Automatic Selection of Bounding Abstractions

Daniel S. Weld*
Department of Computer Science and Engineering, FR-35
University of Washington
Seattle, WA 98195
weld@cs.washinton.edu

Abstract

Since the complexity of model-based reasoning increases drastically with the model size, automated modeling has become an active research area. However, unlike human engineers, few modeling programs introduce approximations that are customized to the question at hand. In this paper, we focus on a single aspect of automated model management: shifting model accuracy. We describe a domain-independent theory of query-directed model simplification that uses *bounding abstractions* to guarantee the accuracy of the simplifications introduced. We have tested our theory by implementing the SUP program which evaluates inequality relations in an approximate model without sacrificing accuracy. These techniques are based on our previously reported work on *model sensitivity analysis* (MSA) and *fitting approximations*. SUP uses MSA, the subtask of predicting how a change in models will affect the resulting predicted behavior, to determine if one model is a bounding abstraction of another. Whenever two models are related by fitting approximations, the MSA computation reduces so the simple problem of computing the sign of partial derivatives in a single model, a task which is easily performed by Mathematica.

Introduction

A central problem in automated reasoning about physical systems is that the complexity of reasoning increases drastically with the size of the system description. However, while automated modeling has become an active research area, many approaches assume that a hierarchy of models is provided as input. Also, few of the modeling programs introduce approximations that are custom generated for the question at hand, even though this is a crucial aspect of a human expert's modeling choices.

In this paper, we focus on a single aspect of automated model management: shifting model accuracy. We suggest that changes of accuracy can be usefully decomposed into shifts that *simplify* and those that *refine* the model. We argue that model simplification should be done in the context of a particular goal and that simplification operators should provide some sort of guarantee on the accuracy of the simplified model. Our approach is based on our previously reported work on *model sensitivity analysis* (MSA) — the problem of predicting how a change in models will affect the resulting predicted behavior over time [16, 18]. We explore the class of model relations called *fitting approximations*[1] for which the problem of model sensitivity analysis is reduced to one of computing the sign of partial derivatives in a single model.

In summary, this paper describes an implemented, domain-independent theory of query-directed model simplification that uses *bounding abstractions* to provide a guarantee on the answers to ordinal queries produced using an approximate model. Model sensitivity analysis is used to compute the bounding abstractions whenever models are related by fitting approximations.

Reasoning about Model Accuracy

The system dynamics view of a MODEL is a finite description of reality, constructed for the purpose of answering particular questions. As a first step in building programs that can choose the most useful model for a given question, it is essential to understand the space of *possible* models. In [18], we argue that this space has several independent dimensions: scope, domain of applicability, resolution, and accuracy. For example, a model of a refrigerator has larger scope than one of

*This work benefited from conversations with (and comments from) Mike Williamson, Ying Sun, J. Scott Penberthy, Dorothy Neville, Pandu Nayak, Leo Joskowicz, Steve Hanks, Oren Etzioni, Tony Barrett, Franz Amador and Sanjaya Addanki. The research was funded in part by National Science Foundation Grants IRI-8902010 and IRI-8957302, Office of Naval Research Grant 90-J-1904, and a grant from the Xerox corporation.

[1]In [16] model-sensitivity analysis was termed "intermodel comparative analysis," a term we have abandoned. Also in [16] the treatment of fitting approximations incorporated an arbitrary vocabulary reformulation (algebraicly, a shift of basis) under the name "approximation reformulation." We believe the terminology in this paper, which agrees with [18], is much easier to understand and use.

the refrigerator's compressor. We say that one model has greater domain of applicability than another if the first is usefully predictive for a wider set of input values for the exogenous parameters. Resolution refers to the precision of the model's output; for example, a qualitative model has lower resolution than a quantitative description. Finally, the accuracy of a model gauges how closely the model's predictions match the reality being modeled.

This paper is just about model accuracy — it describes algorithms for selecting, evaluating, and switching modeling approximations, but says nothing about how a program might switch scope, domain, or resolution. We believe that it is appropriate to consider each of these dimensions in isolation. Any complex modeling switch can be broken up into smaller shifts along each of the independent dimensions. To gain insight into the overall process of automated model management, we suggest focusing attention on each dimension in turn.

We prefer the GRAPH OF MODELS (GoM) framework of Addanki et al. [8, 1, 2] to analyze the dimension of model accuracy. A GoM is a directed graph in which nodes represent models of the system at hand and edges are labeled with the set of simplifying assumptions (i.e. approximations) that distinguish the two models. Given this conceptual framework, reasoning about model accuracy can be analyzed as a search through the graph (figure 1).

