Flexible Model Induction through Heuristic Process Discovery

Pat Langley
Institute for the Study of Learning and Expertise
2164 Staunton Court, Palo Alto, CA 94306

Adam Arvay
Department of Computer Science
University of Auckland, Auckland 1142 NZ

Abstract

Inductive process modeling involves the construction of explanatory accounts for multivariate time series. As typically specified, background knowledge is available in the form of generic processes that serve as the building blocks for candidate model structures. In this paper, we present a more flexible approach that, when available processes are insufficient to construct an acceptable model, automatically produces new generic processes that let it complete the task. We describe FPM, a system that implements this idea by composing knowledge about algebraic rate expressions and about conceptual processes like predation and remineralization in ecology. We demonstrate empirically FPM's ability to construct new generic processes when necessary and to transfer them later to new modeling tasks. We also compare its failure-driven approach with a naive scheme that generates all possible processes at the outset. We conclude by discussing prior work on equation discovery and model construction, along with plans for additional research.

1 Background and Motivation

Research on computational scientific discovery (Shrager and Langley 1990; Džeroski and Todorovski 2007) attempts to reproduce the human ability to infer scientific laws and models, but there is considerable variety in this arena. Initial work on the topic focused on induction of descriptive laws, which typically occurs early in a scientific field's history and appears to involve knowledge-lean mechanisms. More recent research has addressed the construction of explanatory models, which usually occurs at more advanced stages and draws on domain knowledge. Both efforts differ from mainstream work in machine learning by using established scientific formalisms, but their concerns can differ considerably.

An important class of problems in the knowledge-lean tradition is *equation discovery* (e.g., Langley et al. 1987), where the aim is to induce numerical relations among observed variables. An equally important subarea in the knowledge-rich paradigm is *inductive process modeling* (Bridewell et al. 2008), which attempts to construct explanations of multivariate time-series data in terms of interactions among unobserved but plausible processes that produce quantitative changes in measured variables.

At first glance, these two tasks appear closely related, as they both generate equations, but the latter also produces a deeper account that moves beyond empirical descriptions. Nevertheless, process model induction relies heavily on background knowledge that must currently be entered manually. A more flexible approach would attempt to introduce new processes when an initial set is inadequate to explain observations, partially bridging the gap between knowledge-lean and knowledge-rich scientific discovery.

In the following sections, we review prior work on inductive process modeling, including its reliance on background knowledge about domain processes. In response, we define the more challenging task of model induction when process knowledge is incomplete. We describe an approach to generating candidate processes, an exhaustive method for using them in model construction, and a heuristic alternative that holds apparent advantages. We report experiments with these techniques on ecological time series that demonstrate basic functionality and examine their relative efficiency and reliability. In closing, we examine relations to prior research and outline priorities for future work.

Research on process model induction is atypical for the cognitive systems community due to its focus on numeric data and use of statistical analysis. However, work in this area addresses a high-level task that humans find challenging, uses structured representations of background knowledge and models, engages in multi-step reasoning over these structures, and draws on heuristics to guide search. All of these are features of cognitive systems research (Langley 2012), making it highly relevant to the paradigm.

2 Recent Work on Inductive Process Modeling

As noted, process model induction involves constructing an explanation of multivariate time series in terms of background knowledge. The input is a set of typed variables, observations for each variable over time, and a set of generic processes that *might* be responsible. The output is a set of linked differential equations, each composed of one or more processes, that reproduce and explain the data.

Table 1 presents a model for a four-variable predator-prey system, with (a) displaying the five component processes and (b) showing four differential equations. Note that some

Copyright © 2017, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

processes appear in multiple equations; this sharing is a key element of process accounts. Each process has a name, an algebraic rate expression, one or more derivatives that are proportional to this rate, and coefficients that relate these terms. Process rates, which vary over time, are similar to those in chemical reactions. The effects of multiple processes on a given derivative are additive, making translation into differential equations straightforward. Figure 1 displays trajectories for the four variables by these linked equations.

