Automated modeling of complex systems to answer prediction questions

Jeff Rickel [1]

Information Sciences Institute and Department of Computer Science, University of Southern California, 4676 Admiralty Way, Marina del Rey, CA 90292, USA

Bruce Porter

Department of Computer Sciences, University of Texas at Austin, Austin, TX 78712-1188, USA

A question about the behavior of a complex, physical system can be answered by simulating the system — the challenge is building a model of the system that is appropriate for answering the question. If the model omits relevant aspects of the system, the predicted behavior may be wrong. If, on the other hand, the model includes many aspects that are irrelevant to the question, it may be difficult to simulate and explain. The leading approach to automated modeling, "compositional modeling," constructs a simplest adequate model for a question from building blocks ("model fragments") that are designed by knowledge engineers. This paper presents a new compositional modeling algorithm that constructs models from simpler building blocks — the individual influences among system variables — and addresses important modeling issues that previous programs left to the knowledge engineer. In the most rigorous test of a modeling algorithm to date, we implemented our algorithm, applied it to a large knowledge base for plant physiology, and asked a domain expert to evaluate the models it produced.

Key words: automated modeling, reasoning about physical systems, large knowledge bases

1 Introduction

Biologists, ecologists, doctors and engineers share an important skill: each has a deep understanding of a class of complex physical systems, and each

[1] Corresponding author. Fax: (310) 822-0751, E-mail: rickel@isi.edu.

Preprint submitted to Elsevier Preprint *25 November 1996*

can construct and simulate models of these systems to predict the system's response to hypothetical conditions. This skill is required for many tasks, such as evaluating designs and control strategies, predicting the effects of trends (e.g., global warming), testing diagnostic hypotheses, and teaching. While there are well-developed methods for *simulating* models, research on *constructing* models automatically is still in its early stages. Because model construction requires expertise and is often time consuming and error prone, our objective is to automate the modeling task: given domain knowledge (i.e., knowledge of how a complex system works) and a prediction question (i.e., hypothetical conditions and some variables of interest), construct the simplest model of the system that can adequately predict and explain the behavior of the variables of interest.

Current modeling programs shift important modeling decisions to the knowledge engineer. For example, some early programs required a knowledge base of all potentially useful models of the physical system (the "graph of models" approach [1]). These programs perform a relatively easy task: they select, but do not generate, the best model for answering each question. To answer questions about complex systems, this approach is impractical because the knowledge engineer cannot anticipate – let alone, build – all the models required for a wide range of questions. The set of models grows combinatorially with the number of phenomena in the system and the various levels of detail with which each phenomenon can be modeled.

Recent modeling programs take a more practical approach, called "compositional modeling" [12]: the domain knowledge provides models of different aspects of the system ("model fragments"), and the modeling program uses them as building blocks to construct an appropriate model for each question. To build an appropriate model, the program typically faces many difficult decisions. From all the phenomena governing the system's behavior, the program must select, and include in the model, only those that are relevant to the question. If it omits relevant phenomena, the model's predictions will be unreliable; on the other hand, if it includes many irrelevant phenomena, the model might be difficult to simulate and understand. In addition to selecting relevant phenomena, the program must chose an appropriate level of detail for each one. For example, the process of photosynthesis can be described as a single chemical reaction or as a complex sequence of more-detailed reactions (each of which could be similarly decomposed). Because compositional modeling programs automatically choose relevant phenomena and levels of detail for each question, the knowledge engineer need not anticipate and build all the models that might be needed.

Although the compositional modeling approach simplifies the task of encoding domain knowledge, current programs still shift important modeling decisions to the knowledge engineer. First, the knowledge engineer must design the model fragments; that is, he must group domain facts into coherent, indivisible bundles that the program can use as building blocks for constructing models. Second, he must supply most of the criteria for making modeling decisions: he must represent the assumptions underlying each model fragment, the dependencies and incompatibilities among assumptions, and the conditions that require choosing from among different modeling assumptions. Because this is knowledge about constructing models, not about how a physical system works, it is not readily available from domain experts.

This paper describes a new compositional modeling algorithm that does not require such knowledge. Our algorithm constructs models from simple building blocks – the individual variables of the physical system, and the influences [15] among them – and addresses the modeling issues that previous programs left to the knowledge engineer. To address these issues, our algorithm uses novel, domain-independent criteria that define when a model is adequate for answering a particular prediction question and when it is simpler than alternative models. (See Section 3.) With these criteria, a modeler can make decisions while knowing little more than the variables and influences that govern a physical system; the criteria demonstrate the central role of variables and influences in every modeling decision. We prove that our modeling algorithm will build a simplest adequate model (as defined by the criteria) for each prediction question, assuming that one can be built from the building blocks provided by the domain knowledge. (See Section 4.)

We implemented our modeling algorithm in a program called TRIPEL.[2] In addition, we integrated TRIPEL with a qualitative simulation program (the Qualitative Process Compiler [14]), which simulates TRIPEL's models to generate predictions.[3] Our goal is to combine the pieces needed to fully automate the task of answering prediction questions.

We evaluated our algorithm by applying TRIPEL to the task of answering prediction questions in the domain of plant physiology. (See Section 5.) While previous modeling programs have only been tested on examples constructed by their designers, our evaluation is considerably more rigorous, in three ways. First, the domain knowledge was encoded by a botany expert. His goal was to encode fundamental textbook knowledge that can support a wide range of tasks, not just prediction. (In fact, the same knowledge base has been used successfully for other tasks, such as answering description questions and generating English text [31,33,32].) Second, the domain knowledge he encoded

[2] The name TRIPEL is an acronym for "Tailoring Relevant Influences for Predictive and Explanatory Leverage." It is also a style of strong ale made by Trappist Monks in Belgium.

[3] Although TRIPEL has only been used to construct qualitative models [53], we believe that our modeling algorithm is equally capable of building numerical models, consisting of algebraic equations and ordinary differential equations. (See Section 6.)

is extensive: it describes 700 properties of a prototypical plant and 1500 influences among them, including many different levels of detail. Finally, the questions used to evaluate TRIPEL were produced by the botany expert, who judged TRIPEL's models by comparing them to his own models for answering the questions. Our goal is to build a modeling program that is sufficiently robust to answer unanticipated questions using large knowledge bases built by domain experts.

The evaluation identified the most important topics for future research. In particular, it showed that, for some modeling decisions, the expert uses more sophisticated criteria than TRIPEL uses. TRIPEL is designed to easily incorporate new criteria: the criteria for each type of modeling decision are encapsulated in an independent module of TRIPEL, and each module can be improved without requiring other changes to the algorithm.

To lay the groundwork for these topics, the next section describes the input to our modeling algorithm.

2 System Descriptions and Prediction Questions

Our modeling algorithm requires two inputs: domain knowledge about how some physical system works (the *system description*), and a prediction question about the system. The following question, from the domain of plant physiology, illustrates the general form of a prediction question: "How would decreasing soil moisture affect a plant's transpiration[4] rate?" A prediction question poses a hypothetical scenario, consisting of a physical system (e.g., a plant and its soil) and some driving conditions (e.g., decreasing soil moisture), and asks for the resulting behavior of specified variables of interest (e.g., the plant's transpiration rate). The system description for the example would describe the variables and influences that govern the plant and its soil.

2.1 System Descriptions

A system description represents all available domain knowledge about a particular system. Although a system description could be provided to the modeler directly, it is typically generated from general domain knowledge and a description of the physical structure of the particular system [12]. For example, given the physical structure of a particular chemical processing factory, general knowledge of chemical engineering could be used to generate a system descrip-

[4] Transpiration is the process by which water evaporates from the leaves.

tion for the factory. The general knowledge provides principles (e.g., "the rate of any chemical reaction is influenced by the concentration of each reactant") that are instantiated for the particular system, yielding rules governing the behavior of the system (e.g., "the rate of the reaction in the reactor tank is influenced by the concentration of nitric acid"). The system description is the result of exhaustively instantiating the general knowledge.

Various methods are available for generating a system description this way. The method of Falkenhainer and Forbus [12], called "scenario expansion," exhaustively generates the system description before model construction begins. In contrast, we have developed a method that interleaves generating the system description with constructing the model, thereby generating only those parts of the system description that are needed [44]. In this paper, we treat the system description as given, but our modeling algorithm is compatible with either approach.

In the compositional modeling approach, elements of the system description serve as building blocks for model construction. We adopt the approach to compositional modeling started by Qualitative Process Theory (QPT) [15]: the system description consists primarily of variables and influences among them. However, we extend QPT's representation in several ways. Most importantly, we allow systems to be described at multiple levels of detail. At the end of this section, we discuss the differences between our language for system descriptions and the languages used by other compositional modeling programs.

2.1.1 Properties of Entities: Variables

A system description includes a finite set of *variables*, which represent those properties of the system that are subject to change. Because our work focuses on building lumped-parameter, differential equation models, each variable in the system description denotes a real-valued, continuous function of time, such as the amount of water in a plant or its rate of transpiration.

Each variable is defined as a *property* of some conceptual *entity*. For example, many variables in plant physiology are properties of one of three types of entities: a space, a pool, or a process. Examples include the cross-sectional area (a property) of a conduit (a space), the amount (a property) of glucose in a plant (a pool), and the rate (a property) of transpiration (a process). This representation of variables is also used in QPT, where variables are called "quantities" and properties are called "quantity types."

Entities, properties and variables are written as ground terms in Predicate Calculus. For example, photosynthesis in a plant, which is an entity, is written as photosynthesis(plant). The rate of photosynthesis in a plant, which is a variable, is written as rate(photosynthesis(plant)). Similarly, the amount of water in a plant, also a variable, is written as amount(pool(water, plant)), where pool is a function that maps a type of substance (or energy) and a space to the corresponding pool.[5]

2.1.2 Entities at Different Levels of Detail: The Encapsulation Relation

In a complex system, entities typically can be described at multiple levels of detail. One entity may represent an aggregation of other entities, summarizing their properties while encapsulating their details. For example, the water in a plant can be treated as an aggregate pool; or the water in the roots, stem and leaves can be treated individually. Analogously, processes can be aggregated. For example, the chemical formula for photosynthesis summarizes the net effects of many chemical reactions. Similarly, in engineering, a system component is often treated as a black box even though it is constructed from other components. These are examples of *entity encapsulation*, which is ubiquitous in science and engineering because it allows modelers to create abstract descriptions that hide irrelevant details. In our terminology, an abstract (aggregate) entity encapsulates the entities that represent its underlying details.

A system description represents encapsulation relationships among entities with the **encapsulates** relation. For example, encapsulates(pool(water, plant), pool(water, leaves(plant))) specifies that the pool of water in the plant encapsulates the pool of water in the leaves; that is, these pools are alternative levels of description. Of course, the pool of water in the plant also encapsulates the water in the stems and roots; each such relationship is a separate pair within the relation. The encapsulates relation is an ordering relation like <; it is irreflexive (no entity encapsulates itself), asymmetric (no two entities encapsulate each other), and transitive.

Note that the encapsulates relation represents relationships among alternative levels of description, not spatial relationships. The relation is used whenever an entity can be described as a black box or, alternatively, through its components. While spatial relations might form the basis of some such relationships (as with pools and subpools), this need not be the case (as with processes and subprocesses).

[5] A pool consists of the substance or energy of a particular type in a particular space. In AI, the concept of a pool is the basis of the "contained stuff" ontology [6,15]. The term "pool" is common in biology and ecology.

2.1.3 Influences

As in QPT [15], the phenomena governing a system are represented as a finite set of *influences*. An influence is a causally-directed relation among two variables, the *influencer* and the *influencee*. There are two types of influences: differential and functional.

A **differential influence** specifies that the rate of change (first time derivative) of the influencee is a function of the influencer (and perhaps other variables). In QPT, differential influences are called "direct" influences. Typically, differential influences represent the effects of processes. For example, the process of water uptake transports water into the roots of a plant; thus, the amount of water in the roots is differentially influenced by the rate of water uptake. Of course, a variable may be differentially influenced by more than one process; for example, the amount of water in the roots is also differentially influenced by the rate at which water is transported from the roots to the leaves. When the differential influences on a variable are combined, they form a first-order differential equation. We write a differential influence as v1 $\Rightarrow$ v2, where the variable v1 is the influencer and the variable v2 is the influencee.

In contrast, a **functional influence** specifies that the influencee (rather than its derivative) is a function of the influencer (and perhaps other variables). In QPT, functional influences are called "indirect" influences. As with differential influences, there may be multiple functional influences on a variable. When combined, they form an algebraic equation. We write a functional influence as v1 $\rightarrow$ v2, where the variable v1 is the influencer and the variable v2 is the influencee.

Typically, functional influences represent one of three types of phenomena. First, they are used to represent the factors that affect the rate of a process. For example, the rate of photosynthesis is functionally influenced by the amount of carbon dioxide (one of its reactants) in the leaves. Second, they are used to represent definitional relations. For example, concentration is defined as amount per unit volume, so the concentration of sucrose in tree sap is functionally influenced by the amount of sucrose in the sap and by the volume occupied by the sap. Finally, a functional influence may represent a quasi-static approximation. For example, when the level of solutes in a plant cell changes, the process of osmosis adjusts the cell's water to a new equilibrium level over time. If the dynamics of this process over time are irrelevant, the modeler can simply treat the level of water as an instantaneous function of the level of solutes, and this functional dependence can be represented with a functional influence.

In QPT, each influence has a sign (+ or $-$), which specifies the sign of the partial derivative of the influencee with respect to the influencer. The sign of an influence is irrelevant to our modeling algorithm, but it is required for simulation of models.

2.1.4 Activity Preconditions

Sometimes, one variable influences another only under certain conditions. For example, the amount of carbon dioxide in the leaves influences the rate of photosynthesis only when the amount of light energy in the leaves is greater than zero. The *activity preconditions* of an influence specify the conditions under which it is active. As in QPT, the **activity preconditions** of an influence are a (possibly empty) conjunctive set of inequalities between variables or between variables and constants.[6]

2.1.5 Significance Preconditions

Sometimes, the effects of an influence are insignificant for purposes of answering a question. A model can often be greatly simplified when insignificant influences are recognized and omitted. While human modelers use many criteria to determine the significance of influences, knowledge of the *time scale* of different processes is particularly important.

In complex systems, processes cause significant change on widely disparate time scales [2,20,40,47,49]. In a plant, for example, water flows through membranes on a time scale of seconds, solutes flow through membranes on a time scale of minutes, growth requires days, and surrounding ecological processes may occur on a time scale of months or years. Given the *time scale of interest* for a question, any influence that causes significant change only on a slower time scale is insignificant [24,28,51]. For example, to answer the question concerning the effect of decreasing soil moisture on a plant's transpiration rate, a time scale of hours is most appropriate; since the effects of growth are significant only on a time scale of days or longer, they are insignificant for purposes of answering the question.

