Heuristic Induction of Rate-Based Process Models

Pat Langley and **Adam Arvay**
Department of Computer Science, University of Auckland,
Private Bag 92019, Auckland 1142, New Zealand

Abstract

This paper presents a novel approach to inductive process modeling, the task of constructing a quantitative account of dynamical behavior from time-series data and background knowledge. We review earlier work on this topic, noting its reliance on methods that evaluate entire model structures and use repeated simulation to estimate parameters, which together make severe computational demands. In response, we present an alternative method for process model induction that assumes each process has a rate, that this rate is determined by an algebraic expression, and that changes due to a process are directly proportional to its rate. We describe RPM, an implemented system that incorporates these ideas, and we report analyses and experiments that suggest it scales well to complex domains and data sets. In closing, we discuss related research and outline ways to extend the framework.

1 Background and Motivation

Reasoning about scientific domains is a high-level cognitive task that is widely viewed as requiring considerable intelligence. Over the past three decades, research on computational scientific discovery (Shrager and Langley 1990; Džeroski and Todorovski 2007) has addressed a variety of tasks that arise in the pursuit of knowledge. Efforts in this paradigm are distinguished from mainstream work in data mining and machine learning by producing content stated in established scientific formalisms, ranging from componential models in particle physics to causal diagrams in genetics and to reaction pathways in biochemistry.

Much of the research in this area has focused on equation discovery, which has played a key role in the history of science and which has clear applications in many different disciplines. However, this task often arises in the early stages of a field's development, before theoretical knowledge supports the creation of explanatory models that account for observations in deeper terms. The task of explanatory model construction has received less attention from computational researchers, but there has been some progress on this topic.

One important form of such tasks, known as *inductive process modeling* (Langley et al. 2002), accepts generic knowledge about processes that can occur in a domain and time-series data for a number of variables. The aim is to find a concrete model, including numeric parameters, that reproduces their trajectories over time and makes accurate predictions for new values. One can compile this model into a set of differential equations, but it also explains observations in terms of underlying processes. This distinguishes the paradigm from other work on modeling time series, including alternative methods for finding differential equations.

In the sections that follow, we review drawbacks of previous efforts on inductive process modeling, focusing on their computational expense and problems with parameter estimation. In response, we present a new approach to process model induction that adopts representational assumptions which let it carry out heuristic rather than exhaustive search through the space of model structures, making it both more efficient and more reliable. We also describe an implemented system that incorporates these ideas, and we report experimental studies that demonstrate its ability to identify relevant processes, handle noisy observations, and construct accurate models that relate many variables. We conclude by discussing our framework's relation to earlier work and proposing responses to its limitations.

2 Critique of Process Modeling Research

Research on inductive process modeling has made continual progress since its first appearance over a decade ago (Langley et al. 2002). Advances have included adding the ability to incorporate hierarchical knowledge (Todorovski et al. 2005), handle missing values (Bridewell et al. 2006), combine models without losing interpretability (Bridewell et al. 2005), incorporate constraints to reduce search (Bridewell and Langley 2010), and even learn such constraints from sample models (Bridewell and Todorovski 2007). The framework has been applied to model construction in ecology (Asgharbeygi et al. 2006), hydrology (Bridewell et al. 2008), and biochemistry (Langley et al. 2006). Despite this progress, research has been hampered by three key features of the induction algorithms used to date.

The first characteristic is that, even though process models are compositional in nature, current algorithms evaluate only complete model structures. This evaluation involves comparing a parameterized model's predictions with observed time series. Existing systems generate many model structures up to a given complexity level, then parameterize and evaluate each candidate in turn. This approach does not support heuristic search through the space of model structures. Runs on domains that involve small sets of variables and generic processes have produced a few thousand model structures, which is computationally tractable. But the number of structures grows exponentially with the number of variables and generic processes, even with constraints, which means the standard approach will not scale.

