27:861–874.

Friedman, S. E., and Forbus, K. D. 2010. An Integrated Systems Approach to Explanation-Based Conceptual Change. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*, 1523–1529. Atlanta, GA: AAAI Press.

Forbus, K. D. 1984. Qualitative Process Theory. *Artificial Intelligence* 24:85–168.

Jost, C., and Ellner, S. P. 2000. 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.

Lallouet, A.; Lopez, M.; Martin, L.; and Vrain, C. 2010. On Learning Constraint Problems. In *Twenty-Second IEEE International Conference on Tools with Artificial Intelligence*, 45–52, Arras, France: IEEE Computer Society.

Langley, P.; Sánchez, J.; Todorovski, L.; and Džeroski, S. 2002. Inducing Process Models from Continuous Data. In *Proceedings of the Nineteenth International Conference on Machine Learning*, 347–354, Sydney, Australia: Morgan Kaufmann.

O'Sullivan, B. 2010. Automated Modelling and Solving in Constraint Programming. In *Proceedings of the Twenty-Fourth National Conference on Artificial Intelligence*, 1493–1497. Atlanta, GA: AAAI Press.

Shrager, J., and Langley, P. eds. 1990. *Computational Models of Scientific Discovery and Theory Formation*. San Mateo, CA: Morgan Kaufman.

Todorovski, L.; Bridewell, W.; Shiran, O.; and Langley, P. 2005. Inducing Hierarchical Process Models in Dynamic Domains. In *Proceedings of the Twentieth National Conference on Artificial Intelligence*, 892–897. Pittsburgh, PA: AAAI Press.