Available online at www.sciencedirect.com
SCIENCE DIRECT®
Artificial Intelligence 162 (2005) 145–204
Artificial Intelligence
www.elsevier.com/locate/artint

Towards a practical theory of reformulation for reasoning about physical systems

Berthe Y. Choueiry [a,*], Yumi Iwasaki [b], Sheila McIlraith [c]

[a] *Constraint Systems Laboratory, Department of Computer Science and Engineering, University of Nebraska-Lincoln, Lincoln, NE 68588-0115, USA*
[b] *Woodinville, WA, USA*
[c] *Department of Computer Science, University of Toronto, Toronto, Ontario, Canada M55 3H5*

Received 20 November 2001; accepted 14 November 2002

Available online 15 December 2004

The authors would like to dedicate this article to the memory of Robert S. Engelmore. He was a long-time friend and senior colleague of the authors. He was also an important contributor to the initial discussion group at KSL that eventually lead us to write this article. We dearly miss him and regret that he did not live to read this product of the many discussions we had with him.

Abstract

Reformulation is ubiquitous in problem solving and is especially common in modeling physical systems. In this paper we examine reformulation techniques in the context of reasoning about physical systems. This paper does not present a general theory of reformulation, but it studies a number of known reformulation techniques to achieve a broad understanding of the space of available reformulations. In doing so, we present a practical framework for specifying, classifying, and evaluating various reformulation techniques applicable to this class of problems. Our framework provides the terminology to specify the conditions under which a particular reformulation technique is applicable, the cost associated with performing the reformulation, and the effects of the reformulation with respect to the problem encoding.

Keywords: Abstraction; Reformulation; Approximation; Reasoning about physical systems

* Corresponding author.
E-mail addresses: choueiry@cse.unl.edu (B.Y. Choueiry), Yumi_iwasaki@hotmail.com (Y. Iwasaki), sheila@cs.toronto.edu (S. McIlraith).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2004.01.004

1. Introduction

Reformulation plays an important role in various intellectual activities and is ubiquitous in reasoning about physical systems. Reformulation improves the effectiveness of a problem-solving process by recasting a problem into a new one that is tailored to a given task. There are a number of reformulation techniques and the selection of reformulation techniques must be carried out in the context of a problem-solving task.

In this paper we examine the role of reformulation in reasoning about physical systems. To achieve a broad understanding of the space of available reformulations, we study and discuss a number of known reformulation techniques. All the methods discussed in this paper have been presented in the literature, and some are in common use in science and engineering.

Our investigations yield two important contributions. First, they provide the first practical framework for a comprehensive description of reformulation techniques. This framework, based on our analysis of the field, allows us to characterize reformulation techniques including their applicability conditions and effects along with the assumptions and implicit evaluation criteria underlying them. Such a framework is necessary for the future development of an automatic mechanism to select a reformulation technique that is appropriate for a task. Second, our study produces a survey of reformulation techniques in reasoning about physical systems. Our framework was essential for undertaking this comparative analysis. Indeed, without a uniform framework from which to characterize reformulation techniques as diverse as we have considered, a meaningful comparison would have been impossible.

Informally, we define reformulation to be a *transformation* from one encoding of a problem to another, given a particular problem-solving task. A problem-solving task is accomplished by the application of a select sequence of reformulations to an initial problem encoding to produce a final encoding that directly addresses the task. We use the term reformulation to subsume the notions of *abstraction* and *approximation*, while avoiding the implication that such a transformation necessarily generalizes or simplifies the domain theory.

Given a reasoning problem, one may choose to reformulate for any of the following reasons:

1. *Engine-driven problem re-encoding*: There may not be a method to solve the given problem as is. In such a case, one may choose to reformulate the original problem by approximating it by another problem (or a set of problems) for which a solution method is known. For example, if the original problem requires one to solve a set of nonlinear differential equations, engineers often substitute the original problem with a set of linear differential equations that approximate the nonlinear equations.
2. *Performance-driven problem re-encoding*: The given problem may be too expensive to solve by an available method, forcing one to reformulate into a similar problem that is easier to solve. For example, if a given set of linear differential equation is too large to solve with available computational resources, one may choose to aggregate to produce a less detailed but smaller model.
3. *Supporting cognitive insight*: One may choose to reformulate in order to gain cognitive insight into the problem or solution space. Mapping from Cartesian coordinates to

polar coordinates, mapping a time domain to a frequency domain, reformulating the equations of an electric circuit in terms of the voltage and current into ones in terms of power are all examples of reformulation that are sometimes performed to make certain aspects of the system behavior easier to detect and/or analyze.

Note that a given reformulation can serve multiple purposes. A reformulation method applied to reduce the cost of solution may also improve one's understanding of the overall behavior of a complex problem.

For the purposes of this paper, we have elected to focus on a specific class of problems, in an effort to develop a framework for characterizing reformulation methods with sufficient detail to be useful to practitioners and researchers alike. In particular, we examine reformulation techniques that can be exploited to reason about physical systems. The long-term goal of our research is to develop an automatic task-driven capability that selects and applies appropriate reformulation techniques in the course of problem solving. An essential tool towards this ultimate goal is a practical framework for characterizing and evaluating the merits and relevance of various reformulation techniques. This paper proposes such a framework with respect to the restricted class of problems we described above. The motivation for developing such a framework came from the observation that much of the previous work on reformulation (including abstraction and approximation) was not sufficient for our purpose. It was either too specific to account for the large variety of reformulation methods commonly used in reasoning about physical systems, or too general for characterizing them in sufficient details to allow informed selection among them. The framework we present provides a significant step towards our long-term goal by defining general criteria for understanding the properties and assessing the merits of various reformulation procedures. More specifically, the framework provides the means to identify the conditions under which a reformulation is applicable, the cost associated with performing the reformulation, and the effects of the reformulation both with respect to the problem encoding and with respect to the accuracy and cost of reasoning.

The paper is organized as follows. Section 2 discusses the motivations behind our endeavor and its scope. Section 3 describes the processing stages of reasoning about physical systems to define the context for the types of reformulation we are interested in. The framework for characterizing reformulation techniques is introduced in Section 4, where we provide a detailed description of the framework itself and present a set of evaluators for assessing the effects of reformulation. Section 5 uses the framework to actually analyze a number of known reformulation techniques. Section 6 discusses related work. Finally, Section 7 summarizes our contributions and outline directions for future research.

2. Motivation and scope

In this section, we first state the motivations of this paper, then we justify our reasons for using the term *reformulation* in an effort to clear up the ambiguity of previous terminology.

2.1. Motivation

Reformulation is ubiquitous in reasoning about complex systems. Analyzing the behavior of a complex electro-mechanical device in full detail, taking into account all its electrical, mechanical, and thermal aspects is too costly and unnecessary for most purposes. Engineers and scientists are accustomed to reformulating repeatedly the mathematical model of a complex physical system in order to reduce the complexity of analysis, to facilitate the explanation of the problem or a solution to others, or often, even to be able to analyze the system at all. There are numerous known techniques for reformulation that are used by engineers and scientists. AI literature also abounds with such techniques. A library of reformulation techniques and an ability to select from it the most appropriate technique for a given problem would greatly enhance the power of a program for reasoning about physical systems.

To compile such a library or to enable informed selection from it, we need a systematic way to compare and evaluate reformulation techniques, which the field currently lacks. This lack may be due to the fact that many commonly used reformulation techniques are informally executed tools in engineers' 'bag of tricks'. A more significant problem in establishing a systematic classification of reformulation techniques is their inherent diversity; they apply to different forms of models, at different stages of problem solving, and to achieve different goals. We need, as a first step, to articulate a vocabulary of attributes to describe various key aspects of reformulation techniques. This vocabulary needs to be informative enough to enable selection of appropriate techniques for different goals at different stages of problem solving.

The understanding and classification of reformulations have been relatively well-addressed in the literature. While some researchers have focused on specific domains such as theorem proving [25], planning [13,34], or model-based reasoning [33,39], others have tried to construct general theories in order to formalize the general properties of reformulations [4,10,22]. We discuss these contributions in Section 6. While these previous works mentioned above explored new grounds and established key properties of certain types of reformulation techniques, we have found them to be too general and insufficient to characterize many of the techniques common in reasoning about physical systems. They are typically concerned with only one aspect of the reformulation: its effects on the solution, the models, the proof trees, the theorems, or the computational cost. They also do not apply to models in the form of mathematical equations, which limits their usefulness in reasoning about physical systems.

In surveying various abstraction and approximation techniques in kinematics theory of rigid solid objects (KRSO), Davis also observes that abstraction and approximation in KRSO do not fit any one elegant metatheoretic structures such as those proposed by Giunchiglia and Walsh, Nayak and Levy, or Weld. He concludes by stating that he does not see any "overarching metalogical structure that sheds useful light on the relation between these approximation techniques" and conjectures that "none will be found in many domains of physical reasoning" [5]. While we agree with Davis's conclusion that there is no clean meta-theory of reformulation in reasoning about physical systems, we do believe that it is possible to provide a framework that enables a systematic description of many aspects

of reformulation techniques that are critical in their selection. We also believe that such a framework is a pre-requisite for compiling a library of reformulation techniques.

This paper is our attempt to formulate such a framework for systematic description of reformulation techniques to enable their comparison and evaluation with respect to given problems. Such a framework will not take the shape of an overarching metatheory of reformulation. Instead, the goal is to characterize reformulation techniques in such way that one can select a technique that is appropriate for a given problem. We do not offer a formal theory of reformulation with definitions and theorems. Instead, we discuss what aspects of reformulation techniques and the context in which they are used must be made explicit if we were to be able to compare them and make an intelligent choice among them in the course of reasoning about physical systems. The goal is more modest than a construction of a "grand-unified" theory of reformulation, but hopefully more realistic and, ultimately, more useful. This paper proposes a structure for presenting a reformulation technique with a view towards compilation of a library of a broad range of reformulation techniques and enabling intelligent selection. Furthermore, we identify the dimensions of a problem that are relevant for the selection of the reformulation and provide the terminology to describe the effects of its application to the problem. In presenting this framework, we also hope to shed new light on the confusion surrounding reformulation and to serve a pedagogical purpose by explicating a number of reformulation techniques from a unified perspective.

2.2. Scope

This paper focuses on reformulation techniques in the context of reasoning about the behavior of physical systems. We restrict ourselves to problems where the behavior of the physical system in question is expressible as a set of lumped-parameter continuous models, containing algebraic or differential equations, though reasoning need not be limited to direct manipulation of equations. We require that the task be motivated by a specific query, thus constraining the necessary computational machinery to solve the task. Finally, we consider only reformulations of a specific problem instance as opposed to the construction of gross properties of a population of instances.[1]

In Section 1, we have informally defined reformulation as "transformation from one encoding of a problem to another". Review of the literature on reformulation reveals that in addition to transformation, two other general approaches to reformulation exist, namely *selection* and *formulating anew*. When an existing encoding proves lacking in some respect, instead of transforming the original, one could formulate a new encoding from scratch or select another formulation from a set of existing alternatives. In general, since there must be a way to use the knowledge of an encoding's shortcomings to guide the process of creating or selecting an alternative, these three approaches do not represent totally orthogonal dimensions in reformulation. Rather, they represent different points in the whole spectrum of formulation techniques and are likely to share many technical issues.

[1] The entire field of statistics is devoted to the problem of producing and using higher-level descriptions of a large population of instances. Though Amador and Weld discuss this kind of an aggregation method as a type of reformulation [2], we will not consider such methods in this paper.

The apparent differences among transformation, selection, and new formulation tend to be more a matter of focus of the work. Reformulation through selection presumes that there exists a set of alternative formulations to begin with. The works that focus on selection generally do not address the problem of *generating* the alternatives [1,36]. Techniques that formulate anew focus on the problem of *constructing a model* in the first place. While transformation presumes that there exists at least one formulation to start with, the process of altering an existing encoding can require much of the same type of knowledge needed for constructing a new one. Both selection and transformation could be part of the process of a new formulation: In some approaches, the construction of a new model involves generation of possible candidates models followed by the selection among them [6]. In others, the generation of an initial model is followed by the transformation if the initial model is found to be optimizable with respect to some criteria [20,21].

While we recognize that selection and formulating anew are viable approaches to reformulation, we focus, in this paper, on reformulation as a transformation. In practice, reformulation by transformation is an important, if not the most important, category of reformulation techniques, as most of the abstraction and approximation techniques commonly used in engineering fall in this category. Reformulation via transformation is frequently the only alternative in practice. In the real world, one often does not have the luxury of multiple existing models or the necessary resources to formulate a new model from scratch.

2.3. On the terminology

In the previous attempts to capture the nature of reformulation and its effects, the words 'abstraction' and 'approximation' have commonly been used to refer to various instances of reformulation. Based on our observation that the words abstraction and approximation are not appropriate to designate all reformulation procedures that are of interest, we choose to use the term 'reformulation' as a general term that subsumes abstraction and approximation.

Abstraction and approximation. It is rather difficult to clearly distinguish between abstraction and approximation, and the notions of generalization and simplification that they imply. Both 'abstraction' and 'approximation' imply some reduction in the complexity of the representation and of the reasoning process. In other words, they are supposed to 'simplify' the problem in some way. 'Abstraction' is often used to indicate some structural modifications of a representation, whereas approximation refers to differences in numerical values or other mathematical quantities [5]. Abstraction is thought to 'simplify' a description by dropping certain information, retaining only general characteristics, at the expense of introducing some ambiguity. 'Approximation', in contrast, implies a replacement of a detailed, precise and accurate model[2] with one that is less accurate but is easier to compute with. In both cases, the difficulty in producing a precise yet general definition of the term

[2] In logic, model is used to refer to an interpretation of a theory. This is not the meaning we intend in this document. By model we mean a collection of statements that are a (possibly non-mathematical) representation of a given physical situation useful for automated analysis.

lies in defining the meaning of 'simpler'. What is simpler almost always depends on what aspect one is focusing on. For example, consider the case of a periodic square function. It is sometimes used to approximate a sine function of the same periodicity, since the former ranges over only two values instead of the continuum of values for the latter. However, the square wave is in fact much richer and more complex in terms of harmonics than the sine wave and, in this sense, the sine is an approximation of the square function.

Simplification. Most work on abstraction and approximation techniques does not make explicit the precise measure of simplicity used. Even in those cases where the notion of simplicity is precisely defined, the definitions are too narrow to account for the various ways a reformulated problem differs from the original one. Rickel and Porter [28,29] define simplicity in terms of the number of variables involved in the mathematical equations of a model: the fewer variables a model has, the simpler it is. Nayak's definition relies on two things; subset relations among the causal relations implied by model fragments and the approximation relations among model fragments [20]. Neither of these definitions allows one, for example, to compare two mathematical models involving the same set of variables, one being a nonlinear model and the other a linearized approximation of the first.

Generalization. Another important change in a model that is brought about by abstraction or approximation, though it is often related to the notion of simplicity, is generality, which is the range of phenomena captured by a model. Comparing the classical dynamics and relativistic dynamics will help illustrate the subtlety of the notion of simplicity and its relation to that of generality. The classical dynamics is a good approximation of the relativistic dynamics but only when the speeds of objects do not approach the speed of light (i.e., when $v/c \ll 1$). In other words, the classical dynamics is less general than the other.

These difficulties in defining precisely what is abstraction or approximation have led us to use instead the term 'reformulation'. Rather than classifying examples of reformulation as abstractions, generalizations, or approximations, we need to understand fully and articulate clearly what happens when a model is transformed by making explicit the effects of reformulation on specific aspects such as the representation, the computational efficiency, cognitive transparency and the result.

3. Reasoning about physical systems

Before discussing various reformulation procedures, we must set the stage by describing the problem-solving context in which we wish to evaluate such procedures. As we stated earlier, we concentrate on the types of physical systems that can be described with lumped-parameter hybrid models containing algebraic and ordinary differential equations. In this section, we describe the various processes that are executed in reasoning about physical systems, and we illustrate these processes in terms of several examples drawn from the literature. The purpose of this section is to explicate the context in which we wish to employ and evaluate reformulation techniques. Since one of the theses of this article is that such an evaluation cannot be made in a vacuum, it is imperative that we make the context as explicit as possible.

3.1. Stages of modeling physical systems

Starting with a description of the task of interest, we perceive the entire endeavor of reasoning about physical systems as a progression through the three *processing stages* illustrated in Fig. 1, namely: the model, the equation, and the solution processing stages.

Task description. Reasoning about physical systems begins with a *task description*, which describes the problem and establishes the problem solving goal. It consists of the following four elements: the domain theory, the scenario, the query, and the modeling assumptions.

The *domain theory* is a corpus of knowledge about the physical world. It contains heterogeneous, possibly redundant, descriptions of the physical structure and phenomena of interest, including logical statements, symbolic equations, and numerical parameters. In the work on compositional modeling [6], the domain theory is represented in the form of a library of model fragments.

In general, any profitable computational effort spent on reasoning about a domain theory must be motivated by a specific task, where a task can vary from prediction, to explanation, to verification, under a wide spectrum of hypothetical situations and conditions. We define a task by a combination of a *scenario*, a *query*, a *domain theory*, and a set of *modeling assumptions*.

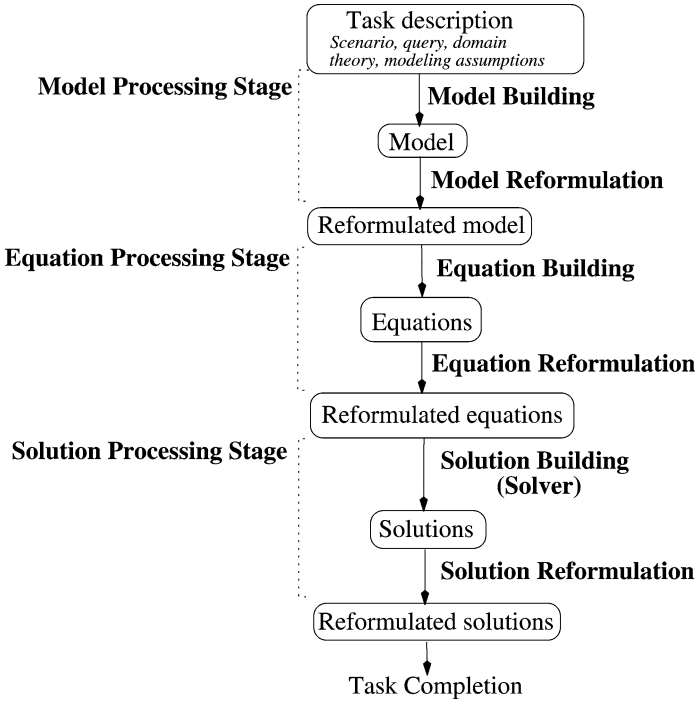

Fig. 1. Reasoning about physical systems. Stages of reasoning and their corresponding processes.