Another drawback is that existing algorithms rely on repeated simulation to estimate parameters for each model structure. They start with values selected randomly within a given range for each parameter, simulate the parameterized model, and calculate error with respect to observed values. These methods use the error gradient to revise parameters, simulate the new model, and continue until improvements cease, indicating they have reached a local optimum. Hundreds of iterations are not uncommon, so that repeated simulations to guide search through parameter space dominate processing. In most runs, 99.99 percent of the CPU time is taken by running simulations for parameter estimation.

Finally, even with this expensive iteration for parameter estimation, the local optima found by traditional techniques may not fit the observed time series very well. To ensure arrival at reasonable parameters, they have resorted to repeated restarts from different random values. For some parameter spaces, ten restarts suffices to produce reasonable results, but informal experiments suggest that, in some cases, more than a hundred restarts are needed for reliable estimation. This further adds to the computational burden and, again, it must be borne separately for each distinct model structure.

Taken together, these features have made previous approaches to process model induction computationally expensive in the extreme, effectively limiting both the complexity of the models and the size of the data sets they can address. Moreover, even with extensive computation, problems with local optima mean that existing methods are not as reliable as one would desire. This suggests the need for a new approach to finding process models that is more tractable and reliable than existing alternatives.

3 Heuristic Induction of Process Models

To address these issues, we have developed a new approach to inducing process models that, we maintain, should be far more efficient and robust than its predecessors. In this section, we present some assumptions that constrain the induction task substantially. After this, we describe RPM, an implemented system that takes advantage of these assumptions in its operation. Finally, we make some high-level theoretical claims about the framework.

3.1 Simplifying Assumptions

To make the model induction task more tractable, we first restrict our notation, assuming that all processes concern changes over time and that they effect these changes at a specific *rate*. Chemical reactions are one obvious example of this idea; a given reaction involves the same substances, but its rate of operation can vary over time. Earlier work on inductive process modeling has included purely algebraic processes that describe instantaneous relations, which we hold are contrary to the notion of a process.

Second, we assume that each process has one or more associated derivatives and, moreover, that these are directly proportional to its rate. For instance, a chemical reaction that combines two substances, C_1 and C_2, to produce a third, C_3, has three derivative expressions of the form $dC_i/dt = k \times R$, where R is the rate and k is a constant parameter. We can distinguish between variables that are *inputs* to a process, which it consumes and thus have negative coefficients in their expressions, and variables that are *outputs*, which a process produces and thus have positive coefficients. This does not mean these variables always increase or decrease over time, since other processes may also influence them.

Third, we assume that the rate of each process is determined by a parameter-free algebraic expression. In some chemical reactions, this expression is the product of the concentrations of the input substances. For instance, the rate at which two substances react, given concentrations C_1 and C_2, might be $C_1 \times C_2$, whereas one in which three substances react might be $C_1 \times C_2 \times C_3$. We assume that rates are always positive and inherently unobservable, so we can adopt any measurement scale we like, avoiding the need for coefficients. Some rate equations, like the 'monod' expression $X/(k+X)$ in biochemical models, include parameters. For now, we will exclude these from our framework, but we will return to them when discussing future research.

Fourth, we assume that, if a variable appears in the rate expression for a process, then it must also appear in a derivative expression associated with that process. For instance, if the rate equation for a process is $X \times Y$, this means it must include derivative expressions for both X and Y, possibly along with ones for other variables. This rules out 'gratuitous' processes with terms that influence rates but are not affected in turn. At first glance, this appears to cause difficulty for chemical reactions that involve catalysts, but we can treat them as special cases in which the proportionality coefficient for derivative equations are near zero.

The simple model for a predator-prey ecosystem in Table 1 illustrates these ideas. Each process has an associated rate expression, one specifying that the rate equals the product of two variables and the others stating that it equals a single variable. Each process also has one or more associated derivatives that are directly proportional to the rate, with parameters detailing this dependence. Finally, every term that appears in the rate expression for a process also appears in some derivative expression. Generic processes take a similar form but utilize typed variables. These assumptions constrain considerably the process models that are possible, although one can still state a large number of such structures.