Research on process model induction has made substantial progress since it appeared over a decade ago. The framework has been applied to aquatic ecosystems (Asgharbeygi et al. 2006), hydrology (Bridewell et al. 2008), and biochemistry (Langley et al. 2006). Recent work (Langley and Arvay 2015) has introduced a more constrained notation for process models, similar to that shown in Table 1, that keeps rate expressions distinct from the derivatives that are proportional to them. Briefly, their RPM system calculates, for each time step, the derivative for each variable and the rates for candidate processes; it then uses multiple linear regression to estimate the coefficients for hypothesized sets of processes that map onto model equations. This approach is both more reliable and less costly than gradient descent, which had been used previously. Their RPM system ran 83,000 times faster than its predecessor and it was far more likely to find accurate and plausible models.

In more recent work, Arvay and Langley (2016) have reported another approach that replaces greedy search through the space of process models with a two-stage strategy. Their SPM system first induces multiple differential equations for each variable, with each term in the linear expression mapping onto a process. It then carries out depth-first search in an effort to find sets of equations that incorporate consistent processes. Experimental results suggest that this technique is much more likely to identify consistent process models in domains like chemistry, where each variable's derivative can be predicted well by many different equations but where only a few of their combinations are coherent.

We have noted that process models move beyond equations to explain observations in terms of the familiar concepts like predation, growth, and loss. This requires background knowledge about processes that might underlie the data. The paradigm assumes this content is stated as *generic* processes, which are similar to those in models but which replace specific variables with typed elements, concrete rate expressions with abstract ones, and numeric cofficients with constraints on their values. Proponents of statistical learning find this dependence on handcrafted structures unappealing, although it is clear that humans draw on domain knowledge when attempting to understand scientific data. Nevertheless, increasing the flexibility of process modeling is a reasonable goal and automating, at least partially, the discovery of generic processes is a natural response. We focus on this challenge in the sections that follow.

3 Constructing New Generic Processes

As just noted, we would like an approach to model induction that is less dependent on handcrafted background knowledge about candidate processes. In this section, we define a revised task – *flexible* process modeling – that includes creating new generic processes from more basic elements. We present a computational technique for composing such building blocks, along with two mechanisms that use these constructs during model induction. One relies on a naive exhaustive scheme, whereas the other employs an incremental heuristic method that seems likely to be more effective.

Table 1: A five-process model for a predator-prey system in which four organisms ($x1$ through $x4$) interact through a linear food chain. Each process has a rate determined by an algebraic expression and one or more derivatives that are proportional to this rate. The table also shows an analogous set of differential equations.

```
(a) growth[x1]
      rate         r = x1
      equations    d[x1] =  0.08 · r
    predation1[x2, x1]
      rate         r = x1 · x2
      equations    d[x1] = -0.03 · r
                   d[x2] =  0.04 · r
    predation1[x3, x2]
      rate         r = x2 · x3
      equations    d[x2] = -0.01 · r
                   d[x3] =  0.01 · r
    predation2[x4, x3]
      rate         r = x3 / x4
      equations    d[x3] = -0.02 · r
                   d[x4] =  0.03 · r
    loss[x4]
      rate         r = x4
      equations    d[x4] = -0.03 · r
```

```
(b) d[x1] = 0.08 · x1 + -0.03 · x1 · x2
    d[x2] = 0.04 · x1 · x2 + -0.01 · x2 · x3
    d[x3] = 0.01 · x2 · x3 + -0.02 · x3 / x4
    d[x4] = 0.03 · x3 / x4 + -0.03 · x4
```

3.1 Task Specification

Like previous authors, we view process modeling as moving beyond replication and prediction of time series to constructing *explanations* of observed values in terms of background knowledge. We can specify the modified task of *flexible* process modeling in terms of its inputs and outputs:

- *Given:* A set of typed variables and observed trajectories of their values over time;
- *Given:* A set of generic processes with type constraints and rate terms that might appear in the explanations;
- *Given:* A set of rate expressions and conceptual relations that can serve as process constituents;
- *Find:* A model that uses these processes to explain variables' trajectories.
- *Find:* New generic processes that are useful for the current and future data.

