A Constraint Language for Process Model Induction

Matt Bravo MBRAVO@STANFORD.EDU
Will Bridewell WILLB@CSLI.STANFORD.EDU
Computational Learning Laboratory, CSLI, Stanford University, Stanford, CA 94305 USA

Ljupčo Todorovski LJUPCO.TODOROVSKI@FU.UNI-LJ.SI
Computational Learning Laboratory, CSLI, Stanford University, Stanford, CA 94305 USA
University of Ljubljana, Faculty of Administration, Gosarjeva 5, SI-1000 Ljubljana, Slovenia

Abstract

We define the inductive process modeling task as the automated construction of *quantitative process models* from time series and background knowledge. In this task, the background knowledge comprises generic processes that along with a given set of entities define the space of candidate model structures. Typically this space grows exponentially with the size of the library, so past research introduced a hierarchical organization on the processes to constrain that space to a limited set of plausible configurations. However, organizing the processes into a hierarchy takes considerable effort, leads to implicit constraints, and creates a complex relationship between the knowledge of what processes exist and the knowledge of how one can combine them. To address these problems, we developed SC-IPM[1], an inductive process modeler that uses declarative constraints to reduce the size of the model structure space. In this paper, we describe the constraint formalism and how it guides SC-IPM's search.

1. Introduction

Scientists build models to explain observations of complex, dynamical systems. This task requires either an implicit or explicit search through the space of plausible models (Langley et al., 1987). However, the difficulty of model development and parameter estimation encourages a greedy search and the scientists may consider only a few alternatives before selecting a final structure. We seek to build tools that help scientists systematically create and evaluate alternative models so that they can improve their understanding of the studied phenomenon. To accomplish this goal, we require a representation for domain knowledge that lets experts interactively guide the system's search.

This paper extends previous work on inductive process modeling (Langley et al., 2002; Langley et al., 2003; Todorovski et al., 2005), by adding a formalism for stating explicit, structural constraints. The general problem of inductive process modeling takes as input a set of time series for observed variables, background knowledge in the form of generic processes and entities, and a set of instantiated entities whose properties may be associated with the data. As output, a learning system should produce a quantitative process model that explains the data in terms of the background knowledge. A basic approach to this task involves instantiating the generic process with the given entities as allowed, exhaustively combining the instantiated components into model structures, fitting the numeric parameters, and returning those models with the best quantitative fits to the data. Typically, this strategy will lead to an search space that is exponential in the number of instantiated processes and that includes a several implausible structures.

To address these problems, one can introduce constraints on the model structures. Previously, Todorovski et al. (2005) developed a formalism based on the decomposition of generic processes and the specification of a process hierarchy. Our recent experience indicates that this structure places a considerable burden on the domain expert, leads to implicit and intertwined constraints, and creates a complex relationship between the knowledge of what processes exist and that of how to combine them.

In this paper we describe a knowledge representation that addresses the shortcomings of the process hierar-

[1]Pronounced "Skip 'em", SC-IPM is an acronym for "Satisfying Constraints to Induce Process Models"

chy. To reduce the effort required to encode a process library, we designed isolated, declarative constraints that experts can add individually without concern for how they fit into a complex hierarchical structure. Due to its piecemeal nature, this formalism also alleviates the problem of obfuscated constraints that are entangled with the individual processes. In the next section, we illustrate and define these constraints using a simplified population dynamics library and discuss how they guide the search. We end the paper with a discussion of related and future work.

2. Declarative Constraints

Table 1 presents a sketch of a constrained library for population dynamics. Notably the generic entities and processes lack the equations, parameter, and variables that define the allowed behaviors. In this example, generic entities have names and processes have both names and placeholders and type information for the instantiated entities that they can relate. Each constraint shares a similar structure comprising a name, a type, and a set of generic entities with an optional entity list. Whereas the name of a constraint serves only descriptive purposes, the type declares a logical relation among the listed generic processes. The formalism supports four types, including *always-together*, *at-most-one*, *exactly-one*, and *necessary*. An entity specified along with a process specializes the constraint based on the entity. For example, "Optional Light Limitation" from Table 1 states that a model may contain no more than one of *exponential_growth*, *logistic_growth*, and *light_limitation*, but it may one instance of *light_limitation* per light source. After describing how we use these constraints, we define their logical semantics.

