Multimodal Reasoning about Physical Systems

Reinhard Stolle and Elizabeth Bradley*
University of Colorado at Boulder
Department of Computer Science
Boulder, Colorado 80309–0430
{stolle,lizb}@cs.colorado.edu

Abstract

We present a knowledge representation and reasoning framework that integrates qualitative reasoning, qualitative simulation, numerical simulation, geometric reasoning, constraint reasoning, resolution, reasoning with abstraction levels, declarative meta-level control, and a simple form of truth maintenance. The framework is the core of PRET, a system identification program that automates the process of modeling physical systems.

Introduction

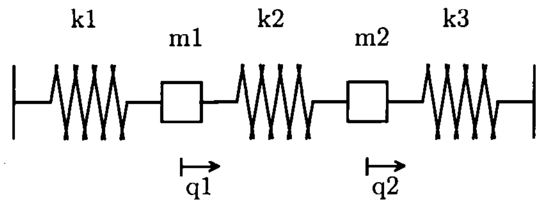
Models are powerful tools that are used to understand physical systems. The process of inferring an internal model from external observations of a system—often called *system identification*—is a routine and difficult problem faced by engineers in a variety of domains (Astrom & Eykhoff 1971; Ljung 1987). Abstract models are simple: they account for major properties of the physical system. Less-abstract models are more complicated, allowing them to capture the features of the physical system more accurately and in more detail, but at the cost of increased complexity during model construction and usage. Typically, in the hierarchy from more-abstract to less-abstract models, the model of choice is the one that is just detailed enough to account for the properties and perspectives that are of interest for the task at hand. The first stage of the system identification proces, *structural identification*, identifies the form of the model, or skeleton of the equation, such as $a\ddot{\theta} + b\sin\theta = 0$ for a simple pendulum. In the second system identification stage, *parameter estimation*, the parameter values a and b are determined.

The program PRET (Bradley & Stolle 1996) automates both stages of the system identification process; its goal is to find a system of ODEs that models a given physical system. Inputs are *observations* about that system, user-supplied *hypotheses* about the desired model, and *specifications*.

Observations are measured automatically by sensors and/or interpreted by the user; they may be symbolic or numeric and take on a variety of formats and degrees of precision. For example, an observation might inform PRET that the system to be modeled is autonomous with respect to the state variable q; another observation could state that q oscillates and that this oscillation is damped. Observations can also be physical measurements made directly and automatically on the system (Bradley & Easley 1997). Hypotheses about the physics involved, e.g., a hypothesis about friction, are supplied by the user; these may conflict and need not be mutually exclusive, whereas observations are always held to be true. Finally, specifications indicate the quantities of interest and their resolutions—a specification might impose a microsecond resolution over 120 seconds of system evolution. Fig. 1 shows a physical system that consists of two masses and three springs and exemplifies how one instructs PRET to construct a model of that system. It is important to note that this simple linear example does not by any means exercise PRET's power.

When modeling physical systems, human engineers make use of several existing modeling techniques at various different abstraction levels. A modeler's reasoning about a given physical system and possible candidate models takes place at an abstract level first and resorts to more detailed reasoning later in the modeling process. PRET's goal is to mimic this strategy and attempt to find the right model at the right abstraction level as quickly as possible. Therefore, the challenge in writing PRET was to design a formalism that meets two requirements: first, it must facilitate easy formulation of the various reasoning techniques; and second, it must allow PRET to reason about which techniques are appropriate in which situation. This paper describes PRET's reasoning modes, how they are integrated, and

*Supported by NSF NYI #CCR-9357740, ONR #N00014-96-1-0720, and a Packard Fellowship in Science and Engineering from the David and Lucile Packard Foundation.



```
(find-model
  (domain mechanics)
  (state-variables <q1> <q2>)
  (point-coordinates <q1> <q2>)
  (hypotheses
    ((<force> (* m1 (deriv (deriv <q1>))))
    ((<force> (* m2 (deriv (deriv <q2>))))
    ((<force> (* k1 <q1>))
    ((<force> (* k2 (- <q1> <q2>)))
    ((<force> (* k3 <q2>))
    ((<force> (* r1 (deriv <q1>)))
    ((<force> (* r2 (square (deriv <q1>))))
    ((<force> (* r3 (deriv <q2>)))
    ((<force> (* r4 (square (deriv <q2>)))))
  (observations
    (autonomous <q1>)
    (autonomous <q2>)
    (oscillation <q1>)
    (oscillation <q2>)
    (numeric (<time> <q1> <q2>)
            ((0 .1 .1) (.1 .1099 .1103) ...  )))
  (specifications
    (resolution <q1> absolute 1e-3 (0 1))
    (resolution <q2> absolute 1e-3 (0 1))))
```