The *scenario* is a description of a particular problem instance (e.g., a set of system components and their physical structure, and the initial conditions of the system). The *query* is an explicit specification of the user's interests in terms of the variables, and their aspects of interest (e.g., the quantitative or qualitative values, direction of change at specific time points, temporal evolution).

The *modeling assumptions* include assumptions that the problem solver makes in order to broadly delimit the scope of the answer to the query (e.g., the temporal and physical extent of its coverage, granularity). See [15] for a more detailed discussion of modeling assumptions.

Model processing. Given a task description, the *model building* process assembles the relevant aspects of the domain theory to produce a model, which is an instantiation of a subset of the domain theory that is both consistent and sufficient to address the query. A model at this point often consists of knowledge of the physical structure (components and their topology, for example) as well as knowledge of the relevant physical phenomena (including the conditions under which they are active), in contrast to the purely mathematical model of the following stage. Examples of model building process include compositional modeling as in [6,15,20] and the modeling algorithm of `TRIPEL` [28].

This process can be followed by a *model reformulation* process. Model reformulation may involve structural consolidation [40], simplification [21] or expansion of a model through aggregation, replacement, deletion or addition of a description of components or phenomena. Specific examples of reformulation carried out at this stage are presented in Section 5.1.

Equation processing. *Equation building* produces an equation model either directly from a task description or by extracting mathematical equations from a model describing the behavior of the system. For example, the Qualitative Physics Compiler [7] converts a model expressed in QP Theory [8] into a set of qualitative differential equations. Once equations are obtained, an *equation reformulation* process may be carried out. An equation reformulation is often motivated by a desire to transform the present equation model into a form that is easier to compute with or that is amenable to a particular problem-solving engine.

We distinguish between non-equational models and equational ones because there is a large class of mathematical techniques that operate solely on equations. In fact, most well-known simplification techniques operate on equations. They include such techniques as dropping insignificant terms, linearizing non-linear equations, aggregating nearly decomposable systems, and reformulating ordinary differential equations as qualitative differential equations or qualitative differential equations as causal orderings. Some of these techniques are discussed in detail in Section 5.2.

Solutions processing. Equations are given to a solver to produce one or more solutions. Some examples of a solver are `QSIM` [14] and Matlab®. Once the equations are solved, a *solution reformulation* process may subsequently be performed in order to make the solution easier to comprehend or to facilitate further reasoning. Examples of such reformulations include categorizing solutions, summarizing a single solution, summarizing a set of solutions, and explanation generation [11].

As the problem becomes larger and more complex, there is an increasing need to facilitate understanding of a solution. In Section 5.3, we discuss two such techniques.

An interesting use of solution reformulation for the purpose of speeding up the overall problem-solving process is demonstrated by Clancy and Kuipers [3]. Clancy and Kuipers interleave `QSIM` simulation with the aggregation of partial solutions corresponding to chatter in a qualitative simulation. In so doing, they significantly improve the overall performance of `QSIM` on some problems.

We note that reasoning need not necessarily proceed through every stage defined above. It is common for a problem solver to omit a stage or to loop over stages. For instance, one may go from a task description directly to the Equation Processing stage, or from the Model Processing stage directly to the Solution Processing stage. QPE [9] is such an example because it produces a solution directly from a model without explicitly generating a set of equations. A problem solver may also loop through one or more of the stages. Ling and Steinberg in [16] describe a three-step procedure for automated modeling that encompasses model generation, simulation, and validation while looping over each of these steps (inner loop) and over the whole process (outer loop). In the example of solution reformulation mentioned above and discussed in Section 5.3.2, Clancy and Kuipers [3] loop over the solution building and solution reformulation processes. Another example of such loops can be found in the work on DME [17] for modeling device behavior. DME cycles through the stages of Equation Processing and Solution Processing during simulation as the operating conditions change and the model used for simulation must be updated.

Earlier, we have informally defined reformulation to be a transformation from one encoding of a problem to another. Having described the stages of modeling physical systems in which to study reformulation, we are now ready to define reformulation more narrowly in this context. In the process of reasoning about physical systems as we have just outlined, what we regard as reformulation is a transformation of an encoding at one stage to another encoding at the same stage. Thus, such reformulation does not change the type of formalism of the encoding but does change the content in such a way that the change is not a mere consequence of making explicit what is implicit in the original.[3] Fig. 1 reflects this narrow definition of reformulation as each processing stage consists of a building step followed by a reformulation step. We believe that restricting the context and narrowing our field enables us to lay out a more concrete and practical framework for informed comparison of reformulation techniques which are relevant in reasoning about physical systems than would be possible otherwise if we adopt a more general definition of reformulation. In the remainder of this paper, we will refer to mechanisms that carry out a building step in any processing stage of Fig. 1 as "solution engines" to distinguish them from reformulation engines. We

[3] If we regard each encoding as a logical theory or as a model in the model theoretic sense, then the two encodings would be two distinct and logically incompatible theories or models. It is open to discussion whether such "models" of physical systems as those we studied in this paper should be interpreted as theories or models in the model theoretic sense. However, such a distinction is not relevant to those of us actually trying to model a physical system.

use the term 'engine' broadly to include anything from an algorithm, to a special-purpose simulation program, to a general-purpose solution package such as Mathematica®.

3.2. Characterizing reformulation

There are three general aspects in which we will characterize a reformulation technique. First, we must specify the types of problems to which the technique is applicable. Second, we must characterize the impact of the reformulation technique in terms of what aspects of a problem it changes and how. Third, we must characterize the reformulation as a computational process. The first and second aspects are rather definitional as they describe the applicability of the procedure and its raison d'être. The third aspect is more descriptive in the sense that it represents a set of ways to evaluate the procedure as a computational tool.

We now briefly introduce three examples of reformulation techniques that are discussed in more detail in Section 5. The three examples are model simplification by Nayak and Joskowicz, aggregation of nearly decomposable systems by Simon and Ando, and behavior abstraction for explanation by Mallory. We will then highlight the key features of these techniques in order to motivate the framework presented in Section 4 for characterizing reformulation.

- Nayak and Joskowicz propose a model reformulation technique that simplifies a compositional model of a device [21]. Their reformulation technique simplifies a model by replacing the model fragments in a given model by alternative simpler fragments whenever possible. The result is a model that has the same explanatory power as the original but is provably most parsimonious.
- Simon and Ando describe a technique for aggregating a system of linear ordinary differential equations to produce a smaller system [32]. The technique only applies to a system that consists of weakly interacting components such that interactions within each component are much stronger than those among components. Their technique produces a smaller system of equations that describes the mid- to long-term behavior of the original system.
- Mallory et al. propose to summarize the results of the qualitative simulation of a physical system in order to help users recognize 'basic patterns of behavior' [19]. Their reformulation procedure summarizes the behavior of the system by generating a behavior graph that retains only those aspects of the behavior tree relevant to the query. The result is a smaller, more abstract tree that retains the essential features that are relevant to the query.

While all the three techniques aim to simplify a formulation, they differ from each other with respect to important aspects, such as the contexts to which they are applicable and the results they produce. Below, we identify the dimensions for characterizing them that are critical for the practical use of these techniques.

3.2.1. What problem does the reformulation apply to?

One of the fundamental assumptions of our analysis is that reformulation must be motivated by a task, and furthermore, that the task itself must be driven by a specific query.

Each of the three example techniques is motivated by a question about a quantity (or a set of quantities) associated with a system. Each however deals with a different formulation of such a query since each technique applies at different stages of reasoning about physical systems of Fig. 1. Nayak and Joskowicz's technique applies at the Model Processing Stage to reformulate a conceptual model. Simon and Ando's technique applies at the Equation Processing Stage to produce a different set of equations. Mallory's technique applies at the Solution Processing Stage and produces a more concise description of a behavior. Thus, our framework requires that the definition of a problem include a specification of its *formulation* as well as the *query*.

3.2.2. What does the reformulation do?

The objective of reformulation is to change some aspect of the formulation of the original problem. Thus, in order to characterize a reformulation technique, we must specify precisely the features of the problem that are affected and the ways in which they are changed.

While the three examples above all purport to affect the simplicity of a formulation, the precise definition of simplicity is different as the problem formulation is different in each case. In Nayak and Joskowicz's example, the notion of simplicity is based on the causal-approximation relation. A model A is simpler than a model B if and only if the set of causal ordering relations implied by model A is a subset of those implied by model B. In Simon and Ando's example, simplicity is defined in terms of the number of equations and variables. In Mallory's case, simplicity can be measured by the number of links and nodes, and the labels of those nodes in a behavior tree. Thus, our framework requires the definition of a set of evaluators to specify the features of a problem (i.e., formulation and query) that are modified by the reformulation technique.

Furthermore, in order to fully specify the effects of reformulation on problems, we must also specify how the affected features are to be compared. *Comparators* are specifications of such means of comparison. Quite often, especially when the affected features are numerical values or sets, comparators are straightforward numerical $(<, =, >)$ or set $(\supset, =, \subset)$ comparators. In the three examples above, the features affected by reformulation can be compared in this manner. In Nayak and Joskowicz's case, the comparator is a subset relation over the sets of causal relations implied by models. In Simon and Ando's example, the numerical comparator, $<$, is sufficient to compare the number of variables and equations. In Mallory's example, subset relations over the links, nodes, and labels are used to compare the features. In other words, in all three cases, the notion of simplicity that each reformulation technique aims to achieve can be precisely defined by specifying *evaluators* to measure some features of the problem and *comparators* to compare these measures.

When a reformulation technique is defined as a mathematical *function* between two sets, the reformulation process can be characterized by the nature of the function such as injective, surjective, one-to-one, etc., and two or more such processes compared for strength or weakness, see [10].

3.2.3. Characteristics of the reformulation technique itself

Aside from describing a reformulation technique in terms of what it does, one needs to be able to characterize it as a computational process using standard criteria. Computational

complexity is one such criterion that is almost always provided by the authors of such techniques. In all our three examples, the procedures are tractable with respect to the class of problems that satisfy the conditions specified by the procedures. In other cases, evaluation of the reformulation technique in terms of the computational complexity is given in slightly different forms, such as best/worst/average case with respect to a simulated/natural population of a certain size.

Although the computational complexity is often provided when reformulation techniques are proposed, there can be a number of alternative ways to characterize these techniques. From the perspective of a person looking for a suitable tool for one's problem at hand, the availability of the code for applying the technique is often an issue in practical settings. If the code for carrying out reformulation is available in several commercial packages, their price or their ease of use could be points of evaluation. As there can be as many angles from which to evaluate reformulation techniques as there are people who wish to use them to solve problems, this can only be a non-exhaustive set of evaluators.

In summary. For practical purposes a reformulation technique must be described along three aspects:

1. The class of problems to which it applies.
2. What it does, in terms of what features of problems it targets and how it changes them.
3. Its characteristics and cost as a computational tool.

Our framework provides the means to define a problem along with several types of evaluators and comparators to characterize those aspects of the reformulation process. The following section formally introduces the components of our framework.

4. Framework for characterizing reformulation

In this section we introduce a framework and a terminology for characterizing and evaluating reformulation techniques. Section 4.1 introduces the components of the framework and reduces the selection of a sequence of reformulations to a planning task. Finally, Section 4.2 introduces the attributes necessary to characterize a reformulation technique, so that selection can be performed.

4.1. Components of the framework

Reformulation replaces an original problem by a new one, as shown in Fig. 2. We distinguish two primary components, the *problem* and the *reformulation*, and two composite components, the *process* and the *strategy*, obtained from composing the former two.

4.1.1. Problem

We define a problem P_i as a three-tuple:

$$P_i = \langle \mathtt{Query_i}, \mathtt{Form_i}, \mathtt{Assmpt_i} \rangle.$$

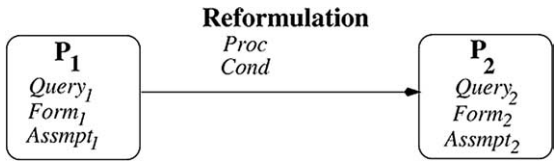

Fig. 2. Reformulation.

- $\texttt{Query}_\texttt{i}$ specifies the question of the user, the one we are trying to answer by manipulating the formulation. The query motivates the whole endeavor.
- $\texttt{Form}_\texttt{i}$ denotes the formulation, i.e., the conceptualization of the domain of, for instance the studied physical artifact. This amounts to a list of objects, and their interrelationships (e.g., functions and relations). Depending on the particular stage of Fig. 1 to which the reformulation method is applied, the particular formulation can be a conceptual model (Model Processing Stage), mathematical equations (Equations Processing Stage), or even solutions (Solution Processing Stage).
- Finally, $\texttt{Assmpt}_\texttt{i}$ designates the conditions under which the formulation is valid, e.g., the domain of applicability and the temporal granularity.

According to this terminology, the task description defined in Section 3.1 and Fig. 1 is a problem P_i where:

- $\texttt{Query}_\texttt{i}$ is the query in the task description,
- $\texttt{Form}_\texttt{i}$ is the domain theory and scenario, and
- $\texttt{Assmpt}_\texttt{i}$ is the set of modeling assumptions.

4.1.2. Reformulation technique

Our ultimate goal is to be able to automatically select a suitable reformulation technique from among a collection of techniques. One step towards this goal is to articulate, for each reformulation, its applicability conditions and algorithmic steps.

A reformulation technique is applied to an original problem P_1 to produce the reformulated problem, P_2, as shown in Fig. 2. By examining a wide collection of reformulation techniques, we have found that reformulation techniques typically modify the problem encoding only: the query and assumptions often remain unchanged before and after reformulation. Counterexamples do exist such as the ones discussed in Sections 5.2.1 and 5.2.3 where assumptions are reformulated. We describe the reformulation technique as a tuple:

$$R = \langle \texttt{Proc}, \texttt{Cond} \rangle,$$

where:

- $\texttt{Proc}$ denotes an effectively computable procedure of which the input is P_1 and the output is P_2. For example, the approximation of the behavior of a nearly-decomposable system by aggregation, discussed in Section 5.2.1 is one instance of such a procedure.
- $\texttt{Cond}$ denotes the applicability conditions. For example, the aggregation of nearly-decomposable system of Section 5.2.1, requires that $\texttt{Form}_1$ be a self-contained set

of nearly-decomposable linear ordinary differential equations and that the system be stable.

It must be noted that Cond is a set of *necessary* conditions to be satisfied by P_1 for the reformulation to be technically applicable. However, whether or not R is an appropriate reformulation to perform depends on the goal of reformulation, which is discussed in Section 4.1.3. If, for example, the goal is to obtain more precise values for the variables in the mid-term behavior of the system than can be produced from the aggregated system, aggregation of the nearly-decomposable system would *not* be an appropriate choice of the reformulation method, though the procedure is technically applicable. In practice, sometimes it is not known whether $\texttt{Form}_1$ satisfies Cond because some parameter values are unknown or even the precise form of the equations is unknown. Even in such cases, one sometimes applies the reformulation method. The validity of the results thus produced will depend on the degree to which Cond is satisfied by $\texttt{Form}_1$. In such cases, Cond becomes the assumptions $\texttt{Assmpt}_2$ underlying $\texttt{Form}_2$. Such conditional reformulation can still be quite useful in providing insight into the shape of the problem space. The caricatural reasoning discussed in Section 5.2.3 demonstrates the usefulness of conditional reformulation.

In Fig. 3 the reformulation R is illustrated as a transition between nodes representing two problems P_1 and P_2.

As mentioned in the introduction, the decision to perform a reformulation could be motivated by the availability of a suitable solution engine and its performance for solving a problem.[4] For example, if we have a slow but general-purpose solver for Ordinary Differential Equations (ODEs) and a high-performance implementation of a numerical simulation algorithm that works only with systems of linear equations, we might reformulate an original nonlinear system of equations, which would have to be solved with the general-purpose solver, into a linear one that can be solved by the fast integrator.

A solution engine is applied to a problem to produce a result as an answer to the query. There could be multiple engines at one's disposal to solve the original or reformulated problems. Alternately, there could be none, when the problem is too difficult. Further, we may use the same means of solution before and after reformulation since the reformulation of a problem, for instance by decomposition, can result in an important improvement of performance for the same solution engine. Aggregation, discussed in Section 5.2.1, illustrates this situation.

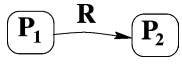

Fig. 3. Reformulation process.

[4] In this paper, we do not address reformulations that apply to the engine itself, as proposed in [10], because such reformulations do not seem to arise in the class of problems we address.

4.1.3. Process and strategy

A problem, a reformulation technique and the problem resulting from applying the reformulation technique to the problem compose *a reformulation process*. Fig. 3 illustrated such a process as a transition between two nodes.

A single reformulation process can be understood as one step *towards* providing an answer to the query. A sequence of reformulations, commencing with the initial problem and possibly including one or more engines, constitutes one *strategy* for addressing a task. The execution of a strategy constitutes problem solving.

Thus, we define a strategy S_i to be a sequence of reformulations, $\langle R_a, \ldots, R_x \rangle$ that is applied to an original problem P_1. The path $\langle P_1, R_a, P_2, \ldots, R_x, P_i \rangle$ in Fig. 4 is an example of the execution of such a strategy. Any subsequence, S_k, of S_i, starting at P_1 and stopping at any intermediary problem P_k, between P_1 and P_i, is also a strategy, and is called a *sub-strategy* of S_i.

We perceive reasoning about physical systems to proceed according to processes identified in Fig. 1. According to this figure, the content of the initial input, i.e., the task description, is gradually modified by a combination of any number of processes culminating in an answer to the query. Given our definition of reformulation, any of these processes is a reformulation. The stage of processing distinguishes whether the reformulation is applied to a collection of model fragments, an equational model, or to one or more solutions. Reasoning about physical systems is thus a successive application of reformulation procedures that transforms an initial problem encoding.