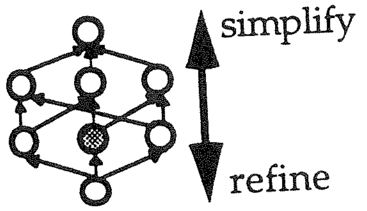

Figure 1: Navigation in a Graph of Models

The primitive operations in the search are the introduction and retraction of approximations. These operators, which we term SIMPLIFICATION and REFINEMENT can be visualized as moving upwards (towards one of the simplest models in the graph) and downwards (towards a maximally complex model) respectively.

When a model is too inaccurate, it must be refined. The objective, then, is to navigate downward in the GoM — i.e. to determine which assumptions to retract. To solve this task, the PROMPT system [1] introduced the methodology of discrepancy-driven refinement: using discrepancies between observations of the system's real-world behavior and the model's predictions to select which approximations to retract. Our work on SAM [16] provided a domain-independent extension of this technique using model sensitivity analysis over fitting approximations.

When a model is too complex, it must be simplified. Here, the objective is to move upward in the GoM, and the danger is that the resulting approximate model might be useless. To avoid shifting to a tractable but worthless model, it is necessary to consider the task at hand when performing simplification. Put another way, the simplification process should be query-directed and should provide a guarantee on the usefulness of the resulting model. This paper describes an implemented way to do exactly this: we simplify the model while providing a query-specific guarantee, bounding abstractions, by exploiting model sensitivity analysis.

Model Sensitivity Analysis

Whether one is trying to evaluate the effect of simplifying a model by including an approximation or of refining a model by retracting a simplifying assumption, it is essential to know the effect of the modeling shift on the behavior predicted. We refer to this key subroutine as MODEL SENSITIVITY ANALYSIS or MSA for short. MSA has three inputs: a source model, a destination model, and a variable of interest; as output the subroutine returns a description of the difference between the source and destination model's predictions of the variable's time-varying behavior. In general, performing MSA is very difficult; however, we have identified a class of modeling relations, called FITTING APPROXIMATIONS, for which the problem of MSA is tractable. Before defining fitting approximations formally, we clarify our terminology.

We consider systems that can be modeled with a set of algebraic and ordinary differential equations, their "dynamical constraints." We distinguish between independent (and assumed constant), boundary and dependent variables of a model and use the functions INDEP, BOUND, and DEPEND to specify them given a model as argument. We use the function RANGE to specify the range of possible values of a variable. We assume that a set of initial values for the independent and boundary variables completely specifies the values for the dependent variables over all time. We call such a solution to the model's dynamical constraints the behavior of the variable; the complete solution to the model's constraints for all variables is called the behavior of the model. If $\mathcal{P}$ is a model with m variables, and v_i is an initial value for variable p_i (or the null value if the i_{th} variable is dependent), then we denote the behavior of $\mathcal{P}$ with $\text{BEHAV}(\mathcal{P}, v_1, \ldots, v_m) = \vec{P} =< P_1, \ldots, P_m>$ Note that we use script letters to denote models, lowercase letters to describe both variables and values, and capital letters to denote behaviors. Thus P_j denotes the behavior of variable p_j as specified by model $\mathcal{P}$ given the initial values $\prec v_1, \ldots, v_m \succ$.

Fitting Approximations

Since fitting approximations make it easy to compute MSA, they are quite important. Intuitively, one model is a fitting approximation of another when the behaviors they predict can be brought arbitrarily close to zero. Thus the first step in defining fitting approximations is making clear the idea of behavior difference.

Definition 1 *Let $\vec{P} =< P_1, \ldots, P_m >$ and $\vec{Q} =< Q_1, \ldots, Q_n >$ be behaviors where $m \leq n$. Define the* BEHAVIOR DIFFERENCE BETWEEN $\vec{P}$ AND $\vec{Q}$ OVER THE TIME INTERVAL $[t_s, t_f]$ *as*

$$\text{BDIFF}(\vec{P}, \vec{Q}, t_s, t_f) = \max_{1 \leq i \leq m} \left(\sup_{t \in [t_s, t_f]} |P_i(t) - Q_i(t)| \right)$$

In other words, for each parameter in the smaller model, we compare corresponding values in the complex model for all times and take the supremum (least upper bound) of the absolute differences. The behavior difference is simply the L_∞ norm from functional analysis, i.e. the maximum value of the suprema. We are now ready to define fitting approximation.