Figure 1: A `find-model` call that instructs PRET to model the `Springs&Masses` system shown above. In this example, the user first sets up the problem, then hypothesizes nine different force terms, makes five observations about the position coordinates q_1 and q_2, and finally specifies resolutions and ranges.

how they solve these problems.

The next section gives a brief overview of PRET. In the following sections, we present the various reasoning modes, their interaction and integration. We also suggest additional reasoning modes for future integration. Finally, we use two examples to illustrate how PRET works.

Reasoning Modes

The high-level control flow of PRET is a variant of "generate-and-test." In the "generate" phase, candidate models are constructed from the user's hypotheses via simple, powerful domain rules such as, for example, Kirchhoff's voltage law for electronic circuits. Since PRET tries to find a model that accounts for all observations without being more complex than necessary, candidate models are generated in order of increasing complexity. In the "test" phase, PRET uses ODE rules to generate new knowledge about the physical system and new knowledge about the current candidate model. A model is valid if the facts about the system that is to be modeled are consistent with the facts about the model. Any inconsistency is a reason to discard the model. The first model in this generate-and-test sequence that is consistent with the observations is currently returned as the result.

Using this "valid if not proven invalid" paradigm, it is important to rule out invalid candidate models as quickly as possible. Therefore, PRET uses abstract knowledge first and detailed knowledge later. In order to achieve this behavior, PRET chooses among and orchestrates several reasoning modes. In the following subsections, we describe these reasoning modes and show how PRET decides which mode is appropriate in which situation.

Qualitative Reasoning

Reasoning about abstract features of the physical system or the candidate model is typically faster than reasoning about their detailed properties. Therefore, PRET uses a "high-level first" strategy: it tries to rule out a model by purely *qualitative* techniques before advancing to more expensive semi-numerical or numerical techniques. Often, only a few steps of inexpensive qualitative reasoning (QR) suffice to quickly discard a model. Some of these qualitative rules in turn make use of other tools; e.g., symbolic algebra facilities from the commercial package Maple. For example, PRET's ODE theory includes the qualitative rule that oscillating linear systems must be of at least second order. If the user's observations imply an oscillation, any model whose order is less than 2 can be discarded without performing more-complex operations such as, for example, a numerical integration of the model.

PRET's QR features are not only important for accelerating the search for inconsistencies between the physical system and the model; they also allow the user to express incomplete information. For example, the user might not know the exact value of a friction coefficient, but he or she might know that it is constant and positive.

Geometric Reasoning

Many inappropriate models can be discarded by reasoning about purely qualitative information that can be inferred from numeric information. In order to achieve this behavior, PRET processes the observations—curve fitting, recognition of linear regions, and so on—using Maple functions and simple phase-portrait analysis techniques, both of which yield high-level results that can then be used much as qualitative observations are (Bradley & Easley 1997).

Qualitative Simulation

Before PRET resorts to the numerical level, it reasons about the qualitative states of the physical system. PRET's qualitative envisioning module constrains the possible ranges of parameters in the candidate model. If the constraints become inconsistent—i.e., the range of a parameter becomes the empty set—the model is ruled out. Currently, the qualitative states contain only sign information $(-, 0, +)$. For example, for the model $ax + by = 0$ the state $(x, y) = (+, +)$ constrains (a, b) to the possibilities $(+, -)$ or $(0, 0)$ or $(-, +)$.

PRET does not do full qualitative simulation (Kuipers 1986). Instead, it only envisions the state space of all possible combinations of qualitative values of state-variables and parameters. This strategy is faster than full qualitative simulation, but it is also less accurate, i.e., it may let invalid models pass the test. These invalid models are later ruled out by the numeric simulator. However, for the models that do fail the qualitative envisioning test, this test is much cheaper than a numeric simulation and point-by-point comparison would be.