Problem solving as plan execution. Problem solving involves the successive application of reformulation procedures to an initial problem encoding to produce a final problem encoding. Clearly there could be multiple sequences of reformulations that could be applied to address the problem-solving task, as illustrated in Fig. 5. Identifying such sequences of reformulation procedures can be viewed as a planning problem in which the *states* are problem encodings, the *transitions* (or actions) are reformulations, and the *plans* are strategies.

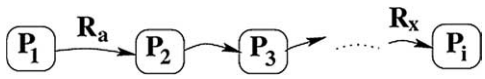

Fig. 4. Strategy.

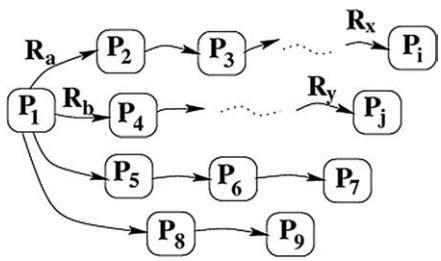

Fig. 5. Problem solving is plan execution.

Hence, problem solving becomes an execution of the selected plan. In Fig. 5 we illustrate a tree of four alternative strategies.

In practice, as for planning, resources may be limited, and one may want to associate a utility or objective function to the problem-solving task. We expect the user to provide the *goal* of the problem-solving task in terms of a *goal test* and of an *objective function* that specifies the importance of some desired features of the problem and the resources available. In this context, selecting an optimal plan or strategy becomes a multi-criteria optimization problem.

4.2. Evaluating and comparing components

Section 4.1 defines the primary components (i.e., problem and reformulation technique) and the composite components (i.e., process and strategy) of our framework. To articulate the *goal* driving the above-mentioned planning process, we identify the features of these components relevant for selecting reformulation techniques and characterizing their effects. These features are divided into sets, relative to the components of our framework. The sets of features are meant to be incrementally augmented and refined as one explores, defines, and proposes new reformulation techniques. There are two main categories of sets:

1. *Evaluators*, denoted `Evals`, evaluate some features of one component or a combination of components of the framework.
2. *Comparators*, denoted `Compars`, assess the change due to reformulation by comparing the values of the evaluators.

The values returned by the evaluators and comparators are not necessarily quantitative; they could be qualitative or logical.

4.2.1. Evaluators

In order to characterize a reformulation technique, we must first define measures or evaluators that capture relevant characteristics of the problem, the reformulation technique and their inter-relationships.

Given a reformulation $R = \langle \texttt{Proc}, \texttt{Cond} \rangle$, and original problem P_o, and the resulting reformulated problem P_r, we distinguish four such sets of evaluators as illustrated in Fig. 6.

We distinguish four sets of such evaluators, $\texttt{Evals}_{\texttt{prob}}$, $\texttt{Evals}_{\texttt{ref}}$, $\texttt{Evals}_{\texttt{p+r}}$, and $\texttt{Evals}_{\texttt{p+r+p}}$ corresponding to evaluators that assess, respectively, aspects of a problem, a reformulation technique, and a combination of an original problem and a reformulation technique and finally a reformulation process. We introduce also a fifth set of evaluators

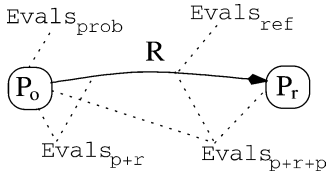

Fig. 6. Evaluators and the components to which they apply.

$\texttt{Evals}_{\texttt{strat}}$ to be computed from the initial four. Below we explain and provide examples for each set of evaluators.

1. $\texttt{Evals}_{\texttt{prob}}(P_i)$ is a set of measures to evaluate some characteristic of a problem P_i. P_i could be the original problem P_o or the reformulated one P_r. In the most general case, the elements in this sets can be defined with respect to any of the three elements of the problem, i.e., the formulation, query and assumptions. An example of an evaluator that assesses the problem as a whole (i.e., formulation, query and assumptions) is the complexity class of the problem (e.g., NP, PSPACE, and EXPSPACE). As we mentioned in Section 4.1.2, we have found that in practice the query and assumptions often remain unchanged before and after reformulation, and that most evaluators are functions applied exclusively to the formulation of P_o.
   The evaluators in the set $\texttt{Evals}_{\texttt{prob}}$ can provide an assessment of some quantitative aspects of the problem (e.g., size) or of its logical properties (e.g., compactness, completeness, and decidability). They can also address qualitative aspects of the problem (e.g., complexity class, or how 'close' the formulation is from answering the query).
   For a system of equations, an example of a quantitative evaluator is the number of equations or their degree, or the number of variables; an example of a qualitative evaluator is adherence of the equations to some canonical form. Other evaluators of the formulation that appear in the literature include scope [38] (which is the range of phenomena that it can describe), expressiveness, syntactic form, simplicity, generality, relevance, absence of irrelevant information, and language restriction to familiar terms [37].
   It is important to define an evaluator in sufficient detail. In the case of simplicity, for example, we must define the specifics of how it is measured (e.g., the number of variables/equations in a equation set, or the number of components in a model). Some of the evaluators in $\texttt{Evals}_{\texttt{prob}}$ are dedicated to assessing the quality of the answer to the query as it is made explicit in $\texttt{Form}_{\texttt{i}}$. The result is often a numerical, quantitative value but it can also be a qualitative (e.g., an interval) or a logical one (e.g., truth or negation of a sentence). It can also be a description of a behavior over time, or sets of partials or global solutions. Examples of such evaluators are the soundness of the answer and its precision. These are typically the evaluators to use in the test that determines whether the goal test of the planning process of Section 4.1.3 is achieved.
2. $\texttt{Evals}_{\texttt{ref}}(R)$ is a set of measures to evaluate some characteristic of the reformulation technique R_t. Examples include but are not limited to:
   (a) the size of the code that implements the procedure,
   (b) the programming language it is written in,
   (c) the price of the commercial software,
   (d) its hardware and software requirements and interfacing capabilities, and
   (e) the type and value of human expertise required for exploiting it.
3. $\texttt{Evals}_{\texttt{p+r}}(R, P_o)$ includes functions that assess the behavior of the reformulation technique *relative* to a given problem. A typical such evaluator is the computational complexity of the procedure $\texttt{Proc}$ when applied to the problem, and that of verifying the conditions of the reformulation, $\texttt{Cond}$.

4. $\texttt{Evals}_{\texttt{p+r+p}}(R, P_o, P_r)$ is a set of measures that characterizes the reformulation process, typically, in terms of the properties of a function (e.g., partial/total, injective, surjective, bijective, or invertible).
   For example, Struss requires reformulation procedures, which he calls representational transformations, to be surjective and not injective mappings [33]. Giunchiglia and Walsh study procedures that are surjective total functions between two formal systems [10]. Other examples include the following: homomorphism, isomorphism, theorem increasing, decreasing, or constant [10], deducibility or negation preserving [10], upward/downward solution [34], upward or downward-failure [39], ordered monotonicity [13], and safety [4].
5. In addition to these sets of evaluators, we introduce $\texttt{Evals}_{\texttt{strat}}(S_i)$, a set of measures for assessing a problem-solving strategy S_i. An element in this set is obtained by considering some combination of the values along S_i of an element in the above-introduced evaluators (i.e., $\texttt{Evals}_{\texttt{prob}}$, $\texttt{Evals}_{\texttt{ref}}$, $\texttt{Evals}_{\texttt{p+r}}$, and $\texttt{Evals}_{\texttt{p+r+p}}$). For instance, the financial cost of the strategy can be computed as the sum of the prices of the individual software packages; its computational complexity as the maximum of their respective complexity; and, when the individual processes are functions, the strategy can be characterized in term of the their composition. Elements of this set is typically used in the objective function of the planning process discussed in Section 4.1.3 to express preferences and manage resources.

4.2.2. Comparators

Comparators characterize the changes in the problem as the result of reformulation. In order to assess the effect of one or more successive reformulations within or across alternative strategies, we define two sets of comparators, $\texttt{Compars}_{\texttt{prob}}$ and $\texttt{Compars}_{\texttt{strat}}$, which compare features of problems and strategies respectively. In order to capture some notion of change or evolution, the elements of $\texttt{Compars}_{\texttt{prob}}$ and $\texttt{Compars}_{\texttt{strat}}$ generally reflect differences or ratios computed from the values returned by the evaluators $\texttt{Evals}_{\texttt{prob}}$ and $\texttt{Evals}_{\texttt{strat}}$. Below we discuss these two sets and provide illustrating examples.

1. $\texttt{Compars}_{\texttt{prob}}(P_i, P_j)$ denotes a set of effects that capture a change in some feature of the problem as expressed in $\texttt{Evals}_{\texttt{prob}}(P_i)$ and $\texttt{Evals}_{\texttt{prob}}(P_j)$.
   As for $\texttt{Evals}_{\texttt{prob}}$, some elements of $\texttt{Compars}_{\texttt{prob}}(P_i, P_j)$ assess the change that one or more reformulations produce on some measure of the answer. Examples of such comparators are a 10% loss in the accuracy of the answer, whether the answer in P_j

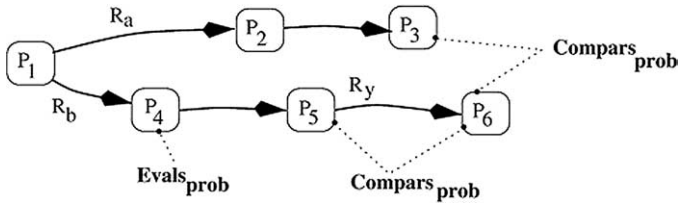

Fig. 7. Evaluating and comparing problems.

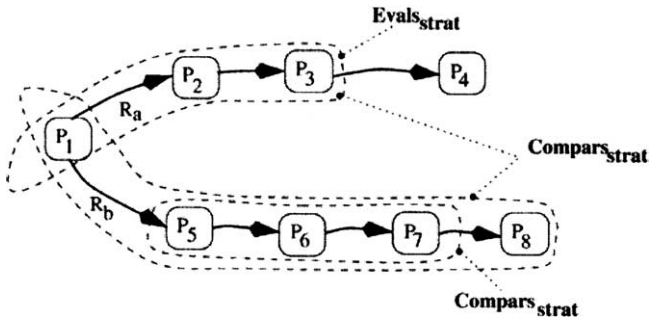

Fig. 8. Evaluating and comparing strategies.

is an overestimate (or an underestimate) of that in P_i, or whether or not the difference between them can be bounded within a threshold.

When P_i and P_j are two problems situated along the same strategy S_k, $\texttt{Compars}_{\texttt{prob}}(P_i, P_j)$ captures the *effect* of applying a reformulation (or, a sequence of reformulations) to P_i.

When the reformulations are defined are functions, this outcome can sometimes be predicted from considering the mathematical property of the reformulation itself (or, the composition of the consecutive reformulations). For instance, when the reformulation is an injective mapping, the size of P_j is bigger than or equal to that of P_i.

One possible effect of reformulation on the problem is to improve cognitive insight; this is common at the Solution Reformulation Stage of Fig. 1. If the original formulation is too complex for a user to understand, reformulation can produce a description better suited to human understanding.

2. $\texttt{Compars}_{\texttt{strat}}(S_i, S_j)$ is a set that denotes the effects to two strategies S_i and S_j. Examples of elements of this set are increase in cost, loss of time, and consumption of available resources.

   When S_i is a sub-strategy of S_j (or vice versa), $\texttt{Compars}_{\texttt{strat}}(S_i, S_j)$ indicates the effects of extending a strategy by one or more reformulation steps. When S_i and S_j are two distinct strategies applied to the same problem, $\texttt{Compars}_{\texttt{strat}}(S_i, S_j)$ assesses their relative merits.

   Giunchiglia and Walsh in [10] introduce the operators "weaker than", "stronger than" and "equivalent" ($\equiv$) to express an order relation determine an order two alternative mappings applied to the same problem.

   A reformulation is said to be cost-wise beneficial, when the cost of reformulating the problem and then solving the reformulated problem does not exceed the cost of solving the original problem. This can be expressed by an element in $\texttt{Compars}_{\texttt{strat}}$.

   Since one of the goals of reformulation is to improve overall problem-solving performance, the reformulation procedure itself should not significantly add to the computational cost. However, sometimes there is no solution engine applicable to the original problem, and consequently any amount of effort to reformulate to make it solvable is justifiable. A *cost-intensive* reformulation may also be justified when it is performed off-line to improve runtime performance of a system.

4.2.3. Summary

Table 1 summarizes the vocabulary and criteria for evaluating and comparing reformulation introduced above. It also situates these terms with respect to the aspects of reformulation that we discussed in Section 3.2.

4.2.4. Remarks

Observe that the evaluators for the problem, reformulation, and strategy are not necessarily independent. For example, the simplification of a set of equations often reduces the size of the formulation (measured by $\texttt{Evals}_{\texttt{prob}}$) and the cost of applying the reformulation procedure (measured by $\texttt{Evals}_{\texttt{p+r}}$ and consequently by $\texttt{Evals}_{\texttt{strat}}$), at the expense of also reducing the precision of the result (measured again by $\texttt{Evals}_{\texttt{prob}}$).

In practice, the framework's components seem often to be characterized and evaluated with respect to a set of non-orthogonal, and sometimes redundant, features. Weld in [38] suggests to assess a reformulation with respect to a set of independent features, which he calls model dimensions. He identifies scope, domain, resolution, and accuracy as four such orthogonal dimensions. Whether or not a set of canonical features exists remains an open problem. Our study, however, indicates that a rich and expressive, although possibly redundant, coverage of the features is desirable in practice.

5. Illustrative examples

In this section, we discuss a number of reformulation developed in the literature of reasoning about physical systems. We review two or more techniques for each of the three stages of reasoning shown in Fig. 1: the Model Processing, Equation Processing and Solution Processing stages. While all examples have been discussed in AI literature, they originate from and are useful in a variety of science and engineering fields. For example, Aggregation (Section 5.2.1) is drawn from econometrics literature. Linearization (Section 5.2.2) is a common technique widely used in mathematics and engineering. Caricatural reasoning (Section 5.2.3) is a formalization of the type of reasoning used in chemistry.

In discussing the reformulation techniques, we conform to the following pattern: First, we briefly summarize a technique in general terms. Then, using the vocabulary introduced in the preceding section, we make explicit the components of the technique in the following order:

1. The original and reformulated problems (`Query`, `Form`, and `Assmpts`).
2. The reformulation technique (`Proc` and `Cond`).
3. The evaluators and effects (`Evals` and `Compars`).

For each example, the first part of the discussion explicates the types of problems that the technique applies to. The second part details the reformulation procedure itself along with its conditions. Finally, the third part describes the effects of the procedure in terms of what aspects of the problems are changed and how. It also discusses the characteristics of the reformulation procedure as a computational process. Additionally, whenever we find it

Table 1
Summary of attributes and metrics

Evaluators		Comparators	
Notation	Examples	Notation	Examples
What does the reformulation do?			
	Problem		
$\texttt{Evals}_{\texttt{prob}}(P_i)$	Logical: • Compactness, completeness, decidability • *Result*: truth/negation of a sentence, soundness Quantitative: • Size, number of equations, degree of equations, number of variables, number of components in a model, • *Result*: behavior over time, numerical or interval value, precision Qualitative: • Complexity class, scope of model, expressiveness of model, syntactic form, simplicity, generality, relevance, restriction of language to familiar terms, adherence of equations to some canonical form, understandability • *Result*: sets of partial or global solutions, result: behavior over time, whether query is answered	$\texttt{Compars}_{\texttt{prob}}(P_i, P_j)$	• Set operators: $\supset, =, \subset$ • Numerical operators: $<, =, >$
	Reformulation process		
$\texttt{Evals}_{\texttt{p+r+p}}(R, P_o, P_r)$	• Necessary, sufficient approximation *Functions*: partial/total, injective, surjective, invertible, morphism, TI/TD/TC, upward/downward solution or failure property, ordered monotonicity, safety property, *etc*.		

(*continued on next page*)

Table 1 (*Continued*)

Evaluators		Comparators	
Notation	Examples	Notation	Examples
Characteristics of the reformulation itself			
	Reformulation technique		
$\texttt{Evals}_{\texttt{ref}}(R)$	• Size of code • Programming language • Price of commercial software • Interfacing capabilities (hardware/software) • Human expertise required		
	Reformulation technique and problem		
$\texttt{Evals}_{\texttt{p+r}}(R, P_o)$	• *Computational complexity*: Best, worst, average case; empirical		
	Strategy		
$\texttt{Evals}_{\texttt{strat}}(S_i)$	Combinations, to be defined, of the values along S_i of an element of the above evaluators. • Financial cost. • Maximal/total computation cost in time/space.	$\texttt{Compars}_{\texttt{strat}}(S_i, S_j)$ (Strategy)	• Weaker, stronger, equivalent abstraction • Cost intensive, Cost-efficient

R: Reformulation technique.
P_i: A problem. Original problem: P_o; reformulated problem P_r.
S_i: A strategy, a sequence of problems are reformulations $\langle P_1, R_a, P_2, \ldots, R_x, P_i \rangle$.

is instructive to do so, we discuss the reformulation technique as part of a larger problem-solving strategy and compare it with other possible strategies.

5.1. Reformulation at the model processing stage

The first step in reasoning about physical systems is to build a model. A model is the conceptual object we study and manipulate instead of studying the real physical device. There can be as many possible models of a given subject as there are reasons for studying it. While there is no one "correct" model, the usefulness of a model depends on the query one tries to answer by constructing and studying the model. For a model to be useful, it must contain enough information to answer the query with sufficient precision and accuracy while avoiding unnecessary detail.

The process of model construction sometimes amounts to model reformulation, where one starts with some existing accepted model and modifies it to suit the task at hand. It is said that in most engineering applications a significant part of the modeling effort is spent on iterative modification of the proposed model [26]. In AI, even works that focus on formulating new models can involve an explicit reformulation step where the model initially generated is modified to meet some optimality criteria.

In this section, we discuss two model-reformulation techniques. The first one, Model Simplification by Nayak and Joskowicz, is an example of a technique developed as part of a model-formulation method. In our discussion, we focus on the last step of the model formulation process, in which the model initially generated is simplified to meet the author's simplicity criterion. Since this reformulation technique is embedded in a modeling approach, called compositional modeling, we must introduce briefly the concept of compositional modeling in order to put our discussion in proper light.

