Topics in Cognitive Science 2 (2010) 36–52

ISSN: 1756-8757 print / 1756-8765 online
DOI: 10.1111/j.1756-8765.2009.01050.x

Two Kinds of Knowledge in Scientific Discovery

Will Bridewell, Pat Langley

Computational Learning Laboratory, Center for the Study of Language and Information, Stanford University

Received 29 July 2008; received in revised form 19 July 2009; accepted 21 July 2009

Abstract

Research on computational models of scientific discovery investigates both the induction of descriptive laws and the construction of explanatory models. Although the work in law discovery centers on knowledge-lean approaches to searching a problem space, research on deeper modeling tasks emphasizes the pivotal role of domain knowledge. As an example, our own research on inductive process modeling uses information about candidate processes to explain why variables change over time. However, our experience with IPM, an artificial intelligence system that implements this approach, suggests that process knowledge is insufficient to avoid consideration of implausible models. To this end, the discovery system needs additional knowledge that constrains the model structures. We report on an extended system, SC-IPM, that uses such information to reduce its search through the space of candidates and to produce models that human scientists find more plausible. We also argue that although people carry out less extensive search than SC-IPM, they rely on the same forms of knowledge—processes and constraints—when constructing explanatory models.

Keywords: Artificial intelligence; Constraints; Creativity; Knowledge representation; Processes; Scientific discovery; Scientific modeling

1. Introduction

As a field, computational scientific discovery seeks to understand the products and processes of science by studying artifacts that engage and assist in knowledge construction. Along these lines, researchers have investigated activities like taxonomy formation, law discovery, and theory development. Their findings have demystified these activities and suggest a strong link between the disciplined practices of scientists and the everyday reasoning skills shared by everyone. Moreover, this work has revealed previously implicit types of knowledge crucial to scientific reasoning.

Correspondence should be sent to Will Bridewell, Cognitive Systems Laboratory, Center for the Study of Language and Information, 210 Panama St., Stanford University, Stanford, CA 94305. E-mail: willb@csli.stanford.edu

The field's earliest and most fundamental result was that programs could actually make discoveries. Up to the late 20th century, the belief that scientific discovery requires a ''divine spark'' peculiar to humanity dominated academic thought on the matter. However, a continual stream of programs that recapitulated historical discoveries and made new ones of their own has served to reshape scientific activity as an understandable, computational process based on problem space search. Some of the earliest support for this view came from the BACON series of programs, which rediscovered Kepler's third law, the ideal gas law, and other numerical relationships from data (Langley, Simon, Bradshaw, & Żytkow, 1987). Later efforts expanded to include the inference of biochemical reactions (e.g., Kulkarni & Simon, 1990), genetic networks (e.g., Zupan et al., 2007), and other forms of scientific knowledge. Some systems targeted other tasks, including the design, execution, and analysis of experiments (Żytkow, Zhu, & Hussam, 1990), the discovery of equations from time-series data (Todorovski & Džeroski, 1997), and the revision of causal models (Mahidadia & Compton, 2001). Additionally, some researchers investigated scientific reasoning by exploring its links with everyday activities (Klahr, 2002; Pazzani & Flowers, 1990), bringing a new perspective to the field.

An unexpected benefit of research on computational scientific discovery has been the recognition of new types of scientific knowledge that are often not discussed in the literature on the history, philosophy, and psychology of science. This knowledge arose from the practical nature of system building, as researchers found it necessary for a program to function. One of the earliest such findings was the chemical fracturing heuristics in DENDRAL (Feigenbaum, Buchanan, & Lederberg, 1971), which interpreted results from a mass spectrometer. On a separate front, work on qualitative physics (Forbus, 1997), in particular qualitative process theory (Forbus, 1984), introduced the role of processes in scientific and common-sense knowledge. More recently, Żytkow (1999) linked scientific theory, processes, and data to suggest a computational method for model construction.

Building on these insights, we defined the task of *inductive process modeling* and developed computational systems to address it (Bridewell, Langley, Todorovski, & Džeroski, 2008; Bridewell, Sánchez, Langley, & Billman, 2006). This task, which we recount in the next section, involves the construction of quantitative models from time-series data and domain knowledge. Systems for inductive process modeling use the knowledge to guide search through a space of candidate models that they, in turn, evaluate with respect to the data. Initially, we believed that knowledge of the generic processes common to a scientific discipline would sufficiently limit this search, but our experience shows that the systems commonly consider implausible candidates. Compounding this problem, some of these nonsensical models ranked higher in quantitative terms than the scientifically plausible ones. This finding led us to extend the task description to include modeling constraints, which are a type of scientific knowledge that often remains implicit.