We built the inductive process modeler SC-IPM to interpret a library that includes such declarative constraints. To search the space of model structures, this system begins by binding the generic processes to the instantiated entities, each of which becomes a literal for propositional formulae. Next, the program transforms each constraint into a corresponding Boolean logic formulae and combines them into a single conjunctive sentence. Valid solutions of this sentence correspond to plausible model structures as defined by the library, such that true literals (i.e., bound processes) become part of the model and false ones do not. Currently, SC-IPM can convert the formula into disjunctive normal form, which is trivial to satisfy, or into conjunctive normal form, for which it uses a version of the WalkSAT (Selman et al., 1993) algorithm to perform local search.

Table 1. A simplified SC-IPM library for population dynamics.

```
ENTITY producer
ENTITY grazer
ENTITY nutrient
ENTITY light

PROCESS exponential_growth
  RELATES P{producer}
PROCESS logistic_growth
  RELATES P{producer}
PROCESS limited_growth
  RELATES P{producer}, N{nutrient}
PROCESS ivlev
  RELATES P{producer}, G{grazer}
PROCESS ratio_dependent
  RELATES P{producer}, G{grazer}
PROCESS nutrient_limitation
  RELATES P{producer}, N{nutrient}
PROCESS light_limitation
  RELATES P{producer}, L{light}
PROCESS exponential_loss
  RELATES P{producer}

PROCESS degradation
  RELATES G{grazer}

CONSTRAINT 'Growth Alternatives'
  TYPE exactly-one
  PROCESSES
    exponential_growth
    logistic_growth
    limited_growth
CONSTRAINT 'Predation Alternatives'
  TYPE exactly-one
  PROCESSES
    ivlev
    ratio_dependent
CONSTRAINT 'Nutrient Limited Growth'
  TYPE always-together
  PROCESSES
    limited_growth
    nutrient_limitation(N)
CONSTRAINT 'Optional Light Limitation'
  TYPE at-most-one
  PROCESSES
    exponential_growth
    logistic_growth
    light_limitation(L)
CONSTRAINT 'Mandatory Loss'
  TYPE necessary
  PROCESSES
    exponential_loss
    degradation
```

2.1. Always Together

In an *always-together* constraint the processes must occur together or not at all. Recall that for a process to

occur in a model the equivalent literal must be true, hence we can write *always-together* constraints such that a disjunction of the literals implies their conjunction. As an example, consider the "Nutrient Limited Growth" constraint in Table 1 when there is one nutrient n and one producer p. SC-IPM will produce the bound processes *nut_lim(n)* and *lim_growth(n)*, and the formula:

$$\neg(nut\_lim(n) \vee lim\_growth(n)) \ \vee \\ (nut\_lim(n) \wedge lim\_growth(n))$$

Since *nutrient_limitation* is specialized on its nutrient argument, when multiple instantiations of that generic entity exist, the program will produce similar a formula for each one.[2]

2.2. Exactly One

The *exactly-one* constraint states that one and only one instantiation of the listed generic processes can occur per model. Logically, one can represent this requirement as a disjunctive logic formula. To illustrate, consider the "Growth Alternatives" constraint in Table 1. When the corresponding processes are bound, SC-IPM produces the following formula.

$$(exp\_growth \wedge \neg log\_growth \wedge \neg lim\_growth) \vee \\ (\neg exp\_growth \wedge \ log\_growth \wedge \neg lim\_growth) \vee \\ (\neg exp\_growth \wedge \neg log\_growth \wedge lim\_growth)$$

2.3. At Most One

The *at-most-one* constraint is similar to *exactly-one* with the added condition that it is satisfied when no instantiations of the listed generic processes appear. The *at-most-one* logical formula is the *exactly-one* formula appended with a conjunction where every literal is negated. This constraint lets one specify optional processes such as *light_limitation* in Table 1. Given a producer p and a light source l, the "Optional Light Limitation" constraint translates as

$$(exp\_growth \wedge \neg log\_growth \wedge \neg light\_lim(n)) \vee \\ (\neg exp\_growth \wedge \ log\_growth \wedge \neg light\_lim(n)) \vee \\ (\neg exp\_growth \wedge \neg log\_growth \wedge light\_lim(n)) \vee \\ (\neg exp\_growth \wedge \neg log\_growth \wedge \neg light\_lim(n))$$