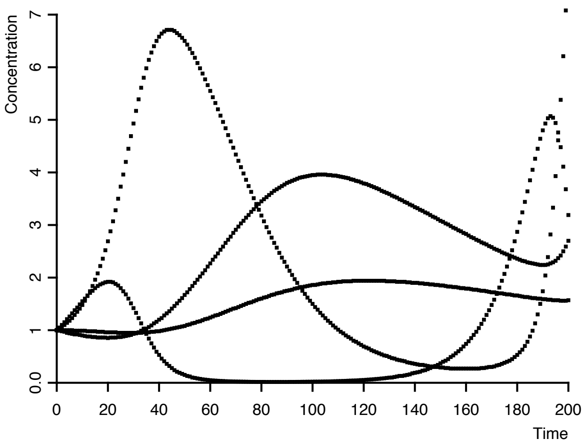

Figure 1: Observed trajectories for a four-variable predator-prey system that obeys the process model in Table 1.

Table 2: Four generic processes that are relevant to modeling predator-prey ecosystems.

```
gain
  :variables    ((x organism))
  :rate          x
  :constraints  ((> par 0)
predation1
  :variables    ((x organism)(y organism))
  :rate          x · y
  :constraints  ((< par 0)(> par 0))
predation2
  :variables    ((x organism)(y organism))
  :rate          x / y
  :constraints  ((< par 0)(> par 0))
loss
  :variables    ((x organism))
  :rate          x
  :constraints  ((< par 0)
```

This statement assumes the induction system is provided with only some generic processes that will prove useful for the observations. Instead, it must discover other, novel processes that aid its explanation on both the current modeling task and ones it may encounter later.

However, one cannot conjure candidate processes from thin air; they can only be constructed from more basic elements. Our notation for processes suggests two types of components: algebraic rate expressions and conceptual relations among variables. For instance, consider the four generic processes shown in Table 2, for gain, predation, and loss, that can be used to induce the model we already examined in Table 1. Each generic process specifies a set of typed variables that should appear as derivatives in equations, an algebraic expression that determines the rate, and constraints on coefficients that relate the rate to each derivative.

The *predation1* and *predation2* processes each involve two organisms, while *gain* and *loss* refer to only one variable of this type. The rate for *predation1* is the product of two organisms' populations, while that for *predation2* is the ratio of the predator's population to that of the prey. In contrast, the rates for *gain* and *loss* are determined by the population of a single organism. The parameter constraints, which indicate whether the process consumes or produces each variable, also differ across the four structures. For instance, the first parameter of *predation1* must be negative and the second must be positive. These relate to coefficients in the *equations* field for instances of this process, as in Table 1.

Table 3: (a) Conceptual relations and (b) algebraic rate templates that serve as components of generic processes.

```
(a) prey
      :variables ((a organism)(b organism))
      :inputs    (b)
      :outputs   (a)
     inflow
      :variables ((a organism))
      :inputs    ( )
      :outputs   (a)
(b) identity
      :expression p
     product
      :expression p · q
     ratio
      :expression p / q
```

We will assume here that the flexible process modeler is provided with conceptual relations that might occur in the domain, like those in Table 3 (a), and algebraic templates for possible rate expressions, like those in Table 3 (b). One can unify such structures to produce the generic processes in Table 2, as well as others. For instance, the *prey* relation and *product* rate in the former can combine to form the *predation1* entry in the latter, while combining *prey* with *ratio* can produce *predation2*. Any generic process in our framework can be broken down into these two elements, which suggests a natural approach to generating the former from the latter.

3.2 Generating Candidate Processes

Given a set of conceptual relations among typed variables and a set of algebraic rate templates like those in Table 3, there is a simple method for combining them to construct generic processes like those in Table 2. This involves examining all possible pairs of conceptual relations and rate templates. For each relation C and template R, one must find all ways in which the variables in R can unify with the typed variables in C. For instance, there are two ways to compose the *prey* relation and the *ratio* template in Table 3, giving the rate expressions a/b and b/a. In the first, the rate is the predator (output) population divided by the prey (input) population; in the second it is the inverse, giving the process *predation2* after replacing variable names.