Parameter Estimation and Numerical Simulation

If the observations contain a numerical time series, the model must match the time series to within the specified resolution. If a candidate model cannot be discarded by qualitative means, PRET integrates the model (an ODE system) with fourth-order Runge-Kutta, comparing the result to the numeric time-series observation. Typically, models contain parameters whose values must be determined before the numeric integration can take place. For example, for the model $a\ddot{\theta} + b \sin\theta = 0$ for a simple pendulum the parameter values a and b must be determined. PRET calculates these values using its nonlinear parameter estimation reasoner (NPER) (Bradley, O'Gallagher, & Rogers 1997), which uses knowledge derived in the structural identification phase to guide the parameter estimation process—for example, choosing good initial values and thereby avoiding local minima in regression landscapes.

Constraint Reasoning

Often, information *between* the purely qualitative and the purely numeric levels is also available. For example, if a physical system oscillates, the imaginary parts of at least one pair of the roots of its model's characteristic polynomial must be nonzero. Thus, if the model $A_1\ddot{x} + A_2\dot{x} + A_3x = 0$ is to match an oscillation observation, the coefficients must satisfy the inequality $4A_1A_3 > A_2^2$. PRET uses expression inference (Sussman & Steele 1980) to merge and simplify such *constraints* (Jaffar & Maher 1994). However, this approach works only for linear and quadratic expressions and some special cases of higher order. We are investigating techniques to reason about more general expressions. For example, if the system $\ddot{x} + a\dot{x}^4 + b\dot{x}^2$ is to be conservative, the coefficients a and b must take on values such that the divergence $-4a\dot{x}^3 - 2b\dot{x}$ is zero, below a certain resolution threshold, for the specified range of interest of x.

SLD resolution

PRET's search for an inconsistency between the observations and a particular candidate model is based on SLD resolution. The observations and the ODE theory are expressed as Horn clauses. The special atomic formula `falsum` may only appear as the head of a clause. Such clauses express fundamental reasons for inconsistencies, e.g., that a system cannot be oscillating *and* non-oscillating at the same time. This concept of *negation as inconsistency* (Gabbay & Sergot 1986) is the only form of negation in our paradigm. A candidate model is ruled out if the inference engine can derive the formula `falsum` from the union of the observations and facts about the candidate model.

The special predicate `scheme-eval`[1] provides the link between the inference engine and all the modules that implement other reasoning modes.

Declarative Meta Level Control

The *control strategy* of a SLD resolution theorem prover is defined by the function that selects the literal that is resolved and by the function that chooses the resolving clause. PRET provides meta-level language constructs that allow the implementer of the ODE theory to specify the *control strategy* that is to be used (Beckstein, Stolle, & Tobermann 1996; Gallaire & Lasserre 1982). For example, a control rule can specify that the inference engine must attempt proofs for formulae with the predicate `order` before it tries to prove formulae with the predicate `oscillation`. The intuition here is, again, that the search should be guided towards a cheap and quick proof of a contradiction.

Reasoning at Different Abstraction Levels

Every rule in the ODE theory is assigned a natural number that indicates the level of abstraction of the rule: the lower the *abstraction level number*, the more abstract the rule. Whereas the meta predicates described in the previous paragraph specify *dynamic* control, the ODE rules' abstraction levels express *static* control information. The theorem prover

[1]PRET is written in Scheme.

proceeds to a higher abstraction level number only if the attempt to prove the `falsum` with ODE rules with lower abstraction level numbers fails. For example, the `scheme-eval`-rule that triggers numerical integration has a higher abstraction level number than the `scheme-eval`-rule that calls the qualitative simulation.

Storing and Reusing Intermediate Results

PRET reuses previously derived knowledge in two ways. First, knowledge about the physical system is global, whereas knowledge about a candidate model is local to that model. Therefore, PRET reuses knowledge that is independent of the current candidate model.