Definition 2 *Let $\mathcal{P}$ and $\mathcal{Q}$ be models with m and n parameters respectively where $m < n$. We say that $\mathcal{P}$ is a* FITTING APPROXIMATION *of $\mathcal{Q}$ if there exists an independent parameter q_f of $\mathcal{Q}$ such that $m < f \leq n$, and there exists an endpoint l of the closure of* RANGE(q_f) *such that forall times t, and for all initial values $\prec v_1, \ldots, v_n \succ$ where* BEHAV$(\mathcal{Q}, v_1, \ldots, v_n)$ *is defined,* BEHAV$(\mathcal{P}, v_1, \ldots, v_m)$ *is also defined, and*

$$\lim_{v_f \to l} \text{BDIFF}((\text{BEHAV}(\mathcal{P}, v_1, \ldots, v_m), \text{BEHAV}(\mathcal{Q}, v_1, \ldots, v_n), 0, t) = 0$$

In this case, the parameter q_f is called the FITTING PARAMETER *of the approximation and l is called its* APPROXIMATION LIMIT.

In other words, model $\mathcal{P}$ approximates model $\mathcal{Q}$ if the difference between the predicted behaviors squeezes to zero as the initial value of the fitting parameter goes to the approximation limit. The natural way to think of the fitting parameter is as an exogenous input to the complex model, $\mathcal{Q}$, that specifies a degree of freedom by which the two models can be made to produce similar predictions. Figure 2 illustrates this abstractly.

P_i is a simple function of time, decreasing from $P_i(0) = p$ to $P_i(t_1) = 0$. Since all variables in the simple model are unaffected by the value of the fitting parameter, P_i can be viewed as a flat plane — movement along the fitting parameter axis does not change its value. The value of Q_i, however, is affected by the value of the fitting parameter so it defines a more complex, curved surface (shaded). As the fitting value approxes the approximation limit, l, Q_i acts more and more like P_i. But when the value of the fitting parameter is far from l, say $v_f = k$, then Q_i is initially higher $Q_i(0, k) = q$ and furthermore Q_i takes longer $(t_2 - t_1)$ to reach zero.

As a simple example, suppose $\mathcal{P}$ is a set of equations modeling a kinematic linkage and $\mathcal{Q}$ contains the

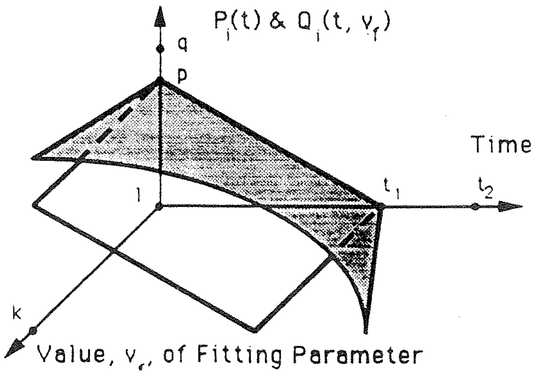

Figure 2: As $v_f \to l$, Q_i converges to P_i.

same equations with the addition of a frictional coefficient, μ, and frictional forces. $\mathcal{P}$ is a fitting approximation of $\mathcal{Q}$ because as $\mathcal{Q}$'s parameter μ tends to zero, the difference between the two model's predictions vanishes. Unfortunately, the previous example is misleading since it makes the concept of fitting approximation seem trivial. In particular, it might appear that there is no need to talk about limits since one could simply set μ to zero and convert $\mathcal{Q}$ to $\mathcal{P}$ with algebraic simplification. However, in a number of fitting approximations the differences in predicted behavior only vanish when the fitting parameter reaches a limit of infinity; substitution is clearly impossible in these cases. In fact infinite approximation limits are actually quite common. For example, the model of a uniform gravitational field is a fitting approximation of the $\frac{1}{r^2}$ law of gravity — as r goes to infinity the field becomes uniform. Economics provides another example: perfect competition approximates Cournot's theory of market as the number of firms approaches infinity. As demonstrated with table 3, fitting approximations are ubiquitous.

Fitting Approximations for MSA

While no algorithm is known for computing MSA in the general case, it is both possible and practical when one model is a fitting approximation of the other. In this case, model sensitivity analysis reduces to an ordinary sensitivity analysis with respect to the fitting parameter in the complex model. For example, the effect of switching from a frictionless model to one with friction is qualitatively equivalent to the effect of increasing μ in the complex model. In general, to compute the difference in the behavior predicted by *eliminating* a fitting approximation, one can simply follow the partial derivative with respect to the fitting parameter *away* from the approximation limit. The effect of the opposite model shift (i.e. *adding* an approximation) is calculated by following the partial *towards* the limit. We express this formally with:

Proposition 1 (MSA Reduction Theorem)

Massless objects	Zero moment of inertia	Elastic collisions
Inelastic string	Zero-radius pulleys	Rotation w/o slippage
Frictionless motion	Zero gravity	Uniform gravity
Nonrelativistic motion	Constant resistivity	Momentumless flow
Ideal thermal insulation	Ideal electrical insulation	Ideal gases
Incompressible fluid	Inviscid flow	Carnot heat pump
Reversible gas expansion	Infinite sources and sinks	Perfect competition