In this paper, we emphasize the importance of constraints that characterize the structure of scientifically plausible models. After describing inductive process modeling, we introduce a new system that uses modeling constraints to guide its search. We then discuss computational experiments that evaluate the utility of the added knowledge. We conclude with

Table 1
The Lotka–Volterra equations for population dynamics (a) unlabeled and (b) labeled with a theoretical interpretation of the component terms

(a)	(b)	
$\dot{p} = \alpha p - \beta pn$	$\dot{p} = \underbrace{\alpha p}_{\text{growth}} - \underbrace{\beta pn}_{\text{predation}}$	$\alpha =$ growth rate, $\gamma =$ death rate
$\dot{n} = \epsilon\beta pn - \gamma n$	$\dot{n} = \underbrace{\epsilon\beta pn}_{\text{predation}} - \underbrace{\gamma n}_{\text{death}}$	$\beta =$ attack rate, $\epsilon =$ efficiency

an examination of related work on constraints in scientific discovery and suggestions for future research in this area.

2. Constructing mechanistic models

In many fields, scientists use *mechanistic models*, which illustrate the relationships among entities and processes, to express causal explanations (Machamer, Darden, & Craver, 2000). In these models the entities are physical objects with spatial extension and have properties of interest such as weight, color, and concentration. Although a particular entity may refer to a single object, such as a bird or a table, it can also represent a group of objects like a flock or a population. The degree of aggregation depends primarily on the driving scientific question and the methods for addressing it. The processes in a model connect entities to each other and are the manifestations of change in the world. Unlike entities, which have measurable properties, processes are unobservable, so scientists infer their existence by observing changes in entities. Theoretical knowledge from the relevant scientific domain guides this inference and influences model construction in general.

2.1. From theory to models

When building mechanistic models, scientists appeal to a mixture of theory and data. Imagining a spectrum of modeling practices, we can place theory-driven or ‘‘analytical’’ approaches at one endpoint and data-driven or ‘‘inductive’’ approaches at the other. Although one can build models analytically, in practice empirical observations play a role during parameter estimation and during evaluation. Likewise, scientists can build purely descriptive models from data, but theoretical concerns still influence variable selection and data collection. Moreover, descriptive models lack explanatory import unless they make contact with theoretical concepts. For example, the Lotka–Volterra equations in Table 1 (a) present a purely mathematical description of the relationship between two variables until we designate their meaning, as in Table 1(b). Although illustrated in quantitative terms, these ideas also apply to qualitative models, which must connect to both data and domain theory to have relevance.

Modern work in the philosophy of science suggests strategies that move from knowledge and data to mechanistic models. For instance, Bechtel and Abrahamsen (2005) argue for the

use of functional and structural decomposition. According to this account, scientists develop mechanisms first by identifying the functional components that could account for observed phenomena and then by determining the structural components that carry out those functions. Darden (2002) lists three other strategies along with historical evidence of their use. First, schema instantiation starts from a rough, general form of an explanation (e.g., $reactant_1 + reactant_2 = product$) that tells scientists what types of entities or processes they should seek out.[1] Second, modular subassembly inverts Bechtel's decomposition strategies by constructing the mechanism from known submodels. Third, forward or backward chaining reasons about entities and activities that spring from or produce known effects of the model.

When proposing these strategies, Bechtel and Darden noticeably avoid the claim that scientists systematically generate mechanisms from domain theory. Punctuating this omission, Suárez and Cartwright (2008) argue against the claim that a ''theory will in general already contain…the good representative models that it spawns.'' In particular, Cartwright (1999) rejects what she calls the *vending machine* view of a theory:

> you feed it input in certain prescribed forms for the desired output; it gurgitates for a while; then it drops out the sought-for representation, plonk, on the tray, fully formed, as Athena from the brain of Zeus. (p. 247)

Instead they claim that scientific theory plays a heuristic role in model construction and not a generative one (Cartwright, Shomar, & Suárez, 1995; Suárez & Cartwright, 2008). Cartwright (1999) admits that the vending machine view has advantages, but that she lacks an ''independent reason to believe that this kind of mechanization is possible.''

We assert that computational model generation is possible. As preliminary evidence, we point to work on *compositional modeling* by Falkenhainer and Forbus (1991). Their system constructs qualitative process models from a domain theory, a description of a physical structure, and a question about the structure's behavior. As further support, Keppens and Shen's (2001) overview of compositional modeling discusses subsequent work in this area, which researchers applied in a variety of domains. Together, these efforts suggest that some version of the vending machine view is feasible and that Cartwright may be too hasty in discarding it.

2.2. Representing theories and models

Our own research makes further strides in this direction, generating models from scientific theory and data. Principally, we claim that building mechanistic models fits the mold of heuristic search through a problem space (Newell & Simon, 1976). This characterization implies representations both for the models and for the knowledge that directs their construction. To this end, we have developed a representation for an important class of models, called *quantitative process models*. In these models, an entity is a collection of the variables and constants that define its state. A process refers to entities and contains algebraic and differential equations that specify its effects. Optionally, each process may contain conditions that signal its applicability (e.g., if a variable's value exceeds a threshold). Table 2 contains an example model that includes processes and entities from population dynamics.

Table 2
A quantitative process model from population dynamics

```
model predator_prey

  entities rabbit{prey}, fox{predator}

  process exponential_growth
    equations d[rabbit.p,t,1]=2.5*rabbit.p

  process exponential_loss
    equations d[fox.p,t,1]=−1.2*fox.p

  process predation_holling_1
    equations
    d[rabbit.p,t,1]=−0.1*rabbit.p*fox.p
    d[fox.p,t,1]=0.3*0.1*rabbit.p*fox.p
```

Note. The entities *rabbit* with type *prey* and *fox* with type *predator* each have a variable *p* that stores their respective population size. The notation $d[X,t,1]$ indicates the first derivative of X with respect to time t.