Our notation borrows heavily from Forbus' (1984) Qualitative Process theory. Rates also played a key role in his framework, with an algebraic rate expression corresponding to a set of indirect influences in a qualitative process. Moreover, each equation that relates a derivative to a rate maps onto a direct influence in a qualitative process. Forbus' framework could have multiple rates per process and ours allows only one, but otherwise they are very similar.

A final issue concerns not the structure of process models but the data available to induce them. For reasons that will become apparent shortly, we assume that observations

Table 1: A three-process model for a system involving the predator *Nasutum* and the prey *Aurelia* that illustrates our framework's assumptions. Each process has a rate determined by an algebraic expression and includes one or more derivatives that are proportional to this rate.

```
exponential_change[aurelia]
  rate          r = aurelia
  parameters    A = 0.75
  equations     d[aurelia] = A × r
exponential_change[nasutum]
  rate          r = nasutum
  parameters    B = −0.57
  equations     d[nasutum] = B × r
holling_predation[nasutum, aurelia]
  rate          r = nasutum × aurelia
  parameters    C = 0.0024
                D = −0.011
  equations     d[nasutum] = C × r
                d[aurelia] = D × r
```

are given for all variables to be included in candidate models. Some early work in the paradigm (e.g., Langley et al. 2002) took the same position, but many recent efforts (e.g., Bridewell et al. 2008) explicitly attempt to handle unobserved terms. This is one reason they require simulation of full models to estimate parameters, as it lets them calculate values for such terms. This makes our approach more limited, but early work on induction of decision trees and Bayesian networks relied on analogous simplifications, and we discuss ways to move beyond it later in the paper.

3.2 A Regression-Guided Process Modeler

We have incorporated these ideas into RPM, a new system for inductive process modeling that uses the assumptions to constrain search and make the discovery task tractable. The system inputs a set of typed variables,[1] a subset of these variables to be predicted, a set of generic processes in the form just described, and an observed time series for each variable. The program returns a model stated as a set of differential equations, one for each dependent variable, that predicts the corresponding derivative as a linear combination of process rates. If it cannot find acceptable equations for some variables, it returns a partial model.

Before it begins to search, RPM takes a number of initialization steps. First it uses the task variables to instantiate the generic processes in all ways that are consistent with the latters' type constraints. For instance, suppose the modeling task involves three variables, A, B, and C, where A has type prey, B has type prey or predator, and C has type predator. Further suppose that the generic process *holling_predation* includes two generic variables, X and Y, where the first has type *prey* and the second *predator*. In this case, RPM would generate three instances of *holling_predation*, one with X bound to A and Y bound to B, one with X bound to A and Y bound to C, and one with X bound to B and Y bound to C. The result is a set of process instances, each with an instantiated rate expression and derivative terms.

Next, the system estimates the derivatives for each dependent variable on each time step. For this purpose, it uses the 'center difference' method, which calculates the derivative for variable X at time t as the average of $X(t) - X(t-1)$ and $X(t{+}1){-}X(t)$. This produces a trajectory, over time, of estimated derivatives for each dependent variable. RPM also uses the observed time series to calculate rates for each such process instance. If the third process instance just discussed has the rate expression $B \times C$, then RPM would multiply B by C at each time to obtain the associated process rate. The assumption that rates are parameter-free algebraic functions of observed variables lets it make this calculation for each candidate process instance. The result is a trajectory, over time, of the rate for each instantiated process.