To represent such knowledge, the **significance preconditions** of an influence are encoded as an inequality relating the time scale of interest and a specific time scale. For example, for an influence representing the effect of growth on the size of a plant, the significance preconditions would be encoded as time-scale-of-interest $\geq$ days. An influence is **significant** for purposes of answering a given question if and only if the question's time scale of interest satisfies the inequality in the influence's significance preconditions.

[6] In QPT, activity preconditions are called "quantity conditions."

Typically, a differential influence represents an effect of a process, so its significance preconditions should specify the fastest time scale on which the effect is significant, as in the growth example above. If the significance preconditions of a differential influence are empty, the modeler must treat the influence as significant for any question. Since functional influences represent instantaneous effects, they are significant regardless of the time scale of interest, so their significance preconditions are always empty.

The modeling methods described in this paper do not depend on this particular criterion for significance. In the future, we plan to incorporate other criteria as well, as discussed in Section 6.1.1. Still, time scale is an important significance criterion in many domains, including biology [19,49], ecology [2,40], economics [51], and many branches of engineering [26,48]. Moreover, empirical results (described in Section 5) show that this criterion is capable of pruning many irrelevant phenomena from models.

2.1.6 Validity Preconditions

Many influences are approximations of the phenomena they represent, and these approximations typically have a limited range of validity. The *validity preconditions* of an influence specify the conditions under which the influence is a valid model of the phenomenon it represents. Contrast validity preconditions with activity and significance preconditions. The latter specify when a phenomenon is inactive or insignificant, and hence need not be modeled at all. Validity preconditions, on the other hand, specify when one particular influence is an invalid approximation of its phenomenon, but they don't obviate the need to model that phenomenon.

As with significance, human modelers use many criteria to assess the validity of influences, but the time scale of interest is particularly important. Therefore, as with significance preconditions, the **validity preconditions** of an influence are encoded as an inequality relating the time scale of interest and a specific time scale. Such a precondition might arise from cases like the following:

- The behavior of an aggregate pool is often used as an approximation to the behavior of one of its subpools. For example, the rate of photosynthesis is functionally influenced by the concentration of carbon dioxide in the mesophyll cells of the leaves. As an approximation, a modeler might say that the rate of photosynthesis is functionally influenced by the concentration of carbon dioxide in the leaves. Such an approximation is reasonable when the subpools equilibrate on a time scale faster than the time scale of interest [24,51]. For example, if diffusion of carbon dioxide throughout the leaves achieves a uniform concentration on a time scale of minutes, the influence of carbon dioxide in the leaves on the rate of photosynthesis is a valid ap-

proximation to the true influence when the time scale of interest is minutes or longer.
- An influence representing a quasi-static approximation is typically valid only if the underlying processes reach equilibrium on a time scale at least as fast as the time scale of interest [24,28,51]. For example, when the level of solutes in a plant cell changes, the process of osmosis adjusts the cell's water to a new equilibrium level. On a time scale of minutes or longer, this process can be treated as instantaneous. Therefore, the functional influence of solute level on water level is valid on a time scale of minutes or longer.

An influence is **valid** for purposes of answering a given question if and only if the question's time scale of interest satisfies the inequality in the influence's validity preconditions. As with significance preconditions, our modeling methods do not depend on this particular criterion, but it has proven very effective.

2.1.7 Influences at Different Levels of Detail: The Explanation Relation

For complex systems, different influences may represent the same phenomenon at different levels of detail. To choose a suitable set of influences on a variable in a model, a modeler must understand which influences represent independent phenomena and which represent different levels of detail for the same phenomenon. Influences on a given variable represent alternative levels of detail in cases like the following:

- The influence of an aggregate process on a pool represents the aggregate effect of its subprocesses on that pool. For example, the influence of photosynthesis on water in the leaves is due to the influence of one of its subprocesses, the light reactions, on water in the leaves. In turn, the influence of the light reactions represents the aggregate effect of two of its subprocesses: the Hill reaction, in which light energy is used to split water molecules into hydrogen and oxygen, and photophosphorylation, in which light energy is converted to chemical energy and water. Thus, the influence of photosynthesis on water in the leaves is explained by the influence of the light reactions, which is explained by the influence of the Hill reaction and the influence of photophosphorylation.
- Analogously, the influence of an aggregate pool on a process represents the aggregate effect of its subpools on that process. For example, in many plants, the influence of carbon dioxide in the leaves on photosynthesis is due to the influences of two subpools: the mesophyll cells and the bundle sheath cells.

To generalize such cases, a system description can specify that one influence is explained by other influences. The *explanation* for an influence, if it has one, relates it to other influences of the same type (i.e., differential or functional) that have the same influencee. (While there may be similar relationships among influences with different types or influencees, our modeling criteria and algorithms do not require a representation of those relationships.) The influence being explained represents the collective effect on the influencee of the influences that explain it, and the influences that constitute the explanation fully explain the aggregate influence. In short, the influence being explained and the influences in its explanation represent the same underlying phenomena at different levels of detail.

Such relationships are represented by the **explanation** relation. The pair (i,i') is an element of this relation if and only if influence i' is an element of the set of influences that explain influence i. The transitive closure of the explanation relation, the explanation* relation, provides an ordering among influences; in addition to being transitive, it is irreflexive (no influence explains itself) and asymmetric (no two influences explain each other). Note that both the explanation relation and the encapsulates relation represent aggregation hierarchies: the former represents a hierarchy of influences, while the latter represents a hierarchy of entities.

2.1.8 Summary

In summary, a system description represents domain knowledge about a particular physical system. The description includes the phenomena that govern the system as well as the levels of detail at which the phenomena can be described. Phenomena are represented by variables and influences, which provide the building blocks for models. For each influence, the system description specifies the conditions under which it is active, valid, and significant; such information helps the modeler decide which influences are relevant to answering a given question. Currently, our criteria for significance and validity are based on time scale [28,22,21], although our model construction algorithm does not depend on any particular criteria. Finally, to help the modeler ensure a coherent model, the system description represents the relationships among different levels of detail using the encapsulates and explanation relations.

2.1.9 Previous Work on System Descriptions

QPT's representation for variables and influences provides the basis for our system description language. However, because QPT was not designed to represent modeling alternatives, it does not include a representation for significance preconditions, encapsulation or explanation relationships, or validity preconditions.

The compositional modeling framework of Falkenhainer and Forbus [12] extends the ideas of QPT to represent modeling alternatives. The building blocks for their models are “model fragments,” which provide individual influences or, more typically, sets of influences (e.g., complete equations). To allow different model fragments to specify different modeling alternatives, each model fragment has associated "assumptions," symbolic labels that characterize the phenomena it represents and its level of detail. To represent the relationships among model fragments, assumptions are organized into "assumption classes"; the assumptions in each class represent mutually incompatible modeling alternatives. Several researchers [23,38,39] define interesting variants of this compositional modeling framework, but the basic ideas are the same.

Our representation differs in two ways. First, the person encoding the domain knowledge need not group influences into model fragments; rather, in our approach, individual influences are the building blocks for models. As will be shown in Section 3, important modeling decisions arise at the level of influences, and we want the modeling program, not the knowledge engineer, to face these decisions. Moreover, in our experience, it is rare for two influences to necessarily occur together in models; there are typically conditions in which only one of the influences is relevant, and alternative levels of detail for each influence are often available. Although our approach – using influences as building blocks for models – differs from other work in automated modeling, the idea is not new: human modelers have taken this approach in a variety of domains [4,18,30,43,46].

Second, Falkenhainer and Forbus require the knowledge engineer to provide more of the criteria for making modeling decisions. In addition to requiring the knowledge engineer to provide model fragments, assumptions, and assumption classes, they require two additional types of knowledge: (1) rules that specify dependencies among assumptions (e.g., which ones are mutually incompatible and which ones require each other) and (2) the conditions in which each assumption class is relevant (i.e., when the modeler must choose one of the alternatives). For our model construction algorithm, the first type of knowledge is unnecessary because the encapsulates and explanation relations sufficiently describe the relationships among alternatives, in a form that we believe will be more natural for domain experts. The second type of knowledge has no counterpart in our approach; our model construction algorithm effectively generates such knowledge automatically, as will be shown in Sections 3 and 4. Our algorithm does require the system description to provide some criteria for modeling decisions, namely significance preconditions and validity preconditions; however, these criteria are properties of individual influences, and hence should be easy for a domain expert to provide. (Our botany expert had no trouble providing such knowledge.)

2.2 Prediction Questions

2.2.1 Driving Conditions and Variables of Interest

A prediction question poses a hypothetical scenario, specified by one or more *driving conditions*, and asks for the behavior of one or more *variables of interest*. Driving conditions specify the behavior or initial condition (or both) of particular variables in the system description. For example, "decreasing soil moisture" is the driving condition in the question "How would decreasing soil moisture affect a plant's transpiration rate?" Any variable in the system description (such as "transpiration rate" in the example) can serve as a variable of interest. The goal in answering a prediction question is to predict and explain the causal effect of the driving conditions on the variables of interest.

We currently use the same language to specify both types of driving conditions (behaviors and initial conditions). Each driving condition is an equality or inequality statement relating a variable (or its derivative) to another variable (or its derivative) or constant. For example, the initial temperature of a plant could be specified precisely as $\mathsf{temperature(plant)} = 67°\mathsf{F}$ or less precisely as $\mathsf{temperature(plant)} > 32°\mathsf{F}$ or $\mathsf{temperature(plant)} > \mathsf{temperature(soil)}$. Its initial rate of change could similarly be specified (using the differential operator D) as $\mathsf{D(temperature(plant))} = \mathsf{zero}$ (thermal equilibrium) or $\mathsf{D(temperature(plant))} > \mathsf{zero}$ (the plant is warming up). These same statements could be specified as behaviors rather than initial conditions, meaning that they hold throughout the temporal extent of the scenario. We also allow a behavior to be described as increasing or decreasing to a new equilibrium value (i.e., increasing or decreasing for an unspecified amount of time and constant thereafter). Our modeling methods are not restricted to this particular language for driving conditions; for instance, we could allow behaviors to be specified as arbitrary functions (e.g., a sine wave) as well. Our methods would simply ignore the extra information provided by such a precise description of behavior.

2.2.2 Time Scale of Interest

As discussed in Section 2.1, a time scale of interest provides an important source of power in modeling. It allows a modeler to (1) treat influences that operate on a slower time scale as insignificant, (2) represent the effects of faster processes using quasi-static approximations, and (3) treat separate pools as a single aggregate when they equilibrate on a faster time scale. Thus, a time scale of interest allows many important model simplifications.

Although the person posing a question may specify a time scale of interest, often a modeler must determine it automatically. Elsewhere [44,45], we describe an algorithm for choosing an appropriate time scale of interest when none is specified in the question. Whether the time scale of interest is chosen by the modeler or provided by the person posing the question, this paper will treat it as part of the question.

3 Scenario Models

Given a system description and a prediction question, our modeling algorithm constructs a scenario model for answering the question. A **scenario model** consists of the following:

- a set of variables (a subset of the variables in the system description) partitioned into *exogenous* variables, whose behavior is determined by influences external to the model, and *dependent* variables, whose behavior is determined by the model, and
- a set of influences (a subset of the influences in the system description), each of whose influencee is a dependent variable in the model and whose influencer is another variable in the model (exogenous or dependent)

For example, the scenario model in Figure 1 shows how a plant regulates the abscisic acid hormone (ABA) in response to changes in turgor pressure (hydraulic pressure) in its leaves (e.g., when it begins wilting). Leaf turgor pressure is the only exogenous variable; all the others are dependent. The model shows that ABA is synthesized and consumed in the leaf mesophyll cells and transported to the guard cells. (The figure also describes the conventions that are used in this and subsequent figures.)

As in previous work [12,23,34], a scenario model is intended to support the entire simulation of the scenario. To make predictions from a particular state of the scenario, the simulator must determine which influences in the scenario model are active in that state. For example, turgor pressure only influences ABA synthesis when the pressure drops below a threshold. The activity preconditions of the influence would represent that fact. To simulate a turgid (not wilting) plant whose turgor pressure is dropping, the simulator would omit this influence until turgor pressure drops below the threshold. A variety of simulators are capable of simulating scenario models in this way [14,15,17]. Using this approach, the modeler need only build one scenario model to answer a question, rather than building a different model for different states of the scenario.

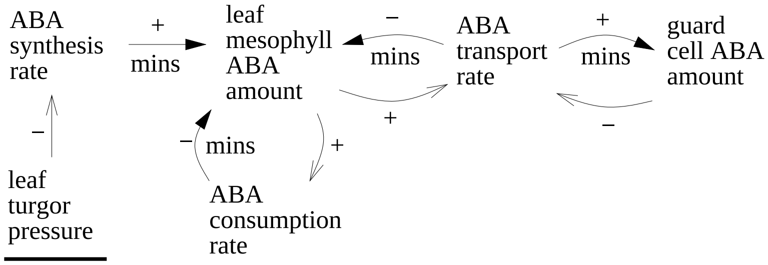


Fig. 1. A scenario model.

This and subsequent figures use the following conventions:

- Arrows with solid tips represent differential influences, while arrows without solid tips represent functional influences.
- Exogenous variables (in this example, leaf turgor pressure) are underlined.
- Differential influences are labeled with the time scale on which they become significant. For example, "mins" is a shorthand for the significance precondition time-scale-of-interest $\geq$ minutes.
- Influences are labeled with the sign of their partial derivative. For example, when leaf turgor pressure decreases, the rate of ABA synthesis increases.
- Activity preconditions of influences are not shown.

3.1 Adequacy

Intuitively, a scenario model is adequate for answering a given prediction question if it satisfies two criteria. First, it must make the desired predictions with sufficient accuracy. Second, to ensure a comprehensible explanation, the model must be a coherent description of the physical system. To automate modeling, we must formalize these two intuitive criteria.

We formalize the criteria as a set of adequacy constraints. Each constraint is a predicate of three arguments: a system description, a prediction question, and a scenario model. A scenario model is adequate for a given system description and question if and only if every adequacy constraint is satisfied. Collectively, the constraints address the key issues in model construction, and they demonstrate the central role of variables and influences in each issue. The key issues include choosing appropriate exogenous variables (Section 3.1.2), choosing appropriate influences on each dependent variable (Section 3.1.3), modeling an appropriate set of system entities (Section 3.1.4), and relating the driving conditions of the question to the variables of interest (Section 3.1.5).

Our objective is to formalize the intuitive criteria that human modelers use to achieve sufficiently accurate, coherent models. For the adequacy constraints we propose, this section explains why each is intuitively necessary. Section 5 discusses the results of empirically evaluating the constraints in the domain of plant physiology.