Second, knowledge is reused within the process of reasoning about one particular model. Every time the reasoning proceeds to a less abstract level, PRET needs all information that has already been derived at the more abstract level. To avoid duplication of effort, PRET stores this information rather than rederiving it. The user declares a number of predicates as *relevant* (Beckstein & Tobermann 1992) which causes all succeeding subgoals with this predicate to be stored for later reuse. Currently, PRET recognizes special cases and generalizations of previously proved formulas but maintains no contexts or labels (de Kleer 1986) for intermediate results. We are investigating the benefits of expanding this module to a full truth maintenance system. For example, PRET may generate explanations (proof trees) for failures of models and use a form of *discrepancy-driven refinement* (Addanki, Cremonini, & Penberthy 1991) to guide the generation of better models.

Future Extensions

There are other modes of reasoning that we are considering adding to PRET. A form of case-based reasoning could help choose promising terms from a set of hypotheses that are generated automatically by power-series expansion. This technique will become important when the model generator runs out of user-supplied hypotheses.[2]

Currently, the control rules are specified by the implementer of the ODE theory. Future incarnations of PRET may monitor typical modeling runs and derive good control rules automatically (Ginsberg & Geddis 1991; Minton 1996).

Example Applications

In this section we trace PRET's actions on the example of Fig. 1. The four friction forces are omitted here for space reasons, and the numeric observation has the form

```
(numeric (<time> <q1> <q2>) (eval *data*))
```

The keyword `eval` causes the variable `*data*` to be evaluated in the calling environment. Bound to this variable is a time series that was generated by Runge-Kutta integration of the system

$$\begin{aligned} \ddot{q}_1 &= -0.1\,q_1 - 0.2\,(q_1 - q_2) \\ \ddot{q}_2 &= 0.2\,(q_1 - q_2) - 0.3\,q_2. \end{aligned}$$

The first candidate model is $k_1 q_1 = 0$. A Scheme function called on the ODE establishes the fact `(order <q1> 0)` which expresses that the order of the highest derivative of q_1 in this model is zero. This fact conflicts with facts inferred from the observation `(oscillation <q1>)`, so this model is ruled out. The way PRET handles this first candidate model demonstrates the power of its abstract-reasoning-first approach; only a few steps of inexpensive qualitative reasoning suffice to let it quickly discard the model.

PRET tries all combinations of `<force>` hypotheses at single point coordinates, but all these models are ruled out for qualitative or numeric reasons. It then proceeds with ODE systems that consist of *two* force balances—one for each point coordinate. One example of a candidate model of this type is

$$\begin{aligned} k_1 q_1 + m_1 \ddot{q}_1 &= 0 \\ m_2 \ddot{q}_2 &= 0 \end{aligned}$$

None of the implemented rules discards this model by purely qualitative means, so PRET invokes its nonlinear parameter estimation reasoner (NPER) which finds no appropriate values for the coefficients k_1, m_1, and m_2 such that any ODE solution matches the numeric time series. Therefore, this candidate model is also ruled out.[3]

After having discarded a variety of unsuccessful candidate models in a similar manner, PRET tries the model

$$\begin{aligned} k_1 q_1 + k_2(q_1 - q_2) + m_1 \ddot{q}_1 &= 0 \\ k_3 q_2 + k_2(q_1 - q_2) + m_2 \ddot{q}_2 &= 0 \end{aligned}$$

Again, it calls the NPER, this time successfully. It then substitutes the returned parameter values for the constants k_1, k_2, k_3, m_1, and m_2 and integrates the resulting ODE system with fourth-order Runge-Kutta,

[2]In this situation the current version of PRET generates ODE terms automatically using a variety of power-series expansions and uses these terms in order of increasing complexity for model generation.

[3]In certain cases, this approach may result in a departure from the paradigm *valid if not proven invalid*; unless we trust that the parameter estimator *always* finds a set of coefficients if such a set exists, this amounts to *negation as failure*. This is the appropriate decision here because a user who supplies a numeric time series is certainly interested in a numerically accurate ODE model.

comparing the result to the numeric time-series observation. The difference between the integration and the observation stays within the specified resolution, so the numeric comparison yields no contradiction and this candidate model and the parameter values are returned as the answer.