Figure 3: Examples of Fitting Approximations

Let $\mathcal{P}$ and $\mathcal{Q}$ be models with m and n parameters respectively, such that $m < f \leq n$. Suppose that $\mathcal{P}$ is a fitting approximation of $\mathcal{Q}$ with fitting parameter q_f and approximation limit l. Let $\prec v_1, \ldots, v_n \succ$ denote a set of initial conditions for $\mathcal{Q}$ such that $\prec v_1, \ldots, v_m \succ$ are consistent initial conditions for $\mathcal{P}$, and let $P_j(t)$ denote the time varying behavior of the j-th variable of $\mathcal{P}$ given these initial conditions, and let $Q_{\bar{j}}(t, v_f)$ denote the value of the j-th variable of $\mathcal{Q}$ as a function of time and the initial value of the fitting parameter as defined by $Q_{\bar{j}}(t, v_f) = \pi_j(\text{BEHAV}(\mathcal{Q}, v_1, \ldots, v_f, \ldots, v_n))$.[2] *For any time t in a bounded interval $[0, t_e]$ if*

$$\lim_{v_f \to l} \frac{\partial Q_{\bar{j}}}{\partial v_f}(t, v_f) = \rho \neq 0$$

then there exists an $\epsilon > 0$ such that forall v satisfying $0 < |v - l| < \epsilon$ if $Q_{\bar{j}}(t, v)$ is defined, then the relation between $Q_{\bar{j}}(t, v)$ and $P_j(t)$ is specified by the following table:

	$l = \inf(\text{RANGE}(Q_j))$	$l = \sup(\text{RANGE}(Q_j))$
$\rho > 0$	$Q_{\bar{j}}(t, v) > P_j(t)$	$Q_{\bar{j}}(t, v) < P_j(t)$
$\rho < 0$	$Q_{\bar{j}}(t, v) < P_j(t)$	$Q_{\bar{j}}(t, v) > P_j(t)$

Proof: See [18].

For example, if the approximation limit is at the bottom of the range (as it is with μ in the example above) and the partial derivative of the j-th parameter is negative with respect to the fitting parameter (as is $\frac{\partial v}{\partial \mu}$), then the detailed model will predict a smaller value than the simple model (e.g., considering friction will decrease velocity). But if the limit of the partial derivative was positive, then the detailed model would predict a greater value than the simple model. If the approximation limit is at the top of the range (e.g. the approximation limit of k was ∞), then the situation is reversed. Unfortunately, if the limit of the partial derivative is zero, then one cannot conclude anything about the two models' relative predictions.

In other words, to compute the difference in the behavior predicted by *eliminating* a fitting approximation, one can simply follow the partial derivative with respect to the fitting parameter *away* from the approximation limit. One important limitation of this theorem results from the unspecified value of ϵ. Although the theorem guarantees that ϵ exists, it does not provide a way to calculate its value; yet the reduction of inter-model changes to an intra-model test is only valid when one evaluates the partial derivative no farther than ϵ from the approximation limit. In many cases, ϵ is very large; for example, in the friction example there is no bound on ϵ and one may evaluate the partial derivative anywhere for a correct result. However, in general, this is not guaranteed to be true; developing techniques to predict the maximum value of ϵ is a very important topic for future research.

The beauty of this result is that it does not commit one to any particular method for computing the sign of the partial derivative. Thus one could perform comparative analysis [17] on a qualitative model, or one could use symbolic algebra techniques on real-valued equations and take the sin at the last minute, or use numerical techniques. In fact, we have implemented the first two of these techniques and are considering the third to handle differential equations.

Simplification with Bounding Abstractions

In this section we describe the details of our query-directed model simplification algorithm and its implementation. A major feature of our algorithm is the fact that it is provably sound: the answers produced by the simple, approximate model are guaranteed to match those of that would have been produced by the complex model. The basis for this property is best described in terms of Giunchiglia and Walsh's theory of abstraction [6], which we review below.

Types of Abstraction

Initially, we consider abstraction from a logicist perspective since this is its context which has been most fully explored. Unfortunately, in the study of logic, the word *model* is defined at variance with the fields of physics and system dynamics. What we refer to as a *model* is called a *theory* by logicians: loosely, a set of axioms and theorems. There are two obvious choices when defining theory $\text{Th}(\Sigma_p)$ to be more abstract than theory $\text{Th}(\Sigma_q)$: either $\text{Th}(\Sigma_p)$ should be satisfied

[2]The obsessive reader will note that $Q_{\bar{j}}(t, v_f)$ is a function of both time and the initial value of the fitting parameter while $Q_j(t)$ is simply a function of time. Thus the *domain* of $Q_{\bar{j}}$ is the cross product of the set of times with the predeclared *range* of Q_f.