Quantitative process models are closely tied to the differential equation models that some scientists routinely use. Notice that in Table 2 the equations for the rabbit and fox populations are split across multiple processes. If we combine these elements using addition, then we produce a pair of equations with the same general form as those in Table 1. With this in mind, scientists can simulate these models to view their behavior just as they would with any system of differential equations. Once a model withstands evaluation, scientists can use it for several purposes. For example, the model could serve as a predictive instrument for anticipating future population dynamics. Alternatively, the model may act as evidence for an argument that certain theoretical interactions hold in a natural system. In other cases, the model can function as a proxy for a natural system, giving scientists a way to investigate complex hypotheses when controlled experiments are impracticable.

Although scientists can use quantitative process models as if they were systems of differential equations, their modularity provides an important distinction. Consider replacing the predation process from the model in Table 2 with an alternative form. Editing the differential equations involves removing the corresponding terms from two equations, adding one or more new terms to both equations, and specifying parameters that properly match. Altering the process model entails removing the old process and adding a new one. The grouped functional forms simplify parameter matching across terms and ensure that further revisions appropriately alter each equation.

As defined, each process represents one interaction that occurs in a particular situation, such as predation in the fox and rabbit dynamics in Champaign, IL. However, the functional forms of these processes often generalize to other scenarios and domains. For instance, the *generic process* for exponential growth is relevant throughout ecology and in other disciplines such as physics and economics. Although the entities and growth rates may differ across specific processes, the mathematical form never varies. The formalism for generic processes resembles that of their instantiated counterparts with the exceptions that they

Table 3
A generic library for population dynamics

```
library population_dynamics
generic entity predator
   variables c

generic entity prey
   variables c

generic process logistic_growth
    relates P{prey}
    parameters gr[0,3], ic[0,0.1]
    equations d[P.c,t,1]=gr*P.c*(1−ic*P.c)

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

generic process exponential_loss
    relates R{predator}
    parameters dr[0,2]
    equations d[R.c,t,1]=−1*dr*R.c

generic process predation_holling_1;
    relates P{prey}, R{predator};
    parameters ar[0.01,10], ef[0.001,0.8];
    equations
   d[P.c,t,1]=−1*ar*P.c*R.c;
   d[R.c,t,1] = ef*ar*P.c*R.c;
generic process predation_holling_2;
    relates P{prey}, R{predator};
    parameters ar[0.01,10], ef[0.001,0.8], ht[1,5];
    equations
   d[P.c,t,1] = −1*ar*P.c*R.c / (1 + ht*ar*P.c);
   d[R.c,t,1]=ef*ar*P.c*R.c / (1+ht*ar*P.c);
```

Note. The variable type constraints are denoted in braces following the variable's name, while parameter bounds are specified within brackets. The notation $d[S,t,1]$ indicates the first derivative of S with respect to t.

contain entity roles instead of entities and parameter ranges instead of real numbers. These ranges specify maximal and minimal values for the parameters, whereas the entity roles are placeholders for the problem-specific entities. Much as processes instantiate generic processes, entities instantiate generic entities that encode their associated properties. Table 3 shows an example library of generic entities and processes for population dynamics that defines a space of mechanistic models.

Taken together, the generic processes and generic entities form the domain theory that guides model construction, but data are necessary for evaluation. As differential equations provide the foundation for quantitative process models, these data will be the trajectories of

continuous variables. With data in hand, one can compare them with simulation trajectories using mean squared error, the frequency of oscillations, or any pertinent fitness measure. If multiple candidate models exist, then one can compare their fitness scores to determine which offers the best available explanation of the phenomena.

The abilities (a) to instantiate generic processes and entities, (b) to revise models by altering their components, and (c) to compare models to each other suggest a practical strategy for searching through a space of quantitative process models. The initial step involves assembling a starting set of models, possibly taken from the related literature, and evaluating their fitness for a particular scenario. Next, one selects those candidates with the best scores and refines them by adding and removing processes. This step initiates future rounds of evaluation and refinement that continue until one or more acceptable models are found. We claim that scientists follow a similar strategy, although in practice, resources may limit them to considering only one or a few candidates.

We follow this fairly general claim about the activity of modeling with a more specific one: the knowledge that scientists bring to the task. When constructing mechanistic models, scientists employ data and knowledge similar to the entities, processes, and generic versions of the same that we have discussed. Clearly, scientists in some fields work with multivariate time series, but they also refer to entities and processes in their publications. For example, Arrigo, Worthen, and Robinson (2003) discuss the processes responsible for ocean physics and biological processes that control plankton concentrations in the Ross Sea. Entities such as phytoplankton and zooplankton populations interact via these processes. Generic processes and entities appear in the literature less frequently than their instantiated versions, but Jost and Ellner (2000) discuss a collection of functional responses for population dynamics models and Atanasova, Todorovski, Džeroski, and Kompare (2006) constructed a library of processes for aquatic ecosystems.