3.1.1 Variables in a Scenario Model

A model is only adequate if it can make the desired predictions. This motivates the following constraint.

Adequacy constraint 1 (include variables of interest)
A scenario model is adequate only if it includes every variable of interest.

As discussed earlier, the simulator must determine which influences in the scenario model are active in each state of the scenario. This requires the ability to evaluate the activity preconditions of influences in the model. The following constraint ensures that the model provides enough information to do so.

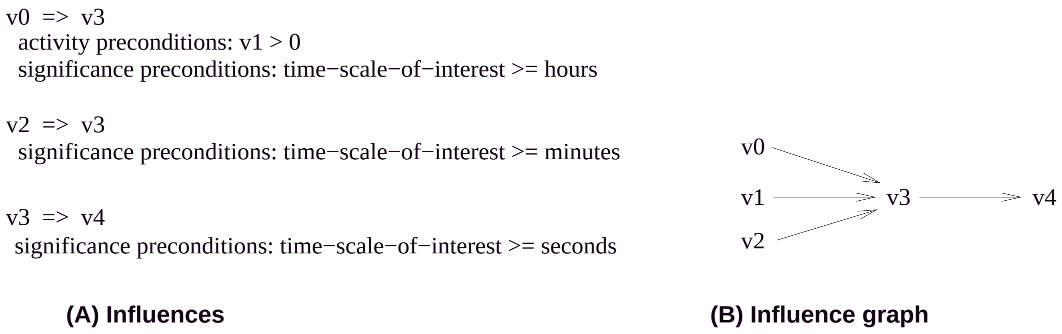
Adequacy constraint 2 (include variables in activity preconditions)
A scenario model is adequate only if it includes every variable appearing in an activity precondition of an influence in the model.

3.1.2 Exogenous Variables

Once a variable is included in a model, the modeler must determine *how* to model it. The first decision is whether to model it as exogenous or dependent.

While the phenomena governing a dependent variable are represented by influences in the model, the phenomena governing an exogenous variable are outside the scope of the model. Conceptually, the model represents a system, and the exogenous variables represent the *system boundary*, the interface between the system and its surrounding environment. Thus, by choosing to model some variables as exogenous, a modeler partitions the system description into two parts: the subsystem that is relevant to answering the given question, and its environment (which is irrelevant). To ensure that a model of a complex system is adequate and as simple as possible, a suitable system boundary is crucial.

Despite the importance of a well-chosen system boundary, few previous automated modeling programs can choose exogenous variables automatically. Moreover, as explained at the end of this subsection, these few programs use criteria that are too weak for answering prediction questions; their choice of exogenous variables can result in either inadequate or unnecessarily complex models.



```
v0  =>  v3
  activity preconditions: v1 > 0
  significance preconditions: time−scale−of−interest >= hours

v2  =>  v3
  significance preconditions: time−scale−of−interest >= minutes

v3  =>  v4
 significance preconditions: time−scale−of−interest >= seconds
```

(A) Influences

(B) Influence graph

Fig. 2. **(A)** A set of influences, along with their activity and significance preconditions. **(B)** The corresponding influence graph.

Human modelers treat a variable as exogenous only if it is approximately independent of the other variables in the model. For example, the rate of precipitation can be treated as exogenous in a model of a single plant; while the behavior of the plant depends critically on the rate of precipitation, the phenomena that govern precipitation do not depend significantly on the behavior of the plant. Thus, to decide which variables can be treated as exogenous, a modeler must be able to determine whether one variable significantly affects another.

The influences in a system description determine which variables affect each other. Clearly, one variable affects another if there is an influence from the first variable to the second. One variable can also affect another by enabling or disabling the influences on it; that is, one variable affects another if there is an influence on the second variable whose activity preconditions reference the first variable.

Therefore, we define the **influence graph** for a system description as follows. The nodes of the graph are the variables. There is a directed edge from one variable to another if and only if there is an influence whose influencee is the second variable and either

- the first variable is the influencer or
- the first variable appears in the influence's activity preconditions.

An **influence path** is a path of non-zero length in an influence graph. One variable **significantly influences** another if and only if there is an influence path leading from the first variable to the second and every influence in the path is valid and significant for the given question.

Figure 2 illustrates these concepts. Part A shows a set of influences, and Part B shows the corresponding influence graph. If the time scale of interest is seconds, only v3 significantly influences v4. However, on a time scale of hours, v4 is significantly influenced by v0, v1, v2 and v3.

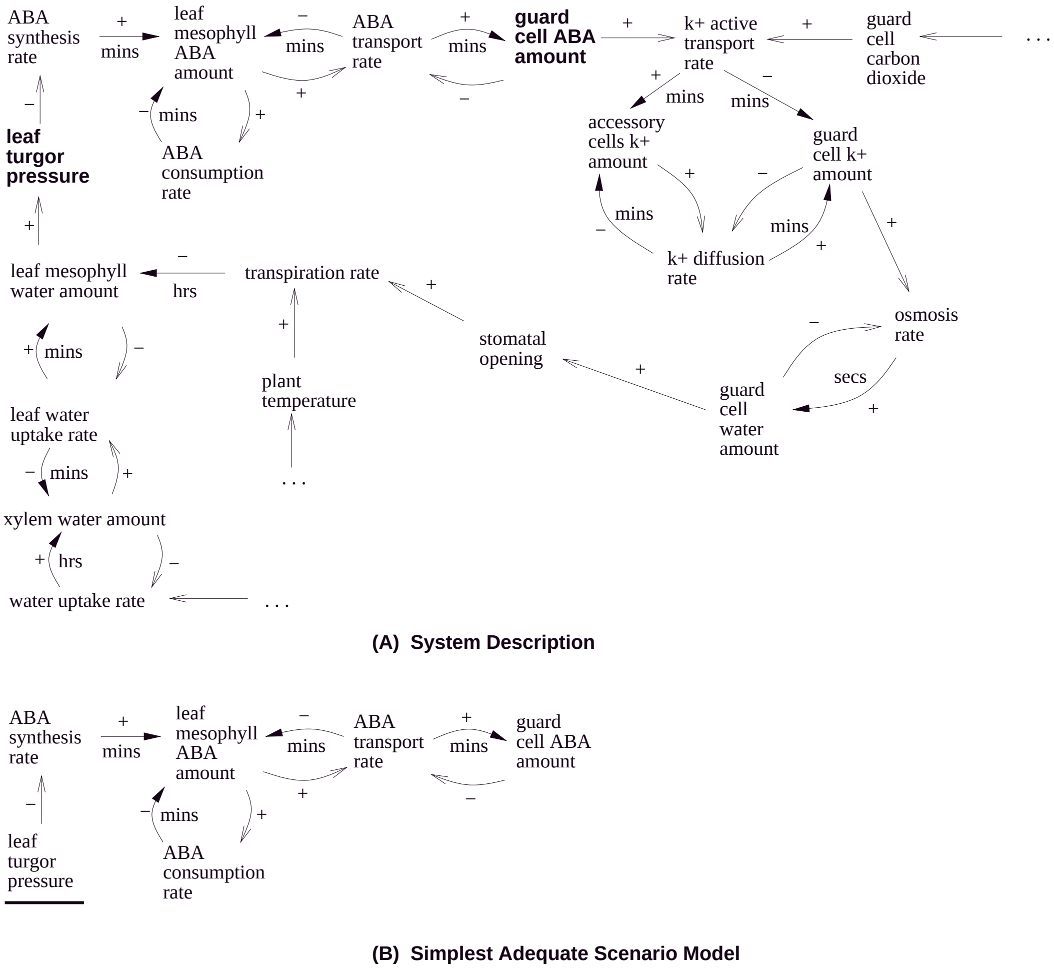
Given the definitions above, the following constraint formalizes the intuition that an exogenous variable is approximately independent of all other variables in the model.

Adequacy constraint 3 (exogenous variables independent of model) *A scenario model is adequate only if none of its exogenous variables is significantly influenced in the system description by another variable in the model.*

While the previous constraint on exogenous variables ensures that they are appropriate for the model that contains them, the next constraint ensures that they are appropriate for the given question. Recall that a prediction question asks for the effects of driving conditions on variables of interest. To answer a prediction question, a modeler includes in the model those variables whose behavior is relevant to determining the behavior of the variables of interest. Therefore, if a variable in the model is significantly influenced by a driving variable (a variable in a driving condition), the model should reflect this so the effects of the driving variable's behavior on that variable can be determined. Thus, to ensure that the exogenous variables do not disconnect the model from relevant driving conditions, a variable cannot be exogenous unless it is approximately independent of the driving variables.

Adequacy constraint 4 (exogenous variables independent of question) *A scenario model is adequate only if none of its exogenous variables is significantly influenced in the system description by a driving variable (other than itself if it is a driving variable).*

Together, these two constraints specify whether a variable in a model can be exogenous. To illustrate these system boundary criteria, consider the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" This question is important because plants send ABA to the guard cells to combat dehydration. The appropriate time scale of interest for this question is minutes. (This time scale can be determined automatically if it is not specified [44,45].) Part A of Figure 3 shows a portion of the system description for a prototypical plant; the driving variable (leaf turgor pressure) and variable of interest (guard cell ABA amount) are shown in bold. Part B shows the simplest adequate model for answering the question. In this model, none of the dependent variables could be exogenous, because each one is significantly influenced (on a time scale of minutes) by the driving variable, leaf turgor pressure (thus violating adequacy constraint 4). Leaf turgor pressure can be exogenous in the model because it satisfies adequacy constraints 3 and 4; that is, as shown in Part A, leaf turgor pressure is not significantly influenced (on a time scale of minutes) by any other variable in the model nor by any other driving variable (there are no others). On a time scale of hours, however, leaf turgor pressure could not be treated as exogenous, because it would be significantly influenced by guard cell ABA amount on that



(A) System Description

(B) Simplest Adequate Scenario Model

Fig. 3. **(A)** A portion of the system description for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" The driving variable and variable of interest are shown in bold. Ellipses indicate connections to the remaining variables and influences in the system description. Alternative levels of detail are not shown. **(B)** The simplest adequate scenario model for answering the question.

time scale via a path passing through guard cell water amount and transpiration. Thus, the time scale of interest allows a tighter system boundary than would otherwise be possible.

Despite its importance, no previous work in automated modeling has provided explicit criteria for choosing exogenous variables. Typically, modeling programs require either the system description or question to specify those variables that can be exogenous. For instance, the modeling algorithms of Williams [54] and Iwasaki and Levy [23] take this approach. Although these algorithms can determine which exogenous variables must be included in the scenario model, neither algorithm can determine exogenous variables automatically. For complex systems, this approach is impractical.

Nayak's modeling algorithm [38] can choose exogenous variables, but it does not have explicit criteria for doing so. Moreover, his definition of an adequate model is suitable for his modeling task, explaining a specified causal relation, but is not sufficient for answering prediction questions. For instance, his definition would allow a scenario model to treat a variable as exogenous even though it is significantly influenced (in the system description) by another variable in the model. Adequacy constraint 3 prevents our modeling algorithm from making this mistake.

The modeling algorithm of Falkenhainer and Forbus [12] largely determines the system boundary by identifying relevant system components. Their algorithm requires, as input, a system decomposition. That is, each system component is also assumed to be a system, and each system can have components that represent its subsystems. To identify the components that are relevant to a question, the algorithm first identifies the smallest set of components that must be modeled to include the immediate influences on the variables of interest; these components are marked as relevant. Next, to ensure that interactions among these components are modeled, the algorithm determines a "minimal covering system," the lowest system down the system decomposition that subsumes all relevant components. That component and its subsystems (down to the level of the initially relevant components) are relevant. Any variable that is a property of a relevant component, but is only influenced by properties of irrelevant components, is exogenous.

Their approach has several limitations. While their modeling algorithm requires a system decomposition, our criteria for choosing a system boundary only require knowledge of the influences. Furthermore, Falkenhainer and Forbus assume that the system decomposition is based on partonomic structure; however, O'Neill *et al.* [40] argue that approximate system boundaries in natural systems arise from differences in process rates (i.e., their time scales) and that these boundaries may not correspond to standard structural decompositions. Even in engineered systems, designed system boundaries cannot be trusted when considering faults or unintended interactions [8]. Reasoning at the level of influences provides more flexibility and overcomes the difficulty of specifying an a priori system decomposition. Additionally, by specifying the criteria for choosing exogenous variables in terms of influence paths, we ensure that the chosen system boundary will be sufficiently sensitive to the connections between driving conditions and variables of interest.

3.1.3 Influences on a Dependent Variable

Exogenous variables, which lie on the system boundary, are governed by phenomena outside the scope of the model. In contrast, for every dependent variable in a model, the modeler must choose a set of influences to represent the amount(pool(water, guard-cells)) ⇐ rate(osmosis(accessory-cells, guard-cells))

amount(pool(water, guard-cells)) ← amount(pool(ABA, guard-cells))
    validity preconditions: time-scale-of-interest ≥ hours

amount(pool(water, guard-cells)) ← amount(pool(CO_2, guard-cells))
    validity preconditions: time-scale-of-interest ≥ hours

Fig. 4. Influences on the amount of water in a plant's guard cells.

phenomena that govern it. The four constraints in this subsection ensure that every dependent variable in a model has an adequate set of influences.

For simulation of a model, the influences on a variable are combined to form an equation. Human modelers use two types of equations: algebraic equations, composed of functional influences, and differential equations, composed of differential influences. The following constraint ensures that the influences on each dependent variable correspond to one of these two types.

Adequacy constraint 5 (influences homogeneous)
A scenario model is adequate only if the influences on any given dependent variable are all the same type (i.e., differential or functional).

For example, Figure 4 shows a set of influences on the amount of water in a plant's guard cells. The first influence represents the fact that the amount of water is regulated by osmosis from neighboring accessory cells. The remaining two influences represent quasi-static approximations; changes in the levels of ABA or carbon dioxide cause osmosis to adjust the level of water to a new equilibrium. The amount of guard cell water can be modeled by the differential influence or the two functional influences, but it would be incoherent to mix them.

A model must also be sufficiently accurate. For this reason, each of its influences must be a valid approximation of the phenomenon the influence represents. That is, the validity preconditions of each influence must be satisfied for the given question, as specified in the following constraint.

Adequacy constraint 6 (influences valid)
A scenario model is adequate only if each of its influences is valid for the given question.