2.4. Necessary

The *necessary* constraint forces processes to occur in a model. The corresponding logical formula consists of a conjunction of all the bound processes related to the constraint. To illustrate, consider the "Mandatory Loss" constraint from Table 1, which is represented as

$$(exponential\_loss \wedge degradation)$$

Using these four constraint types we give domain experts powerful building blocks to translate their knowledge of valid model structures into precise, machine readable information.

[2] Although both entities bind to the generic process, we show only those entities on which the constraint is specialized to highlight the relationship.

3. Related and Future Work

SC-IPM reflects influences from many sources. In particular, the task of inductive process modeling combines research on qualitative process theory (Forbus, 1984) and equation discovery (Langley et al., 1987) as they are applied in scientific and engineering domains. For example, Falkenhainer and Forbus's (Falkenhainer & Forbus, 1991) work on compositional modeling emphasizes the construction of qualitative models from fragments similar to our own generic processes. Similarly, Żytkow et al's (Żytkow et al., 1990) FAHRENHEIT identified quantitative regularities from electrochemical experiments by searching through a space of candidate functional forms for which it then fit parameters.

More recently, the PRET system (Bradley et al., 2001) used general mathematical knowledge along with more specific domain expertise to constrain the construction of quantitative models of dynamic systems. Unlike the constraints in SC-IPM, those used by PRET ruled out systems of equations based primarily on qualitative behavior rather than process-level knowledge. In this respect, Todorovski's LAGRAMGE (Todorovski, 2003) most closely resembles the current work in that it uses constrained processes to organize the search space. However, that system shares the representational burdens of the hierarchical process libraries (Todorovski et al., 2005) that the current work alleviates.

To extend this work, we intend SC-IPM to provide the foundation for an interactive modeling and discovery environment. The first step in this direction involves the system's incorporation into PROMETHEUS (Bridewell et al., 2006), which supports the construction, evaluation, and revision of quantitative process models. The next step involves the development of tools to help scientists identify new domain-level constraints. To this end, our recent work on learning declarative bias (Bridewell & Todorovski, 2007) indicates one possible approach.

In this paper, we defined a modular constraint language for inductive process modeling. Moreover, we described how to transform the supported restrictions on model structure into a satisfaction problem that

one can solve with readily available tools. We expect this approach to ease the acquisition of domain-level knowledge when compared to the use of hierarchical process libraries, and we plan to perform a more careful analysis of the two formalisms in the future.

Acknowledgments

This research was supported by Grant No. IIS-0326059 from the National Science Foundation. We thank Pat Langley and Stuart Borrett for discussions that influenced this paper.

References

Bradley, E., Easley, M., & Stolle, R. (2001). Reasoning about nonlinear system identification. *Artificial Intelligence*, *133*, 139–188.

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. *Proceedings of the Seventeenth International Conference on Inductive Logic Programming* (p. to appear). Corvallis, WA.

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

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

Langley, P., George, D., Bay, S. D., & Saito, K. (2003). Robust induction of process models from time-series data. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 432–439). Washington, D.C.

Langley, P., Sánchez, J., Todorovski, L., & Džeroski, S. (2002). Inducing process models from continuous data. *Proceedings of the Nineteenth International Conference on Machine Learning* (pp. 347–354). Sydney.

Langley, P., Simon, H. A., Bradshaw, G. L., & Zytkow, J. M. (1987). *Scientific discovery: Computational explorations of the creative process*. Cambridge, MA, USA: MIT Press.

Selman, B., Kautz, H. A., & Cohen, B. (1993). Local search strategies for satisfiability testing. *Proceedings of the Second DIMACS Challange on Cliques, Coloring, and Satisfiability* (pp. 521–532). Providence, RI.

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. *Proceedings of the Twentieth National Conference on Artificial Intelligence* (pp. 892–897). Pittsburgh, PA.

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