RPM then iterates through the dependent variables, attempting to find a differential equation for each one that predicts its observed changes as a function of process rates. For each variable D, it considers only those process instances P that include D as a dependent term, as others would not make appropriate predictors. The system then examines, in turn, each linear equation of the form $D = a \times P$, using linear regression to calculate the parameter a. If the r^2 for any of these equations exceeds a user-specified threshold, then RPM adds the best candidate to its model and examines the next dependent variable. If not, then it considers all pairs of process instances, P_1 and P_2, in which D is a dependent term and considers each linear equation of the form $D = a \times P_1 + b \times P_2$. For this it uses multiple linear regression to estimate the parameters a and b. Again, if some equation's r^2 exceeds the threshold, the system adds the best to the model. Otherwise, it examines triples of processes, and so on, continuing until reaching a user-specified maximum.

In summary, RPM carries out a sequential form of greedy search through the space of process models, in that once it selects an equation for one variable, it cannot change its mind. This means that, if more than one equation produces acceptable r^2 values, the system may make an incorrect choice that will cause it difficulty later. Nevertheless, greedy search is highly efficient when its heuristics guide search down useful paths, and more efficient induction of process models is one of our primary concerns.

Knowledge about processes modulates search for individual equations in four complementary ways. First, RPM groups generic processes into types, such as *predation* and *organism loss*. Different processes in the same class relate the same number and types of variables but differ in their algebraic rate expressions. RPM only considers models that contain at most one process instance of each type with the same arguments. Suppose we have two generic predation processes that relate a predator population X with a prey population Y, one with rate $X \times Y$ and another with X/Y. Suppose we have two variables, A and B, the first a predator and the second a prey. This gives two process instances – $predation(A, B)$ with rate $A \times B$ and $predation(A, B)$ with rate A/B – which RPM treats as mutually exclusive, ensuring no model structure contains more than one.

[1] Type constraints are disjunctive, in that a given variable may have more than one possible type, say either predator or prey.

Second, generic processes may include numeric constraints on the parameters that appear in proportionality equations. For instance, predation processes specify that the predator's coefficient must be positive, whereas that for the prey must be negative, indicating predation causes the predator's population to increase and the prey's to decrease. Once RPM finds an equation for a variable with an acceptable fit, it checks the equation's parameters to see if they satisfy its associated processes' constraints. If not, then it abandons the equation and continues to look for other candidates. In cases where an organism may be either a predator or a prey, the system will find two equations that fit the data equally well, but only one will have parameters with the correct signs.

Third, if RPM incorporates a process P into the equation for one dependent variable, say D_1, that includes another variable, say D_2, then it must include P in all equations that it later considers for predicting D_2. These constraints reduce the number of process combinations RPM considers when searching for later equations. In some cases, given the user-specified limit on the number of processes allowed, it fully determines an equation's form, eliminating search entirely.

Finally, to make such constraints more effective, the system utilizes information about earlier equations to decide which variable to focus on next. For each variable not yet incorporated into the model, it calculates the number of processes in which it already appears. RPM then selects the one that takes part in the most processes (breaking ties at random), as it will have more terms in its equation determined. This is similar to selection of the most constrained variable in work on constraint satisfaction. Unless the data set can be partitioned into subsets of variables that do not interact, this heuristic further reduces the search needed to find equations.

Together, these techniques limit greatly the number of model structures that RPM entertains during its discovery efforts. They make the difference between tractable heuristic search and the intractable methods used in previous systems.

3.3 Theoretical Claims

Based on our earlier observations, we can make a number of theoretical claims about the new approach to process model induction and our implementation of its ideas.

- RPM should duplicate previous methods' abilities to identify the structure of process models and distinguish irrelevant processes from ones that, taken together, account for the observed trajectories.
- Because the system estimates parameters for each equation separately, and without relying on repeated simulation of complete models, it should be far more efficient computationally.
- RPM's reliance on multiple linear regression, a statistical technique with an ability to tolerate noise and low variance, should let it avoid local optima at the equation level and find parameters that fit the data well.
- Heuristic search through the structure space should let the system scale well to models of high complexity, providing they are sparsely connected in that each variable is influenced by only a few processes.