We have chosen the simple spring-mass example of Fig. 1 to make this presentation brief and clear. Linear systems of this type are easy to model; no engineer would use a software tool to do generate-and-test and guided search to find an ODE model of a system so simple and well-understood. This example is representative neither of PRET's power nor of its intended applications—nonlinear, high-dimensional, black-box systems. Modeling these types of systems is where PRET's mixture of exact and approximate techniques, quantitative and qualitative reasoning, and precise and heuristic knowledge becomes truly powerful.

The following paragraphs describe PRET's application to a real-world example: the radio-controlled (R/C) cars used in the University of British Columbia's soccer-playing robot project. These commercially acquired cars cannot be controlled without an accurate ODE model of their dynamics—something that is not part of the specifications sheet. The sensor data consists of the car's position in an (x, y)-plane and its orientation θ. PRET goes through the structural identification process that we have already seen in the `Springs&Masses` example: it uses force balances to assemble hypotheses into models; it examines hypothesis combinations in order of increasing complexity; and it discards models that are inconsistent with the observations, always taking advantage of its abstract-reasoning-first approach. The result, in this example, is the model:

$$\begin{aligned}\dot{x} &= v\cos\theta \\ \dot{y} &= v\sin\theta \\ \dot{\theta} &= \rho v \\ \dot{v} &= \alpha + \gamma v\end{aligned}$$

where v is the car's velocity, θ its orientation, ρ the position of the steering wheel, α acceleration, and γ friction.[4] Following the structural identification phase, PRET calls the NPER; as in the previous example, qualitative knowledge derived during the former are used in the latter—for example, symbolic algebra, constraint propagation, and divided differences are used to compute the initial values and bounds that are passed to the local least-squares solver that lies at the heart of the NPER.

[4]This formulation of the model makes use of two different reference frames; PRET's formulation of the model looks more complicated. Allowing the user to express hypotheses in appropriate reference frames is a deep and important problem that we will address in future work.

PRET's solution to the modeling problem surprised the University of British Columbia analyst. The model did not match his intuition because he had omitted some important information from the specification—including the fact that the car started from rest. In order to work around noise in sensor data, PRET's NPER is designed to filter data and adjust boundary conditions; in this case, this reasoning led to a numerically successful ODE model with a negative initial condition for the velocity. Further reflection on the discrepancy led the analyst to realize that the system dynamics might include a delay. Thus, the correct `find-model` call for this example should contain an `observation` that the initial velocity was zero and a `hypothesis` that incorporates a delay between the application of force and the acceleration of the car. We use this anecdote to emphasize that PRET is an engineer's tool, not a scientific discovery system. Its goal is to construct the simplest ODE that accounts for the observations and specifications that are *explicit* in the `find-model` call, not to infer physics that the user left implicit. In this example, the returned model enabled the expert to identify what was wrong with the observations and model fragments he suggested. For a more detailed discussion see (Bradley, O'Gallagher, & Rogers 1997).

Related Work

Most automated modeling programs (Forbus 1990; Addanki, Cremonini, & Penberthy 1991; Falkenhainer & Forbus 1991; Amsterdam 1992; Kuipers 1993; Nayak 1995) use domain knowledge in order to map a structural description of the system to model fragments. PRET , however, uses only general mathematics in its analysis and does not rely on knowledge specific to the domain in question. Also, PRET's aim is not to discover the underlying physics of the system, but rather to find the simplest ODE that is consistent with the observed behavior.

In other automated modeling systems one of the important features of the formalized domain theory is that it suggests how structural components of the physical system map to candidate model fragments. PRET's true targets, however, are physical systems that do not have a well-defined domain theory, such as complex industrial devices and processes. The R/C car was a first step in this direction; the next test case, on which we are currently working, is a complex machine tool that takes a plastic blank and a prescription and automatically produces an eyeglass lens.

PRET aims to integrate quantitative and qualitative information. Many good papers have reported work in this area, among which are (Farquhar & Brajnik 1994; Williams 1991; Kay & Kuipers 1993).

Reasoning about physical systems is an extremely active research area. Space restrictions precluded the inclusion of many other papers that should have been mentioned here.