Compositional modeling is an effective method for automatically formulating a behavior model represented by a system of ordinary differential equations for a physical system [6,15,21,29]. The basic idea in compositional modeling is to formulate a model of a given situation by putting together pieces of descriptions, called *model fragments*, of physical phenomena in the domain. Each model fragment describes one aspect of a component behavior or a physical process. A system formulates a model of a given situation by selecting applicable model fragments and composing them. The main advantages of compositional modeling are modularity and reusability of knowledge. It is, at least in principle, easier to write and reuse model fragments than complete models. Since there can be multiple ways to describe a given situation, each with varying degrees of details or with different modeling assumptions, it is possible to have several alternative model fragments describing the same physical phenomenon. Nayak and Joskowicz's reformulation technique involves replacing the model fragments in the existing model by simpler alternatives that describe the same phenomenon.

The second example we discuss is Critical Abstraction by Williams [40]. As in Nayak and Joskowicz's work, the goal of the reformulation in Critical Abstraction is to produce a model that is the simplest yet sufficient to provide a causal explanation of the behavior of interest. Critical Abstraction could very well be part of an overall model formulation method, even though Williams did not present it as such.

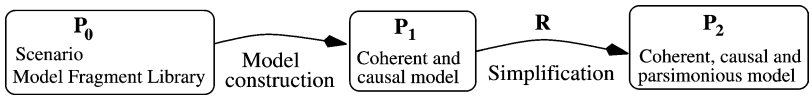

Fig. 9. Model simplification.

5.1.1. Model simplification

Nayak and Joskowicz propose a model reformulation technique that simplifies a compositional model of a device, while maintaining its ability to provide a causal explanation of the expected behavior of the device [21].

The primary objective of their work is to perform efficient compositional modeling for generating parsimonious causal explanations of the functioning of a device. They provide tools for model-fragment library indexing and selection to support the construction of device models. The reformulation procedure is applied to the model thus built in order to simplify it. The entire model formulation procedure is portrayed as a sequence of steps in Fig. 9. While the compositional modeling contributions of this paper are particularly interesting, it is the reformulation procedure following the initial model formulation that is of relevance to us. Thus, we focus on this simplification process in our discussion.

Given a device description, the expected behavior of a device, its structural and behavioral constraints, and a library of model fragments organized and indexed in a particular fashion, their model-building algorithm composes an initial *coherent* model of the device. This initial model is coherent in that it explains the expected behavior, it is internally consistent, and the set of equations implied by the model are not over-constrained. However, it may not be as *parsimonious* as it could be; that is, it may be possible to further simplify the model while maintaining the structural and behavioral constraints of the device, and the ability of the model to explain the expected behavior. Their reformulation procedure performs this simplification by removing unnecessary model fragments from the model and also by replacing model fragments by their approximations whenever it is possible to do so without compromising the coherence of the model.

The organization of the library of model fragments makes explicit the causal approximation relations among alternative descriptions of a given phenomenon. Thus, alternative simpler descriptions can be found without search, which enables an efficient execution of the approximation.

The metric for evaluating and comparing problems is parsimony of the formulation. A parsimonious model contains only those model fragments necessary to explain the expected behavior while satisfying the structural and behavioral constraints. The reformulation algorithm guarantees that the resulting model is the most parsimonious in the sense that no model fragment in the model can be removed or replaced by any of its causal approximations without violating the structural and behavioral constraints.

The problems, P_1 and P_2

The original problem, P_1, and the reformulated problem, P_2, consist of the three-tuples, $\langle \texttt{Query}_1, \texttt{Form}_1, \texttt{Assmpt}_1 \rangle$ and $\langle \texttt{Query}_2, \texttt{Form}_2, \texttt{Assmpt}_2 \rangle$.

$\texttt{Query}_1 = \texttt{Query}_2$: Generate a causal explanation of the specified expected behavior.

$\texttt{Form}_1$ consists of:

1. A library of model fragments, where
   - model fragments are grouped into assumption classes,
   - each assumption class has one best model fragment of which the rest are causal approximations, and
   - the causal approximation relations are acyclic.
2. The expected behavior of the device in terms of an expected causal relations among a pair of variables, represented as $\texttt{causes}(v_1, v_2)$.
3. The structural and behavioral constraints.
4. A coherent causal model of the device comprised of instances of model fragments.

$\texttt{Form}_2$: The same (1) through (3) as in $\texttt{Form}_1$. As for (4), a simplest, coherent and causal model of the device.

$\texttt{Assmpt}_1 = \texttt{Assmpt}_2$: The assumptions underlying the library of model fragments if any.

The reformulation technique

Proc: Input to the reformulation procedure is a compositional model that may contain unnecessary model fragments and may not be as simple as possible. The reasons for this are that the model formulation algorithm initially selects the most accurate model fragment in each assumption class and also that it satisfies some constraints by adding model fragments that may later prove unnecessary.
The reformulation procedure exploits two operators:
1. Replacement of a model fragment by one of its immediate causal approximations, as defined in the model-fragment library; and
2. Removal of an unnecessary model fragment.

The first operator is applied repeatedly while ensuring that the resultant model can explain the expected behavior. This is achieved by an order of magnitude reasoner. The second operator is then applied, again ensuring that the expected behavior can be explained and that all the structural and behavioral coherence constraints are satisfied. Note that the reformulation procedure generates one simplest adequate model. More than one may exist but the procedure stops after finding the first.

Cond: The model-fragment library must be organized in a manner that makes the causal approximation relations among alternative descriptions of the same phenomena explicit and alternative simpler model fragments readily available. This is achieved as follows: Model fragments that describe the same physical phenomenon but are based on different assumptions are grouped into an assumption class. Each assumption class is a collection of mutually contradictory descriptions of the same phenomenon. For example, the behavior of a resistor can be described as a constant resistance resistor or a temperature-dependent resistance. Each assumption class is organized in a tree with model fragments as nodes and causal approximation relations as links. Each assumption class has a single most accu-

rate description at the root, and all other descriptions are causal approximations of it. Given two model fragments m_i and m_j, m_i is a causal approximation of m_j if and only if m_i is an approximation of m_j and the causal ordering relations among variables of m_j is a superset of those of m_i. Approximation relations among model fragments are given *a priori* in the library.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}$, $\texttt{Compars}_{\texttt{prob}}$: Nayak and Joskowicz evaluate a formulation—more specifically the model part of a formulation—in three ways, namely coherence, causality, and parsimony of a model. Coherence refers to the consistency and completeness of a model with respect to the expected behavior and other given behavioral and structural constraints. Causality refers to the ability of a model to explain the expected causal relations among variables. Parsimony refers to the simplicity of a model.

Coherence is in $\texttt{Evals}_{\texttt{prob}}$ since it is a characteristics of a formulation—more specifically, that of a model. Coherence of a model is defined in terms of consistency and completeness of a model. A model is coherent when it is both consistent and complete. A model is consistent when it does not contain model fragments with mutually contradictory assumptions. A model is complete when the set of equations entailed by its model fragments are complete. A set of equations is complete if it contains the same number of equations as variables and no subset is overconstrained.

Causality is in $\texttt{Evals}_{\texttt{prob}}$ as it is a characteristic of a formulation. More specifically, it is a characteristic of the model as well as of the expected behavior, which is expressed in terms of causal dependency relations between a pair of variables. A model is a causal model if and only if the causal ordering generated from the equations of the model entails the expected behavior.

The most important characteristics of the reformulation procedure is that it preserves the coherence and causality properties of a model while making it most parsimonious. Thus, the effect of the reformulation is expressed in terms of parsimony.

Parsimony is in $\texttt{Compars}(P_1, P_2)$ as it is defined in terms of a binary relation, `simpler-than`, between models instead of some measure of the degree of parsimony of an individual model. The `simpler-than` relation in turn is defined in terms of a binary relation, $\texttt{approximation}(m_1, m_2)$, between model fragments. The approximation relations are given explicitly in the model-fragment library. A model M_1 is simpler-than M_2 if for each model fragment m_1 in M_1, either m_1 is also in M_2 or there is another model fragment m_2 in M_2 such that m_1 is an approximation of m_2. A model M_1 is said to be parsimonious if it is coherent and there is no other model strictly simpler than M_1 that is also coherent. The reformulation procedure is guaranteed to produce a parsimonious model. Though there may be more than one, the procedure stops after producing one.

$\texttt{Evals}_{\texttt{p+r}}$: The authors evaluate the reformulation procedure in terms of its computational complexity, which is in $\texttt{Evals}_{\texttt{p+r}}(R, P_1)$. The complexity of the refor-

mulation procedure relative to the problem encoding is polynomial *provided* the order of magnitude reasoning used to verify that the simplified model still explains the expected behavior is polynomial. The tractability of the procedure is ensured by the upward failure property regarding the need to explain the expected behavior. The upward failure property in this case means that, if a model fails to explain the expected behavior, any of its approximation will also fail, thus allowing one to prune large parts of the search tree. The upward failure property is ensured by the requirements that: (1) the approximation relations among model fragments in the library are causal approximations, (2) the causal approximation relations are acyclic, and (3) each assumption class has only one root, i.e., the most accurate model fragment.

Discussion. The success of Nayak and Joskowicz's elegant and efficient model formulation as well as simplification procedures hinges entirely on the strong restrictions placed on the organizations of the model-fragment library as well as on the types of constraints it can handle. In other words, the model-fragment library (and to a large extent, the problem itself) must be carefully crafted for the procedure to run efficiently. This criticism also applies also to other model composition techniques that rely on an elaborately structured model-fragment library such as the one used in [15]. Because of the restrictions placed on the problems and model-fragment library, the practicality and generality of the simplification procedure described here remains questionable. For example, the requirement that all approximation relations must be causal approximations seems to leave out many other useful types of approximations, such as linearization discussed in Section 5.2.2. Further, the requirement that the builder of a model-fragment library must specify approximation relations a priori is also restrictive. As we have illustrated in Section 2.3 with the example of square and sine waves, given two different descriptions, the question of which is a useful approximation of the other may very well depend on the question being asked.

As Pos points out [26], modeling is a naturally dynamic and iterative process, during which problems themselves, including the assumptions, constraints, and formulations, are likely to change. In light of this dynamic and iterative nature of the modeling endeavor, reformulation approaches that involve selection from pre-specified set of model fragments using hard-wired approximation relations among them could prove too limiting. Furthermore, the requirement for a large, carefully crafted model-fragment library to be constructed makes the approaches that depend on such libraries less practical in most cases. In the following section, we discuss a technique that does not require the use of a model-fragment library, at least for the purpose of reformulation.

5.1.2. Critical abstractions

Williams proposes a reformulation technique, called *Critical Abstraction*, for simplifying a model of a physical device to the furthest extent possible while preserving its ability to answer a given query [40]. The information retained in the model should be necessary and sufficient to answer the query while supporting a causal explanation of the answer. There are two inputs to the procedure: a model of the physical behavior, and a query on a variable in the model. A model contains two types of information: physical components and their behavior equations. The behavior equations describe the interactions among the

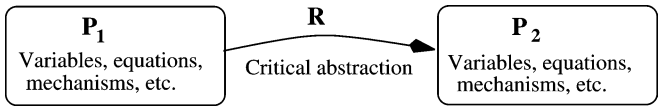

Fig. 10. Critical abstraction.

components as well as within the components. The output is a simplified version of the model that is sufficient to answer the query. See Fig. 10.

The reformulation is a three-step procedure that modifies the model according to the given query. First, superfluous interactions among mechanisms are eliminated. This is achieved by first building the causal ordering of the variables then eliminating the equations that are not causally upstream of the query variable. Second, the behavior descriptions of each mechanism are simplified by combining equations to eliminate the variables and interactions that are only internal to each mechanism. Third, individual variables and equations are modified to retain only those features that contribute to that determination of the answer to the query.

As in the case of Nayak and Joskowicz's reformulation technique, the metric for evaluating and comparing problems is parsimony of the formulation, though the definitions of parsimony differ. While Williams' concept of parsimony of a model is similar to that of Nayak and Joskowicz's in that it requires the set of variables and causal relations entailed by a more parsimonious model to be a subset of those entailed by the original model, it does not involve approximation relations. As a result, his reformulated model produces the exact same answer to the query as the original model. Furthermore, Williams requires that reformulation preserve the connection between equations and the mechanisms that produce them. This requirement ensures that the resulting model contain enough information to explain the answer in terms, not only of the causal dependency relations among variables, but also of the actual physical mechanisms that are responsible for the relations. Williams' reformulation algorithm guarantees that the resulting model is the most parsimonious in the sense that no model that is strictly more abstract can produce the same answer to the query while maintaining the same causal and teleological explanatory power.

The problems, P_1 and P_2

The original problem, P_1, and the reformulated problem, P_2, consist of the three-tuples, $\langle \texttt{Query}_1, \texttt{Form}_1, \texttt{Assmpt}_1 \rangle$ and $\langle \texttt{Query}_2, \texttt{Form}_2, \texttt{Assmpt}_2 \rangle$.

$\texttt{Query}_1 = \texttt{Query}_2$: Find, or verify, the value of a qualitative variable x.

$\texttt{Form}_1$ and $\texttt{Form}_2$ consist of the following:

1. A set of variables V.
2. A set of independent variables $IV \subset V$.
3. A set of equations I (called *interactions*) on the variables of V, such that
   - each member of V appears in at least one equation,
   - there is no simultaneity, and
   - the equations are not redundant.

4. A set of mechanisms M (such as components, connections, and processes).
5. An onto function $IM : I \to M$, which associates each interaction with the mechanism that produces it.

In addition, $\texttt{Form}_2$ is an *abstraction* of $\texttt{Form}_1$, and $\texttt{Form}_2$ is the most abstract possible in the sense that no formulation that is strictly more abstract than $\texttt{Form}_2$ can produce the same answer to the query while maintaining the same causal explanatory power. Williams defines abstraction as follows:

Definition 1 (*Abstraction*). $\texttt{Form}_2$ is an abstraction of $\texttt{Form}_1$ if $V_2 \subset V_1$, $IV_2 \subset IV_1$, $M_2 \subset M_1$, and for every m in M_1, its interactions in $\texttt{Form}_1$ entail its interactions in $\texttt{Form}_2$. In other words, for each i in I_2 and its corresponding mechanism m, i is satisfied whenever m's interactions in $\texttt{Form}_1$ are satisfied.

$\texttt{Assmpt}_1 = \texttt{Assmpt}_2$: Whatever assumptions that underlie P_1.

The reformulation technique, R

Williams places the following restrictions, Cond, on the set of equations:

- The equations describe static interactions.
- The equations are non-redundant.
- The equations do not contain simultaneity.

The reformulation procedure, Proc, consists of the following steps:

1. Determine the causal ordering among variables using the equations. The conditions that there is no simultaneity and that the equations are non-redundant guarantee that a causal ordering can be found efficiently and that it is unique. Remove all the variables, interactions, and mechanisms upon which the query variable does not causally depend.
2. Simplify the remaining set of equations by eliminating the variables that are internal to one mechanism and do not participate in interactions with other mechanisms. Again, the lack of simultaneity in equations makes this step straightforward.
3. Since the query asks for the qualitative value of a variable, the last step turns the remaining equations into qualitative equations to the furthest extent possible without losing the ability to answer the query. This step, which ultimately produces the final set of equations in the mixed qualitative and quantitative algebra, is achieved roughly as follows. With each equation on the causal ordering starting from the query variable and moving towards the independent variables on which the query variable causally depends, the sign operator is pushed inwards as far as possible using the homomorphism of the sign operator with respect to multiplication, division and negation.[5] This process has the overall effect of pushing the sign operator from the query variable to

[5] The sign operator [] is homomorphic with respect to multiplication, division and negation but not with addition and subtraction. In other words, $[a \times b] = [a] \times [b]$, $[a/b] = [a]/[b]$, $[-a] = -[a]$, but the same does not hold in general for $+$ and binary $-$.

causally upstream variables and expressions as far as possible. At the same time, this process identifies the expressions whose qualitative values can only be determined by evaluating them quantitatively in order to answer the query.

The requirement for static equations is essential for the claim that the procedure identifies every landmark value for each variable necessary to answer the query and that those are the only ones needed. Since the equations are static, simulation is not necessary for such identification. If the equations were dynamic, one would have to perform simulation over time to identify relevant landmark values. The requirement that the equations be non-redundant ensures that the causal ordering can be found and is unique. Otherwise, one would have to select a non-redundant subset of the equations to determine the causal ordering. Finally, the requirement that the equations should not contain simultaneity ensures that the procedure can always associate a unique mechanism with an equation, which Williams regards as important for a causal explanation.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}, \texttt{Compars}_{\texttt{prob}}$: Williams evaluates a formulation according to three aspects, namely its ability to answer the query, causality, and parsimony. The evaluators seem similar to those of Nayak and Joskowicz's on the surface but, in fact, they are quite different in their actual definitions. The overriding concern in Critical Abstraction is that reformulation should not change the answer produced by the formulation. The goal of reformulation here is to simplify the formulation to the furthest possible extent without altering any aspect of the answer to the query. This is a significant difference from reformulation in Nayak and Joskowicz's work, where reformulation produces a more approximate model, which presumably will produce an answer that is different from the original. Likewise, the concept of parsimony in Nayak and Joskowicz's work relies on that of approximation while approximation plays no role in the concept of parsimony in Critical Abstraction. As for causality, an important difference between the two is that Williams includes the connection between the interactions and mechanisms that produce the interactions in the overall concept of causality in a formulation while Nayak and Joskowicz do not.

The ability to answer the query is in $\texttt{Evals}_{\texttt{prob}}(P)$, and the means of comparison for this evaluator is that they must be exactly the same. The answer to the query entailed by the two formulations must be exactly the same in all respects, including the accuracy, precision, and ambiguity.[6]

$\texttt{Evals}_{\texttt{p+r+p}}$: The concept of causality Williams is concerned with includes the connection between interactions and mechanisms in addition to the causal dependency relations among variables. This notion of causality is embodied in the last part of the definition of abstraction presented above, which says that the abstraction must preserve the connection between interactions and individual mechanisms that pro-

[6] If the initial formulation is qualitative, the reformulation must not change the ambiguity in the answer.

duce them. We consider this evaluator to be in $\texttt{Evals}_{\texttt{p+r+p}}(R, P_1, P_2)$ since it is defined as a relation between two formulations.
Parsimony here is also in $\texttt{Evals}_{\texttt{p+r+p}}(R, P_1, P_2)$ as Williams' concept of parsimony is mainly that of abstraction as defined above. It also includes a measure of simplicity of the quantity space of variables. Informally, the less distinctions one needs to make in the quantity space of a variable, the simpler the quantity space.[7] In the mixed quantitative-qualitative algebra employed in Critical Abstraction, this means that for each variable, the fewer quantitative expressions involving the variable there are in the equations, the simpler its quantity space is. However, Williams does not present a complete formal definition of parsimony.