If these claims hold true, then RPM should be far more effective at inducing process models than systems that have previously addressed this challenging class of problems.

4 Evaluation of the Approach

A complexity analysis of the new approach is encouraging even in the worst case. Let V be the number of variables, P the number of generic processes, S the number of samples, k the maximum number of variables in a process, and j the maximum number of processes in an equation. The total number of process instances $I = P \times V!/(V-K)!$, but RPM only considers process instances in which a given variable appears, meaning the number in play for each equation is at most $I' = P \times (V!/(V-K)! - (V-1)!/(V-K-1)!)$. This gives $J' = I'!/(J! \times (I'-J)!)$ as the maximum number of equations considered for each variable, leading to $S \times V \times J'$ as the total time complexity. Assuming $k = 3$ and $j = 3$, which are reasonable values, J' is approximated by $P^3 \times V^{7.4}$, giving an effective complexity of $S \times P^3 \times V^{8.4}$. However, this analysis ignores our heuristics' role in guiding the selection of variables and search for equations, so we have also studied our method's empirical behavior.

4.1 Empirical Behavior on Natural Data

To demonstrate RPM's relevance to natural science, our initial runs focused on published data about a simple predator-prey ecosystem (Veilleux 1979). These involve 26 samples of two protozoa – *Nasutum* and *Aurelia* – with the former preying on the latter, as their populations vary over time. Earlier methods for inductive process modeling have reported reasonable results on these time-series data (Bridewell et al. 2008). Our purpose here was not to claim superiority, but simply to show that our new mechanism for constructing process models produces accurate and plausible explanations for actual scientific data.

When provided with these data and two generic processes, one for predation and another for exponential loss/growth, RPM induces the model shown earlier in Table 1. This maps onto the equation $d[aurelia] = 0.75 \times aurelia + -0.011 \times nasutum \times aurelia$, for which r^2 is 0.84, and $d[nasutum] = 0.0024 \times nasutum \times aurelia + -0.57 \times nasutum$, for which r^2 is 0.71. Figure 1 plots the observed trajectories and the simulated ones produced when we provided these equations to a differential equation solver. The model does not reproduce the observed trajectories perfectly, but it does capture the peaks and troughs, offset slightly. This result is encouraging, as RPM evaluates equations based on prediction of observed derivatives, not on the trajectories themselves, as did its predecessors. If reproducing trajectories is important, one could use gradient descent through parameter space with estimated coefficients as starting points.

4.2 Induction with Irrelevant Processes

We also desire to demonstrate that RPM does not confuse relevant process instances with irrelevant ones when searching the space of model structures. To this end, we provided the system with eight additional generic processes for the predator-prey domain. These differed from the original ones in the algebraic expression that determines each process rate.

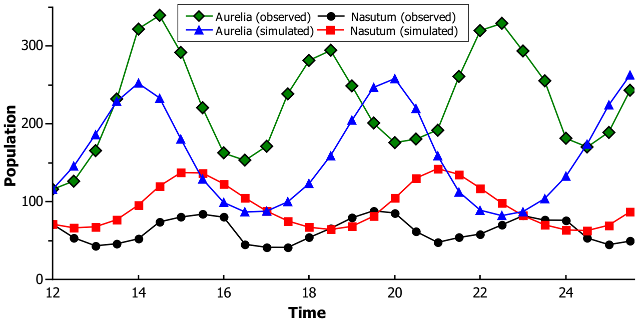

Figure 1: Observed and simulated trajectories for a predator-prey system involving the protozoa *Nasutum* and *Aurelia*.

We included four alternative algebraic forms for individual loss/growth of an organism and another four alternative forms for predation. On runs with the natural data for *Nasutum* and *Aurelia*, as well as on synthetic data generated using the X and $X \times Y$ rate terms, RPM found the same process models as before. This suggests the system can distinguish relevant processes from irrelevant ones effectively.