For example, the two functional influences in Figure 4, which represent quasi-static approximations, are only valid on a time scale of hours, because the mechanisms that restore equilibrium operate on a time scale of minutes. Therefore, for any question whose time scale of interest is less than hours (e.g., seconds or minutes), a scenario model that includes these influences is inadequate.

amount(pool(CO_2, leaves)) $\Leftarrow$ rate(CO_2-diffusion(atmosphere, leaves))
    significance preconditions: time-scale-of-interest $\geq$ seconds

amount(pool(CO_2, leaves)) $\Leftarrow$ rate(photosynthesis(leaves))
    significance preconditions: time-scale-of-interest $\geq$ minutes

amount(pool(CO_2, leaves)) $\Leftarrow$ rate(dark-reactions(leaves))
    significance preconditions: time-scale-of-interest $\geq$ minutes

Explanation(amount(pool(CO_2, leaves)) $\Leftarrow$ rate(photosynthesis(leaves)),
        amount(pool(CO_2, leaves)) $\Leftarrow$ rate(dark-reactions(leaves)))

Fig. 5. Influences on the amount of carbon dioxide in a plant's leaves. The first two are the maximally aggregate influences. The influence of photosynthesis is explained by the influence of the dark reactions (and not by any other influences).

To further ensure that a model is sufficiently accurate, the influences on each dependent variable should represent all the phenomena that affect the variable. Such a set of influences is *complete*. Given a system description, a dependent variable, and a type of influence (i.e., functional or differential), we define a **complete set of influences** as follows:

- The set of all "maximally aggregate" influences of the specified type on the variable is complete. A maximally aggregate influence is one that does not explain any other influence (i.e., a maximal element of the explanation relation).
- The result of replacing an influence in a complete set with the set of all influences that explain it (as specified by the explanation relation) is a complete set.

For example, Figure 5 shows a set of influences on the amount of carbon dioxide in a plant's leaves. As shown, the influence of photosynthesis is explained by the influence of the dark reactions (and not by any other influences). The first two influences in the figure constitute a complete set because they are the maximally aggregate influences. Also, the first and third influences constitute a complete set, since the photosynthesis influence is fully explained by the more-detailed influence of the dark reactions.

Of course, the model need only be sufficiently accurate for the given question. Therefore, the influences on each dependent variable need only represent all the *significant* phenomena that affect the variable. For a given question, a set of influences on a variable is **approximately complete** if and only if it is a subset of a complete set of influences and none of the omitted influences is significant for the question. For example, in Figure 5, the first influence alone constitutes an approximately complete set on a time scale of seconds. However, on a time scale of minutes or longer, either the second or third influence must be additionally included.

Given these definitions, the following constraint ensures that the model represents all phenomena that significantly affect each dependent variable.

Adequacy constraint 7 (influences complete)
A scenario model is adequate only if the set of influences on each dependent variable is approximately complete for the given question.

Finally, to ensure that the influences on a dependent variable are coherent, a modeler must avoid mixing different levels of detail for the same phenomenon. The following constraint enforces this requirement.

Adequacy constraint 8 (influences not redundant)
A scenario model is adequate only if the influences on each dependent variable do not include two influences related by the explanation* *relation).*

If a model's influences on a dependent variable satisfy the four constraints in this subsection, we say that the influences are adequate. Recall that adequacy of a model must address two issues: accuracy of predictions, and coherence. Constraints 6 (influences valid) and 7 (influences complete) help ensure that the influences provide a sufficiently accurate representation of the governing phenomena, and constraints 5 (influences homogeneous) and 8 (influences not redundant) help ensure that the representation is coherent.

Most previous work in automated modeling does not enforce explicit constraints like these for the influences on a dependent variable. For those modeling programs that use the assumption class representation of Falkenhainer and Forbus [12], the person encoding the model fragments and the constraints among assumptions must ensure that each compatible combination of model fragments yields an adequate set of influences.

Some previous modeling programs are given a complete equation for a dependent variable and they identify and discard negligible terms in the equation [11,36,55,57]. This is analogous to identifying an approximately complete set of influences given a complete set. However, these programs do not consider alternative levels of detail for the elements of the equation.

3.1.4 Entities in a Model

A scenario model is a model of selected entities in a system. Each variable in a model is a property of an entity, so the entities in a scenario model consist of all the entities whose properties are represented by the model's variables. The entities in a model are important because they indicate the model's view of the system.

To ensure consistent predictions and a comprehensible explanation, that view must be coherent. More specifically, while entities can typically be described at multiple levels of detail, a modeler must avoid mixing levels. In the system description, entities at different levels of detail are related by the encapsulates relation. Thus, the following constraint prevents a model from mixing levels of detail.

Adequacy constraint 9 (entities coherent)
A scenario model is adequate only if it does not include two entities related by the encapsulates *relation.*[7]

The driving variables of a question also constrain the choice of entities in a model. A scenario model need not necessarily include all driving variables, because some may be irrelevant to the variables of interest. However, the model should respect the level of aggregation specified in the driving variables, for two reasons. First, these variables indicate the level of detail in which the user is interested. Second, if the modeler encapsulates these variables or chooses variables at a lower level of detail, the information in the driving conditions will be lost.[8] The following constraint ensures that the model respects the level of aggregation specified in the driving variables.

Adequacy constraint 10 (entities compatible with driving variables)
A scenario model is adequate only if it does not include an entity that encapsulates an entity of a driving variable and it does not include an entity that is encapsulated by an entity of a driving variable.

Elsewhere [44], we formulate additional adequacy constraints based on the entities in a model. The constraints ensure that the model is appropriate for the user's level of knowledge and desired level of detail. While useful, such constraints are tangential to the focus of this paper.

3.1.5 Influence Paths in a Model

A prediction question asks for the causal effect of driving conditions on variables of interest. Therefore, a scenario model is adequate for answering the question only if the variables of interest are significantly influenced (in the model) by the driving variables. Additionally, in order to predict the behavior of the variables of interest beyond the initial state, the influence paths relating the driving variables to the variables of interest must be capable of predicting changes in the variables of interest.

[7] Recall that the relation is transitive.

[8] It may be possible to infer driving conditions at the abstract or more-detailed levels from the given driving conditions, but we have no general method for making such inferences.

Through an individual influence, one variable can cause change in another variable in two ways: (1) with a differential influence, a specified value for the influencer (along with values for other influencing variables) provides the rate of change of the influencee; (2) in contrast, a functional influence can cause change only if the influencer is changing [15]. Thus, a model can predict the changes in a variable of interest caused by a driving variable only if the influence path connecting them contains a differential influence or the driving conditions specify how the driving variable is changing (in which case a path of functional influences will propagate the change). If either case is satisfied, the influence path is a **differential influence path**.

For example, the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" specifies that turgor pressure is decreasing, so any influence path from turgor pressure to another variable is a differential influence path, capable of causing change. In contrast, if the question only specified that turgor pressure is above the "yield point" (above which the pressure causes cell growth), an influence path leading from turgor pressure is differential only if it contains a differential influence (as is the case with the influence of turgor pressure on cell size).

Motivated by the above discussion, the following constraint ensures that a model can predict the effect of the driving conditions on the variables of interest.

Adequacy constraint 11 (variables of interest differentially influenced) *A scenario model is adequate only if, for every variable of interest, the model includes a differential influence path leading to it from some driving variable such that every influence in the path is valid and significant for the given question.*

The requirement that a scenario model relate driving variables to variables of interest is not new, although previous work has not required differential influence paths. Nayak [38] requires an adequate model to provide a causal path linking the driving variable to the variable of interest. Amsterdam [3] requires an adequate model to provide "interaction" paths (i.e., not necessarily causal) linking every variable of interest to some driving variable. Williams's method for generating a "critical abstraction" [54] is designed to ensure that the chosen scenario model causally links the driving variables (in his framework, the exogenous variables of the system) to the variables of interest. We only require differential influence paths because they are appropriate for answering prediction questions; our model construction algorithm would work equally well if adequacy constraint 11 only required valid, significant influence paths (not necessarily differential) from driving variables to variables of interest.

3.1.6 Other Possible Adequacy Constraints

Some previous modeling programs define a model as adequate only if its predictions match the "correct" behavior (within a specified tolerance). These programs either address tasks in which the correct behavior of the variables of interest is known [1,52] or they assume that the approximate error introduced by different approximations can be estimated [9–11,50]. However, a prediction question does not provide the correct behavior, and error estimates are not available in the domains we have studied, so we exclude such a constraint. In Section 4.5, we suggest how TRIPEL could be extended to handle such a constraint.

3.2 Simplicity

To answer a prediction question, a modeler should construct the simplest adequate scenario model, minimizing irrelevant phenomena and details, because a model with irrelevant information is more difficult to analyze (e.g., simulate) and explain. Thus, a modeler requires criteria for determining whether one candidate model is simpler than another.

Human modelers probably use a combination of many criteria to assess the complexity of a model. Nevertheless, the number of variables in a model is a simple measure that correlates well with most other measures of complexity, and it has proven to be an effective heuristic in our experience. Simulation complexity tends to increase with the number of variables, and a model with more variables is generally more difficult to understand and explain. Furthermore, most simplification techniques used by human modelers reduce the number of variables in a model. Thus, we define one model as simpler than another as follows:

– For any two scenario models m and m′, m is **simpler** than m′ if and only if m has fewer variables than m′.

In contrast to our measure of simplicity, Nayak [38] and Iwasaki and Levy [23] define one scenario model as simpler than another if, for every model fragment in the first, either that model fragment or a more-detailed alternative is in the second.[9] This is a reasonable criterion when it holds, but it leaves too many models incomparable. For example, consider two models, one with only a few variables and influences (i.e., representing a few phenomena), and one with many variables and influences (i.e., representing many phenomena, some in

[9] Actually, Iwasaki and Levy's definition is in terms of "composite model fragments" rather than model fragments, but the distinction is irrelevant to our discussion.

great detail); if the first model treats some aspect of the system in more detail than the second model, the two models are incomparable under their criterion. Thus, although the first model is intuitively simpler, a modeling algorithm based on their simplicity criterion would be content to choose the second model as the simplest adequate model.

3.3 Summary

In summary, we define a scenario model as adequate for a given prediction question if and only if the model satisfies the following constraints:

- Its variables include every variable of interest (adequacy constraint 1) and every variable appearing in an activity precondition of its influences (adequacy constraint 2).
- Its system boundary is adequate (adequacy constraints 3 and 4).
- Its influences on each dependent variable are adequate (adequacy constraints 5, 6, 7, and 8).
- Its entities are coherent (adequacy constraint 9) and appropriate for the question (adequacy constraint 10).
- It relates the driving variables of the question to the variables of interest (adequacy constraint 11).

Among the adequate scenario models for a question, those with the fewest variables are the simplest, and the modeler's objective is to find one of these simplest adequate models.

4 Model Construction Algorithm

Together, Sections 2 and 3 define the model construction task: given a system description and a prediction question, construct a simplest adequate scenario model for answering the question. This section presents algorithms for performing the task and its subtasks.

4.1 Extending Partial Models

There are many possible models of a complex system, so finding a simplest adequate model is difficult. To find such a model efficiently, TRIPEL searches the space of *partial* models of the system, ruling out most models without ever generating them.

A partial model satisfies the definition of a scenario model with one possible exception: in addition to exogenous and dependent variables, it may contain *free variables*. After a modeler has chosen to include a variable in a model, but before the modeler has decided whether to treat it as exogenous or dependent, the variable is free. Thus, a partial model with free variables represents a model still under construction.

Formally, a **partial model** consists of the following:

- a set of variables (a subset of the variables in the system description) partitioned into exogenous variables, dependent variables, and free variables
- a set of influences (a subset of the influences in the system description), each of whose influencee is a dependent variable in the model and whose influencer is another variable in the model (exogenous, dependent or free)

Note that a scenario model, as defined in Section 3, is simply a special type of partial model, one with no free variables.

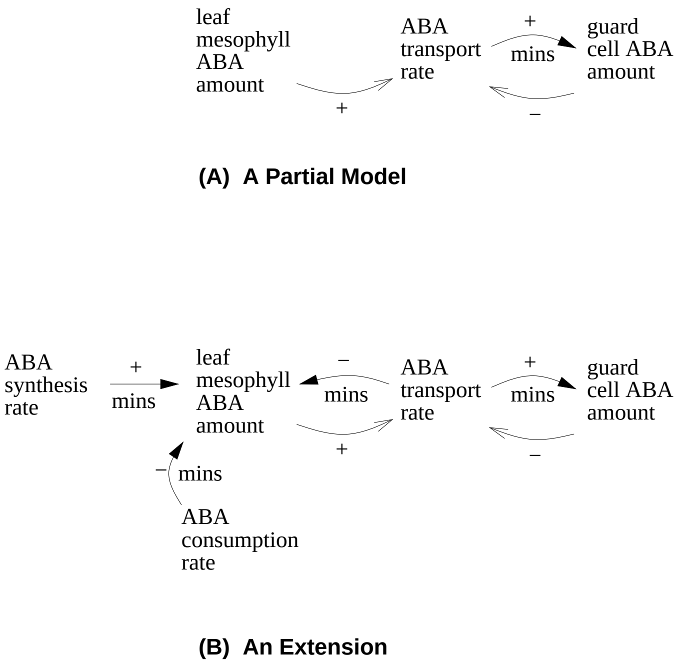
Partial models are ordered by an *extension* relation. Intuitively, a partial model $\mathsf{m_e}$ is an extension of a partial model m if and only if $\mathsf{m_e}$ can be constructed from m by making additional modeling decisions. More precisely, $\mathsf{m_e}$ is an **extension** of m if and only if m and $\mathsf{m_e}$ are not identical and all of the following conditions are satisfied:

- every variable in m is also in $\mathsf{m_e}$
- every exogenous variable in m is an exogenous variable in $\mathsf{m_e}$
- every dependent variable in m is a dependent variable in $\mathsf{m_e}$
- the set of influences on the dependent variables of m are identical in m and $\mathsf{m_e}$

These conditions allow a partial model to be extended by adding variables, by deciding to treat a free variable as exogenous or dependent, and by adding influences on free variables or new variables. For example, Part A of Figure 6 shows a partial model in which the amount of leaf mesophyll ABA is a free variable, and Part B shows an extension. In the extension, the amount of leaf mesophyll ABA is a dependent variable, the influences on it are included, and two new free variables (the influencers) are included.

The extension relation is an ordering relation like <. That is, it is irreflexive (no partial model is an extension of itself), asymmetric (no two partial models are extensions of each other), and transitive. The definition of simplicity used for scenario models applies to partial models as well, so a partial model is at least as simple as any of its extensions, because any extension has at least as many variables.

One key to efficient model construction is the ability to recognize that a given



(A) A Partial Model

(B) An Extension

Fig. 6. **(A)** A partial model. The variable leaf mesophyll ABA amount is free. **(B)** An extension of that partial model. The variables ABA synthesis rate and ABA consumption rate are free.

partial model cannot be extended into an adequate scenario model. The adequacy constraints in Section 3.1, although defined in terms of scenario models, apply to partial models as well. A partial model that violates an adequacy constraint can sometimes be extended to remedy the violation; for example, if a partial model violates adequacy constraint 1 (include variables of interest), it can be extended to include the variables of interest. However, a partial model can be eliminated from consideration when it violates a *monotonic constraint.* A **monotonic constraint** is an adequacy constraint which, when violated for a partial model, is violated for each of its extensions. For instance, when a partial model includes mutually incoherent entities, so will all its extensions. By pruning such a partial model from consideration, TRIPEL avoids generating any of its extensions, effectively pruning a large chunk from the search space. (Remember, the extension relation is transitive, so a single partial model may have many extensions.) Section 4.3.3 lists those adequacy constraints that are monotonic.