2.3. Inductive process modeling

Having stated representations for mechanistic models and the scientific theories that support them, we can now characterize the task of building mechanistic models:

- *Given*: Generic entities that have properties relevant to the observed dynamics
- *Given*: Generic processes that specify causal relations among entities using generalized functional forms
- *Given*: A set of entities present in the modeled system
- *Given*: Observations for the continuous properties of those entities as they change over time
- *Find*: A quantitative process model that, when given initial values for the modeled variables and values for any exogenous (i.e., forcing) variables, explains the observed data and predicts unseen data accurately

We refer to this task *inductive process modeling* (Langley, Sánchez, Todorovski, & Džeroski, 2002). A system that addresses this problem produces one or more explanations cast in

the language of a scientific domain. Importantly, that model connects the scientists' conceptual knowledge about processes and entities with a familiar mathematical formalism.

We developed a computational discovery system, IPM, to carry out this task using heuristic search through the space of process models. This search operates in two stages, the first building a candidate structure and the second estimating its numerical parameters. Initially, IPM binds the generic processes to the problem-specific entities. These *bound processes* are more concrete than their generic counterparts but still lack numerical parameters. Processes in this form serve as the atomic elements of a model's structure. Candidate solutions consists of the entities, a set of bound processes, and the numerical parameters for those processes and entities.

The space of candidate models grows exponentially with the number of bound processes, so IPM uses heuristic search to explore the space of candidate solutions.[2] Our implementation of heuristic search incorporates a simplicity bias, which means that IPM considers models with fewer processes first. In general, this strategy follows the procedure described earlier in that it takes a set of candidate solutions, keeps a few that perform the best, and adds processes to those during the next round of exploration. Search halts when exploration fails to produce any solutions better than those previously considered. In many cases, this *beam search* approach quickly finds high-quality solutions, even though its simplicity bias is naive for IPM's nonlinear modeling task.[3]

We have applied IPM with considerable success in task domains as varied as aquatic ecosystems, biochemical kinetics, and molecular biology (Asgharbeygi, Bay, Langley, & Arrigo, 2006; Langley, Shiran, Shrager, Todorovski, & Pohorille, 2006), but here we discuss an example that illustrates one of its weaknesses (Bridewell et al., 2008): The system sometimes constructs scientifically implausible models and prefers them over more plausible ones that have lower fitness scores. The modeling problem involves predicting the water level in Denmark's Ringkøbing Fjord. The data consist of a trajectory for the observed water level recorded hourly for roughly 1 year, which the model should predict, and trajectories for the water level of the open sea, the influx of fresh water, wind velocity and direction, and the number of gate segments that were opened to allow water exchange between the estuary and the open sea.

We gave IPM a partial solution and required it to explain the gate and wind influences using a selection of 16 generic processes. The library's size leads to over 65,000 model structures whose evaluation would amount to several days of computation. Although beam search would explore a reduced set of candidates, it cannot rule out the implausible structures. Left to the task, IPM will spend hours or days of computation on these models from which a practically meaningless solution could eventuate. This drawback suggests that the system lacks knowledge that scientists would bring to the task.

To determine what type of knowledge to add, we thought about how generic processes limit the space of candidate models. Consider an equation discovery system, such as LAGRAMGE (Todorovski, 2003), that searches the space of ordinary differential equations. Whereas IPM generates solutions by adding processes to a model, LAGRAMGE adds terms to individual equations. As a result, when applied to the problem of foxes and rabbits from Table 2, it would turn out candidate solutions where the fox population increases through predation,

but the rabbit population grows unchecked. By operating at the level of processes instead of individual terms, IPM ensures not only that predation terms appear in either both equations or neither of them but also that the numeric parameters in those terms correspond correctly. Taking this view, the generic processes act as constraints that rule out theoretically implausible sets of equations.[4]

Now imagine constraints that control which processes can appear in a model. For example, we could state that exactly one predation process must appear in the model of foxes and rabbits. In the Ringkøbing Fjord example, we asserted that certain wind-related processes mutually excluded others. This additional knowledge reduced the number of candidate structures from over 65,000 to 1,280. Of course, these constraints could have lowered the solution's fitness. To evaluate this, we compared scores from IPM with those reported for LAGRAMGE (Todorovski, 2003), which searched a larger space of differential equations. For a model trained over the first half of the data set, the best model found by IPM had a root mean squared error (RMSE) of 0.052 and a coefficient of determination (r^2) of 0.421. In comparison, LAGRAMGE's model, which was trained over the full data set, had an RMSE of 0.059 and an r^2 of 0.434.[5] The difference in scores was minimal, with a negligible decrease in error, while the savings in search was marked. A series of similar results encouraged us to develop new types of constraints to guide IPM's heuristic search.

3. Constraints on inductive process modeling

The advantages of constraints that we observed when using IPM led us to consider both a new approach to model generation and an additional form of domain knowledge. Specifically, we designed modular constraints that complement the generic process formalism and let scientists control the structure of the models that are considered. We incorporated these constraints into a new system called SC-IPM—an acronym for ''Satisfying Constraints to Induce Process Models''—that produces only those models supported by this new knowledge.