Because we are centrally concerned with efficient induction, we also measured RPM's run time when presented with different numbers of irrelevant processes. Figure 2 plots the CPU seconds required as a function of the number of generic processes, starting with two relevant ones and adding two irrelevant ones on each increment. The growth in computation time appears to be greater than linear, but the overall behavior still scales reasonably well with the number of irrelevant processes and much better than our worst-case analysis.

4.3 Handling Noise and Complex Models

In addition, we examined RPM's ability to handle noisy observations. This is especially important given the system's focus on finding equations that predict estimated derivatives, as taking differences between succssive observed values can magnify noise substantially. For instance, when we added five percent noise to synthetic data for the five predator-prey system, 0.28 was the highest r^2 value found for any derivative, and RPM could distinguish only some of the target equations from alternative forms. However, when we smoothed the observed values using locally weighted regression (Cleveland 1979), the system found all target equations without difficulty, even when the noise level was ten percent and we provided the irrelevant processes described above. This suggests that preprocessing with simple and inexpensive smoothing techniques will suffice for the approach to handle substantial amounts of observational noise.

Moreover, because we wanted to demonstrate that our approach can induce process models that involve many variables, we ran RPM on a synthetic predator-prey data set with 20 organisms. The target model corresponded to a linear food chain in which organism 20 preys on organism 19, which in turn preys on organism 18, and so forth, with organism 1 at the bottom. The model augmented these 19 predation processes with an exponential growth process for organism 1 and an exponential loss process for organism 20.

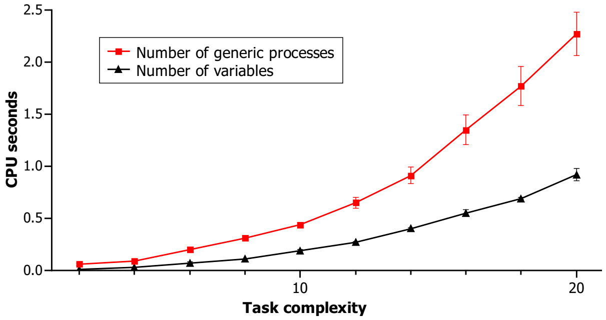

Figure 2: RPM's processing time as a function of the number of generic processes, given five variables, and as a function of the number of variables, given two generic processes.

RPM identifies correctly the structure of the target model from noise-free data involving 236 samples. The system generates 400 process instances, but it considers only 1,483 equation structures during search because those found for early variables constrain both which processes to incorporate and which variables to examine next. Earlier systems that carry out nonheuristic search through the structure space would have great difficulty finding models of this complexity. We also systematically varied the number of variables and recorded CPU time. Figure 2 shows this grows at a reasonable rate, indicating the heuristics provide average-case behavior far better than our worst-case analysis.

4.4 Comparison to SC-IPM

Finally, we ran a comparison between RPM and SC-IPM (Bridewell and Langley 2010), an earlier system for inducing process models from time series, on synthetic data generated from a known target. The target model involved three organisms that interact through two predation processes in a single food chain, with one process for growth and another for loss at either end. We provided both systems with 100 observations for each of these simulated variables and analogous background knowledge about generic processes.

SC-IPM has two parameters that RPM lacks: the number of model structures it considers and the number of times it restarts gradient descent during parameter estimation. Higher values improve the system's chances of finding a good model but also increase its processing time. We held the first parameter constant at 600, which seemed sufficient to ensure SC-IPM examines the target structure, and varied the second from 10 to 150 restarts. In each case, we recorded run time and mean squared error on the training data.

Figure 3 plots SC-IPM's performance on these dimensions, along with the time and error for ten RPM runs. The results are umambiguous. RPM finds very accurate models reliably, whereas SC-IPM induces ones with comparable accuracy rarely even on its most expensive runs. Moreover, although both are written in Lisp, RPM finds these models more than *83,000 times faster* than even the most rapid SC-IPM runs. Rather than examining 600 model structures, RPM considered only 21 equations, each of which it fit using multiple linear regression rather than gradient descent.