We illustrate TRIPEL's model construction algorithm using the familiar question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" Figure 3 (p. 19) shows a portion of the system description for this question. As mentioned earlier, the appropriate time scale of interest is minutes.

To construct an adequate scenario model, TRIPEL starts with a partial model

```
Find-adequate-model (S, Q)
  /* S is a system description, and Q is a prediction question */
  agenda ← $\emptyset$
  let initial be a partial model consisting of the variables of interest, each free
  if initial satisfies all monotonic constraints
    then add initial to agenda
  while agenda is not empty
    remove the simplest partial model m from agenda
    if m is an adequate scenario model
      then return m
      else for each partial model m' in Extend-model(m, S, Q)
              if m' satisfies all monotonic constraints
                then add m' to agenda
  return failure
Extend-model (m, S, Q)
  /* m is a partial model, S is a system description, and Q is a prediction question */
  if all free variables in m can be exogenous
  then mark all free variables in m as exogenous
       return {m}
  else let v be a free variable in m that must be dependent
       models ← $\emptyset$
       for each $m_v$ in Dv-models(v, S, Q)
         m' ← extend m with $m_v$
         add m' to models
       return models
```

Fig. 7. TRIPEL's model construction algorithm

consisting only of the variables of interest, and it incrementally extends this model until it satisfies all the adequacy constraints. At each step, there may be alternative ways of extending the model, so it must search through the possibilities.

The model construction algorithm, shown in Figure 7, can be viewed as graph search. Each node in the search graph is a partial model. The initial node in the search is a partial model consisting only of the variables of interest, each a free variable. For instance, the initial node for the example is a partial model consisting of one free variable, guard cell ABA amount. As will be described below, a partial model's successors in the search graph consist of some of its extensions. The goal of the search is to find a simplest adequate scenario model for the question. (Unlike some graph search problems, the path by which a goal node is found is irrelevant.)

A best-first strategy guides the search, using the simplicity criterion as the evaluation function. That is, TRIPEL always extends the search by removing the simplest partial model (i.e., the one with the fewest variables) from the search agenda. If the partial model is an adequate scenario model, it is returned as a simplest adequate scenario model; every other partial model on the agenda has as many or more variables, so they and their extensions cannot be simpler. In the example, the initial partial model is the simplest one on the agenda (in fact, the only one), so it is removed. Because it contains a free variable, it is not a scenario model, hence it is not an adequate scenario model.

If the partial model is not an adequate scenario model, its successors replace it on the search agenda. The function Extend-model returns the successors of a given partial model m. To generate these successors, the function extends m with alternative ways of modeling one of m's free variables.

To accomplish this, Extend-model first asks the System Boundary Selector (discussed in Section 4.3) whether all of m's free variables can be exogenous (i.e., whether they satisfy adequacy constraints 3 and 4). If so, Extend-model marks each free variable as exogenous and returns the resulting scenario model as the only successor. In our example, this is not the case. The free variable in the initial partial model (guard cell ABA amount) cannot be exogenous because it violates adequacy constraint 4; specifically, as shown in Figure 3 (p. 19), it is significantly influenced by the driving variable (leaf turgor pressure) on the time scale of interest (minutes).

When the System Boundary Selector's response is "no", it also tells Extend-model which variable v must be dependent (in the example, guard cell ABA amount). In this case, Extend-model asks the function Dv-models (described in Section 4.2) for those combinations of influences on v that might be adequate for the question (i.e., satisfy adequacy constraints 5, 6, 7, and 8). In our example, Dv-models simply returns the only influence on guard cell ABA amount, the influence of the ABA transport rate. In general, Extend-model returns a set of new partial models, each the result of extending m with one of these combinations of influences.

To extend m with a combination of influences, Extend-model marks v as dependent, adds the influences on v to the model, and adds any new free variables. A free variable is added in the following cases:

- If the influencer of a new influence is not already in m, it is added as a free variable.
- If a variable in the activity preconditions of a new influence is not already in m, it is added as a free variable (to satisfy adequacy constraint 2).

Before adding a partial model to the agenda (whether the partial model is the initial node in the search or a successor returned by Extend-model), TRIPEL checks whether the model violates a monotonic constraint. If so, it is pruned from the search, since none of its extensions is an adequate scenario model.

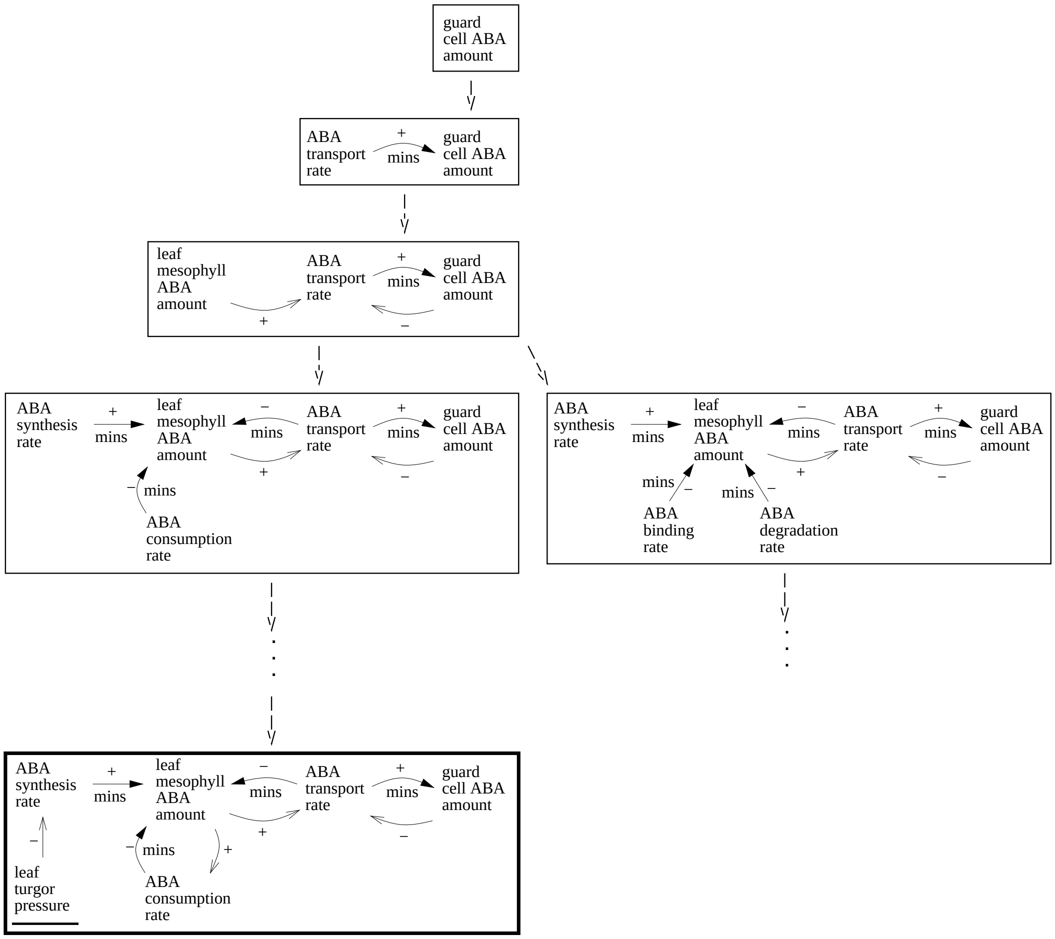


Fig. 8. The search graph for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" Boxes indicate partial models, and dashed arrows point from a partial model to its successors. The heavy box indicates the simplest adequate scenario model (the goal node returned by the model construction algorithm).

The partial model in our example does not violate any monotonic constraints, so it is added to the agenda.

The search ends with success when a simplest adequate scenario model is found. In contrast, the search ends with failure when the search agenda becomes empty, because this indicates that no adequate scenario model exists.

Figure 8 illustrates the search graph that TRIPEL generates for the example. The third node from the top has two successors because there are two adequate combinations of influences on leaf mesophyll ABA amount: the first includes the influence of ABA consumption, and the second includes the influences of ABA binding and ABA degradation that explain it (for simplicity, those influences were not shown in Figure 3).

4.2 Choosing Influences on Dependent Variables

A modeler must choose an adequate set of influences on each dependent variable in a model. In TRIPEL, this task arises in the function Extend-model and it is performed by the function Dv-models. After deciding to model a variable as dependent, Extend-model asks Dv-models for an adequate set of influences on the variable. As illustrated in Figure 7, the inputs to Dv-models include a system description, a prediction question, and a variable whose influences are desired.

There may be more than one adequate set of influences for a dependent variable. For instance, it may be possible to use either differential influences, which represent the dynamic effects of processes, or functional influences, which represent a quasi-static approximation of those processes. Also, one adequate set may contain the influences that explain an influence in another adequate set, as with the variable leaf mesophyll ABA amount in Figure 8. Dv-models must return each alternative set of influences for consideration by the model constructor. Extend-model creates a new partial model for each one, and the function Find-adequate-model tests each new partial model to see which ones represent a potentially adequate extension of the current partial model.

Section 3.1.3 specified the criteria for determining whether a set of influences on a dependent variable is adequate:

- The influences must be approximately complete; that is, they must represent all significant influencing phenomena at some level of detail (adequacy constraint 7).
- The influences must represent valid approximations (adequacy constraint 6).
- The influences must be mutually coherent (adequacy constraints 5 and 8).

Because adequacy constraint 5 requires the influences on a dependent variable to have the same type (i.e., differential or functional), Dv-models can separately consider sets of functional influences and sets of differential influences. After separately generating the adequate sets of influences that contain only differential influences and those that contain only functional influences, it returns the union of these two sets. The remainder of this section presents the algorithm for generating the adequate sets of influences for a given influence type (either one).

Given a system description, a prediction question, and a dependent variable to be modeled, Dv-models generates the adequate sets of influences of a given type (differential or functional) as follows:

(i) It generates every complete set of influences (of the specified type) on the dependent variable (i.e., those sets of influences that represent all

amount(pool(CO_2, leaves)) $\Leftarrow$ rate(CO_2-diffusion(atmosphere, leaves))
    significance preconditions: time-scale-of-interest $\geq$ seconds

amount(pool(CO_2, leaves)) $\Leftarrow$ rate(photosynthesis(leaves))
    significance preconditions: time-scale-of-interest $\geq$ minutes

amount(pool(CO_2, leaves)) $\Leftarrow$ rate(dark-reactions(leaves))
    significance preconditions: time-scale-of-interest $\geq$ minutes
Explanation(amount(pool(CO_2, leaves)) $\Leftarrow$ rate(photosynthesis(leaves)),
        amount(pool(CO_2, leaves)) $\Leftarrow$ rate(dark-reactions(leaves)))

Fig. 9. Influences on the amount of carbon dioxide in a plant's leaves. The first two are the maximally aggregate influences. The influence of photosynthesis is explained by the influence of the dark reactions (and not by any other influences).

the phenomena that affect the variable). Section 3.1.3 defined these as follows:
  - The set of all maximally aggregate influences of the specified type on the variable (i.e., those that do not explain any other influence) is complete.
  - The result of replacing an influence in a complete set with the set of all influences that explain it (as specified by the explanation relation) is a complete set.

(ii) From these sets, it removes any influences that are insignificant for the given question. Each resulting set is approximately complete (as defined in Section 3.1.3), so each satisfies adequacy constraint 7.
(iii) It discards any set that contains an influence that is invalid for the given question. Any such set of influences is inadequate because it violates adequacy constraint 6.
(iv) It discards any set that is incoherent. A set is incoherent if it violates adequacy constraint 8 (i.e., it includes two influences related by the explanation* relation).

For example, consider the influences shown in Figure 9 (previously shown as Figure 5) and assume that seconds is the time scale of interest. The algorithm proceeds as follows:

(i) As discussed in Section 3.1.3, there are two complete sets: (1) the first and second influences and (2) the first and third influences.
(ii) In the first set, the photosynthesis influence is insignificant (on the time scale of interest, seconds), so it is removed. Similarly, in the second set, the dark reactions influence is insignificant, so it is removed. This leaves two identical sets, each of which includes only the diffusion influence. Because the sets are identical, one is pruned and the other is passed to step iii.
(iii) The set does not include an invalid influence, so it is not discarded.
(iv) The set is coherent, so it is not discarded. Therefore, it is returned by Dv-models.

cross-section-area(stomates) $\leftarrow$ amount(pool(water, guard-cells))

cross-section-area(stomates) $\leftarrow$ amount(pool(ABA, guard-cells))
validity preconditions: time-scale-of-interest $\geq$ hours

cross-section-area(stomates) $\leftarrow$ amount(pool(CO_2, guard-cells))
validity preconditions: time-scale-of-interest $\geq$ hours

Explanation(cross-section-area(stomates) $\leftarrow$ amount(pool(ABA, guard-cells)),
cross-section-area(stomates) $\leftarrow$ amount(pool(water, guard-cells)))

Explanation(cross-section-area(stomates) $\leftarrow$ amount(pool(CO_2, guard-cells)),
cross-section-area(stomates) $\leftarrow$ amount(pool(water, guard-cells)))

Fig. 10. Influences on the cross sectional area of a plant's stomates. The second and third influences are each explained by the first influence.

As another example, consider the influences shown in Figure 10 and assume that hours is the time scale of interest. The algorithm proceeds as follows:

(i) The algorithm generates four complete sets: the second and third influences (the maximally aggregate influences), the first and third influences (since the first explains the second), the first and second influences (since the first explains the third), and the first influence alone (generated from either of the previous two).

(ii) None of the influences is insignificant, so no set is changed.

(iii) None of the influences is invalid, so no set is changed. However, if the time scale of interest were less than hours (e.g., seconds or minutes), any set containing the second or third influence would be discarded.

(iv) Two of the four sets are incoherent (i.e., they violate adequacy constraint 8): the one that includes the first and second influences, and the one that includes the first and third influences. These two sets are discarded, and Dv-models returns the two surviving sets: the one that includes the second and third influences, and the one that includes only the first influence.