3.1. Constraints in satisfying constraints to induce process models

Analysis of the implausible models produced by IPM suggested four types of constraints that we incorporated into SC-IPM: *necessary*, *always-together*, *at-most-one*, and *exactly-one*. Each constraint expresses a relationship among generic processes that refines the space of candidate models. Of these, the *necessary* constraint asserts that at least one instantiation of a given generic process must appear in a model. For example, this type of constraint could ensure that a predation model includes an exponential loss process or an instantiation of a particular predation process (e.g., Holling type I). The *always-together* constraint creates a relationship among its generic processes by declaring that a model must instantiate either all of them or none of them. This relationship is useful in ecosystem models that include nutrient-limited primary production where the nutrient-limited growth process appears if and only if there is a corresponding nutrient absorption process.

Table 4
Constraints from an SC-IPM library for ecosystem models

```
constraint growth_alternatives
    type exactly-one
    processes exponential_growth(P), logistic_growth(P), limited_growth(P)

constraint grazing_alternatives
    type at-most-one
    processes ivlev, ratio_dependent(P,G), holling_type_1(P,G), holling_type_2(P,G)

constraint nutrient_limited_growth
    type always-together
    processes limited_growth(P), nutrient_limitation(P,N)

constraint optional_light_limitation
    type at-most-one
    processes exponential_growth(P), logistic_growth(P), light_limitation(P,L)

constraint mandatory_loss
    type necessary
    processes exponential_loss(P), degradation(P)
```

Note. The arguments to the processes map to instantiations of generic entities. P is primary producer, N is nutrient, G is grazer, and L is light.

SC-IPM's other two constraints let users assert that two or more generic processes are mutually exclusive. As suggested by its name, the *at-most-one* constraint ensures that no more than one instantiation of a generic process appears within a model. This addition makes the inclusion of alternative functional forms within a process library more practical. For example, one can preclude ecosystem models where the same two species engage in both symbiotic and parasitic interactions. The final constraint, which we call *exactly-one*, combines *at-most-one* and *necessary* to define a mutually exclusive group of generic processes where one of the alternatives must appear in the model. Table 4 gives an example process library for ecosystem models, omitting the details of the generic processes to focus on the constraints.

As described, SC-IPM's four constraints limit the instantiation of generic processes within a model without qualification, which has limited usefulness in models with more than one instantiation of a generic process. For instance, an unqualified version of the growth_alternatives constraint in Table 4 would limit each model to one of these three processes even if there were multiple primary producers. To relax the constraints, users state that processes must be uniquely instantiated over some set of entities. Reflecting this, the growth_alternatives that appears in Table 4 allows one growth process for each producer. Likewise, the grazing_alternatives constraint ensures that each producer–grazer pairing has at most one grazing process.

To develop a system that respects these constraints, we extended the part of IPM that generates candidate model structures. Specifically, we turned to approaches for solving constraint satisfaction problems. Methods from this paradigm begin by translating statements in

higher order logics to a propositional representation. For SC-IPM, this takes place when entities are bound to the generic processes, creating potential components for the process models. From the perspective of propositional logic, each component is a Boolean variable, and the constraints define the sentences that contain these variables. Given a logical sentence, the constraint satisfier attempts to find values for all the variables that will make the sentence true. If a variable is *false*, then the associated component must not appear in the candidate model, if *true*, then it must appear, and otherwise the component is considered *free* and its inclusion does not affect the plausibility of the model structure. For example, given the bound processes $p_1,\ldots,p_5$, a set of constraints could lead to the Boolean sentence $p_1 \wedge \neg p_2 \wedge (p_4 \vee p_5)$. In this case, p_1 must appear in all plausible models, p_2 can never appear, at least one of p_4 and p_5 must appear, and p_3 is a free variable, which leads to six valid structures.

After SC-IPM transforms the constraints into a propositional sentence, it can use available methods for constraint satisfaction that employ either traditional heuristic search (Davis, Logemann, & Loveland, 1962) or local search (Selman, Kautz, & Cohen, 1996). To be efficient in time, both technologies expect the logical sentence to be in conjunctive normal form (CNF).[6] In general, transforming an arbitrary logical sentence into CNF can result in an exponential increase in clauses particularly in the case of mutually exclusive relationships. However, we developed specialized translation functions for the constraints in SC-IPM that guard against an excessive number of clauses. After translating the constraints into CNF, the system applies the WalkSAT algorithm (Selman et al., 1996) to generate candidate model structures.

This strategy for model construction differs from the heuristic search used by IPM, but it enables a modified version of beam search. To implement this approach, we added a new, two-stage procedure for model generation. In the first stage, SC-IPM generates a candidate structure that satisfies all the constraints, leaving the free variables set to false. This step yields a plausible model that includes a minimal subset of the available components. The second stage expands this structure by including subsets of the unconstrained components, essentially adding processes one at a time to balance model complexity with predictive accuracy. By merging techniques for constraint satisfaction and beam search, we reduce search both by ruling out scientifically implausible models and by focusing the exploration on candidates that are potentially accurate.