by strictly more interpretations than $\mathrm{Th}(\Sigma_q)$ (which means it will have fewer theorems) or $\mathrm{Th}(\Sigma_p)$ should contain strictly more theorems than $\mathrm{Th}(\Sigma_q)$(in which case it will have fewer interpretations). Giunchiglia and Walsh call these two definitions TD-abstraction (for "theorem decreasing") and TI-abstraction ("theorem increasing") respectively [6]. TI-abstractions are useful because they guarantee that the absence of a theorem in the abstract theory implies the absence in the concrete theory. For example, considering the mutilated checkerboard as 30 white and 32 black squares is a TI-abstraction — since the abstract board is untilable it follows that the concrete board also cannot be tiled.

TD-abstractions provide the opposite guarantee, which, surprisingly, is equally useful: a TD-abstraction of a concrete theory has fewer theorems and therefore more interpretations than the concrete theory. For example, the qualitative equation $y = M^+(x)$ is a TD-abstraction of the real-valued equation $y = x^3$ because every solution of the real equation satisfies the qualitative equation, but the converse is not true [7, 20].[3]

In summary, a TD-abstraction lets one conclude less (but all the predictions are correct) while a TI-abstraction lets one conclude more (but some of the conclusions may be wrong). However, if one knows which type of abstraction one has, then all errors can be avoided by mapping down only proof failures for TI-abstractions and only mapping down proof successes for TD-abstractions.

TD-Abstractions from Approximations

Although approximations are commonly used in engineering analyses, especially with respect to continuous systems, there is no precise characterization of what an approximation *is*. Common sense suggests that the approximation should be simple and that the difference between original and approximation should be "small," but this informal description is insufficient for automated reasoning. Typical strategies for creating approximations are replacing a complex curve with a straight line and dropping low order terms. For example, the real equation for the molar ideal gas law$P = \frac{RT}{V}$ is an approximation (in fact a fitting approximation) of the Van der Waals equation $P = \frac{RT}{V-b} - \frac{a}{V^2}$. Each equation allows the calculation of pressure, but for nonzero values of a and b, the equations predict different values.

In certain cases, however, it is possible to *interpret* an approximation as a TD-abstraction. In other words, with extra information about an approximation, it is possible to map an approximate result into a guaranteed result in the original model. Elevation of an approximation to a TD-abstraction is possible in two cases:

- If the approximation always overestimates or always underestimates.
- If the approximation's error is bounded.

For example, compared to the Van der Waals equation the ideal gas law provides an upper bound for pressure. If the GoM link between the Van der Waals model and the ideal gas approximation is annotated with this information (or if that information is derivable), then a program can reason in terms of the region beneath the approximate curve, rather than in terms of the curve itself. Since the solution to the original equation is guaranteed to be in the region beneath the upper bound, we have satisfied the definition of a TD-abstraction: a strict increase in the set of solutions. Figure 4a illustrates the abstraction generated by an upper bound approximation. Figure 4b shows the abstraction produced by intersecting the upper and lower bounding regions.[4] Figure 4c illustrates the abstraction generated by an approximation with a percent-bound guarantee.

Query-Directed Simplification

We now return to the problem of answering quantitative, analytic queries that might occur during the verification of a proposed design. In particular, we consider the most basic step in verification: solving inequalities. Inequality reasoning is extremely common when ensuring that a design meets its specification. For example, whether an aircraft wing is strong enough to resist shearing in turbulence can be determined by solving as an inequality, as can the majority of verification tasks. While the algorithms behind inequality reasoners have been carefully analyzed [13, 10], the modeling questions have not. Given that there are a number of models in which one could verify a design, there are major advantages in choosing one that is a TD-abstraction. This strategy allows verification in an inexpensive, approximate model with the guaranteed same results as if the most complex model were used.

The trick is to use the inequality whose truth is being queried to select the approximate model. For example, if the query is "Necessarily $x > y$?" then one attains the guarantee by choosing an approximate model that underestimates x and overestimates y. If the approximate answer is "Yes" then the detailed model would certainly agree, but if the answer is "No" then no guaranteed information has been gained and backtracking must return reasoning to a more detailed model.

[3] In the terminology of physics, modeling *equations* take the place of a logical theory and *solutions* to the equations correspond to logical interpretations.

[4] The intersection, union or composition of two TD-abstractions is a TD-abstraction [18, proposition 2]. Note that the shaded region of the graph (representing the uncertainty inherent in the abstraction) has shrunk.