Dv-models can recognize when there are no adequate sets of influences on a variable. For example, consider the influences shown in Figure 10, but suppose the first influence is not in the system description (i.e., that level of detail is missing). If the time scale of interest is less than hours (e.g., seconds or minutes), no set will survive step iii, so Dv-models will return the empty set (i.e., no adequate sets of influences). Thus, Extend-model will return the empty set (i.e., no successors); the partial model under consideration cannot be adequately extended. The key is that each influence represents a phenomenon to be modeled; if the phenomenon is significant, Dv-models must find a valid way of modeling it, either with that influence or an alternative level of detail.

4.3 Choosing Exogenous Variables

System boundary decisions arise in the successor function Extend-model. Given a system description, a prediction question, a partial model and one of its free variables, Extend-model asks the System Boundary Selector whether the variable can be exogenous. Such decisions are important; if the variable must be dependent, the model must be extended to include additional influences (on that variable) and variables (referenced by those influences).

The System Boundary Selector's response is either "yes" (the variable can be exogenous) or "no" (the variable must be dependent), interpreted as follows:

- If the response is "yes," then the variable can be exogenous in any extension of the partial model that does not contain additional variables.
- If the response is "no," then the variable must be dependent in every extension of the partial model. That is, no extension in which the variable is exogenous is an adequate scenario model.

Recall from Section 4.1 how Extend-model uses the System Boundary Selector's response. If the response is "no" (the variable must be dependent), Extend-model marks the variable as dependent and extends the partial model to include influences on it. In contrast, if the response is "yes" (the variable can be exogenous), Extend-model only marks the variable as exogenous if all other free variables can also be exogenous. The System Boundary Selector's response justifies Extend-model's actions.

The criteria for choosing exogenous variables were specified in Section 3.1.2:

- Adequacy constraint 3 — A variable in a scenario model cannot be exogenous if it is significantly influenced in the system description by another variable in the model.
- Adequacy constraint 4 — A variable in a scenario model cannot be exogenous if it is significantly influenced in the system description by a driving variable (other than itself if it is a driving variable).

Although these constraints are stated in terms of scenario models, they apply to partial models as well. As shown in Appendix A.1, both constraints are monotonic; that is, if a variable in a partial model violates one of the constraints, the variable cannot be exogenous in any extension of the partial model either. In this case, the System Boundary Selector can answer "no" (the variable cannot be exogenous). On the other hand, if a variable in a partial model satisfies both constraints, it can be exogenous in any extension with the same variables. (The variable might not satisfy adequacy constraint 3 in an extension with additional variables.) In this case, the System Boundary Selector can answer "yes" (the variable can be exogenous). Thus, the system boundary selection task simply requires the ability to test these two constraints.

These constraints can be tested by a graph connectivity algorithm. Recall from Section 3.1.2 that one variable significantly influences another if and only if there is an influence path (in the system description) leading from the first variable to the second and every influence in the path is valid and significant for the given question. Thus, a free variable in a partial model can be exogenous if and only if the graph algorithm finds no such path leading to the variable from any driving variable of the question or any other variable in the model.

However, it would be inefficient to run the graph algorithm anew for each system boundary decision. Each run of the graph algorithm will repeat much of the search performed by previous runs. To avoid this problem, TRIPEL performs a *system boundary analysis* before beginning the search for an adequate scenario model. The system boundary analysis determines all variables and influences that *might* be relevant to the question, and it computes and caches connectivity relations among the variables. These potentially relevant variables and influences constitute the space that would be repeatedly searched by the graph algorithm. The algorithm for system boundary analysis is given in Section 4.3.1.

The result of the system boundary analysis is a Boolean *connectivity matrix.* This matrix records the connectivity between every pair of potentially relevant variables. That is, the *ith* variable significantly influences the *jth* variable for purposes of answering the given question if and only if the (i,j) cell of the matrix contains a 1.

Once system boundary analysis is complete, TRIPEL begins its search for the simplest adequate scenario model as described earlier. Using the connectivity matrix, system boundary decisions that arise during model construction are trivial. A free variable in a partial model must be dependent if, according to the connectivity matrix, the variable violates adequacy constraint 3 or 4. In this case, the System Boundary Selector returns "no" (the variable cannot be exogenous). Otherwise, it returns "yes."

4.3.1 System Boundary Analysis

The variables in the connectivity matrix are called the *potentially relevant variables* because they include all variables that *might* be relevant to answering the question. More precisely, they include any variable that might be added to a partial model during model construction. Similarly, the *potentially relevant influences* include any influence that might be added to a partial model during model construction. We define the potentially relevant variables and influences as follows:

- The variables of interest are each potentially relevant.
- If a variable is potentially relevant, any influence on it that is valid and significant (for the given question) is a potentially relevant influence.
- The influencer of every potentially relevant influence is potentially relevant.
- Any variable appearing in the activity preconditions of a potentially relevant influence is potentially relevant.

This definition mirrors the steps that add variables and influences to partial models during model construction.

The System Boundary Selector finds the potentially relevant variables and influences using a breadth-first search through the influence graph. First, each variable of interest is marked as potentially relevant and placed on the search agenda. On each iteration of the search, a variable is removed from the agenda, and each valid, significant influence on that variable is marked as potentially relevant. For each such influence, its influencer and the variables in its activity preconditions are marked as potentially relevant. Each newly marked variable is placed on the agenda unless it had previously appeared on it. The search ends when the agenda is empty; the terminal variables in the search are those that are not significantly influenced (e.g., those that are regulated on time scales slower than the time scale of interest) and those that are significantly influenced only by variables discovered earlier in the search (i.e., through feedback loops). When the search ends, all potentially relevant variables and influences have been marked.

To illustrate this algorithm, consider the familiar question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" Part A of Figure 11 repeats a portion of the system description for this question. The search for potentially relevant variables and influences begins with the influences on guard cell ABA amount. The influences of transpiration on leaf mesophyll water (middle of left side) and water uptake on xylem water (lower left) are insignificant on the time scale of interest (minutes); removing these two influences disconnects the potentially relevant variables from the remaining variables and influences, including the feedback loop through transpiration, thus allowing TRIPEL to ignore those other variables and influences. Part B shows the result, the potentially relevant variables and influences for the example. For comparison, Part C shows the simplest adequate model for the question (as described in Section 4.1).

As illustrated by the example, the search for potentially relevant variables and influences will typically have to traverse only a fraction of the influence graph. In complex systems, such as plants, animals, and ecosystems, modularity arises from the widely disparate time scales at which processes cause change [2,28,40,47,49]. The result is a hierarchy of nearly decomposable subsystems; processes acting *within* a subsystem cause significant change quickly, while

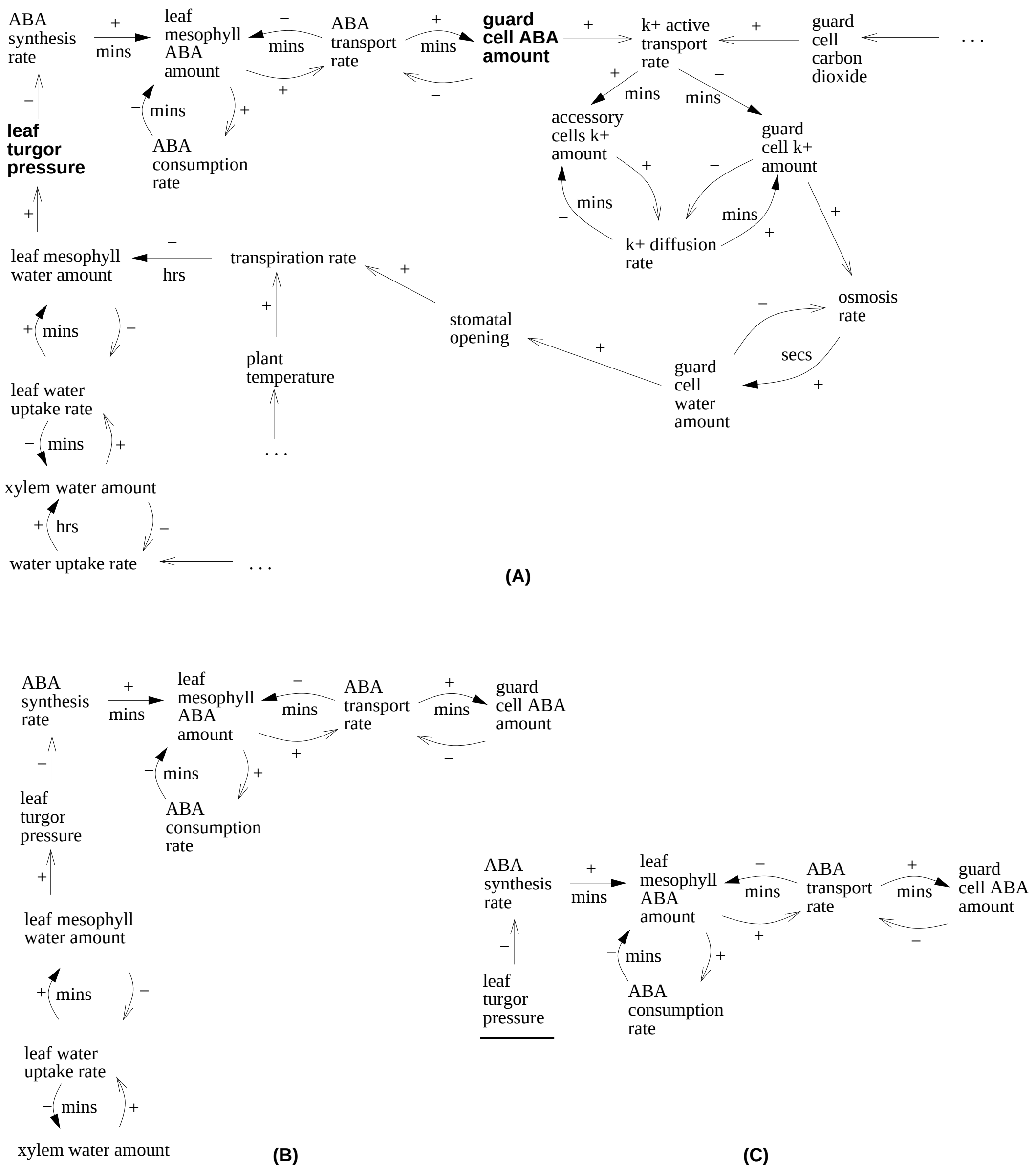


Fig. 11. **(A)** A portion of the system description for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" The driving variable and variable of interest are shown in bold. Ellipses indicate connections to the remaining variables and influences. Alternative levels of detail are not shown. **(B)** The potentially relevant variables and influences for the question. **(C)** The simplest adequate scenario model for the question.

processes acting *across* subsystems cause change more slowly [2,28,40,51]. The time scale of interest filters out influences that are significant only on slower time scales, thus isolating the variables of interest in their own nearly decomposable subsystem. The search for potentially relevant variables and influences is confined to this subsystem because the influences from other subsystems are insignificant.

After determining the graph of potentially relevant variables and influences, the System Boundary Selector constructs the connectivity matrix. First, it constructs the subgraph of the influence graph corresponding to the potentially relevant variables and influences. Analogous to the definition in Section 3.1.2, the nodes of this subgraph are the potentially relevant variables, and there is a directed edge from one variable to another if there is a potentially relevant influence whose influencee is the second variable and for which the first variable is the influencer or appears in the activity preconditions. The connectivity matrix is simply the adjacency matrix for the transitive closure of this subgraph. Given the subgraph, the connectivity matrix can be computed efficiently; the Floyd-Warshall algorithm computes it in $\Theta(n^3)$ time, where n is the number of nodes (potentially relevant variables) in the subgraph [7].

As discussed earlier, the System Boundary Selector decides whether a variable in a partial model can be exogenous by checking cells in the connectivity matrix. The connectivity matrix is guaranteed to include every variable for which a system boundary decision might be required, because the definition of potentially relevant variables and influences mirrors the steps that add variables and influences to partial models during model construction. Furthermore, the following theorem ensures that the connectivity matrix accurately reflects whether one variable significantly influences another.

Theorem 1 (Connectivity matrix is correct) *For a given system description and prediction question, cell* (i,j) *of the connectivity matrix contains a 1 if and only if the ith variable significantly influences the jth variable for that question.*

Proof See Appendix A.2. □

To determine whether a variable can be exogenous, the System Boundary Selector must ensure that the variable is not significantly influenced by any driving variable (adequacy constraint 4). However, the definition of potentially relevant variables does not ensure that every driving variable is potentially relevant, so some driving variables may not appear in the connectivity matrix. Nevertheless, variables in the connectivity matrix are only significantly influenced by other variables in the matrix. Therefore, when deciding whether a variable can be exogenous, the System Boundary Selector knows that the variable is not significantly influenced by any driving variable that is not in the matrix.

4.3.2 Extensibility

Choosing exogenous variables is an important part of constructing a simple yet adequate model. Our design encapsulates such decisions in the System Boundary Selector, an independent module of TRIPEL. This allows changes in the criteria for choosing exogenous variables without requiring changes in the model construction algorithm. Similarly, the System Boundary Selector does not depend on the particular criteria for determining whether an influence is valid and significant. TRIPEL uses a time scale of interest, but other criteria could be used instead or in addition.

For complex systems, in which variables are highly interconnected, the ability to recognize insignificant influences is crucial to achieving a suitable system boundary. This ability is also required to keep the number of potentially relevant variables (and hence the size of the connectivity matrix) small. Therefore, the performance of the System Boundary Selector will improve as more sophisticated significance criteria are incorporated into TRIPEL, as will be discussed in Section 5.

4.3.3 The Role of Each Adequacy Constraint

The adequacy constraints from Section 3.1 serve different roles in the model construction algorithm. Adequacy constraint 1 (include variables of interest) is used to construct the initial partial model on the agenda, and adequacy constraint 2 (include variables in activity preconditions) is used by Extend-model to identify new free variables for a partial model being extended. These constraints are both *propagation constraints*: when violated in a partial model, they specify the elements that must be added for the constraint to be satisfied (analogous to constraint propagation).

Some monotonic constraints serve as filters. As shown in Appendix A.1, adequacy constraints 9 (entities coherent) and 10 (entities compatible with driving variables) are both monotonic, and adequacy constraint 11 (variables of interest differentially influenced) is monotonic when applied to models that have no free variables. A partial model is added to the search agenda only if it satisfies these constraints. (If it has free variables, it need only satisfy the first two constraints.)

The remaining constraints, although monotonic (as shown in Appendix A.1), are folded into the subroutines of Find-adequate-model. Adequacy constraints 3 (exogenous variables independent of model) and 4 (exogenous variables independent of question) are tested by the System Boundary Selector, and adequacy constraints 5 (influences homogeneous), 6 (influences valid), 7 (influences complete), and 8 (influences not redundant) are enforced by the function Dv-models.

For extensibility, TRIPEL is designed to easily accommodate new monotonic constraints and propagation constraints. This allows TRIPEL to incorporate additional sophistication in its modeling criteria, such as new criteria for determining whether models are coherent, without changes in its model construction algorithm.