$\texttt{Evals}_{\texttt{p+r}}$: The computational complexity of the reformulation is in $\texttt{Evals}_{\texttt{p+r}}(R, P_1)$. Although Williams does not present an analysis of the computational complexity, all the steps in the procedure appear linear with respect to the number of equations and variables.

Discussion. The strength of Critical Abstraction is that it does not require a pre-enumerated set of pieces of alternative descriptions, as do the model formulation, or reformulation approaches that rely on model-fragment libraries. Based solely on the query, Williams simplifies the formulation by removing parts that are irrelevant to the query and identifying the minimal set of quantitative distinctions that are necessary. Furthermore, this reformulation is performed without compromising the causal explanatory power of the formulation or its ability to answer the query. Williams accomplishes this without incurring a large computational cost.

Also noteworthy is Williams' attempt to abstract the quantity space of the variables in accordance with the query. The last step in his reformulation procedure automatically identifies the necessary quantitative distinctions that must be made to answer the query. This is in contrast to the more common practice in qualitative reasoning where landmark values are, for most part, pre-enumerated.

In both of the approaches discussed in this section for reformulating models, the professed goal of reformulation is a better explanation, where "better" means causally consistent and simpler, even though the end product in both cases is another formulation and not an explanation. A working assumption in both cases is that a simpler model will produce a better explanation. It is also worth noting that their goal is not reduction of the computational effort required to answer the query though one might expect such a reduction, especially given both approaches produce simpler models in the sense of "more approximate" or "requiring less quantitative distinctions". This suggests that what one perceives as a satisfying causal explanation of a physical system involves more than just quantities and equations; it must somehow establish connections between quantities and equations on one hand and the components of the cognitive model of the physical system, may it be model fragments (as in the Nayak and Joskowicz's case) or mechanisms (as in Williams'). Examples of reformulation whose primary goal is reduction of computational cost are found in the next section on reformulation at the Equation Processing Stage.

[7] A quantitative variable can be considered to an infinite number of distinctions.

5.2. Reformulation at the equation processing stage

We now turn to examples of reformulation techniques at the Equation Processing Stage. Reformulation of a system of equations is a common practice when the system is too large or complex to be solved or analyzed directly. Thus, most of the techniques employed at this stage, including the first three discussed in this section, simplify the system through such means as aggregating variables and equations (Section 5.2.1), substituting nonlinear equations by linear ones (Section 5.2.2), and decomposing the problem space (Section 5.2.3). However, simplification is not the goal of the last technique discussed in this section: In Exaggeration (Section 5.2.4), a system of qualitative equations based on standard numbers is replaced by one based on nonstandard numbers, producing a more complex system exhibiting more complex behavior.

5.2.1. Aggregation of nearly decomposable systems

Simon and Ando provide a formal basis for a reformulation technique that aggregates variables in dynamic systems to reduce the size of the system [32]. The aggregation procedure they describe takes a set of linear differential equations that is nearly decomposable and produces a smaller aggregate. A nearly decomposable system is a system composed of subsystems such that the interactions among subsystems are much weaker than the interactions within each subsystem. If a given linear system is nearly decomposable, and if the system is stable, then their procedure defines an aggregate variable for each subsystem as a linear function of the original variables in the subsystem, and reformulates the entire system in terms of the aggregate variables and equations. The result is a smaller system set that adequately describes the mid-to-long term behavior of the original set. The primary goal of aggregation is reduction of the cost of solving the equations since the size of the system has a direct impact on the cost of finding the solution.

The basis for the aggregation procedure is the observation that the behavior of a nearly decomposable system can be characterized in the following four stages [32]:

1. *Short-run dynamics*, where variables in each subsystem are moving towards their relative equilibrium independently of other subsystems.
2. *Short-run equilibrium*, where the most significant root of each subsystem dominates the behavior of the subsystem.
3. *Long-run dynamics*, where the variables in each subsystem move together towards overall equilibrium.
4. *Long-run equilibrium*, where the most significant root of the entire system dominates.

Aggregation produces a system that ignores the first stage but models the remaining stages with increasing accuracy as time goes on.

Simon provides a good example of aggregation of variables in the case of heat flow within a building [31]. Consider a building divided into a large number of rooms that are in turn divided into a number of offices by partitions. The outside walls provide perfect thermal insulation from the environment. The walls between rooms are good but imperfect insulators while the partitions within rooms are poor insulators. In this situation, the temperature equilibrium among offices within one room will be reached very rapidly while

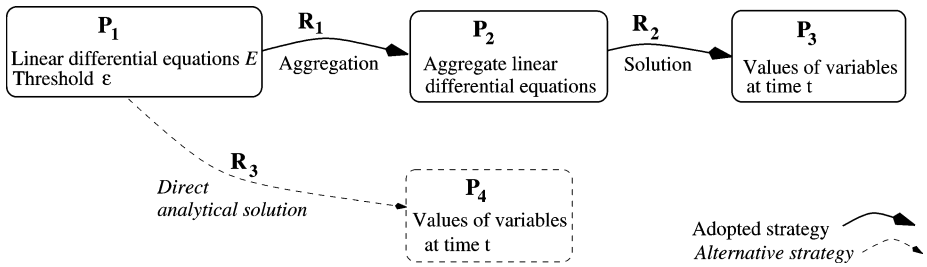

Fig. 11. Solving linear ordinary differential equations by aggregation.

equilibrium across rooms will be reached only slowly. Therefore, as long as we are not interested in modeling rapid temperature fluctuations within one room, a useful aggregation is to have one temperature variable for each room and to assume that equilibrium within a room is reached instantaneously.

Fig. 11 shows a strategy involving aggregation to solve linear differential equations. It is depicted as a two-step strategy of aggregation (R_1) followed by solution of the aggregate system (R_2). It also shows an alternative strategy of directly solving the original system as the path $\langle P_1, R_3, P_4 \rangle$.

The problems, P_1 and P_2

The original problem, P_1, and the reformulated problem, P_2, consist of the three-tuples, $\langle \texttt{Query}_1, \texttt{Form}_1, \texttt{Assmpt}_1 \rangle$ and $\langle \texttt{Query}_2, \texttt{Form}_2, \texttt{Assmpt}_2 \rangle$.

$\texttt{Query}_1 = \texttt{Query}_2$: What is the value of the variable x at time t, where x is a variable in E.

$\texttt{Form}_1$ consists of the following:

1. A set E of linear ordinary differential equations involving n variables and n equations.
2. A small value ε, to be used as the threshold value for classifying coefficients in E into those representing strong or weak interactions.

$\texttt{Form}_2$ consists of the following:

1. A set E' of linear ordinary differential equations involving m variables and m equations, where $m < n$.
2. n equations of the form $x_i = f_i(y_j)$, one for each variable x_i in E, where f_i is a linear function of y_j, and y_j is a variable in E'.

$\texttt{Assmpt}_1$: None.

$\texttt{Assmpt}_2$: The values of x for a relatively small t are not important (i.e., the short-run dynamics is not of interest).

The reformulation techniques, R_1 and R_2

The aggregation technique R_1 requires the following conditions to be satisfied by the original system:

$\texttt{Cond}_1$:

- The system is nearly decomposable with respect to the threshold value ε. This means that given ε, the matrix consisting of the coefficients in E can be put in a nearly block-diagonal form, where the elements outside the diagonal sub-matrices are all less than ε. This can be achieved by rearrangement of rows and columns if necessary. However, no other manipulations, such as Gaussian elimination, are allowed.
- The behavior of the system represented by $\texttt{Form}_1$ is stable.

The reformulation procedure, $\texttt{Proc}_1$, consists of the following steps:

1. For each diagonal sub-matrix in $\texttt{Form}_1$,
   - solve the sub-matrix by itself,
   - choose the most significant root,
   - define the aggregate variable as a linear function of the most significant root. The definition of the aggregate variable and the eigenvector associated with the most significant root determines the linear relation between the aggregate variable and each of the variables in the subsystem.
2. Rewrite the original matrix by substituting each variable with its expression in terms of the respective aggregate variable.

The solution step R_2 follows R_1. R_1 has no conditions. Its procedure, $\texttt{Proc}_2$, consists of the following steps:

1. Solve E'.
2. Using the solutions for the variables y_j's in E' and the n equations of the form $x_i = f_i(y_j)$ in $\texttt{Form}_2$, compute the values of x_i's.

Iwasaki and Bhandari later generalized the original aggregation procedure to cases where there are multiple significant roots per system [12], but the intuition remains the same.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}$, $\texttt{Compars}_{\texttt{prob}}$: The immediate effect of aggregation is to reduce the size of the system of differential equations that must be solved. Aggregation also has an effect of making the solution less accurate. The size of a formulation is in $\texttt{Evals}_{\texttt{prob}}(P)$. It is measured in terms of the number of variables in the system. The size strictly decreases as a result of aggregation.

Another effect of aggregation is on the accuracy of a solution, which is also in $\texttt{Evals}_{\texttt{prob}}(P)$. However, the comparator for this evaluator ($\texttt{Compars}_{\texttt{prob}}(P_1$,

P_2)) is more complicated than the comparator for the size. The magnitude of the discrepancy between the two solutions in general decreases as t grows.

$\texttt{Evals}_{\texttt{p+r}}$: One can evaluate the reformulation procedure in terms of its computational complexity. Solving the matrix analytically requires computing eigenvalues and eigenvectors of the non-singular square matrix whose elements are the linear coefficients in the equations. Given ε, the complexity of all the steps in $\texttt{Proc}_1$ is constant or linear, except for the step 1, whose complexity is $O(m_i^3)$, where m_i is the size of the ith sub-matrix. Thus, the overall complexity of $\texttt{Proc}_1$ is $O(m_{\max}^3)$ where $m_{\max}$ is the size of the largest sub-matrix.

$\texttt{Evals}_{\texttt{strat}}$: Likewise, strategies can be evaluated with respect to their computational complexity. The complexity of the strategy represented by the path $\langle P_1, R_1, P_2, R_2, P_3\rangle$ is the greater of $O(m_{\max}^3)$ and $O(m^3)$, where $m_{\max}$ is the size of the largest sub-matrix and m is the size of the aggregate matrix. The complexity of the strategy represented by the path $\langle P_1, R_3, P_4\rangle$ is $O(n^3)$, where n is obviously larger than m or $m_{\max}$. The cost of computing the numerical solution decreases[8] in general.

Discussion. Aggregation reduces computational cost at the expense of some loss in accuracy, though the discrepancy diminishes as t increases. That is why this reformulation method is appropriate only when the short-run dynamics are not of interest. Of course, if the short-run dynamics are of primary interest, one can always solve the subsystems independently to obtain a reasonably accurate solution for a small t.

Finally, aggregation can have an effect on the cognitive transparency of a formulation. Though in the work of Simon and Ando the issue of cognitive transparency of a model did not arise, it is often an important consideration as we have seen in Section 5.1 on reformulation of models. Depending on the ways in which aggregate variables are defined, aggregation can make a formulation more compact and thus easier to understand or it can make the physical meaning of a formulation more difficult to comprehend. If the original variables in a subsystem all have the same physical dimension, one could define the aggregate variable so that it has an easily discernable physical significance such as the sum or the average. In this case, the resulting aggregate formulation will be easier to understand as a description of a physical behavior. However, in other situations, for example if the original variables have different dimensions, the aggregate formulation may be difficult to interpret physically even if it is useful computationally.

5.2.2. Stability analysis of nonlinear systems through linearization

Nonlinear systems can exhibit entirely new types of behavior compared to linear systems, but rarely are there tools available to analyze these behaviors. Fortunately, it is often unnecessary to obtain detailed numerical or analytical solutions but is sufficient to characterize only some aspect of the system behavior. Existence of equilibrium points and their stability are examples of aspects frequently studied about nonlinear systems [18].

[8] The cost will be proportional to $m \times t$, where m is the size of the system and t is the number of time steps at which variables are evaluated.

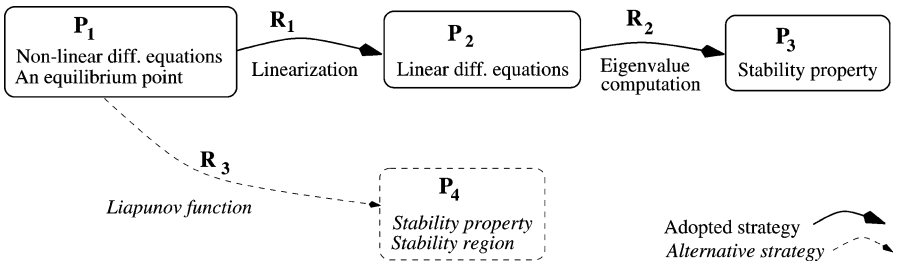

Fig. 12. Stability by linearization.

Linearization is a strategy commonly used to evaluate the stability of a nonlinear system near equilibrium points. Linearization is justified because over a small time interval, the performance of the system is approximately governed by the linear terms. Provided that the linear terms do not vanish near the equilibrium point, these terms dominate and thus determine stability around the point [18]. If the linear terms do vanish, a separate analysis is required.

Once linearized, the stability of the system is determined by the location of the eigenvalues of the system matrix in the complex plane. Thus, the problem-solving strategy consists of the two sequential steps shown in Fig. 12—R_1, which transforms a nonlinear system to a linear system, followed by R_2, which determines stability. This strategy is motivated by the lack of a general method to directly determine the stability of a nonlinear system. For the purpose of stability analysis, there is actually an alternative to linearization. The alternative involves a Liapunov function and is shown in Fig. 12 as a strategy represented by the path $\langle P_1, R_3, P_4\rangle$. R_3 requires no computation with the condition that a Liapunov function is known for the nonlinear system in P_1, since the existence of such a function immediately implies that the system is stable and the region over which the function is defined is the region of stability. However, ways to construct a Liapunov function are known only for limited classes of nonlinear systems.

The problems P_1, P_2, and P_3

$\texttt{Query}_1 = \texttt{Query}_2 = \texttt{Query}_3$: Determine the stability of the system near an equilibrium point.

$\texttt{Form}_1$ comprises:

1. A set of nonlinear, time-invariant[9] differential equations $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t))$ of n variables.
2. An equilibrium point for the nonlinear system.

$\texttt{Form}_2$ is a set of time-invariant linear differential equations of n variables that approximate $\texttt{Form}_1$ at an equilibrium point. $\texttt{Form}_2$ has the same set of variables and the same

[9] The dynamic behavior of a continuous system of n variables is described by a set of differential equations of the following general form: $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), t)$. The system is said to be *time-invariant* when the functions **f** do not depend explicitly on time, i.e., $\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t))$.

number of equations as $\texttt{Form}_1$. The system matrix of $\texttt{Form}_2$ is the Jacobian of $\texttt{Form}_1$ evaluated at the equilibrium point.

$\texttt{Form}_3$ is $\texttt{Form}_2$, the positions of its eigenvalues in the complex plane, and the result (i.e., one symbol of {`stable`, `unstable`, `unknown`}).

$\texttt{Assmpt}_1$: None.

$\texttt{Assmpt}_2 = \texttt{Assmpt}_3$: $\texttt{Form}_2$ is a valid approximation of $\texttt{Form}_1$ near the equilibrium point, i.e., linear terms in $\texttt{Form}_1$ do not vanish in its vicinity.

The reformulation techniques R_1 and R_2

R_1 transforms a nonlinear system to a linear system. It has no conditions, and the procedure is as follows:

$\texttt{Proc}_1$: Compute and evaluate the Jacobian of $\texttt{Form}_1$ at the equilibrium point. Construct a linear system with the Jacobian as the system matrix.

R_2 determines the stability of $\texttt{Form}_2$. It has no conditions, and the procedure is as follows:

$\texttt{Proc}_2$: Compute the eigenvalues, λ_i, of the system matrix of $\texttt{Form}_2$. Then, the stability is inferred as follows:

- If at least one eigenvalue is found to be in the right-hand side of the complex plane ($\exists i, \mathrm{Re}(\lambda_i) > 0$), the system is unstable.
- If all eigenvalues are in the left-hand side of the complex plane ($\forall i, \mathrm{Re}(\lambda_i) < 0$), the system is stable.
- If all eigenvalues are in the left-hand side of the complex plane, but at least one has a zero real value ($\exists i, \mathrm{Re}(\lambda_i) = 0$), then no conclusions can be drawn about the stability, and one must analyze the higher-order terms of the function $\boldsymbol{f}$.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}$: As linearization is motivated by the lack of general method for determining stability of a nonlinear system, we can categorize it as an example of engine-driven reformulation. More specifically, an evaluator of a problem ($\texttt{Evals}_{\texttt{prob}}(P)$ for P_1 and P_2) is the existence of a general, computable procedure for determining stability.

Other aspects of the problem that are changed by linearization are accuracy and generality of a formulation as a description of a physical system. Accuracy and generality of a formulation with respect to the physical behavior that it describes are elements of $\texttt{Evals}_{\texttt{prob}}(P)$ for $P = P_1$ and P_2. Presumably, P_1 is more accurate than P_2, but the discrepancy between the two descriptions decreases as the equilibrium point is approached. Accuracy is closely related to generality if generality is defined as the subspace in the n-dimensional space in which a formulation is valid as a description of the system. P_1 is a valid description in the

entire space in which the system is defined, while P_2 is only valid in the vicinity of the equilibrium point.

$\texttt{Evals}_{\texttt{p+r}}$: The two reformulation-steps, R_1 and R_2, in the linearization strategy can be evaluated with respect to the complexity of their procedure, $\texttt{Proc}_1$ and $\texttt{Proc}_2$, relative to the problem encoding, $\texttt{Form}_1$ and $\texttt{Form}_2$, respectively. The worst-case complexity of $\texttt{Proc}_1$ is $O(n^3)$ where n is the number of variables in the system. For R_2, which requires the computation of the eigenvalues of the system matrix, it is also $O(n^3)$ in general.