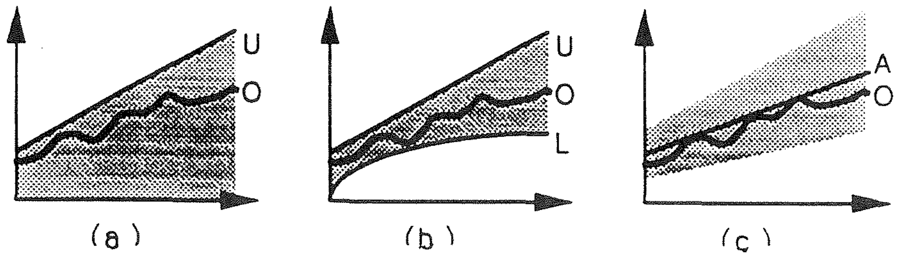

Figure 4: Approximations considered as TD-abstractions.

Implementation

To test the practicality of query-directed simplification, we have used Mathematica [22] to build the SUP program. To implement the simplification algorithm, we needed to commit to a specific backtracking strategy and also a method for computing MSA.

- **Implementing Model Sensitivity Analysis**
  SUP uses MSA to determine when a model shift is sounds relative to a query. If a GoM link has not been analyzed relative to any query before, SUP checks if it is tagged as a fitting approximation and uses Mathematica's symbolic partial derivative primitive to compute an expression for the MSA.[5] The resulting expression is simplified and cached so that subsequent calls to MSA on this link with respect to this parameter need not recompute the partial. Finally, the expression is coerced into a qualitative value by determining the range of possible values it may take on given the range restriction on exogenous parameters and any known values.[6] We note that this use of caching subsumes the mechanism behind parameter change rules [1], since SUP can reason about links in a GoM that do not correspond to fitting approximations if a user prestores values in the cache.

- **Search Strategy**
  SUP performs a variant of depth-first search as it traverses the GoM trying to prove an inequality. If a model has a link to an approximate model which is sound relative to the query, SUP pushes the current model on the stack and shifts to the approximate model until there is no sound simplification. If the algebraic solver fails to prove the inequality true in this minimal model, then SUP pops the next simplest model off the stack and calls the algebraic solver on that model. This process continues until the inequality is proven true, or the base model is reached. Clearly, this simple strategy is suboptimal; section discusses decision-theoretic control of modeling shifts.

Example

To illustrate SUP's behavior, consider the analysis of a hypothetical chemical reactor (figure 5) which calls for a positive displacement pump to maintain extreme pressure and temperature of the reactant gas, ammonia, as it flows through the catalytic bed [4]. The design verification step must determine whether the proposed lobe pump will be able to achieve the necessary temperature and pressure to enable the catalytic process. To verify that the design meets the specifications involves a choice of models for gas thermodynamics, pump performance, and load across the catalytic bed. In particular, when modeling gas behavior, one has a choice [14] of the ideal gas law, the Van der Waals equation, or the more complex Redlich / Kwong equation: $P = \frac{RT}{V-b} - \frac{a}{T^{\frac{1}{2}}V(V+b)}$. Similarly, there are several possible models for the pump: an idealized PDP model [15], a detailed second-order polynomial model relating backflow to pressure and fluid viscosity, and a simple linear lower bound. Even if we assume a fixed equation for load, the GoM for this example contains six models.

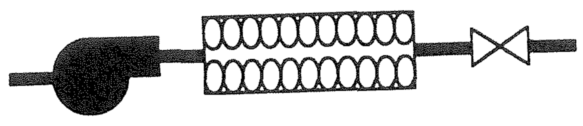
Figure 5: Simple design for verification

SUP is given constants for the maximum RPM for the pump, the volume of the reactor, the necessary temperature for catalysis, the real gas constant, and parameters of the gas: critical temperature, critical pressure, and viscosity.[7] With these values, SUP has enough information to evaluate the inequality "Can the pump produce enough pressure to enable catalysis in the gas?" in the most detailed model. However, SUP instead first simplifies the models.

[5]Note that this approach does not allow SUP to reason about differential equations, although it can handle polynomial and transcendental functions.

[6]Although the best way to do this would be with the BOUNDER inequality reasoner [10], the impossibility of linking Lisp with Mathematica version 1.2, caused us to use a simple stochastic bounder instead.

[7]The GoM would be even bigger if equations encoding the temperature dependence of viscosity were included.

First SUP notices that the ideal gas law is a fitting approximation of the Redlich / Kwong equation with two fitting parameters, a and b.[8] Computation of MSA shows that the ideal gas law provides an upper bound on the pressure required to reach a given temperature so that approximation is made. Next, SUP notes that a prestored cache entry marks the linear pump model as a lower bound,[9] so that approximation is made as well. At this point SUP has reached the simplest model, so it proceeds to calculate the achievable and required pressures. Since the results are 47.3197 and 46.6624 bar respectively, the inequality is guaranteed true, not only in this model but in the more detailed models as well and the analysis is complete. For example, if SUP had performed the analysis with the most detailed equations it would have estimated achieveable pressure at 48.3099 and the catalysis threshold as 44.4865. We can verify the use of a bounding abstraction by observing: $48.3099 \geq 47.3197 > 46.6624 \geq 44.4865$.