The number of processes generated in this manner can be substantially greater than $c \cdot r$, where c is the number of conceptual relations and r is the number of rate templates. Despite its combinatorial character, automating this mechanism is both straightforward and tractable for reasonable numbers of relations and rate templates, say 20 of each type. We will refer to the algorithm as GCP, for 'generation of candidate processes'. The only subtlety, which we will not detail here, concerns the notion of symmetry sets. If two inputs or outputs, a and b, for a conceptual relation have the same type, and if both participate in a symmetrical rate expression like $p \cdot q$, one should not distinguish between a generic process in which the rate expression is $a \cdot b$ and another in which it is $b \cdot a$, as they are equivalent mathematically.

We should note that it is also possible to generate a set of conceptual relations like those in Table 2 from known variable types, provided there is a limit on the number of variables in each structure. Similarly, one could automatically construct rate templates from a set of arithmetic operators like multiplication, division, addition, and subtraction, again provided some limit on expression complexity. This scheme would depend slightly less on user input, but it would not differ materially from the approach we have described.

3.3 A Naive Approach to Process Discovery

Once a process modeler has generated a set of generic processes in the manner just described, it can pass them to an existing system, such as SPM (Arvay and Langley 2016), that searches a space of candidate models. However, this involves instantiating the generic structures with observed variables to generate process instances. When the data set involves many such variables, the number of process instances produced in this manner can be daunting. For instance, ten variables and five generic processes that each relate two variables of the same type will produce $180 \cdot 5 = 900$ process instances for consideration during model induction.

A more reasonable approach would initially attempt to find an explanation for the data using a set of user-provided generic processes and, if these fail to produce a consistent model, only then generate new structures by composing conceptual relations with rate templates. We have implemented a naive system of this sort that first invokes the SPM program to find one or more process models with available processes. If this does not succeed, it calls on the GCP module to generate an expanded set of generic processes from components like those in Table 3. The approach is analogous to work on model revision (Asgharbeygi et al. 2006), but it occurs at the level of generic processes rather than process instances.

Given enough computational resources, this technique should, in principle, induce reasonable models even when its initial set of generic processes omits some relevant structures. However, when many process instances are being considered, the SPM module must sample sets of processes to identify relevant terms for its differential equations. Considering all generic processes that are composable from conceptual relations and rate templates will produce many irrelevant process instances that reduce the chances of finding model equations or require large numbers of samples.

3.4 A Heuristic Approach to Process Discovery

In response, we have also developed a heuristic approach, which we call FPM (for Flexible Process Modeling) that uses information about where model induction fails to focus attention. Recall that the SPM module operates in two stages. The first phase induces one set of alternative differential equations for each variable whose trajectory it aims to explain. The second attempts to find sets of these equations, each containing one equation per variable, that include consistent sets of processes. For instance, the *predation2* process instance in Table 1 influences both *d[x3]* and *d[x4]*. This means that, for it to appear in a final model, it must contribute to the equations for both derivatives.

The SPM routine carries out depth-first search through the space of candidate models defined by all combinations of component equations. A path downward through the search tree can fail at level d either because induction has not found any equation for variable x_d or because none of the equations for x_d are consistent in terms of processes with equations earlier along the path. By collecting the variables at these failed nodes, FPM can identify where new processes might break an impasse. For this, it adopts the heuristic of selecting the variable associated with the deepest node as the focus of attention, as it came closest to a consistent model.

FPM then uses this variable to constrain the process instances considered during future processing. It must still examine all new generic processes generated by the GCP method specified earlier, but it passes to SPM only those process instances that include the focus variable. For instance, suppose that, on a ten-variable modeling task, FPM finds no consistent sets of equations beyond variable x_5 and suppose GCP has produced candidate generic processes that influence two *organism* variables each. During the next attempt at model induction, FPM would add to the pool only process instances that mention x_5, which are approximately one ninth of the total number. Given SPM's reliance on sampling to induce individual equations, this should give much higher chances of finding candidates that contribute to a consistent model. If FPM finds complete models or cannot proceed beyond the problem variable, then it halts. Otherwise, it retains generic versions of the process instances that helped extend its partial models to include more variables.