Conclusion

System identification is a necessary first step in many control-theory problems; without a reasonable model of the dynamics, very few systems are amenable to control. PRET automates this process by building an AI layer on top of a set of the kinds of traditional SID techniques that human experts use to solve these problems: e.g., regression, curve-fitting, matrix methods, fast-fourier transforms and filtering, root-locus plots, etc. Integrating this collection of techniques, which are diverse both in methods and in reasoning levels, was an important design goal for the inference system described in this paper. The successful design of that framework allows PRET to intelligently assess the task at hand, and then automatically choose, invoke, and interpret the results of appropriate lower-level methods.

Acknowledgements

Matt Easley, Apollo Hogan, Brian LaMacchia, Agnes O'Gallagher, Janet Rogers, Ray Spiteri, and Tom Wrensch contributed ideas, code, and/or R/C car data to this project. We would also like to thank the QR community for helpful comments.

References

Addanki, S.; Cremonini, R.; and Penberthy, J. S. 1991. Graphs of models. *Artif. Intell.* 51:145–177.

Amsterdam, J. 1992. *Automated Qualitative Modeling of Dynamic Physical Systems.* Ph.D. Dissertation, MIT.

Astrom, K. J., and Eykhoff, P. 1971. System identification — a survey. *Automatica* 7:123–167.

Beckstein, C., and Tobermann, G. 1992. Evolutionary logic programming with RISC. In *4th Intl. Workshop on Logic Programming Environments.* Washington, D.C. Tech. Rep. TR 92-143, Center for Automation and Intelligent Systems Research at Case Western Reserve University.

Beckstein, C.; Stolle, R.; and Tobermann, G. 1996. Meta-programming for generalized horn clause logic. In *META-96.*

Bradley, E., and Easley, M. 1997. Reasoning about sensor data for automated system identification. In *IDA-97.*

Bradley, E., and Stolle, R. 1996. Automatic construction of accurate models of physical systems. *Annals of Mathematics and Artif. Intell.* 17:1–28.

Bradley, E.; O'Gallagher, A.; and Rogers, J. 1997. Global solutions for nonlinear systems using qualitative reasoning. In *QR-97.*

de Kleer, J. 1986. An assumption-based truth maintenance system. *Artif. Intell.* 28:127–162.

Falkenhainer, B., and Forbus, K. D. 1991. Compositional modeling: Finding the right model for the job. *Artif. Intell.* 51:95–143.

Farquhar, A., and Brajnik, G. 1994. A semi-quantitative physics compiler. In *QR-94.*

Forbus, K. D. 1990. The qualitative process engine. In Weld, D. S., and de Kleer, J., eds., *Readings in Qualitative Reasoning About Physical Systems.* San Mateo CA: Morgan Kaufmann.

Gabbay, D. M., and Sergot, M. J. 1986. Negation as inconsistency I. *J. Logic Progr.* 3(1):1–36.

Gallaire, H., and Lasserre, C. 1982. Metalevel control for logic programs. In Clark, K. L., and Tärnlund, S. A., eds., *Logic Programming.* London: Academic Press.

Ginsberg, M., and Geddis, D. 1991. Is there any need for domain-dependent control information? In *AAAI-91.*

Jaffar, J., and Maher, M. J. 1994. Constraint logic programming: A survey. *J. Logic Progr.* 20:503–581.

Kay, H., and Kuipers, B. 1993. Numerical behavior envelopes for qualitative models. In *AAAI-93.*

Kuipers, B. J. 1986. Qualitative simulation. *Artif. Intell.* 29(3):289–338.

Kuipers, B. J. 1993. Reasoning with qualitative models. *Artif. Intell.* 59:125–132.

Ljung, L., ed. 1987. *System Identification; Theory for the User.* Englewood Cliffs, N.J.: Prentice-Hall.

Minton, S. 1996. Is there any need for domain-dependent control information? A reply. In *AAAI-96.*

Nayak, P. 1995. *Automated Modeling of Physical Systems.* Berlin: Springer. LNCS 1003.

Sussman, G. J., and Steele, G. L. 1980. CONSTRAINTS — a language for expressing almost hierarchical descriptions. *Artif. Intell.* 14:1–39.

Williams, B. C. 1991. A theory of interactions: unifying qualitative and quantitative algebraic reasoning. *Artif. Intell.* 51:39–94.