In addition to the example above, SUP has been tested on several dozen problems using five different GoMs. In all cases, SUP performed both MSA and the eventual calculations in a matter of seconds. However, as discussed below, more work is necessary to transform SUP into a practical tool.

The Cost of Model Shifting

One issue that is conspicuously neglected in the design of SUP is consideration for the cost of shifting models. Yet an intelligent modeling system should evaluate the computational tradeoff between time spent analyzing a model and time spent reasoning about which model to analyze. Decision theory is the natural framework for such a system, but several problems must be addressed to make this practical, chief among them is a means for cheaply estimating run time.

Note that since bounding abstractions provide a guarantee of correctness, there is no need to decide between the relative worth of an extra percent error vs an extra unit of CPU time. Utility can be simply defined as the multiplicative inverse of run time. The expected cost, C_e, of trying to shift from the base model, $\mathcal{Q}$, to a simpler model, $\mathcal{P}$, along a link $\mathcal{Q} \rightarrow \mathcal{P}$ when analyzing the query $Q_i < Q_j$ given initial conditions $\vec{v}$ can be written as:[10]

$$\begin{aligned} C_e(\mathcal{Q}\rightarrow\mathcal{P}, Q_i < Q_j, \vec{v}) = \; & C_e(\text{MSA}(\mathcal{Q},\mathcal{P},i)) + \\ & C_e(\text{MSA}(\mathcal{Q},\mathcal{P},j)) + \\ & \alpha C_e(\text{ANAL}(\mathcal{P}, P_i < P_j, \vec{v})) + \\ & (1-\alpha\beta) C_e(\text{ANAL}(\mathcal{Q}, Q_i < Q_j, \vec{v})) \end{aligned}$$

whether $\mathcal{P}$ is sound relative to $\mathcal{Q}$ and the query. The third term of the equation represents the cost of analyzing the inequality in the approximate model, and is only performed some of the time: α denoting the probability that $\mathcal{P}$ is an appropriate bounding abstraction. The fourth term accounts for the cost of analyzing the inequality in the original detailed model; this happens when $\mathcal{P}$ is not the desired bounding abstraction (with probability $1-\alpha$) or when the approximate model fails to prove the inequality (with probability $(1-\beta)$ of the times the model is tried, i.e. $\alpha(1-\beta)$) and backtracking is necessary. It only makes sense to try and shift models when this total expected cost is less than the cost of simply analyzing the inequality in the base model:

$$C_e(\text{ANAL}(\mathcal{Q}, Q_i < Q_j, \vec{v}))$$

Thus the SUP approach makes good sense if α and β are high (i.e. there is a good chance of finding a bounding abstraction that is unambiguous on the query), if the cost of analysis in the simple model is much less than that of the complex model and if the cost of computing MSA is less than the cost of analysis. We believe that compositional modeling (section) will ensure this latter condition, but runtime evaluation of the cost equation is likely to be difficult until a means is found for estimating the computational savings of an approximation.

Compositional Modeling

Throughout this paper we have talked in terms of a graph of models because it affords a clean conceptual foundation. On pragmatic grounds, however, the GoM approach is problematic (e.g., a space requirement that is exponential in the number of assumptions). The obvious solution is to represent a GoM implicitly and to dynamically instantiate only the models that are actually useful for analysis. Pioneering work on this approach, termed compositional modeling, suggests its substantial promise [5]. Query-directed approximation with bounding abstractions and MSA computation via fitting approximations fit perfectly into the compositional modeling framework. Since MSA can be computed on model fragments as easily as on complete GoM models, one can reason about bounding abstractions, fragment by model fragment. This is almost certain to make the expected cost of MSA negligible compared to that of model analysis. We are working on implementing a compositional modeling successor to the SUP program using a variant of the QPC algorithm [3].

[8]The quantities a and b are positive constants that can be computed from the critical temperature and pressure using $a = \frac{.42748 R^2 T_c^{(2.5)}}{P_c}$ and $b = \frac{.08664 R T_c}{P_c}$.

[9]The polynomial viscosity model does not have a fitting approximation lower bound, but it does have an upper bound which is a fitting approximation.

[10]Following [18] we use "MSA($\mathcal{Q}$, $\mathcal{P}$, i)" to denote the process of computing the change in the ith parameter (i.e. $Q_i - P_i$) when shifting from model $\mathcal{Q}$ to $\mathcal{P}$. At the expense of extra complexity, the cost equation can be extended to cover the case in which $\mathcal{Q}$ has multiple approximation links, and the possibility that $\mathcal{P}$ might be simplified as well.