This last step is important because the original set of generic processes may omit structures essential for more than one variable. If so, then FPM will make multiple calls on SPM before it augments its library enough to find complete models. For example, it might start with one generic predation process whose rate expression is $a \cdot b$, but find this insufficient to find an equation for $d[x_5]$. Expanding the set to include newly generated processes may produce an equation for this variable with an alternate predation process with the rate a/b that it useful for $d[x_7]$, but SPM may still encounter dead ends when it considers variable x_8. Further search may reveal a third process that lets it find an equation for this variable and a complete model. Both generic processes would be added to the default set for use in future modeling tasks, with those not used kept in the background.

3.5 Implementation Details and Assumptions

We have implemented both of these approaches in Steel Bank Common Lisp, using Arvay and Langley's (2016) SPM program as a subroutine for inducing consistent process models and their associated differential equations. They also both call on the GCP algorithm described earlier. We have already labeled the heuristic version as FPM (Flexible Process Modeling); we will refer to the naive variant as FPM/N. In comparison runs, we provide them with the same generic processes, conceptual relations, and rate templates.

Both systems assume, like other recent work in the area, that all variables are observed. This lets them compute the rates of candidate process instances on each time step, which make possible their use of multiple linear regression to find differential equations. For the same reason, they also assume that expressions for process rates are algebraic combinations of observed variables, like $(a \cdot b)/c$, that may include constants but not unknown parameters. These are important assumptions that we aim to overcome in future work, but we cannot address all issues at once. Finally, many of our runs use synthetic noise-free data from known target models. This issue is less problematic, as Langley and Arvay (2015) have shown that simple smoothing of trajectories let systems like SPM handle up to ten percent noise.

4 Empirical Studies

Our approach to flexible process modeling appears promising, but its effectiveness is an empirical question. In this section, we report results from a number of studies with the two systems. Because the task we have addressed is novel, we cannot carry out experimental comparisons with other programs, even ones that focus on inductive process modeling. Neither would comparisons to equation discovery methods be appropriate, as they are not concerned with inferring underlying explanations. However, we can demonstrate that FPM exhibits the basic functions for which we intend it, compare its behavior with the naive approach, and examine its support for transfer across modeling tasks.

4.1 Basic Functionality and Generality

Our first aim was to demonstrate that the FPM system, which implements the heuristic approach described in the previous section, operates as intended. To reiterate, the system inputs a set of typed variables, time series for their values, a set of generic processes for use in modeling them, and – unlike its predecessors – conceptual relations and rate templates for introducing new generic processes if needed.

The initial run examined FPM's behavior on a natural data set, reported by Veilleux (1979), that displays a classic predator-prey cycle in which the protist *Nasutum* feeds on *Aurelia*, another protist. The two differential equations

$$\begin{aligned} d[aur] &= \quad 1.48 \cdot aur \; + \; -0.0230 \cdot aur \cdot nas \\ d[nas] &= -1.12 \cdot na \; + \quad 0.0047 \cdot aur \cdot nas \end{aligned}$$

describe the trajectories reasonably well, giving r^2 scores of 0.81 and 0.73, respectively. The SPM system finds a model that translates into these equations without difficulty when given relevant generic processes, but it fails otherwise. In contrast, when we presented FPM with growth and loss processes but omitted any generic processes for predation, it examined the candidate structures generated by the GCP module and immediately found the same model as SPM when given more background knowledge. This modeling task is quite simple compared to others, but it shows that our ideas work on actual scientific data.