$\texttt{Evals}_{\texttt{strat}}$: The two strategies represented in Fig. 12 as the path $\langle P_1, R_1, P_2, R_2, P_3\rangle$, involving linearization, and the path $\langle P_1, R_3, P_4\rangle$, involving a Liapunov function, can be evaluated in terms of their computational cost, which is in $\texttt{Evals}_{\texttt{strat}}$. The cost of the former is proportional to n^3 as can be seen from the complexity of its component procedures, while the cost of the latter is a constant. The two strategies can also be characterized in terms of their generality, also in $\texttt{Evals}_{\texttt{strat}}$: Linearization is justified in cases where the linear terms do not vanish while Liapunov function can be used only in cases where one is known.

Discussion. Linearization is one of the most important approximation tools for modeling the behavior of complex systems. Though basic equations of physics are nonlinear, linearization is a practical necessity since so little of general nature is known about nonlinear systems. Also, there are many situations where linear approximation is adequate for most purposes. Linearization is particularly useful for stability analysis since one is interested in the behavior of the system in a limited region where the linear terms tend to dominate. Since it is motivated by lack of other methods to analyze nonlinear systems, we can classify this reformulation as engine-driven.

Though linearization is performed only around an equilibrium point in the case discussed in this section, linear approximation can be used more generally to reason about the global characteristics of nonlinear dynamic systems. In piecewise linear approximations of nonlinear ordinary differential equations, the entire space of behavior is divided up into regions such that a different linear equation is used to approximate the behavior in each region (e.g., [23,30]). The characteristics of the behavior of the original system can be inferred by analyzing each linear region separately and by piecing the inferred behaviors together. Sacks' PLR system [30] analyzes nonlinear dynamic systems using piecewise linear approximations. Given a nonlinear system, PLR semi-automatically breaks up the space into regions and proposes a linear approximation for each region using information about the original function (such as local extrema) as well as user input. Then, it analyzes the behavior within each region and pieces them together to draw a phase diagram of the entire space. In the process, PLR refines the initial linear approximations as necessary to detect all the qualitatively significant features of the behavior. Reasoning by PLR is expensive; the computational complexity of the inequality reasoning employed by PLR is exponential in the length of its input. However, the expense is justifiable since nonlinear dynamic systems can otherwise resist analysis altogether.

Breaking up a complicated behavior into smaller pieces each of which can be approximated by a simpler behavior is a common strategy employed by human experts analyzing

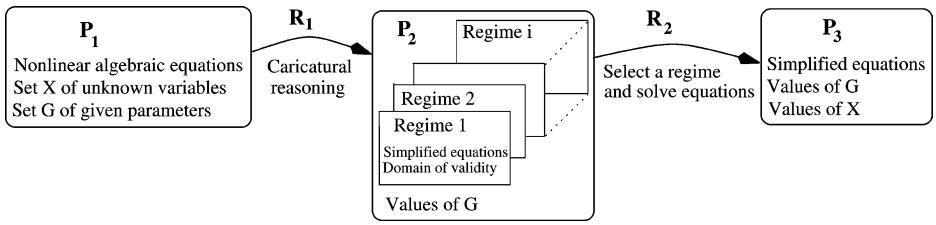

Fig. 13. Caricatural modeling.

a complex system. The reformulation technique discussed next also follows this strategy in solving high-order algebraic equations.

5.2.3. Caricatural reasoning

In [41], Williams and Raiman study a reformulation technique used by analytical chemists for analyzing a set of nonlinear, high-order algebraic equations such as those describing the equilibrium behavior of a chemical reaction. High-order nonlinear equations[10] are hard to solve in general, and, furthermore, the complexity of such systems of equations makes it difficult to gain clear insight into the space of possible solutions. They introduce *Caricatural reasoning* as a technique for decomposing a problem space into a set of *regimes*, each of which is defined by a set of assumptions about the relative orders of magnitude of certain terms. Such decomposition not only facilitates understanding of the general shape of the solution space but also gives rise to a set of simplified equations for each regime to approximate the solution in the regime. Since this reformulation operates solely on equations and variables, we place it among reformulation techniques for the Equation Processing Stage.

Fig. 13 depicts the entire problem-solving strategy as a two-step process of decomposition (R_1) followed by solution of a simplified set of equations (R_2). Once the problem is decomposed into regimes (as shown in P_2), one can select a regime whose conditions are satisfied by the set of givens and solve the simplified equations.

The problems P_1, P_2, and P_3

$\texttt{Query}_1 = \texttt{Query}_2 = \texttt{Query}_3$: Determine the values of the unknown variables.

$\texttt{Form}_1$: A set E of nonlinear algebraic equations involving a set G of given parameters and a set X of unknown variables.

$\texttt{Assmpt}_1$: None.

$\texttt{Form}_2$: A set of sets of simplified equations. Each set of equations is accompanied by a set of validity conditions about the relative orders of magnitudes of the parameter values.
Given values of the parameters in G.

[10] When the degree of the equations is equal to or higher than five, there exist no general closed form solutions (by Galois).

$\texttt{Assmpt}_2$: None.

$\texttt{Form}_3$: A set of simplified equations in a chosen regime.
Computed values of the unknowns in X.

$\texttt{Assmpt}_3$: The given values of the parameters in G satisfy the validity conditions of the chosen regime.

The reformulation techniques, R_1 and R_2

The entire problem-solving strategy consists of a sequence of two reformulation processes, R_1 and R_2.

R_1 replaces the original equation model by a patchwork of sets of simplified equations and determines their validity conditions. R_1 has no conditions. The procedure of R_1 relies heavily on a process called *qualitative resolution* to simplify and solve equations using orders of magnitude information. R_1 employs the machinery of Minima [40] for mixed qualitative and quantitative algebraic reasoning and Estimates [27] for reasoning about orders of magnitude information.

$\texttt{Proc}_1$ consists of the following five steps:

1. For each equation in E, a set of *caricatural assumptions* is generated. An assumption is generated by extracting from the equation the relative strength of the terms and by exaggerating this feature. Each assumption is in the form of *term1* $\ll$ *term2*, where *term1* and *term2* are terms appearing on the opposite sides of the equation.
2. A set C of all consistent combinations of caricatural assumptions is generated. Each such combination is called a *dominance regime*.
3. For each dominance regime $c \in C$, a set E' of simplified equations is generated from c and E. Each set of simplified equations holds under the assumptions defining the dominance regime.
4. For each dominance regime, a qualitative resolution mechanism is applied to the set of simplified equations to generate equations each containing only one unknown variable.
5. The validity conditions for each dominance regime are computed. Validity conditions are a set of relations involving only the parameters in G, and they are conditions that can be tested once the values of G are provided to select an appropriate dominance regime. Validity conditions are derived from c and E' using repeated qualitative resolution to eliminate all the appearances of unknown variables from c.

Once a problem is simplified by R_1, it is solved by R_2 for a specific case. Given a specific set of values for the parameters in G, $\texttt{Proc}_2$ selects an appropriate dominance regime and solves the simplified equations for the unknown variables using any available mechanism for solving algebraic equations. R_2 has no particular conditions.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}$, $\texttt{Compars}_{\texttt{prob}}$: The most immediate and obvious effect of caricatural reasoning is on the compactness of the formulation, which is in $\texttt{Evals}_{\texttt{prob}}$. Caricat-

ural reasoning makes a formulation less compact. While the original formulation describes the entire behavior with one set of equations, reformulation results in a long conjunction of conditionalized sets of equations.

A more important effect of caricatural reasoning is to simplify the equations. Simplicity of equations, which is also in $\texttt{Evals}_{\texttt{prob}}$, is measured in this case by the number of terms and the degrees of equations. Caricatural reasoning simplifies equations in these aspects since the addition of assumptions often results in the removal of terms after the qualitative resolution of equations, though in some cases an additional assumption results in no such simplification.

Simplification also has obvious implication on the solvability of a set of equations (also in $\texttt{Evals}_{\texttt{prob}}$). Simplicity in the above sense will generally make equations easier to solve and even make hitherto unsolvable ones to have closeform solutions. Note, however, that there is no guarantee that simplification will in fact make the equations solvable.

Another aspect of a formulation that Williams and Raiman emphasize is the cognitive transparency, which is also in $\texttt{Evals}_{\texttt{prob}}$. Cognitive transparency of a formulation refers to ease of deriving from the formulation insight into the behavior of the whole system. Though this aspect is clearly subjective and difficult to define precisely, it is an important objective in many examples of reformulation. In the case of caricatural reasoning, Williams and Raiman argue that decomposition of the problem space and simplification of equations lead to more cognitive transparency—an argument that seems to be borne out by the chemistry example that they present.

Since caricatural reasoning is an approximation technique, the answer to the query will be less accurate as a result. Accuracy of an answer is in $\texttt{Evals}_{\texttt{prob}}$. The magnitude of the error will depend on the actual parameter values and the regime selected to compute the answer. Williams and Raiman note that they are unable to bind the error, and they recognize the importance of extending their method in this direction.

$\texttt{Evals}_{\texttt{p+r+p}}$: An aspect of caricatural reasoning that is repeatedly mentioned by Williams and Raiman is that it preserves nonlinearity of equations, which is in $\texttt{Evals}_{\texttt{p+r+p}}$. However, their own example shows that nonlinearity is not necessarily preserved in all regimes. Stated more broadly, their objective is that "approximation should preserve essential characteristics of the behavior". This is another qualitative evaluator that is difficult to define precisely.

$\texttt{Evals}_{\texttt{p+r}}$: Caricatural reasoning could be evaluated in terms of its computational cost, but the actual cost will depend on the types and size of the system of equations. The authors do not provide complexity analysis of the steps in the reformulation procedure. Even if the decomposition step (R_1) is expensive, it may be cost-effective if the cost can be amortized over a large number of actual problems solved (R_2).

Discussion. The most significant contribution of caricatural reasoning is that it proposes an automatic method to identify a meaningful decomposition of the problem space where such decomposition is not immediately obvious. In contrast, in both linear approximation

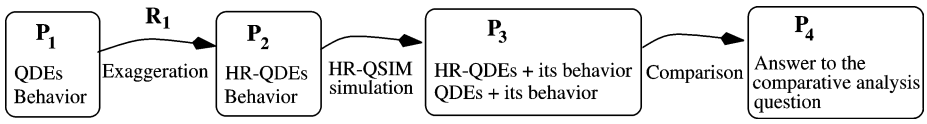

Fig. 14. Comparative analysis by exaggeration.

of nonlinear systems (discussed in Section 5.2.2) and aggregation of nearly decomposable systems (discussed in Section 5.2.1), a reasonable way to decompose is fairly obvious from the original system. Furthermore, despite the fact that decomposition does not necessarily guarantee that the resulting equations will be solvable, the decomposition itself is of much value in this case since it provides a person with insight into the space of solutions and helps her identify interesting families of solutions.

Caricatural reasoning achieves such decomposition through exaggeration of relative magnitudes of terms in equations. Exaggeration is also the modus operandi of the reformulation technique we turn to in the next section. However, exaggeration in the next section has the effect of complicating a model, while it is used here for the purpose of simplification.

5.2.4. Exaggeration

Exaggeration is a technique proposed by Weld [36] for performing comparative analysis [35]. Comparative analysis predicts how the behavior of a system will change as a result of the perturbation, given a model of the system, its behavioral description and a perturbation to the model. Weld proposes two techniques for comparative analysis, namely Qualitative Differential Analysis and Exaggeration in [35]. Exaggeration is of particular interest here since it involves a unique type of reformulation based on a change in the underlying mathematics.

Exaggeration answers a comparative analysis question by taking perturbation to the limit and comparing the resulting behavior with that of the original. It proceeds in three steps, which Weld calls transformation, simulation, and scaling. First, the transformation step reformulates a given qualitative model into a qualitative model in the space of hyper-real numbers with some of the variables being given extreme values such as infinite or infinitesimal according to the given perturbation. Second, the simulation step carries out a qualitative simulation using `HR–QSIM`, which is `QSIM`, a widely used qualitative simulation program [14], extended with hyper-real numbers. Finally, the scaling step infers the effects of the perturbation on the behavior by comparing the predicted behavior of the transformed model to that of the original model obtained by QSIM.

We focus on the transformation step in our discussion. Unlike any other reformulation techniques discussed in this paper, this reformulation involves very little syntactic change since the essence of the transformation is a change in the model of numbers underlying the formulation from the standard model to the nonstandard one. Exaggeration is also unique in that the reformulation results in a more complex formulation than the original one while most other reformulation techniques result in simplification. Complication of the formulation serves the present goal of comparative analysis by enabling comparison of the behaviors of the two formulations.

The problems

$\texttt{Query}_1 = \texttt{Query}_2$: What is the consequence of the given perturbation on the behavior of the system?

$\texttt{Form}_1$ comprises:

1. A mathematical model in the form of qualitative differential equations (QDEs),
2. Initial state description, and
3. The perturbation.

$\texttt{Form}_2$: QDEs in the hyper-real space, where some of the variable values are given extreme values. The choice of such variables and their values depends on the given perturbation.

$\texttt{Assmpt}_1 = \texttt{Assmpt}_2$: None.

The reformulation technique

$\texttt{Proc}_1$: Reformulation of a standard QDE to a non-standard QDE requires little syntactic change in the formulation. An important change in the formulation takes place in the semantic level since, in $\texttt{Form}_1$, all the variables and operators, which used to be defined over the reals, are now defined over the hyperreals. The difficult part of the reformulation procedure is deciding how the perturbation is to be reflected in a value of a variable in the nonstandard QDE. Weld does not provide a deterministic procedure for making this decision in general. His implementation works correctly for cases where the decision of which and how parameters should be perturbed is straightforward, and it will not work for cases where the decision requires more sophisticated reasoning. Weld describes the three parts of this as follows:

1. Choose a parameter to exaggerate. This choice is obvious when the perturbation is differential in nature (i.e., question of the type 'what happens if the value of x is increased (or decreased)?') However, when the perturbation is of a different nature, such as a change in the physical configuration, there is no established procedure for the selection. Weld's implementation handles only differential perturbations.
2. Choose the direction in which the parameter should be perturbed. There is no well-defined procedure for selecting the direction, but there are only two possibilities, increase or decrease.
3. Select the final value for the perturbed parameter. In some cases, this decision is trivial, either infinity or infinitesimal depending on the direction of perturbation. However, there are cases for which this does not work. Weld states 'the parameter should be transformed to the shortest distance that causes some parameter to reach an infinite or infinitesimal value' without providing a general procedure for doing so.

$\texttt{Cond}_1$: None.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}$: An immediate effect of Exaggeration on the formulation is on its complexity, which is in $\texttt{Evals}_{\texttt{prob}}$. Though there is little syntactic change in the qualitative differential equations themselves, the shift to a nonstandard model increases the number of landmark values in the quantity space of each qualitative variable. For each landmark value l that is not $+inf$ or $-inf$ in the original quantity space of a variable, the new quantity space include $(HALO\, l-)$ and $(HALO\, l+)$[11] in addition to l. Therefore, the complexity of the formulation measured in terms of the size of the quantity space of variables increases as a result of the reformulation.

$\texttt{Evals}_{\texttt{p+r}}$: This increase in complexity of the quantity space results in a more complex predicted behavior. The state tree produced by QSIM from $\texttt{Form}_1$ and HR–QSIM from $\texttt{Form}_2$ are in general not identical because the former is operating in the space of standard numbers and the latter in the space of hyperreals, which results in a tree with a higher branching factor. Consequently, care must be taken to determine what differences in the predicted behavior are indicative of the effects of the perturbation and what are due to the difference between QSIM and HR–QSIM. The computational complexity of this scaling step is exponential in the depth of the behavior tree but Weld states that it does not in general constitute a problem since the constant factor is very low compared to that of simulation.

$\texttt{Evals}_{\texttt{strat}}$: Exaggeration can be evaluated as a strategy for comparative analysis with respect to its cost, its soundness, and the quality of explanations it provides ($\texttt{Evals}_{\texttt{strat}}$). The overall cost of the Exaggeration can be computed as the combination of the costs of the three steps. Although the reformulation step itself is tractable, both QSIM[12] and HR–QSIM[13] have a worst-case exponential complexity. In his extensive analysis and comparison of Exaggeration and Qualitative Differential Analysis [35], Weld points out that the inference made by the exaggeration method is unsound. This is not to say that HR–QSIM is unsound but that the inference made to answer the comparative analysis question by comparing the two formulations is unsound since it assumes that the system responds monotonically to perturbations, an assumption that does not necessarily hold. He also shows that while DQ analysis has many computational advantages over exaggeration, such as soundness and computational tractability, exaggeration produces more intuitive and simpler explanations.

[11] $(HALO\, l-)$ and $(HALO\, l+)$ are the notations that Weld used to represent the qualitative values corresponding to the left and right half of the halo of l. They are the sets of all nonstandard numbers that are infinitely close to l but that are less than or greater than l.

[12] Cost of the qualitative simulation by QSIM: In general, the time to produce the set of successor states for a given state is linear with respect to the size of the model. However, since qualitative simulation can suffer from intractable branching, the number of states produced for a given depth T of the tree can be exponential with respect to T.

[13] The cost of the qualitative simulation by HR–QSIM is similar to that of QSIM except that the branching factor of the tree can be higher than that of QSIM. (However, Weld states that intractable branching occurred only in a few pathological cases out of 50 problems tried.)

Discussion. One may question whether the transformation phase in Exaggeration should in fact be regarded as reformulation since nothing on the surface of the formulation changes. We claim that it is a bona fide example of reformulation since a profound transformation takes place in the semantics of the formulation.[14] Another example of reformulation involving a change in the underlying model of numbers is switching between discrete and continuous domains in interpreting equations. Approximating a discrete model by a continuous one (or vice versa) is a practice that is often useful but sometimes dangerous. Switching between standard and nonstandard models of numbers appears less dangerous by virtue of the transfer principle, which allows inferences made in the nonstandard universe to be applied to the standard universe,[15] but as the difficulties in the scaling phase of Exaggeration discussed by Weld suggest, care must be taken in interpreting the results obtained using nonstandard models.

5.3. Reformulation at the solution processing stage

In this section, we review two reformulation techniques that operate with `QSIM` to make its results more understandable. Given a set of qualitative constraints called qualitative differential equations (QDE) governing the behavior of a system, `QSIM` predicts its possible behaviors by iteratively generating a node that describes its qualitative state then using the qualitative constraints to infer the set of next qualitative states that could follow the current one [14]. The paths in the behavior tree correspond to possible behaviors of the dynamic system, and thus represent solutions to the qualitative differential equations. The biggest problem that limits the usefulness of a qualitative simulator such as `QSIM` is the complexity of the results produced. While the ability to predict behavior from an imprecisely specified model is quite valuable, the imprecision also leads to a high degree of ambiguity in the result. A typical `QSIM` output is a highly branching tree of states representing a large number of possible courses of behavior. Complexity makes it difficult for a person to extract useful patterns of behavior from such output.