Related Work & Conclusions

Shirley and Falkenhainer [12] suggest reasoning about the percent error of an approximation and Nayak [9] shows how it can be performed using the less expensive of a pair of model fragments in certain cases. Interestingly, Nayak concludes that MSA is necessary even when calculating the percent error of a modeling shift. Selman and Kautz [11] describe algorithms that compute the equivalent of bounding abstractions for theories in propositional logic. Williams [21] defines the critical abstraction of a quantitative component-connection model with respect to a query to be the maximally simple model that still is capable of answering the query. Cast in Williams' terminology, the SUP program's bounding abstractions could be considered to be "critical approximations" — maximal upper and lower bounds. Unlike Williams' program, SUP does not need to do a full analysis in the detailed model before simplification.

Our motivating goal is to build programs that can reason efficiently about complex systems by dynamically choosing simplifying assumptions and perspectives that are appropriate to the task at hand. This paper presented a framework for reasoning about model accuracy in terms of simplification and refinement that is based on model sensitivity analysis. Our main result is the notion of bounding abstraction, and a sound, MSA-based algorithm for performing query-directed model simplification when analyzing inequality relations.

References

[1] S. Addanki, R. Cremonini, and J. S. Penberthy. Reasoning about Assumptions in Graphs of Models. In *Proceedings of IJCAI-89*, August 1989. Reprinted in [19].

[2] S. Addanki, R. Cremonini, and J. S. Penberthy. Graphs of Models. *Artificial Intelligence*, 51(1-3):145–178, October 1991.

[3] J. Crawford, A. Farquhar, and B. Kuipers. QPC: A Compiler from Physical Models into Qualitative Differential Equations. In *Proceedings of AAAI-90*, pages 365–372, August 1990.

[4] K. Denbigh and J. Turner. *Chemical Reactor Theory*. Cambridge University Press, 1984.

[5] B. Falkenhainer and K. Forbus. Compositional Modeling: Finding the Right Model for the Job. *Artificial Intelligence*, 51(1-3):95–144, October 1991.

[6] F. Giunchiglia and T. Walsh. Abstract Theorem Proving. In *Proceedings IJCAI-89*, pages 372–377, August 1989.

[7] B. Kuipers. Qualitative Simulation. *Artificial Intelligence*, 29, September 1986. Reprinted in [19].

[8] S. Murthy and S. Addanki. PROMPT: An Innovative Design Tool. In *Proceedings of AAAI-87*, pages 637–642, August 1987.

[9] P. Pandurang Nayak. Validating Approximate Equilibrium Models. In *Proceedings of the 1991 AAAI Model-Based Reasoning Workshop*, July 1991.

[10] E. Sacks. Hierarchical Reasoning about Inequalities. In *Proceedings of AAAI-87*, pages 649–654, August 1987. Reprinted in [19].

[11] Selman, B. and Kautz, H. Knowledge Compilation using Horn Approximations. In *Proceedings of AAAI-91*, 1991.

[12] M. Shirley and B. Falhenhainer. Explicit Reasoning about Accuracy for Approximating Physical Systems. In *Proceedings of the AAAI workshop on Automatic Generation of Approximations and Abstractions*, July 1990.

[13] R. Simmons. "Commonsense" Arithmetic Reasoning. In *Proceedings of AAAI-86*, pages 118–124, August 1986. Reprinted in [19].

[14] J. M. Smith and H. C. Van Ness. *Introduction to Chemical Engineering Thermodynamics, 4th ed.* McGraw-Hill, 1987.

[15] J. Thorpe. *Mechanical System Components*. Simon and Schuster, Needham Heights, MA, 1989.

[16] D. Weld. Approximation Reformulations. In *Proceedings of AAAI-90*, pages 407–412, August 1990.

[17] D. Weld. *Theories of Comparative Analysis*. MIT Press, Cambridge, MA, May 1990.

[18] D. Weld. Reasoning about Model Accuracy. *Artificial Intelligence*, To Appear 1992.

[19] D. Weld and J. de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, San Mateo, CA, August 1989.

[20] M. Wellman. Qualitative simulation with multivariate constraints. In *Proceedings of the Second International Conference on Knowledge Representation*, May 1991.

[21] B. Williams. Capturing How Things Work: Constructing Critical Abstractions of Local Interactions. In *Proceedings of the AAAI workshop on Automatic Generation of Approximations and Abstractions*, July 1990.

[22] S. Wolfram. *Mathematica: A System for Doing Methematics by Computer*. Addison-Westley, Redwood City, CA, 1988.