Another run involved synthetic data from a four-variable predator-prey ecosystem in which the rate for most predation processes was the function $a \cdot b$, but in which one was the ratio a/b. Figure 1 displays the time series we provided to the system. When we gave it both forms in generic processes, FPM found the target model without special effort, since the SPM subroutine returned a consistent set of equations. However, when we omitted the generic process with the ratio-based rate expression, it failed to find an equation with sufficiently high r^2 for either x_3 or x_4. At this point, the system focused on x_3 and introduced additional process instances, generated from the GCP-produced generic structures, that mentioned this term. Using this expanded set, SPM found a complete set of equations with the same structure, and nearly the same coefficients, as the target model.

We also ran FPM on synthetic time series for an aquatic ecosystem from Arvay and Langley (2016) that included phytoplankton, zooplankton, nitrogen, iron, and detritus. They provided their SPM system with five generic processes – for grazing, organism loss, nutrient absorption, remineralization, and nutrient influx – that let it reconstruct the equations used to generate the data. We ran our system on the same data but withheld two of these generic processes, while also giving it conceptual relations analogous to the five processes and a set of four algebraic rate templates. On its first pass, FPM failed to find any equations for zooplankton, so it imported candidate processes the GCP routine had generated. This let SPM find equations for zooplankton on the next pass, but it still found none for nitrogen, so it incorporated more processes. These were sufficient not only to induce equations for nitrogen but also to find a model with the same structure as the one used to synthesize the data.

4.2 Scaling Experiments

In the previous section, we described two alternative methods for responding to incomplete sets of generic processes. Both relied on the GCP routine to generate candidate structures from conceptual relations and algebraic rate templates. However, the heuristic approach – implemented in the FPM system – uses points of failure to focus attention on one variable at a time, reducing the number of process instances it considers. In contrast, the naive method – implemented in FPM/N – instead attempts to use all generated processes whenever it cannot find a consistent model.

We have reported that FPM induces a number of ecological models even when we omit key generic processes from its background knowledge, and FPM/N succeeds on the same tasks. Because they rely on sampling, their behavior is nondeterministic and they occasionally fail, but they are typically successful. However, we should also examine how each system scales to increasing difficulty. Dealing with more missing processes is not a serious issue. The time FPM requires to induce a model should grow at most linearly with this factor, as it constructs new processes as needed, and informal studies are consistent with this prediction. But two other factors – the number of variables being modeled and the number of rate templates – pose more serious challenges.

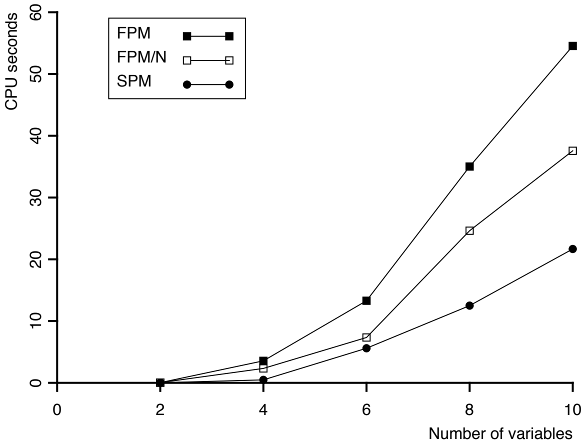

Figure 2: Computation times for FPM and FPM/N as functions of the number of variables they modeled. Background knowledge omitted one relevant generic process, but three conceptual relations and four rate templates were available. All relevant generic processes were given for the SPM curve.

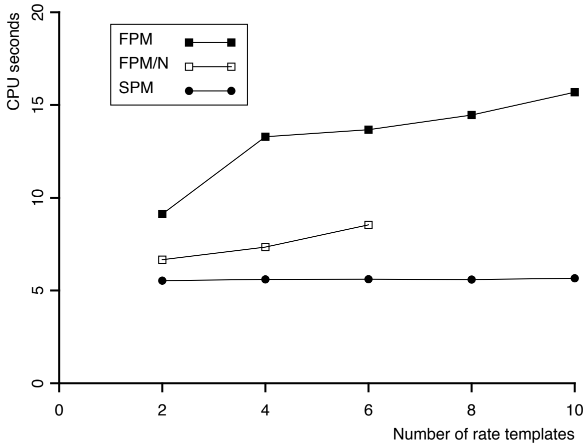

Figure 3: Computation times for FPM and FPM/N as functions of the number of rate templates on a six-variable modeling task. FPM/N could not find models for higher numbers of templates. Background knowledge omitted one generic process, but three conceptual relations were available.