In Section 2.2, we claimed that scientists construct explanatory models by creating an initial set of models and refining the members of that set by adding or removing processes. SC-IPM extends this claim to suggest that scientists have broadly defined constraints that ensure that their models maintain scientific plausibility. However, we expect that they use incremental strategies to develop their models as opposed to the local search approaches that the system implements. We also suggest that when presented with models, scientists carry out a form of constraint checking to ensure that the relationships among the entities and processes match their knowledge of the problem domain. To this extent, SC-IPM refines the cognitive model of IPM to make a more specific claim about the scientific activity.

3.2. Computational experiments on constrained model induction

In Section 2.3, we described computational experiments that showed how constraints on the structure of candidate models affected the amount of search in IPM and the predictive accuracy of its solutions. As a follow-on, we also ran experiments with an early prototype of SC-IPM that incorporates the same type of constraints (Todorovski, Bridewell, Shiran, & Langley, 2005). Like SC-IPM, this system starts with minimal plausible structures and refines them using beam search. We report results on three problems: one that uses synthetic data and two others that use actual scientific data. In every case, the constraints reduced the total amount of search and prevented the acceptance of implausible models. In five of eight trials, the extra knowledge also increased the predictive accuracy of the best models.

The first set of experiments used synthetic data generated from an ecosystem-style model that included four entities and nine processes. In every scenario, the structural constraints reduced search by at least a factor of seven and improved predictive accuracy on separate test data. In addition, all the constrained models had plausible structures, whereas the best models produced by IPM were implausible. Furthermore, the SC-IPM prototype accurately reconstructed the original model structure in a majority of the trials. In the cases where the system did not reconstruct the source model, its solution differed by only a single process.

The second set of computational experiments used data from a predator–prey system involving microscopic species (Veilleux, 1979), whereas the third used data related to phytoplankton dynamics in the Ross Sea (Arrigo et al., 2003). For these domains, the overall reduction in search varied between 7- and 19-fold, much as we had expected, but the error measures told different stories. In the Ross Sea domain, the constraint-aware system substantially outperformed IPM in accuracy with r^2 values ranging from 0.06 to 0.16 points higher. In contrast, the constraints reduced r^2 in the predator–prey domain by more than 0.2 points in each case. Curious about this result, we looked more closely at the models that each system produced. Doing so revealed that none of the models produced by IPM were plausible given expert-derived domain knowledge because they either included mutually exclusive processes or excluded necessary ones. Our tentative conclusion was that the library of generic processes either lacked the appropriate functional forms or that the parameter ranges were too restrictive. Subsequent studies with the same data sets support the latter explanation.

SC-IPM extends previous work on inductive process modeling to include structural constraints that help prevent the consideration of implausible models. Results with a precursor to SC-IPM support the hypothesis that these constraints add quantitative and qualitative value to the modeling system. Although we left out several details as we recounted these results, each case shows that the addition of constraints on generic processes substantially reduces search, leads to scientifically plausible models, and can improve the predictive accuracy of proposed solutions.

4. Related work

Inductive process modeling is a recent contribution to the literature on computational scientific discovery. Inspired by previous work, we investigated this task with a system that carries out heuristic search through a problem space of candidate models. Guided by time-series data, the system composes mechanistic models from a library of generic processes that delineate its search space. Unfortunately, this approach entertained scientifically implausible candidates that sometimes fit the data quite well. Regardless of their quantitative accuracy, these models provide poor explanations of the phenomena of interest and should be excluded from consideration.

The need to further restrict the search space led us to introduce constraints on model structures. Specifically, the additional knowledge restricts the processes that may appear in a candidate solution. This type of information forms a key part of scientific knowledge and helps focus a discovery system's attention on plausible models. To take advantage of these constraints, we developed a language for representing them and incorporated it into a new program called SC-IPM.

Inductive process modeling builds on a long line of work in fields as varied as equation discovery and qualitative physics. Our research grew directly out of Todorovski and Džeroski's (1997) work on equation discovery and the LAGRAMGE system. Starting from the premise that discovery systems spend considerable time exploring implausible equations, they investigated ways to incorporate domain knowledge that excludes nonsensical candidates. Their approach treated equations as if they were sentences, appealing to context-free grammars as a means to limit the search space. Although LAGRAMGE met this goal, its knowledge representation was foreign to domain scientists, which led to an emphasis on process models (Langley et al., 2002).

Our focus on processes as an important type of scientific knowledge stems both from their use in the everyday language of scientists (Haefner, 2005; Jørgensen & Bendoricchio, 2001) and from work in the area of qualitative physics (Forbus, 1984) and compositional modeling (Falkenhainer & Forbus, 1991). In particular, the latter also constructed models from a library of processes and entities and included support for quantitative simulation. However, it created these models only in response to a query about the behavior of a specified physical system, and it searched for the simplest model that could answer this query. By contrast, our methods for inductive process modeling create models that aim to explain how observed variables change over time, which raises quite different challenges.