The goal of the first example, discussed in Section 5.3.1, is to generate an explanation. The goal of the second one, discussed in Section 5.3.2, is to eliminate insignificant branching in qualitative simulation. The former fits in a straightforward fashion into our framework for reasoning about physical systems (Fig. 1): `QSIM` is used to solve the qualitative differential equations (Equation Processing Stage), and the reformulation procedure is applied to the generated behaviors (Solution Processing Stage). The problem solving process involving the second reformulation technique is more complex: the reformulation of the solution is interleaved with equation solving, producing a more compact behavior

[14] The surface similarity between the two formulations before and after transformation is a simple artifact of the practice in nonstandard mathematics to use the same set of symbols for arithmetic operators as those in standard mathematics.

[15] A general form of transfer principle can be stated informally as follows (from "Standard and Nonstandard Analysis" by R.F. Hoskins, Ellis Hoswood Limited, 1990): "Every properly formulated proposition with bounded quantifiers about standard objects [...], will be true in the standard universe if and only if the corresponding proposition (suitably re-interpreted if necessary) about their nonstandard counterparts [...] is true in the nonstandard universe".

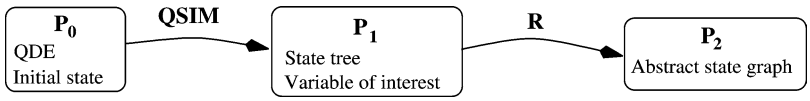

Fig. 15. Solution reformulation.

tree and enabling the simulation engine to carry out simulation further than otherwise possible.

5.3.1. Behavior abstraction for explanation

In [19], Mallory et al. make the results produced by QSIM easier to understand by summarizing along user-specified dimensions. Their goal is to support individual users in interpreting QSIM results. Fig. 15 illustrates their approach. Given the aspects of behavior the user is interested in (such as a combination of variable values and their directions of change), the reformulation procedure extracts a graph of abstract states from a QSIM state tree. The procedure examines the states in the original tree, discards unnecessary information from them, and aggregates adjacent nodes and arcs that are indistinguishable from the perspective specified by the user. The result is an abstract graph that reveals only information that the user wants to see. The result is also much smaller so that it can be easily inspected visually to identify significant patterns of behavior such as oscillation.

The problems, P_1 and P_2

$\texttt{Query}_1 = \texttt{Query}_2$: Predict the qualitative behavior of the given system over time.

$\texttt{Form}_1$ comprises:

1. A state tree produced by QSIM. Each node of the tree represents a qualitative state of the system and an arc a transition. Each path through the tree corresponds to a possible behavior over time.
2. A method for labeling nodes. This method specifies the dimensions of the behavior along which aggregation is to take place. A typical method may specify the values and directions of change of a subset of the variables, but it could potentially be any method for labeling.

$\texttt{Form}_2$: A graph of abstract states. Each node corresponds either to a node or to an aggregate of nodes in $\texttt{Form}_1$. Likewise, for the arcs. In addition, each node contains only information specified by the labeling method.

$\texttt{Assmptn}_1 = \texttt{Assmptn}_2$: The same assumptions underlying QSIM.

The reformulation technique, R

Proc comprises the following steps:

1. Label each node in the tree by the given labeling method.

2. Generate a graph by aggregating nodes that are adjacent and have the same labels. Arcs are also aggregated as necessitated by aggregation of the nodes. The exact conditions for merging nodes and arcs are specified in [19], but roughly speaking, two nodes are considered adjacent if "one is the successor of the other or if they share common immediate predecessors or successors".

`Cond`: None.

Evaluators and effects

$\texttt{Evals}_{\texttt{prob}}$, $\texttt{Compars}_{\texttt{prob}}$: The primary objective of Mallory's technique is to improve the understandability of the generated behavior by projecting it over the dimensions specified by the user, which also reduces significantly its size. With respect to `Size`, which is be measured by the number of nodes and arcs in the graph, the size of the abstracted behavior graph is guaranteed to be equal to or smaller than that of the original one. `Understandability` is more difficult to quantify, but one aspect of it is the ratio of the amount of information that is pertinent to the user's specified interest to the total amount of information in the formulation. This measure is also in $\texttt{Evals}_{\texttt{prob}}(P)$, and its value for P_2 is always 1 as the authors prove that the abstracted behavior graph is guaranteed to keep only those states pertinent to the query, while the value for P_1 is likely to be much smaller. Finally, as an observation with respect to understandability, the authors report that the user has to experiment with different specifications of the labeling method in order to reach a satisfactory summary of the behavior of interest.

$\texttt{Evals}_{\texttt{p+r+p}}$: The authors evaluate the reformulation process with respect to `Soundness` and `Completeness`. The authors define soundness to be that any path in the abstract graph should correspond to at least one path in the original tree, and completeness to be that any path in the original tree should correspond to some path in the abstract graph. They prove that an abstract graph produced by their procedure is always sound and complete with respect to the original tree.[16]

$\texttt{Evals}_{\texttt{p+r}}$: The reformulation procedure is evaluated with respect to its complexity, which is polynomial in this case with respect to the size of the state tree, assuming that the complexity of the labeling method is no more than polynomial with respect to the number of variables.

Discussion. Most solution reformulation techniques aim to facilitate the interpretation of complex results produced by other programs through additional processing such as summarizing, graphing, generating natural language explanations, and extracting specific aspects of the results [11]. The work discussed in this section is a clear example of such techniques, and though what is produced is another graph representing behaviors, it provides a good basis for generating high-level explanations. In fact, the authors state that their ultimate goal is to produce a natural language explanation of qualitative behavior.

The technique is based on a simple idea but the approach is quite general and allows a wide range of perspectives for summarizing since one can basically provide any piece of

[16] Thus, the procedure defines a surjective mapping between the original tree and the reformulated graph.

code as a method for labeling the nodes. It places the burden of specifying the perspective squarely on the user's shoulder, which may be inevitable since it is impossible to anticipate all the different perspectives from which users may wish to examine a given result.

5.3.2. Chatter elimination in qualitative simulation

In [3], Clancy and Kuipers address the problem of chatter in qualitative simulation. The phenomenon of chatter in qualitative simulation is one of the important factors that unnecessarily undermine the applicability qualitative simulation and complicate its results. Chatter happens when the direction of change in a variable is constrained only by continuity. Since variables in such a situation are free to alternate among the three possible directions of change, namely increasing (`inc`), steady (`std`), and decreasing (`dec`), this may lead to intractable branching even though such branching reveals no significant information about the possible behaviors. The situation can be even worse since any number of variables may be unconstrained. Branching due to chatter may cause the size of the behavior tree to become infinite, limiting the range of behaviors that can be explored by `QSIM` [14]. Clancy and Kuipers propose two techniques, Chatter Box Abstraction and Dynamic Chatter Abstraction, each of which can be used to eliminate all occurrences of chatter in the behavior tree. We focus on Chatter Box Abstraction in our discussion since it can be more clearly viewed as reformulation than Dynamic Chatter Abstraction.

To eliminate chatter during `QSIM` simulation, Chatter Box Abstraction is carried out every time simulation enters a potentially chattering state. Simulation is first suspended, analysis is carried out to determine the chattering variables, and abstract states are inserted into the behavior tree, before simulation is resumed. Fig. 16 illustrates the strategy of alternating simulation and reformulation as a path leading from P_0 to P_n. As a result of Chatter Box Abstraction, regions of a behavior tree that can include a large number of chattering states and, thus, potentially represent an infinite number of distinct behaviors, are replaced by abstract nodes, extending the range of models that can be simulated by `QSIM`.

The problems, P_1 and P_2

$\texttt{Query}_1 = \texttt{Query}_2$: Predict the behaviors of the physical system.

$\texttt{Form}_1$ comprises:

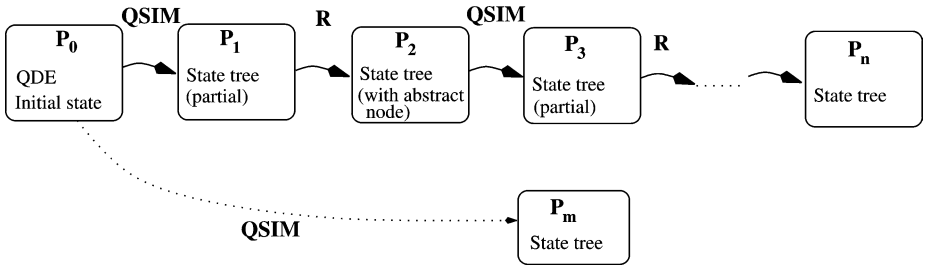

Fig. 16. Chatter elimination.

1. A set of Qualitative Differential Equations (QDEs).
2. A partial behavior tree in which one of the states indicates the possibility of chattering. (The condition for detecting this possibility is given below as the condition of the reformulation procedure.)

$\texttt{Form}_2$ comprises:

1. The same set of Qualitative Differential Equations (QDEs).
2. A partial behavior tree that is the same as above except for new abstract states that replace the chattering states and their successors.

$\texttt{Assmptn}_1 = \texttt{Assmptn}_2$: The same assumptions underlying QSIM.

The reformulation technique, R

Before simulation starts, the constraints in the QDE are analyzed to identify the variables that can potentially chatter and those that can never chatter throughout simulation. Variables that can chatter are further grouped into sets called *chatter equivalency classes*, such that if any one variable in a class starts to chatter, all the remaining variables in the class will also chatter.

Proc: The reformulation procedure consists of the following two steps. The first step determines which variables are actually chattering, and the second step creates abstract states when chattering is detected.
   1. Determination of variables that exhibit a chattering behavior is accomplished by a recursive call to QSIM to explore a limited region of the state space. In the recursive call, only the directions of change of the potentially chattering variables are allowed to change. The result is analyzed for existence of chatter. Chattering is detected if there are cycles in the graph of states that are only distinguished by the directions of change in some variables.
   2. Once chattering is detected, the states involved in the cycles are summarized into one abstract state, where the direction of change of each chattering variable is represented by a list of possible directions (inc, std, dec). The successor states (i.e., the states that are successors of the states included in the cycles but are not themselves in any cycle) are also abstracted in the same manner. In other words, if any successor states are different from each other only in the direction of change in the chattering variables, they are summarized into one abstract state. Abstract states are inserted into the original behavior graph and simulation resumes.

Cond: The reformulation procedure is triggered when a potentially chattering variable is changing within a time-interval state.

Evaluators and effects

For the purpose of assessing the benefits the reformulation, what should be evaluated in the case of Chatter Box Abstraction is not the changes brought about by individual deployment of the reformulation procedure but the cumulative effects of repeated deployment

on the final result as well as on the computational effectiveness. Thus, we must compare P_n and P_m as well as the strategies represented by the paths leading to them from P_0 in Fig. 16.

$\texttt{Evals}_{\texttt{prob}}$, $\texttt{Compars}_{\texttt{prob}}$: The main effect of Chatter Box Abstraction is on the compactness of the state tree. Compactness can be measured in terms of the number of distinct behaviors (i.e., paths through a state tree). Clancy and Kuipers empirically evaluate the effect of the technique with respect to this measure and demonstrate 1 to 3 orders of magnitude reduction in the number of behaviors predicted (Table 1 in [3]). Such reduction is significant in itself, but even more significant is the fact that it makes the result of simulation by QSIM much easier for a person to comprehend. Cognitive transparency is also in $\texttt{Evals}_{\texttt{prob}}$, and even though it is difficult to quantify, it is an important aspect of a formulation that could determine the usefulness the overall problem solving effort. In addition, Clancy and Kuipers evaluate the abstracted tree P_n by its correctness with respect to P_m. They guarantee that the set of trajectories in P_n is equal to the set in P_m with respect to the non-chattering variables and that it is a super-set with respect to the chattering variables.

$\texttt{Evals}_{\texttt{p+r}}$: The evaluator here is the computational complexity of Proc applied to Form. Each time Proc is invoked, the number of states explored in step 1 of the procedure is exponential in the worst case with respect to the number of chattering variables.

$\texttt{Evals}_{\texttt{strat}}$: By eliminating the behaviorally insignificant distinctions among states, Chatter Box Abstraction cuts down on the branching factor of the behavior tree and enables QSIM to explore a much larger portion of the state space than otherwise possible for given computational resources. Interestingly, the whole computational process of simulation remains intractable with or without reformulation. However, Chatter Box Abstraction improves computational effectiveness by enabling QSIM to avoid the futile effort of expanding infinite sequences of chattering states, thus using resources more efficiently. One way to measure computational effectiveness is would be to measure the ratio of the amount of time wasted on expanding chattering states versus the overall processing time.

Discussion. Unlike Chatter Box Abstraction, discussed above, Dynamic Chatter Abstraction [3] does not make recursive calls to QSIM to determine the chattering variables. Instead, it relies on the current state and the constraints to determine if the current state can lead to chattering. Dynamic Chatter Abstraction is more tightly integrated into the qualitative simulation process, and though the algorithm of Dynamic Chatter Abstraction is still exponential with respect to the number of chattering variables, the experimental results reported by the authors show that the it can outperform Chatter Box Abstraction by a factor of 4 to 10. Though it is devised to accomplish the same goal as Chatter Box Abstraction, Dynamic Chatter Abstraction basically eliminates the need to reformulate by spending more resources analyzing the current state and the constraints to avoid generating chattering states.

Chatter Box Abstraction and Dynamic Chatter Abstraction along with the behavior abstraction technique discussed in Section 5.3.1 present an interesting spectrum of techniques for reducing complexity. On one extreme, behavior abstraction tries to simplify after a large complex result is produced. On the other extreme, Dynamic Chatter Abstraction tries to avoid producing unnecessarily complex results in the first place by doing more analysis during simulation. Chatter Box Abstraction is somewhere in between as it interleaves simulation and reformulation, trying to limit the area of behavior space that needs to be reformulated at any one time.

5.4. Discussion

Reformulation techniques vary widely with respect to their goals, the types of formulations they operate on, their actual effects on these formulations, their cost, etc. Having reviewed a variety of reformulation techniques, we now summarize our findings and present general observations. Table 2 summarizes five key aspects of the reformulation techniques examined in this paper. The first column indicates the purpose of a technique, which is the ultimate goal of the authors of the technique. The second column indicates the type of formulation to which each technique is applicable. The effect column describes the principle change brought about by reformulation. The procedure column summarizes the major steps for realizing the change. The last column lists the key information, besides a representation of the system being modeled, that each technique relies on in order to carry out the steps. In addition, the table indicates the reasoning stage at which each technique is applied.

5.4.1. Purpose of reformulation

The purposes listed in Table 2 for each technique are the ones stated by its authors. In general, the purpose of a reformulation falls in one of two categories; namely, facilitating human understanding and facilitating problem solving. The former category encompasses the techniques that seek better explanations and cognitive insight, and the latter includes those aimed at reducing computational cost and making hitherto unsolvable problems into solvable ones. The goals of these two categories are not mutually exclusive and some reformulation techniques actually achieve the objectives of both categories. For example, a technique that reduces the size of a formulation may decrease the cost of finding the solution and make the formulation easier to understand at the same time. Exaggeration is an oddball among all the techniques. Its purpose, which is to compare the behavior of a reformulated model with that of the original one, does not fall in either category. We are not aware of other examples of this type although they may exist.

We must point out that there is sometimes a leap of faith between the purpose of a reformulation technique as stated by its authors and what it actually achieves. This is especially true in cases where improvement of human understanding is the claimed purpose of reformulation. None of the works we surveyed actually demonstrates improved understandability of the resulting formulation. Those that aim to generate better explanations do not actually generate explanations, but it is hoped that their reformulation improves the quality of the explanations to be generated.

Table 2
Summary of the reformulation technique

Technique	Purpose	Formulation type	Effects	Procedure	Key information	Stage
Model Simplification (Section 6.1.1)	Better explanation	Model fragments	More approximate model	Replace by approximate model fragments	Causal relation of interest. Assumption classes.	Model
Critical Abstraction (Section 6.1.2)	Better explanation	Components, Behaviors, Interactions	Less equations/ variables	Aggregate paths. Eliminate parts	Causal relation of interest	
Aggregation (Section 6.2.1)	Cost reduction	Equations	Less equations/ variables	Decompose. Aggregate	Threshold value ε	Equation
Linearization (Section 6.2.2)	Solvability	Equations	Linear equations	Focus. Replace by simpler model	Equilibrium point	
Caricatural Reasoning (Section 6.2.3)	Solvability & cognitive insight	Equations	Lower degree for equations	Decompose. Exaggerate	Given variables	
Exaggeration (Section 6.2.4)	Comparative analysis	Equations	Nonstandard model that allows exaggeration of values to extreme	Replace the underlying semantics	Disturbed variable and variable of interest	
Behavior Abstraction (Section 6.3.1)	Better explanation & cognitive insight	State graph	Smaller behavior graph	Project. Aggregate (after simulation)	Labeling method	Solution
Chatter Elimination (Section 6.3.2)	Cost reduction & cognitive insight	State graph	Smaller behavior graph	Aggregate (during simulation)		

5.4.2. Effects of reformulation

The effects of reformulation are the actual changes a particular technique brings about in a formulation, distinct from the stated purpose discussed above. In the discussion of each technique, we have identified what and how aspects of formulations are changed in terms of a set of evaluators and comparators. The most common effect in all techniques, except Exaggeration, is what one might generally call 'simplification'. However, as we argued in Section 2.3, this labeling is insufficient since the exact metrics of simplicity varies in every case. We list below factors of simplicity used in different techniques. The precise definition of simplicity sometimes involves multiple factors. For example, the definition of simplicity in Model Simplification relies both on the knowledge of approximation relations as well as that of subset relations of causal orderings among variables.

- Simplicity based on the size of the formulation:
  - Fewer variables, equations, and/or causal orderings.
    Examples: Critical abstraction, Aggregation, Model simplification.
  - Fewer states and transitions.
    Behavior abstraction, Chatter elimination.