Figure 2 plots the CPU times for FPM and FPM/N as one increases the number of variables. Here we provided synthetic time series for linear food chains with two to ten organisms. We held constant the numbers of missing processes (one), conceptual relations (three), and rate templates (four). The SPM module took 5,000 samples of ten processes each, found up to ten equations for each variable, and used 0.98 as its r^2 acceptance threshold. Computation for both systems grows with the number of variables, but the slopes differ little from that when all relevant generic processes are available, which is encouraging. The main surprise is that FPM is slower than the naive FPM/N because it makes multiple passes through the variables when positing new processes.

Increasing the number of rate templates, as seen in Figure 3, produces rather different results. Here we held constant the number of variables (six), missing processes (one), and conceptual relations (three), with SPM using the same settings as in the first experiment. The slopes for both systems are quite low, suggesting they scale well on this dimension. However, FPM/N could not find complete models when given eight or ten rate templates, even though it ran faster when fewer were present. Analysis suggests it generates so many process instances at higher levels that SPM's sampling method cannot find good equations. In contrast, FPM's heuristics focus its attention to make search tractable.

4.3 Transfer to New Modeling Tasks

The heuristic approach to process discovery should also lend itself to transfer across modeling tasks. Once the FPM system has found a generic process that lets it induce a consistent model, it adds this structure to its background knowledge, where it is available for use by the SPM subroutine on new tasks. At the very least, this should reduce the effort required to find process models in the new setting, but in some cases including content gleaned from earlier runs could mean the difference between inducing a complete process model and failing completely.

To demonstrate this ability, we ran FPM on a synthetic time series for a four-organism predator-prey system in which predation rates were determined by $a \cdot b$, but omitted the relevant generic process. The system introduced a new predation process with this rate term and, because it helped explain the observations, stored it for future use. Next we ran FPM on trajectories for another predator-prey system that involved both this process and a new predation relation with a/b as its rate expression. As before, FPM introduced the necessary generic process and induced the target model. However, when we presented it with the second modeling task without exposing it to the first one, it required multiple passes, with different variables as its focus, before it finally arrived at the same model through additional effort.

5 Related Research

Our research on flexible process modeling builds on a number of earlier traditions. The most obvious is the paradigm of inductive process modeling (Bridewell et al. 2008), from which we have adapted both our problem and many elements of our solution. This framework in turn owes a strong intellectual debt to Forbus' (1984) early work on qualitative process theory, which also specifies scientific models in terms of interacting processes. Falkenhainer and Forbus' (1991) progress on compositional modeling, although focusing on qualitative explanations, is especially relevant.

We argued earlier that our problem is distinct from equation discovery (e.g., Bradley et al. 2001; Džeroski and Todorovski 1995; Schmidt and Lipson 2009; Srividhya et al. 2007), in that we focus on *constructing explanations* in terms of unobserved processes, not merely finding empirical relations. Nevertheless, our system must address related issues along the way, as our framework relies on equation induction. We have also borrowed freely from established techniques for statistical analysis, differential equation solvers, and heuristic search that have been used by many others in the field, although our combination of these methods to support scientific model construction is novel. Recent work by Mai et al. (2016) focuses on inducing differential equations, but their method also finds a small set of basis functions to predict derivatives. This is similar in spirit to discovering processes, although the inferred bases do not appear to map onto an interpretable scientific formalism.