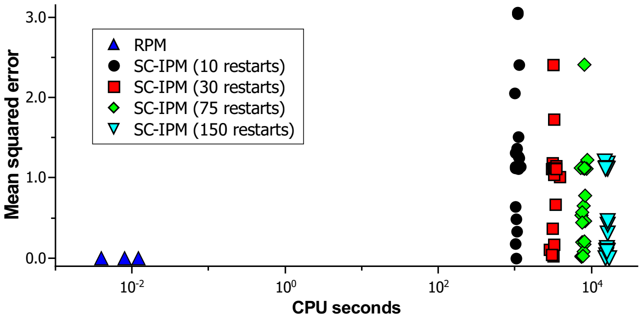

Figure 3: Processing time vs. error for RPM and SC-IPM with different number of restarts for gradient descent search.

These experimental results, taken together, offer support for our claims that the new approach induces accurate process models in a computationally tractable manner. The system responds well to irrelevant processes, noisy observations, and substantial model complexity, and it compares very favorably to a representative earlier system in both reliability and processing time.

5 Related Research

The approach to inductive process modeling we have reported here incorporates ideas from earlier efforts, but it also introduces novel features that distinguish it from predecessors. The framework draws directly from Forbus' (1984) early work on qualitative process theory, which also encodes scientific accounts as sets of interacting processes. Our reliance on quantitative processes is not unique; the representational innovation lies in reformulating processes in terms of algebraic expressions that determine rates and in terms of derivatives that are proportional to these rates. The resulting class of models still have broad coverage, but they are far more amenable to computational discovery.

We have already discussed earlier work on inductive process modeling, but not related research on discovering differential equation models from time series. We will not review traditional work on system identification, which assumes functional forms and focuses on parameter estimation. However, Džeroski and Todorovski's (1995) LaGrange system searches the space of functional forms, and the closely related LaGramge (Todorovski and Džeroski 1995) uses grammatical knowledge to inform this search. Bradley et al.'s (2001) PRET addresses the same task, analyzing qualitative regularities in data to limit the structures it considers. Genetic programming methods (e.g., Koza et al., 2001; Schmidt & Lipson, 2009) use evolutionary search techniques, with more recent systems sampling data to avoid overfitting, but they search a larger space of differential equations, making them less scalable and more likely to produce uninterpretable models. LaGrange comes closest to our approach, constructing equations for each derivative as linear functions of algebraic expressions, but, like other systems, does not incorporate processes to constrain search.

We should also note that Srividhya et al. (2007) report a related approach to inducing differential-equation models for biochemical pathways, which we discovered only after developing RPM. This uses knowledge about types of reactions to generate candidate equation structures, which it then fits to estimated derivatives of chemical concentrations. Their method also invokes multiple linear regression to find coefficients and pursues greedy search for model structures. Their system also favors equations with fewer terms, but it does not use heuristics to order variables or require that later equations be consistent with earlier ones. It also relies on chemical constraints about numbers of molecules in reactions, making it less general than our framework.

6 Limitations and Future Work

Our experimental results suggest that the new approach to process model induction can find accurate and plausible models in reasonable amounts of time. However, this behavior relies on some assumptions that we should aim to relax in our future analyses and implementations.

The framework's requirement that processes comprise an algebraic rate expression and derivatives proportional to this rate is useful but not very limiting. However, the assumption that rate expressions be parameter-free, which it uses to calculate rates from variables, means it cannot encode some models that earlier systems considered. In future work, we plan to weaken this restriction by allowing one parameter per rate equation and using the reliable LMS algorithm to estimate their values. One parameter per process means each linear differential equation has at most one free parameter per term. For a given equation, the extended RPM could make an initial guess about these parameter values, then use the LMS method to iteratively revise them, using r^2 as an error metric. This should let us handle many parameterized rate expressions, including the monod function that is used widely in ecology and biochemistry.