4.4 Properties of the Model Construction Algorithm

To ensure an efficient search for a solution, a search algorithm must avoid redundancy. Typically, a graph search algorithm avoids redundancy by maintaining a record of nodes it has visited. However, Find-adequate-model does not keep a record of partial models that it has visited because of the following theorem.

Theorem 2 (Search is not redundant) *In the search graph constructed by* Find-adequate-model*, a given partial model cannot be reached via more than one path from the initial partial model.*

Proof A partial model has multiple successors only when one of its free variables is chosen as dependent (by definition of Extend-model). Each successor in this case contains a different set of influences on that variable. Since an extension of a partial model cannot change the influences on that model's dependent variables, no two successors of a partial model can share a common extension. Thus, if a partial model is viewed as representing itself and all its extensions, its successors represent disjoint subsets of its extensions. Viewed this way, Find-adequate-model starts with a single set (the initial partial model) and repeatedly splits one set into disjoint subsets. Therefore, it is not possible for any two partial models in the search graph to have a common descendant. □

Thus, Find-adequate-model is a version of the well-known "split and prune" search algorithm [41], and the search graph it constructs is a tree. Subsequent theorems and proofs rely on this "split and prune" view of the algorithm.

Conceptually, Find-adequate-model operates by repeatedly pruning parts of the search space from consideration. When each iteration of the while loop begins, part of the search space has been pruned from consideration and part remains. Specifically, the partial models on the agenda, along with all their extensions, are still under consideration. This set of partial models is the **consideration set**. The following theorem ensures that the search will always terminate by showing that each iteration of the while loop decreases the size of the consideration set.

Theorem 3 Find-adequate-model *always terminates.*

Proof Every individual step in the algorithm always terminates because the system description is finite. Thus, Find-adequate-model will terminate if its while loop terminates. For a finite system description, there are only a finite set of unique partial models, so the initial consideration set is finite. Every iteration of the while loop removes the simplest partial model on the agenda from the consideration set, decreasing its size. Therefore, the while loop must eventually terminate. □

Most importantly, Find-adequate-model is an *admissible* search algorithm. A search algorithm is **admissible** if it is guaranteed to return an optimal solution whenever a solution exists [41]. Find-adequate-model is admissible because it is guaranteed to return a simplest adequate scenario model whenever an adequate scenario model exists. Conceptually, the algorithm is admissible because it uses the following strategy:

- From its initial consideration set, which includes all adequate scenario models, it repeatedly prunes away models until only a single scenario model (if any) remains.
- It never prunes a scenario model unless either (1) the model is inadequate for the question or (2) if the model is adequate, there is an adequate scenario model still in the consideration set that is at least as simple.

Theorem 4 (Model construction algorithm is admissible) *Given a system description and a prediction question for which some scenario model is adequate,* Find-adequate-model *will return a simplest adequate scenario model.*

Proof See Appendix A.3. □

4.5 Previous Model Construction Algorithms

Falkenhainer and Forbus [12] take a knowledge-based approach to model construction. Each model fragment has associated "assumptions," symbolic labels that characterize the phenomena it represents and its level of detail. Domain knowledge provides constraints on the use of assumptions:

- Assumptions are organized into "assumption classes." The assumptions in an assumption class represent mutually incompatible modeling alternatives.
- The domain knowledge provides domain-specific constraints among assumptions, such as that one assumption requires another.

- For each assumption class, the domain knowledge must specify the scenario conditions under which it is relevant. An adequate scenario model must include one alternative from each relevant assumption class.

In their modeling task, a question specifies terms (e.g., variables) of interest. Their objective is to find a minimal set of assumptions that satisfy all the domain constraints and ensure that the model includes the terms of interest. They accomplish this with a constraint satisfaction algorithm ("dynamic constraint satisfaction" [37]).

In their framework, unlike ours, most criteria for model adequacy are implicit in the domain knowledge. Because they have no counterpart of our function Dv-models, they require the domain knowledge to group influences into coherent bundles (model fragments). Also, our algorithm does not require the domain knowledge to provide relevance conditions or domain-specific constraints among modeling alternatives. Formulating such "modeling knowledge" so that it ensures an adequate model could be a difficult, error prone task. Moreover, it is not clear how to encode some constraints, such as adequacy constraint 11 (variables of interest differentially influenced), in their language. In place of domain-specific modeling knowledge, TRIPEL relies on domain-independent criteria that specify when a model is adequate. When a model violates these criteria, the particular violation tells TRIPEL how to extend the model. Removing the need for domain-specific modeling knowledge has been a driving motivation for our work.

A second approach to model construction is to start with the most detailed model and repeatedly simplify it. Williams's method for generating a "critical abstraction" [54] simplifies the detailed model in three ways: (1) the method removes influences on which the variables of interest do not causally depend (such influences are never introduced into a scenario model by our algorithm), (2) the method algebraically eliminates certain intermediate variables if they are neither driving variables nor variables of interest, and (3) the method algebraically abstracts quantitative details that are not needed to answer the question. Yip's modeling algorithm [56,57] simplifies the detailed model by removing insignificant terms in the equations (analogous to eliminating insignificant influences). Nayak's modeling algorithm [38] repeatedly simplifies the detailed model by (1) eliminating irrelevant phenomena or (2) replacing one model fragment with another that represents a "causal approximation" of it (typically, this corresponds to omitting some of the influences in the original model fragment).

For complex systems, which include many phenomena that can be described at many levels of detail, the approach of repeatedly simplifying the most detailed model is impractical. To find a simplest adequate model of a complex system, the number of elements that would have to be removed from the most-detailed model is far greater than the number of elements that would have to be added to an empty model. For this reason, TRIPEL takes the latter approach.

Recent work by Nayak and Joskowicz [39] addresses the impracticality of simplifying a most-detailed model. Their method generates an initial, overly detailed model and then applies Nayak's algorithm [38] to repeatedly simplify it. Their hope is that the initial model will be far simpler than the most-detailed model. Their method requires the domain knowledge to provide rules that specify the ways in which different components of the physical system can interact ("component interaction heuristics"). Starting with a model consisting of the driving variables and variables of interest, their algorithm constructs the initial, overly detailed model by repeatedly adding aspects of the physical system that can interact with those aspects currently in the model. Unfortunately, when applied to complex systems such as a plant, their approach will result in a very detailed initial model, because most aspects of a complex system interact either directly or indirectly. To achieve a simpler initial model, they will require component interaction rules that are sensitive to the question, the available levels of detail in the system description, and criteria for determining significance of interactions, just as our adequacy criteria and model construction algorithm are.

Nayak [38] proves that his model simplification algorithm will reach a simplest adequate model in time polynomial in the size of the system description. However, his results depend on several assumptions that are inappropriate for the modeling task we address. First, as discussed in Section 3.2, his simplicity criteria leave many models incomparable, even though some of these models are intuitively much simpler than others. His algorithm exploits his simplicity criteria by using a hill-climbing search. If more of the models were comparable, as they are under our simplicity criterion, this search strategy would not be guaranteed to find a simplest adequate model. Second, his hill-climbing search strategy requires that every phenomenon in the system description has its own set of modeling alternatives and that the modeler can choose an alternative for modeling one phenomenon independent of how the other phenomena are modeled. However, our modeling framework is built around aggregation of phenomena: one entity can aggregate several other entities, and one influence can aggregate several other influences. Aggregation hierarchies are crucial to achieving simple models of complex systems, but they violate Nayak's assumption. We investigated the possibility of extending Nayak's approach to handle aggregation, but it would require assuming that, for every level of description for a phenomenon, there is a compatible level of description for every related phenomenon; this requires a level of completeness in the system description that seems impractical. Finally, his proofs currently place restrictions on the use of influences in model fragments, and these restrictions would seriously diminish the advantages of using influences as the building blocks for models.[10]

TRIPEL's algorithm for model construction is most similar to the one used by Iwasaki and Levy [23]. Their algorithm starts with a partial model consisting of the variables of interest, and it repeatedly extends the model to include the influences on free variables. There are three major differences between the two algorithms. First, their algorithm has no method for automatically choosing exogenous variables. Second, their algorithm has no counterpart of our function Dv-models; the person encoding the model fragments and the constraints among assumptions must ensure that each compatible combination of model fragments yields an adequate set of influences. Finally, like Nayak [38], their simplicity criteria leave many models incomparable, even though some of these models are intuitively much simpler than others; if more of the models were comparable, as they are under our simplicity criterion, their search strategy would not necessarily find a simplest adequate model.[11] In addition to these primary differences, there are other differences:

- They allow the activity preconditions of a model fragment to include predicates in addition to inequalities among variables. Thus, while TRIPEL's algorithm always extends a model by considering the influences on a free variable, their algorithm can also extend a model to include influences on these predicates. This is a natural and useful extension of TRIPEL's approach.
- In their representation, influences in the system description do not have a causal direction. The direction of causality is only assigned after the model is complete, using a causal ordering algorithm [24]. This requires their algorithm to extend models to include all variables that could "possibly influence" the chosen free variable, which will generally result in larger models with more irrelevant phenomena. The question of whether influences should be given a causal direction before model construction begins is still open [16]. However, our approach has worked well in the plant physiology domain, and we expect similar success in other domains. Elsewhere [44], we argue that, regardless of the domain, most influences can be given a causal direction before model construction, and we show how TRIPEL could be extended to handle influences for which this is not possible.
- Their algorithm relies on a strong assumption about the system description (the "library coherence assumption") to guarantee that the equations in an adequate scenario model are complete (i.e., have the same number of

[10] Nayak (personal communication) believes that the proofs could be extended to accommodate our use of influences.

[11] In fact, despite Levy's proof [34], their algorithm does not necessarily find a simplest adequate model even by their own criteria; their algorithm adds more elements to models than are required by their definition of an adequate model. To repair the proof, they are currently modifying the algorithm and extending the adequacy definition (Levy, personal communication).

equations as dependent variables). In contrast, our modeling algorithm is designed to ensure that the models it constructs are complete.

- Their algorithm is guaranteed to run in time polynomial in the size of the system description [34]. However, that result does not apply to our task since it relies on the same assumptions as the similar result of Nayak [38] discussed earlier.

Several people have explored an approach to model construction called "discrepancy-driven refinement" [1,3,52]. After constructing an initial model, the modeler compares its predictions against the known behavior of the system. Discrepancies suggest refinements to the model, and the process is repeated until a sufficiently close match is obtained. We have not used this approach because we do not assume that the correct behavior is known. However, when it is, these algorithms are complementary to TRIPEL, because TRIPEL provides a more sophisticated approach to constructing the initial model than these algorithms currently use. Thus, TRIPEL could serve as a valuable subroutine in these algorithms.

5 Empirical Evaluation

5.1 Introduction

There are two important issues that must be empirically evaluated. The first issue concerns the quality of the models TRIPEL constructs. Section 4.4 proved that TRIPEL always returns a simplest adequate model when there is one. However, the proof says nothing about whether the definition of a simplest adequate model matches our intuitive notions of simplicity and adequacy. The second issue concerns TRIPEL's efficiency. For complex systems, the system description and the space of possible models are very large. TRIPEL will only be practical if it can cope with such complexity. This section describes our empirical evaluation of these two issues.

Previous automated modeling programs were tested on handcrafted examples. That is, the program's designers built a knowledge base and constructed examples to demonstrate the program's capabilities. Our goal was a more rigorous evaluation that could expose the strengths and weaknesses of our methods. To accomplish this goal, we evaluated TRIPEL using knowledge and questions constructed independently by a domain expert.

The knowledge was provided by the Botany Knowledge Base (BKB) [42]. The BKB is an ideal test bed for evaluating TRIPEL for three reasons. First, its knowledge is extensive. It currently contains about 200,000 facts covering plant anatomy, physiology, and development. Second, it was independently developed by a domain expert, whose main objective was a faithful and unbiased representation of botany knowledge. Finally, it was developed to support a wide range of tasks besides prediction; that is, the BKB encodes fundamental, textbook knowledge, and the representation of that knowledge was not chosen to facilitate its use for any single task such as prediction. (Lester and Porter [31,33,32] describe results on using the BKB to answer other types of questions.)

Using the BKB, the domain expert constructed a system description for a prototypical plant and its environment (i.e., surrounding soil and atmosphere). Most elements of the description were generated via automated inference (i.e., inheritance and inference rules) from the general principles in the BKB. In addition, the expert manually added missing elements and repaired erroneous elements. The resulting system description includes 691 variables and 1507 influences among them. It includes 47 different spaces (e.g., roots, stems, leaves) and 172 different pools of substances in those spaces (e.g., oxygen in the leaves). It includes 313 processes, covering water regulation, metabolic processes like photosynthesis and respiration, temperature regulation, and transportation of gases and solutes. Moreover, the variables, influences, spaces, pools and processes cover many different levels of detail. Thus, this system description meets the most important requirement for evaluating TRIPEL: it includes many phenomena at many levels of detail.

Next, we asked the domain expert to construct a large set of prediction questions concerning a prototypical plant. From these, we randomly chose a small subset to use for evaluating TRIPEL. For each of these questions, the expert generated his answer (model and predictions) before looking at TRIPEL's model.[12] Next, he evaluated TRIPEL's model for each question by comparing it to his own. Finally, after he evaluated TRIPEL's performance on the entire subset of questions, he presented his assessment and discussed the knowledge he used to reach his conclusions. Appendix B lists the questions used in the evaluation.

5.2 Adequacy and Simplicity

The evaluation results show that TRIPEL is very effective at constructing adequate models. In every case where TRIPEL was given a question (including an appropriate time scale) for which an adequate scenario model exists, it constructed an adequate model. That is, according to the domain expert, each

[12] One question had to be thrown out, because the expert was not sure how to answer it. Therefore, he could not, with confidence, determine which elements of the system description were relevant.