However, none of these connections are especially relevant to the two ideas emphasized in this paper. The first is that one can devise more flexible knowledge-rich systems by adding mechanisms for incrementally extending cognitive structures. Another is that one can make this elaboration more effective by identifying failure points and focusing on repairing them. Both ideas are reminiscent of two earlier efforts within the cognitive systems tradition. Sleeman (1984) adopted a similar approach to discovering 'malrules' for use in diagnosing students' algebra errors, guiding the mechanism by detecting incomplete parts of an explanation. Similarly, VanLehn, Jones, and Chi (1992) reported a model of physics problem solving that acquires new rules at points where it fails to explain worked-out solutions. Despite the different contexts, these systems have much in common with the framework we have presented.

6 Closing Remarks

In this paper, we reviewed earlier work on process model induction, which involves constructing differential equation accounts of time series from generic processes, and specified a new task – flexible process modeling – in which some background knowledge may be absent and one must infer the missing structures to construct an acceptable model. We described a simple combinatorial method for creating generic processes from more basic conceptual relations and rate templates, and we presented two mechanisms for using the resulting structures for filling knowledge gaps during model induction. We reported successful results with the more sophisticated approach, which uses failures in combining equations into consistent models, to focus attention and make tractable the identificaton of promising candidates. We also showed that the heuristic approach fared better than the naive alternative and that it benefits from transfer of inferred processes across successive modeling tasks.

Despite this encouraging progress, there remain open questions that demand additional research. Our framework assumes that all variables are observed and that rate expressions include no parameters, both of which limit its applicability in scientific fields like biochemistry. Moreover, the task of process discovery requires search through a space of candidate structures. We have developed one method for guiding this search, but we should test it more fully, identify situations in which it does not suffice, and devise improved mechanisms that respond appropriately in these contexts. We should also explore interactive solutions to these challenges. Few scientists desire to be replaced, and they can often offer useful guidance even in cases where they cannot specify their reasons. Computational scientific discovery remains an intriguing research topic with many open problems that we will solve only with concerted effort.

Acknowledgements

The research reported in this paper was supported in part by Grant No. N00014-11-1-0107 from the US Office of Naval Research, which is not responsible for its contents.

References

Arvay, A.; and Langley, P. 2016. Selective induction of rate-based process models. *Proceedings of the Fourth Annual Conference on Cognitive Systems*. Evanston, IL.

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

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

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

Džeroski, S.; and Todorovski, L. (Eds.) 2007. *Computational discovery of communicable scientific knowledge*. Berlin: Springer.

Džeroski, S.; and Todorovski, L. 1995. Discovering dynamics: From inductive logic programming to machine discovery. *Journal of Intelligent Information Systems* 4: 89-108.

Falkenhainer, B.; and Forbus, K. 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.

Langley, P. 2012. The cognitive systems paradigm. *Advances in Cognitive Systems* 1: 3–13.

Langley, P.; and Arvay, A. 2015. Heuristic induction of rate-based process models. *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, 537–543 Austin, TX: AAAI Press.

Langley, P.; Simon, H. A.; Bradshaw, G. L.; and Żytkow, J. M. 1987. *Scientific discovery: Computational explorations of the creative processes*. Cambridge: MIT Press.

Mai, M.; Shattuck, M. D.; and O'Hern, C. S. 2016. Reconstruction of ordinary differential equations from time series data. *arXiv:1605.05420v1* [physics.data-an].

Schmidt, M.; and Lipson, H. 2009. Distilling free-form natural laws from experimental data. *Science* 324: 81–85.

Shrager, J.; and Langley, P. (Eds.) 1990. *Computational models of scientific discovery and theory formation*. San Mateo, CA: Morgan Kaufmann.

Sleeman, D. 1984. An attempt to understand students' understanding of basic algebra. *Cognitive Science* 6: 387–412.

Srividhya, J.; Crampin, E. J.; McSharry, P. E.; and Schnell, S. 2007. Reconstructing biochemical pathways from time course data. *Proteomics* 7: 828–838.

VanLehn, K.; Jones, R. M.; and Chi, M. T. H. 1992. A model of the self-explanation effect. *Journal of the Learning Sciences* 2: 1–60.

Veilleux, B. G. 1979. An analysis of predatory interaction between paramecium and didinium. *Journal of Animal Ecology* 48: 787-803.