We should also explore replacing the greedy search method through the space of model structures with beam search. Experience with RPM suggests that, when observed trajectories are reasonably flat, multiple equations have similar r^2 scores. Beam search would let the system retain multiple equations for each variable, delaying decisions about which to retain until it has acquired information about equations for other variables. This should also reduce reliance on the r^2 threshold, which we currently tune manually to a level appropriate for the data set being analyzed.

We have seen that RPM's adoption of heuristic search lets it induce reasonably complex models, but it still uses an exhaustive method to find individual equations, at least in its early stages. True scalability to hundreds or thousands of equations will also require heuristic construction of differential equations for each dependent variable. The natural response is to adapt greedy forward-search techniques for feature selection (Blum and Langley 1997), but relations among process rates, which can share variables, make r^2 an unreliable metric for the usefulness of individual terms. We must identify more informative guides to support effective heuristic search through the space of component equations.

Finally, we must extend the framework to induce process models even when some variables remain unobserved. We assumed fully observable domains because, combined with parameter-free rate expressions, it let our mechanism use linear regression to find each equation in a model separately from others. This was the key source of RPM's scalability to complex models, but in future work we should move beyond this restriction. Following Bridewell et al. (2006), we can handle occasional missing values using an iterative-optimization scheme, but the more challenging case concerns entirely unobserved variables. Our assumption that derivatives depend on linear combinations of process rates should let us extend that approach to such cases, but whether it retains good convergence properties is an open question.

7 Concluding Remarks

In this paper, we critiqued earlier research on inductive process modeling, which involves constructing differential equation accounts of time series from knowledge about generic processes. We also presented a new approach to this task, and its implementation in the RPM system, that builds on some important assumptions. These included associating with each process a rate that is determined by a parameter-free algebraic equation, along with one or more derivatives that are directly proportional to this rate. We argued that this constrained framework should be more tractable computationally and more robust on parameter estimation.

We presented a complexity analysis that shows the new approach should scale reasonably to induction tasks that involve irrelevant processes and target models with many variables. More important, we showed encouraging experimental results on both natural and synthetic data. In the latter case, we demonstrated that RPM can ignore irrelevant processes, handle noisy data when aided by standard smoothing techniques, and scale well enough to induce process models with 20 variables. In addition to these promising findings, we outlined steps to overcome our simpliying assumptions that should broaden the coverage and applicability of the modeling framework's future implementations.

Acknowledgements

This research reported here was supported in part by ONR Grant N00014-11-1-0107. We thank Will Bridewell, Sašo Džeroski, Rich Morin, Son To, and Ljupčo Todorovski for discussions that led to the approach we have described.

References

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

Blum, A.; and Langley, P. 1997. Selection of relevant features and examples in machine learning. *Artificial Intelligence* 97:245–271.

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

Bridewell, W.; Bani Asadi, N.; Langley, P.; and Todorovski, L. 2005. Reducing overfitting in process model induction. In *Proceedings of the Twenty-Second International Conference on Machine Learning*, 81–88. Bonn, Germany.

Bridewell, W.; Langley P.; Racunas, S.; and Borrett, S. R. 2006. Learning process models with missing data. In *Proceedings of the Seventeenth European Conference on Machine Learning*, 557–565. Berlin:Springer.

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.

Cleveland, W. S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74:829-836.

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.

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

Koza, J. R.; Mydlowec, W.; Lanza, G.; Yu, J.; and Keane, M. A. 2001. Reverse engineering of metabolic pathways from observed data using genetic programming. *Pacific Symposium on Biocomputing* 6:434–445.

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. San Francisco: Morgan Kaufmann.

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

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.

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

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: AAAI Press.

Todorovski, L.; and Džeroski, S. 1997. Declarative bias in equation discovery. In *Proceedings of the Fourteenth International Conference on Machine Learning*, 376–384. Nashville, TN: Morgan Kaufmann.

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