Although the value of search constraints is well known, their role in modeling has rarely been addressed. Nersessian's (2008) latest book serves as a welcome exception. She identifies several types of constraints that inform model construction, distinguishing them based on their source. Reflecting our background in computational modeling, we instead draw a distinction between constraints that refer to a model's behavior and those that refer to its structure. For example, SC-IPM can quickly tell whether a model contains all the necessary processes, but determining whether it produces oscillations requires parameter estimation and simulation, which accounts for over 99% of the computational time in SC-IPM. Bradley

and Stolle's (1996) PRET, an equation discovery system, bridges this distinction with constraints that relate an equation's structure to its behavior.

Perhaps the closest in spirit to the reported research is MECHEM (Valdés-Pérez, 1995), which inputs knowledge of chemistry in the form of solution constraints and produces chemical reaction pathways. As in SC-IPM, that system's constraint language was modular, but the input was problem specific in that users would enter constraints relevant to a particular chemical reaction. By contrast, SC-IPM focuses on theory-level constraints rather than problem-level ones. This distinction is important in that constraints expressed at the theory level are general enough to transfer across modeling tasks.

The importance of theory-level constraints and the little attention they have received leave the door open for future research. For instance, we intend to follow MECHEM's example and include problem-level constraints in SC-IPM that can refine the space of candidate models based on scenario-specific considerations. Additionally, we are investigating approaches for learning modeling constraints. We have obtained promising initial results in this direction (Bridewell, Borrett, & Todorovski, 2007; Bridewell & Todorovski, 2007), and we are currently exploring inferred constraints that generalize across problems and across task domains.

In this paper, we highlighted the role of generic processes and model constraints in scientific discovery. The distinction between the components that comprise models and the constraints that inform their construction suggest new avenues for the development of discovery systems, the acquisition of expert knowledge, and the analysis of human scientific activities. Moreover, from the perspective of cognitive modeling, we discussed how these types of knowledge relate to the information that scientists use when they model dynamic systems and how the search methods in IPM and SC-IPM relate to scientific discovery by humans.

Notes

1. Thagard (2003) developed a similar approach based on *explanation schemas* that suggest both the pattern of the explanation and the questions it addresses.
2. The program also supports exhaustive search, which is often impractical due to the number of candidate models.
3. Minor changes in a nonlinear model may result in dramatic changes in its predictions. As a result, in general we can expect little correlation between a model's fitness and its fitness after adding a process.
4. To avoid misinterpretation, we note that the explanatory content of the model stems from its relationship to scientific concepts and not from the equations themselves. Equations without a theoretical interpretation provide a description of system dynamics, but we are reluctant to call them explanations.
5. The root mean squared error is the average absolute error of each prediction. The coefficient of determination for IPM is the square of the correlation coefficient; so, values

range between 0 and 1, with larger values indicating that the model explains more of the observed variance.

6. Conjunctive normal form is a sequence of disjunctive (i.e., logical 'or') clauses combined with the logical 'and' operator.

Acknowledgments

This research was supported by grant IIS-0326059 from the National Science Foundation. We thank Matt Bravo and Ljupčo Todorovski for their contributions to the IPM and SC-IPM systems and their role in developing the constraint language.

References

Arrigo, K. R., Worthen, D. L., & Robinson, D. H. (2003). A coupled ocean–ecosystem model of the Ross Sea: 2. Iron regulation of phytoplankton taxonomic variability and primary production. *Journal of Geophysical Research – Oceans*, *108*, 24-1–24-17.

Asgharbeygi, N., Bay, S., Langley, P., & Arrigo, K. (2006). Inductive revision of quantitative process models. *Ecological Modelling*, *194*, 70–79.

Atanasova, N., Todorovski, L., Džeroski, S., & Kompare, B. (2006). Constructing a library of domain knowledge for automated modelling of aquatic ecosystems. *Ecological Modelling*, *194*, 14–36.

Bechtel, W., & Abrahamsen, A. (2005). Explanation: A mechanist alternative. *Studies in the History and Philosophy of Biology and Biomedical Sciences*, *36*, 421–441.

Bradley, E., & Stolle, R. (1996). Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artificial Intelligence*, *17*, 1–28.

Bridewell, W., Borrett, S., & Todorovski, L. (2007). Extracting constraints for process modeling. In D. H. Sleeman & K. Barker (Eds.) *Proceedings of the fourth international conference on knowledge capture* (pp. 87–94). Whistler, British Columbia: ACM Press.

Bridewell, W., Langley, P., Todorovski, L., & Džeroski, S. (2008). Inductive process modeling. *Machine Learning*, *71*, 1–32.

Bridewell, W., Sánchez, J. N., Langley, P., & Billman, D. (2006). An interactive environment for the modeling and discovery of scientific knowledge. *International Journal of Human–Computer Studies*, *64*, 1099–1114.

Bridewell, W., & Todorovski, L. (2007). Learning declarative bias. In H. Blockeel, J. Ramon, J. Shavlik & P. Tadepalli (Eds.), *Proceedings of the seventeenth annual international conference on inductive logic programming* (pp. 63–77). Corvallis, OR: Springer.

Cartwright, N. (1999). Models and the limits of theory: Quantum Hamiltonians and the BCS models of superconductivity. In M. Morrison & M. S. Morgan (Eds.), *Models as mediators* (pp. 241–281). Cambridge, England: Cambridge University Press.