- Simplicity based on the form of equations:
  - Linear vs. nonlinear.
    Example: Linearization.
  - Degree of equations.
    Example: Caricatural reasoning.
- Simplicity based on the concept of approximation:
  - Example: Model simplification.[17]

5.4.3. Reformulation procedure

While reformulation procedures are varied, some general types of manipulation emerge from the examples, which we will call here *generic methods*. The generic methods we have observed in our examples are replacement, decomposition, aggregation, and focusing. A reformulation procedure often consists of the application of one or more generic methods:

Replacement: Examples are Model simplification, Linearization, Exaggeration. The reformulation consists in replacing a part or the entire formulation by another formulation. The alternative used must have a well-defined relationship to the original but may not have been generated by the reformulation technique itself (e.g., model simplification).

Decomposition: Division of the entire problem space into smaller ones. Examples are Aggregation and Caricatural reasoning.

Aggregation: Parts that are "adjacent" according to some criterion are merged. Examples are Aggregation, Critical abstraction, Behavior abstraction, and Chatter elimination.

Focusing: Removing parts of the formulation from consideration to emphasize the remaining parts. Examples are Critical abstraction, Behavior abstraction, and Linearization. The labeling method of Behavior abstraction provides the projection mechanism to filter out uninteresting information about each component. Linearization focuses on the area around the equilibrium points, while ignoring everything else.

5.4.4. Key information

Every reformulation is guided by a specific query. All the techniques discussed, except Chatter elimination, require some additional information, beyond the representation of the system modeled, to address only the aspect of the formulation that is relevant to the query. Three techniques (i.e., Model simplification, Critical abstraction, and Exaggeration) require the specification of the variables of interest. Model simplification and Critical abstraction require also the specification of causal relations among these variables. Linearization requires the selection of an equilibrium point. The labeling method of Behavior abstraction is a procedural way to specify one's interest. Aggregation requires the specification of a threshold value (ε), which allows one to focus on mid- to long-term behavior.

[17] The approximation relations among model fragments are stated a priori by the builder of the knowledge base.

In Chatter elimination, it is implicit that the chattering behavior is uninteresting and needs to be eliminated.

5.4.5. Processing stage

Since reformulation is simply one step in the larger endeavor of reasoning about physical systems, it is important to localize where a particular reformulation technique fits in the entire problem-solving process. Doing so enables one to discover other reformulation opportunities for achieving the same purpose and to explore the trade-offs of reformulating at different stages.

On the one hand, reformulating at an early stage has the advantage of having one model-equations-solution sequence where the relations between the formulations at consecutive steps are clear. Indeed, if one chooses to reformulate at a later stage, the causal relations between the original model and the end result of the physical system becomes more complex, less direct, and can be harder to explain.

On the other hand, if the primary goal of applying a reformulation is to affect the performance of a particular reasoning stage, it may be preferable to reformulate the encoding closest to the targeted reasoning stage. Reformulating at an earlier stage may prevent us from controlling the effects of reformulation we are seeking as precisely as we would wish.

For example, Model Simplification (Section 5.1.1), which reformulates a model, results in a consistent model-equations-solution sequence that will be easy to trace and explain. However, if the primary goal were to produce equations that are easier to solve or solutions that are less complex to interpret, the benefits of Model Simplification could be less predictable.

6. Related work

Various general theories of reformulation, including abstraction and approximation, have been proposed in the literature. Some of these theories provide an encompassing high-level characterization. Others restrict their scope to some specific aspect (e.g., cost or faithfulness of results). These theories proved to be essential to our understanding of reformulation, but we found them to be of limited practical use in automating the selection of reformulation techniques.

Giunchiglia and Walsh [10] introduced a general theory of abstraction applicable to theorem proving in formal systems, and illustrated their theory in planning and commonsense reasoning problems. They introduced a formal classification of reformulation and its properties, as well as highlighting their use for both provability and refutation. Importantly, their theory also encompasses those techniques that may yield inconsistencies. Giunchiglia and Walsh acknowledged that their terminology does not capture all desirable properties of abstraction, such as the upward and downward-solution properties proposed by Tenenberg [34] based on the structure of the proof tree in theorem proving. Plaisted [25], Cremonini et al. [4], and Nayak and Levy [22] also explored abstraction theories that are restricted to logical systems and to techniques that preserve consistency and correctness of proofs.

Each of these general theories of abstraction deals with only one, mainly logical, aspect of reformulation and addresses only a few aspects of change that can be cleanly character-

ized formally. We find that those logical theories are often not sufficient, and sometimes not useful, for characterizing the types of reformulation used by engineers or those proposed in the literature about model-based reasoning. Another serious problem with those logical theories is that they ignore the relevance of problem-solving goals of reformulation, which we believe is crucial in understanding why and how aspects of a formulation are affected by reformulation.

The goal-directed nature of reformulation is addressed by Weld and Addanki in [37,39]. Weld and Addanki [39] take a task-driven approach to reformulation and adopt Tenenberg's vocabulary [34] for describing the effects of the reformulation on the formulation. In his paper on approximation reformulations [37], Weld discusses the idea of switching between two models[18] in order to find the model whose predictions best match some observations. In this work, models are organized as nodes in a graph, called a graph of models (GoM), whose edges represent the approximation relations among models obtained by the elimination of one assumption. These works are more relevant to reasoning about physical systems than the general, logical theories of abstraction, since many, but not all, reformulations used by engineers can be seen as approximations. However, they only deal with only one type of reformulation, namely approximation. In addition, the goal of approximation reformulations is not to *dynamically* reformulate, but rather to exploit a fixed set of models and identify the one best that satisfies given criteria.

None of the theories discussed so far makes extensive analysis of complexity issues, nor do they provide the terminology for quantitatively evaluating the effects of reformulation. In contrast, the body of research on approximations in the computational complexity community [24], provides rigorous evaluation criteria with respect to cost. More specifically, the reformulation procedure is constrained to logarithmic space mappings, and the goal pursued is strictly the improvement of the computational cost, while providing a guaranteed bound on the quality of the result.[19] While focusing on the computational aspects, these approaches are not concerned with the important aspects of representation and expressiveness.

Our work comes closest in spirit to [5], where Davis studies approximation and abstraction and focuses on the practical application of reformulation techniques applied to reasoning about solid object kinematics. Davis also stresses that the selection of the reformulation technique must be task-driven, in order to satisfy some well-defined criteria. Our work goes further in providing a practical framework for characterizing and comparing many different aspects of reformulation techniques in the context of a specific type of reasoning about physical systems.

7. Conclusions

In this paper we have provided a practical framework for characterizing, evaluating, and selecting reformulation techniques in the context of reasoning about physical systems.

[18] In his paper, a model is a set of equations.

[19] In our terminology, this bound can be represented as a comparator the output of two strategies, the one that computes the exact result and the one that approximates it.

Our framework allows one to express the relevant quantitative and qualitative aspects of a reformulation process, whereas previous theories addressed either logical properties or computational aspects. Another important feature of our framework is that it requires one to explicitly state the effects of reformulation in terms of *what* and *how* aspects of the formulation are changed, which is essential in the selection and evaluation of reformulation techniques with respect to a given goal.

Using our framework, we have presented an analysis of eight examples of reformulation techniques applicable at various stages of reasoning. Though we have limited ourselves to these eight examples in this paper, we are aware of the vastness of the space of reformulation techniques in reasoning about physical systems. In mathematics, one transforms geometry problems into algebra and vice versa, real analysis into non-standard analysis, number theory into set theory, and so forth. In engineering, one transforms a time domain into a frequency domain, continuous problems into discrete problems, and so forth. While all the examples in this paper have been discussed in the AI literature, their choice does not reflect our belief that they are somehow the most common or the most important. Rather, they were chosen because the set represents breadth in terms of the stages of application, and also because descriptions of their goals, algorithms, along with examples of their execution were readily available to us. This is central to this work since our goal is to characterize reformulation techniques as a specific computational procedure with well-defined scope and effects. Furthermore these examples demonstrate the generality of our framework since they originate from and are useful in a variety of science and engineering fields. Both our framework and our narrow focus on the process of reasoning as set out in Section 3 were essential in undertaking this study since a meaningful comparison of such a diverse set of techniques would have been impossible without a uniform platform.

From our investigations, we have learned that characterizing a reformulation technique in terms of *what* aspects of a formulation are affected and *how* these aspects are modified reveals the hidden assumptions underlying the technique. It is essential to uncover and clearly articulate these hidden assumptions since they determine which aspects of the original formulation are targeted by reformulation, which is a prerequisite for automation. We have learned that it is important to place the reformulation step in the context of the overall process of reasoning about physical systems in order to precisely predict to what extent the reformulation step contributes to realizing the goal of the problem-solving task. Indeed, we have noticed that there is often a leap of faith between the stated goal of a reformulation technique as presented in a technical paper and how well it achieves the goal. We have also observed that, despite the diversity of the reformulation procedures that we have studied, four distinct types of manipulation emerged as generic methods of reformulation.

Our analysis of the eight examples has also shed light on the question of how to select a reformulation method in a specific context and for a specific goal. The selection and design of a reformulation method requires the consideration of the following issues:

1. What is the goal of the reformulation?
2. What effect should the reformulation have on a formulation of the problem in order to achieve the goal? In other words, which aspect of the original formulation makes it difficult to achieve the desired goal? Which components and what aspects need to be changed and how?

3. At what stage of the problem-solving process is it most effective to reformulate in order to achieve the goal?
4. What information, beyond the representation of the system being manipulated, is available to help focus the reformulation?
5. What cost are we willing to pay for reformulation?

Reformulation is ubiquitous and techniques for reformulation are varied. Given a broad interpretation of reformulation as any change in the formulation of a problem, every problem-solving step can be conceived as a reformulation. Nevertheless, in the practice of reasoning about physical systems, it is useful to distinguish reformulation from other problem solving steps that make explicit the implications of an existing formulation. Thus, in the work presented here, we have made the distinction. We were able to do so by restricting the context of reformulation to a particular type of reasoning about physical systems with three distinct stages. We believe that restricting the context and narrowing our field has enabled us to lay out a concrete and practical framework for informed comparison of reformulation techniques that are relevant in reasoning about physical systems.

While our work is still one step away from a formal specification of a language for representing attributes of reformulation to enable automatic selection, we believe that the analysis of techniques we present here identifies and articulates the features essential to such automatic selection. There are several directions in which this work can be extended:

- Develop a method for automatically selecting and composing reformulation techniques, relative to a given goal.
- Identify a reasoning task for which a multitude of reformulation techniques exists. Design and implement a system that consists of a formal representation of the task and an encoding of these techniques together with an extended set of evaluators. Use this system to validate our framework and to test its robustness under a variety of conditions to be selected.
- Extend our framework to encompass engine reformulation. We have deliberately ignored in our research all reformulations that apply to the engine[20] itself (be it a heuristic reasoner, or formal inference rules), because such reformulations do not seem to arise in the class of problems we address. However, this may not hold for the most general case, and one may have to examine this issue more closely and decide whether more attributes are indeed required.

Acknowledgements

This work emerged from the discussions of a reading group on reformulation at the Knowledge Systems Laboratory of Stanford University. The group also included William Buchanan, Robert S. Engelmore, Richard Fikes, Tony Loeser, and Todd Neller. We would

[20] In [10], Giunchiglia and Walsh consider the deductive machinery, denoted Δ, to be part of the formal system to which the reformulation is applied. They distinguish between Δ-variant and Δ-invariant reformulation depending on whether or not the same inference engine is used to solve both the original and reformulated problems.

like to thank Ernest Davis, Alon Y. Halevy, and the anonymous reviewers for their constructive comments. Berthe Y. Choueiry was supported by a fellowship for advanced researchers from the Swiss National Science Foundation. Yumi Iwasaki was supported in part by DARPA and the National Institute for Standards and Technology, Rapid Development, Exploration and Optimization (RaDEO) program, under cooperative agreement 70NANBH0075. Sheila McIlraith was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and by Xerox Palo Alto Research Center.

References

[1] S. Addanki, R. Cremonini, J. Penberthy (Eds.), Reasoning about Assumptions in Graphs of Models. Readings in Qualitative Reasoning about Physical Systems, Morgan Kaufmann, San Mateo, CA, 1990.
[2] F.G. Amador, D.S. Weld, Multi-level modeling of populations, in: 4th International Workshop on Qualitative Physics, Lugano, Switzerland, 1990.
[3] D.J. Clancy, B. Kuipers, Static and dynamic abstraction solves the problem of chatter in qualitative simulation, in: Proc. of AAAI-97, Providence, RI, 1997, pp. 118–125.
[4] R. Cremonini, K. Marriott, H. Søndergaard, A general theory of abstraction, in: Proc. of the 4th Australian Joint Conference on Artificial Intelligence, Australia, 1990, pp. 121–134.
[5] E. Davis, Approximation and abstraction in solid object kinematics, TR706, 1995.
[6] B. Falkenhainer, K.D. Forbus, Compositional modeling: finding the right model for the job, Artificial Intelligence 51 (1991) 95–143.
[7] A. Farquhar, A qualitative physics compiler, in: Proc. of AAAI-94, Seattle, WA, 1994, pp. 1168–1174.
[8] K.D. Forbus, Qualitative process theory, Artificial Intelligence 24 (1984) 85–168.
[9] K.D. Forbus, The qualitative process engine, in: D.S. Weld, J. de Kleer (Eds.), Readings in Qualitative Reasoning about Physical Systems, Morgan Kaufmann, San Mateo, CA, 1990, pp. 220–235.
[10] F. Giunchiglia, T. Walsh, A theory of abstraction, Artificial Intelligence 57 (1992) 323–389.
[11] T.R. Gruber, P.O. Gautier, Machine-generated explanations of engineering models: a compositional modeling approach, in: Proc. of IJCAI-93, Chambéry, France, 1993, pp. 1502–1508.
[12] Y. Iwasaki, I. Bhandari, Formal basis for commonsense abstraction of dynamic systems, in: Proc. of the National Conference on Artificial Intelligence (AAAI-88), Saint Paul, MN, Morgan Kaufmann, San Mateo, CA, 1988.
[13] C.A. Knoblock, J.D. Tenenberg, Q. Yang, Characterizing abstraction hierarchies for planning, in: Proc. of AAAI-91, Anaheim, CA, 1991, pp. 692–697.
[14] B. Kuipers, Qualitative simulation, Artificial Intelligence 29 (1986) 289–338.
[15] A.Y. Levy, Y. Iwasaki, R. Fikes, Automated model selection for simulation based on relevance reasoning, Artificial Intelligence 96 (1997) 351–394.
[16] R. Ling, L. Steinberg, Automated modeling in computational heat transfer, in: Proc. of the AAAI Fall Symposium on Scientific Modelling and AI, Cambridge, MA, 1992.
[17] C.M. Low, Y. Iwasaki, Model generation and simulation of device behavior with continuous and discrete changes, Intelligent Systems Engineering 1 (2) (1992) 115–145.
[18] D.G. Luenberger, Introduction to Dynamic Systems: Theory, Models, and Applications, 1979.
[19] R.S. Mallory, B.W. Porter, B.J. Kuipers, Comprehending complex behavior graphs through abstractions, in: Tenth International Workshop on Qualitative Physics, Fallen Leaf Lake, CA, 1996, pp. 137–146. AAAI Technical Report WS-96-01.
[20] P.P. Nayak, Causal approximations, Artificial Intelligence 70 (1994) 277–334.
[21] P.P. Nayak, L. Joskowicz, Efficient compositional modeling for generating causal explanations, Artificial Intelligence 83 (1996) 193–227.
[22] P.P. Nayak, A.Y. Levy, A semantic theory of abstractions, in: Proc. of IJCAI-95, Montréal, Québec, 1995, pp. 196–203.
[23] T. Nishida, K. Mizutani, S. Doshita, Automated analysis of qualitative behaviors of piecewise linear differential equations, New Gen. Comput. 11 (2) (1993) 159–177.

[24] C.H. Papadimitriou, Computational complexity (1994) 299–328.
[25] D.A. Plaisted, Theorem proving with abstraction, Artificial Intelligence 16 (1981) 47–108.
[26] A. Pos, Automated Redesign of Engineering Models, The Dutch Graduate School for Information and Knowledge Systems, 1997.
[27] O. Raiman, Order of magnitude reasoning, Artificial Intelligence 51 (1991) 11–38.
[28] J. Rickel, B. Porter, Automated modeling for answering prediction questions: selecting the time scale and system boundary, in: Proc. of AAAI-94, Seattle, WA, 1994, pp. 1191–1198.
[29] J. Rickel, B. Porter, Automated modeling of complex systems to answer prediction questions, Artificial Intelligence 93 (1997) 201–216.
[30] E.P. Sacks, Automatic qualitative analysis of dynamic systems using piecewise linear approximations, Artificial Intelligence 41 (1990) 313–364.
[31] H.A. Simon, The architecture of complexity, nearly decomposable systems, in: The Sciences of the Artificial, Cambridge, MA, 1969, pp. 99–101.
[32] H.A. Simon, A. Ando, Aggregation of variables in dynamic systems, Econometrica 29 (1961).
[33] P. Struss, On temporal abstraction in qualitative reasoning (a preliminary report), in: Proceedings of the Seventh International Workshop on Qualitative Reasoning about Physical Systems, Orcas Island, WA, 1993, pp. 219–227.
[34] J.D. Tenenberg, Inheritance in automated planning, in: First International Conference on Knowledge Representation and Reasoning, Toronto, Ontario, 1989, pp. 475–485.
[35] D.S. Weld, Comparative analysis, Artificial Intelligence 36 (1988) 333–373.
[36] D.S. Weld, Approximation reformulations, in: Proc. of AAAI-90, Boston, MA, 1990, pp. 407–412.
[37] D.S. Weld, Exaggeration, Artificial Intelligence 43 (1990) 311–368.
[38] D.S. Weld, Reasoning about model accuracy, Artificial Intelligence 56 (1992) 255–300.
[39] D.S. Weld, S. Addanki, Task-driven model abstraction, in: 4th International Workshop on Qualitative Physics, Lugano, Switzerland, 1990, pp. 16–30.
[40] B.C. Williams, Critical abstraction: generating simplest models for causal explanation, in: Proceedings of the Fifth International Workshop on Qualitative Reasoning about Physical Systems, Austin, TX, 1991, pp. 77–92.
[41] B.C. Williams, O. Raiman, Decompositional modeling through caricatural reasoning, in: Proc. of AAAI-94, Seattle, WA, 1994, pp. 1199–1204.