Cartwright, N., Shomar, T., & Suárez, M. (1995). The toolbox of science. In W. E. Herfel, K. Krajewski, I. Niinilvoto & R. Wojcicki (Eds.), *Theories and models in scientific processes* (pp. 137–149). Amsterdam: Rodopi.

Darden, L. (2002). Strategies for discovering mechanisms: Schema instantiation, modular subassembly, forward/backward chaining. *Philosophy of Science*, *69*, S345–S365.

Davis, M., Logemann, G., & Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, *5*, 394–397.

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

Feigenbaum, E. A., Buchanan, B. G., & Lederberg, J. (1971). On generality and problem solving: A case study using the DENDRAL program. *Machine Intelligence*, *6*, 165–190.

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

Forbus, K. D. (1997). Qualitative reasoning. In A. B. Tucker (Ed.), *The computer science and engineering handbook* (pp. 715–733). Boca Raton, FL: CRC Press.

Haefner, J. W. (2005). *Modeling biological systems: Principles and applications*. New York: Springer.

Jørgensen, S. E., & Bendoricchio, G. (2001). *Fundamentals of ecological modelling*. New York: Elsevier.

Jost, C., & 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.

Keppens, J., & Shen, Q. (2001). On compositional modeling. *Knowledge Engineering Review*, *16*, 157–200.

Klahr, D. (2002). *Exploring science*. Cambridge, MA: MIT Press.

Kulkarni, D., & Simon, H. A. (1990). Experimentation in machine discovery. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 255–273). San Mateo, CA: Morgan Kaufmann.

Langley, P., Sánchez, J. N., Todorovski, L., & Džeroski, S. (2002). Inducing process models from continuous data. In C. Sammut & A. G. Hoffman (Eds.), *Proceedings of the nineteenth international conference on machine learning* (pp. 347–354). Sydney: Australia.

Langley, P., Shiran, O., Shrager, J., Todorovski, L., & Pohorille, A. (2006). Constructing explanatory process models from biological data and knowledge. *Artificial Intelligence in Medicine*, *37*, 191–201.

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

Machamer, P. K., Darden, L., & Craver, C. F. (2000). Thinking about mechanisms. *Philosophy of Science*, *67*, 1–25.

Mahidadia, A., & Compton, P. (2001). Assisting model-discovery in neuroendocrinology. In K. P. Jantke & A. Shinohara (Eds.), *Proceedings of the fourth international conference on discovery science* (pp. 214–227). Washington, DC: Springer.

Nersessian, N. (2008). *Creating scientific concepts*. Cambridge, MA: MIT Press.

Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the ACM*, *19*, 113–126.

Pazzani, M., & Flowers, M. (1990). Scientific discovery in the layperson. In J. Shrager & P. Langley (Eds.), *Computational models of scientific discovery and theory formation* (pp. 403–435). San Mateo, CA: Morgan Kaufmann.

Selman, B., Kautz, H., & Cohen, B. (1996). Local search strategies for satisfiability testing. In D. S. Johnson & M. A. Trick (Eds.), *Cliques, coloring, and satisfiability*. (pp. 521–532). Providence, RI: American Mathematical Society.

Suárez, M., & Cartwright, N. (2008). Theories: Tools versus models. *Studies in History and Philosophy of Modern Physics*, *39*, 62–81.

Thagard, P. (2003). Pathways to biomedical discovery. *Philosophy of Science*, *70*, 235–254.

Todorovski, L. (2003). *Using domain knowledge for automated modeling of dynamic systems with equation discovery*. Doctoral dissertation, Faculty of computer and information science, University of Ljubljana, Ljubljana, Slovenia.

Todorovski, L., Bridewell, W., Shiran, O., & Langley, P. (2005). Inducing hierarchical process models in dynamic domains. In M. Veloso & S. Kambhapati (Eds.), *Proceedings of the twentieth national conference on artificial intelligence* (pp. 892–897). Pittsburgh, PA.

Todorovski, L., & Džeroski, S. (1997). Declarative bias in equation discovery. In D. H. Fisher (Ed.), *Proceedings of the fourteenth international conference on machine learning* (pp. 376–384). Nashville, TN: Morgan Kaufmann.

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

Veilleux, B. G. (1979). An analysis of predatory interaction between *Paramecium* and *Didinium*. *The Journal of Animal Ecology*, *48*, 787–803.

Zupan, B., Bratko, I., Demšar, J., Juvan, P., Kuspa, A., Halter, J. A., & Shaulsky, G. (2007). Discovery of genetic networks through abduction and qualitative simulation. In S. Džeroski & L. Todorovski (Eds.), *Computational discovery of scientific knowledge* (pp. 228–247). Berlin: Springer.

Żytkow, J. M. (1999). Model construction: Elements of a computational mechanism. In S. Cotton (Ed.), *AISB'99 symposium on AI and scientific creativity*. Edinburgh, Scotland.

Żytkow, J. M., Zhu, J., & Hussam, A. (1990). Automated discovery in a chemistry laboratory. T. Dietterich & W. Swertout (Eds.), *Proceedings of the eighth national conference on artificial intelligence* (pp. 889–894). Boston: AAAI Press.