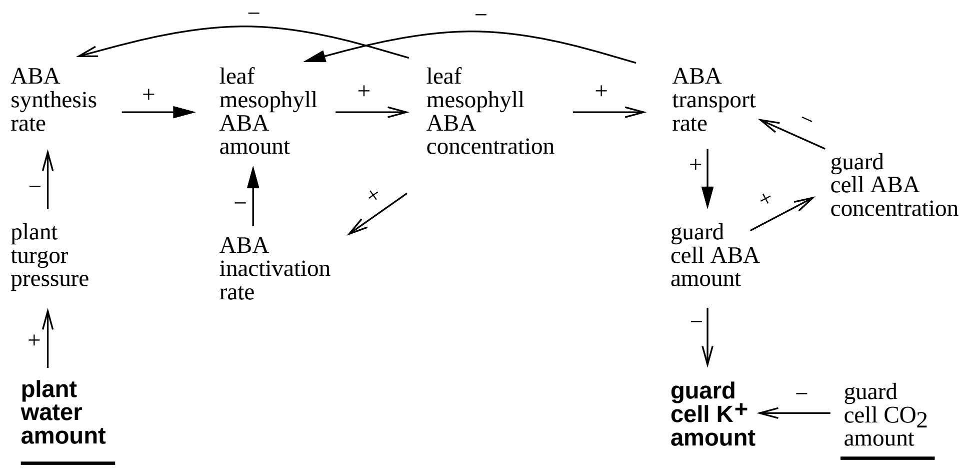


Fig. 12. The model TRIPEL constructed to answer the question "How does a decreasing amount of water in a plant affect the amount of K^+ in its guard cells?"

of these models includes all the information needed to generate the right predictions and explanations. For instance, Figure 12 shows the model TRIPEL constructed to answer the question "How does a decreasing amount of water in a plant affect the amount of K^+ in its guard cells?" The model correctly shows the mechanisms by which decreasing plant water causes increased synthesis of the ABA hormone, and how ABA is transported to the guard cells, causing potassium ions to leave.

The expert also assessed whether these models include irrelevant information. Column three of Table 1 shows the number of variables and influences in each model, and column four shows how many of them the expert judged irrelevant to answering the corresponding question. In comparison with the size of the system description (691 variables and 1507 influences), these models are quite small, and most of the models have few if any irrelevant elements.

The cases where TRIPEL included irrelevant elements are most interesting, because they suggest opportunities for improving its methods. The raw numbers of irrelevant elements are somewhat misleading; one error in TRIPEL's judgement typically forces it to include many irrelevant elements. Most of the irrelevant elements in these models were included because TRIPEL overestimated the significance of an influence or influence path. Most of TRIPEL's errors result from three differences between TRIPEL's criteria for significance and the expert's criteria:

- The expert uses a finer gradation of time scales than those in the system description. For each time scale in the system description (e.g., minutes or hours), the expert considers a variety of more specific time scales (e.g., a few minutes versus many minutes). For example, when the expert chooses "few minutes" as the time scale of interest, he ignores processes operating

#	Time Scale of Interest	TRIPEL's Model (variables, influences)	Irrelevant Elements (variables, influences)
i	instantaneous	6, 5	none
ii	hours	6, 7	none
iii	minutes	11, 14	none
iv	hours	16, 25	5, 8
v	minutes	19, 28	6, 9
vi	hours	25, 40	none
vii	minutes	25, 41	none
viii	minutes	36, 60	18, 34
ix	minutes	41, 70	29, 55
x	minutes	82, 147	64, 121

Table 1
The number of irrelevant elements in TRIPEL's models. Each row represents a question. The first column shows the question number (corresponding to Appendix B). The third column shows the number of variables and influences in the simplest adequate model found by TRIPEL. The last column shows the number of these variables and influences that are not relevant to answering the question.

on a time scale of many minutes. Because the system description does not distinguish these two time scales, TRIPEL treats the slower processes as significant.

- When assessing an influence's significance, TRIPEL does not try to anticipate the behavior of the physical system in the scenario. In contrast, the expert sometimes determines that an influence is insignificant because it is significant only under conditions that will not arise in the scenario. For example, oxygen is rarely a limiting reactant for respiration; therefore, when the expert can see that the driving conditions of a question will not cause oxygen to become limiting, he omits the influence of oxygen on respiration.
- The expert's criteria for determining whether an influence path is significant are more sophisticated than TRIPEL's. As discussed in Section 3.1.2, TRIPEL judges an influence path as significant for a given question if each influence in the path is valid and significant on the time scale of interest. However, the expert's reasoning indicates that an influence path might be significant only on a slower time scale; the expert reasons about extra time lags due to the length of the path or the spatial distance it covers. Therefore, TRIPEL sometimes includes feedback loops that the expert recognizes as insignificant.

In summary, TRIPEL's performance on these ten questions indicates that it is capable of constructing simple, adequate models despite the complexity of the system description. The most important area for improvement is in TRIPEL's criteria for recognizing insignificant influences and influence paths. TRIPEL is designed to easily incorporate additional significance criteria without requiring changes to the model construction algorithm, so the main challenge for future research is in formulating the criteria. We discuss this issue further in Section 6.

In addition to the ten questions discussed so far, we also tested TRIPEL on four questions for which the BKB cannot provide an adequate scenario model. This situation can arise in several ways: (1) the driving conditions of the question may have no significant effect on the variables of interest (question xii in Appendix B), (2) the system description may be missing the processes by which the driving conditions affect the variables of interest (question xi), or (3) the question may require a model of several phenomena for which the system description does not provide compatible levels of detail (questions xiii and xiv). The first case reflects a property of the question, while the other cases reflect gaps in the system description.

Ideally, TRIPEL should recognize that no adequate scenario model exists for these four questions. It did correctly report such a conclusion for questions xiii and xiv. However, for the other two, TRIPEL returned what it believed to be an adequate model. In each of these two cases, TRIPEL found what it identified as a significant influence path relating the driving conditions and variables of interest. In contrast, the expert judged these paths insignificant. As discussed earlier in this section, the expert's assessment differs from TRIPEL's because he additionally considers cumulative delays along an influence path. Thus, by extending TRIPEL's criteria to include such considerations, we can improve its ability to recognize inadequate models as well as irrelevant phenomena.

5.3 The Importance of a Time Scale of Interest

A time scale of interest is an important source of TRIPEL's power. TRIPEL uses the time scale of interest to identify insignificant influences, allowing it to prune them from its models. Its ability to identify insignificant influences is also a crucial part of its ability to choose appropriate exogenous variables. Finally, the time scale of interest allows TRIPEL to use some levels of detail that are not valid on faster time scales (e.g., influences representing quasi-static approximations). Clearly, TRIPEL's ability to recognize insignificant influences and valid approximations plays an important role in its success.

To quantify the importance, we ran TRIPEL without a time scale of interest on the ten questions for which an adequate model exists. Without a time scale of interest, all influences are treated as significant, and influences that are valid only for certain time scales are treated as invalid (forcing TRIPEL to use more-detailed influences instead). This experiment yielded two observations. First, the simplest adequate model that TRIPEL found for each question was significantly larger; on average, each model included 65 more variables than when TRIPEL exploited a time scale of interest. Second, there were two questions for which TRIPEL determined that no adequate model exists, even though it found an adequate model when using a time scale of interest. The reason is simple: without using a time scale of interest, TRIPEL is forced to model more phenomena, so it is more likely to need two phenomena for which the system description does not provide compatible levels of detail. Thus, a time scale of interest not only results in smaller models, but also makes TRIPEL less sensitive to gaps in the system description.

5.4 Efficiency

5.4.1 Model Construction

In the theoretical worst case, the model construction algorithm (Find-adequate-model) has a running time that is exponential in the size of the system description. In practice, however, it performs quite efficiently. For the expert's questions where TRIPEL constructed an adequate model, column 2 of Table 2 shows the amount of time the algorithm took to find a simplest adequate model.[13] These numbers are consistent with our informal experience using TRIPEL.

To appreciate TRIPEL's efficiency, consider the size of the search space. Any combination of influences defines a legal scenario model: the model's dependent variables are the influencees of the influences, and all other variables referenced by the influences are exogenous. Furthermore, each of these scenario models is different since they include different influences. Thus, since the system description for a prototypical plant includes over 1500 influences, the search space includes over 2^{1500} possible scenario models.

TRIPEL searches this space efficiently because it avoids generating most of these models. By pruning a partial model, TRIPEL avoids generating any of its extensions. Therefore, one way to measure the efficiency of model construction is to determine how many partial models TRIPEL explicitly generates and considers for each question. Find-adequate-model terminates when it finds an adequate model, so all the partial models that it generates fall in one of three classes: the simplest adequate model, models that were pruned by monotonic

[13] The timing data pertains to Harlequin Lispworks 3.2 Common Lisp running on a DEC 3000/500 workstation.

#	Time (seconds)	Models Pruned	Models Left on Agenda
i	.01	1	0
ii	.04	4	0
iii	.1	13	3
iv	.2	11	9
v	.6	14	18
vi	1.3	60	27
vii	.8	10	9
viii	11	120	49
ix	2	45	14
x	80	740	121

Table 2
The efficiency of model construction. The first column shows the question number. The second column shows the amount of time TRIPEL spent during model construction (i.e., the amount of time to execute the function Find-adequate-model). The third column shows how many partial models TRIPEL generated and pruned with monotonic constraints. The fourth column shows how many partial models were left on the agenda when TRIPEL found a simplest adequate model.

constraints, and models left on the agenda at termination. For each question, columns 3 and 4 of Table 2 show the number of partial models falling in the latter two classes. The numbers indicate that TRIPEL only generates a manageable number of partial models, especially compared to the size of the search space.

5.4.2 System Boundary Analysis

Before calling Find-adequate-model, TRIPEL performs a system boundary analysis. As described in Section 4.3.1, system boundary analysis consists of two steps. First, TRIPEL uses a breadth-first search to identify the potentially relevant variables and influences. Second, it uses the Floyd-Warshall transitive closure algorithm to compute a connectivity matrix. The time required to perform the system boundary analysis is dominated by the transitive closure algorithm, which requires $\Theta(n^3)$ time (where n is the number of potentially relevant variables) [7].

One of the biggest surprises during the empirical evaluation was the number of potentially relevant variables TRIPEL found for each of the expert's questions. The number is nearly independent of the question; it depends primarily on the time scale of interest. When the time scale of interest is seconds or faster, there are one or two dozen potentially relevant variables, and system boundary analysis finishes in less than one second. However, when the time scale of interest is minutes, there are always about 450 potentially relevant variables, and there are always about 650 on a time scale of hours. Since the entire system description includes 691 variables, these numbers represent a significant fraction. Such a high number of potentially relevant variables makes the transitive closure algorithm expensive; the algorithm requires about 30 minutes to handle 450 variables and about two hours to handle 650. Even though we could expect significant improvements from an optimized implementation in a more efficient language, this situation is unacceptable.

The root of the problem is TRIPEL's criteria for determining whether an influence path is significant, as already discussed in Section 5.2. As long as every influence in a path is valid and significant, TRIPEL considers the path significant. When identifying potentially relevant variables and influences, this criterion causes TRIPEL to include variables that influence the variables of interest through very long paths. The expert can tell that these paths are insignificant because he considers cumulative delays along influence paths. Thus, the same problem that causes TRIPEL to include irrelevant elements in models causes inefficiency during system boundary analysis.

There is a simple solution to this problem for some cases. Often, a wide variety of questions can be answered from the same system description; each question is distinguished by different driving conditions and variables of interest. This is the case with all the expert's questions concerning a prototypical plant. It would also be the case for a chemical processing facility, the human body, or an ecosystem. Given a system description, TRIPEL can generate a complete connectivity matrix (i.e., including all variables) for each possible time scale. Then, to answer a question, system boundary analysis simply selects the matrix corresponding to the time scale of interest. We have implemented this strategy, and it allows plant physiology questions to be answered very quickly.

Nevertheless, this strategy has limitations. It does not allow TRIPEL to efficiently answer questions until all necessary connectivity matrices are built. Moreover, this strategy requires a complete system description, preventing the possibility of generating only those parts of the system description needed for model construction [44]. To make system boundary analysis efficient, as well as to improve other areas of TRIPEL's performance, we must improve TRIPEL's criteria for determining whether an influence path is significant.

6 Future Work

6.1 Modeling Criteria

The empirical results show that TRIPEL is effective at building simple, adequate models of complex systems. Nonetheless, its criteria for making decisions can be improved.

6.1.1 Significance Criteria

The ability to recognize insignificant influences is an important source of power in modeling. Currently, TRIPEL uses a time scale of interest to determine whether an influence is significant. However, its model construction algorithm does not depend on this particular criterion; TRIPEL can be extended to include other criteria as well. The evaluation suggests that additional criteria would make TRIPEL more efficient and would reduce the number of irrelevant elements in its models.

In addition to time scale, human modelers use other criteria to recognize insignificant influences. For example, the concentration of a reactant significantly influences the rate of a chemical reaction only if the reactant is limiting (i.e., not available in abundance); if the reactant will not become limiting in the context of the question, the influence can be ignored. In other cases, one influence can be ignored because, in the context of the question, it will be dominated by other influences. Ultimately, TRIPEL should take into account the time scale of interest, desired accuracy, expected range of behavior, and dominance relations to determine which influences are significant. Similar comments apply to the problem of determining whether an influence is valid.

Applied mathematicians have developed formal (albeit heuristic) methods for recognizing insignificant terms (i.e., influences) in equations [25,35]. These methods are interesting because they combine the considerations mentioned above. In these methods, the modeler first "scales" the equations; that is, he uses scales of interest (e.g., a time scale of interest) to put the equations in nondimensional form so that the order of magnitude of each term is apparent. Next, the modeler drops terms whose order of magnitude is very small. Finally, the modeler solves the equations and checks whether the discarded terms are in fact negligible. Yip [56,57] has automated this procedure. However, Yip's program starts with a complete, detailed set of equations and repeatedly simplifies them. A program that could use such methods to construct an initial model would be even more valuable. Ling's MSG program [36] is a promising start in this direction.

As discussed in Section 5, TRIPEL could also benefit from more sophisticated criteria for determining whether an influence path is significant. Currently, it treats an influence path as significant if every influence in the path is valid and significant on the time scale of interest. The evaluation suggests that TRIPEL should also consider extra time lags due to the length of the path or the spatial distance it covers. To maintain efficiency while searching for significant influence paths, TRIPEL uses graph algorithms (such as the Floyd-Warshall algorithm) that do not record each path from one variable to another. However, these algorithms are based on a very general algebraic framework (closed semirings) [7] that allows them to efficiently summarize the properties of paths from one variable to another. Like the expert that evaluated TRIPEL, these algorithms can use properties of paths such as length and spatial distance in assessing whether one variable significantly influences another. Determining how these factors should be used in the assessment is an important area for future work.

6.1.2 Coherence Criteria

Although the explanation and encapsulates relations are typically sufficient for determining coherence of models, they are not, by themselves, sufficient in general. For example, a plant can be decomposed into roots, stems, and leaves or, alternatively, into apoplast (roughly, the network of dead parts of the plant) and symplast (roughly, the network of living parts of the plant). The pool of water in the roots and the pool of water in the symplast are not comparable by the encapsulates relation, since neither encapsulates the other, yet they seem mutually incoherent. A similar problem arises with influences; two influences may represent overlapping phenomena, yet neither explains the other.

One solution is to extend these two relations to represent multiple decompositions of entities and influences. Given a method for recognizing that two entities or two influences in a model come from incompatible decompositions, monotonic constraints can be implemented to prune such models. As stated earlier, TRIPEL can incorporate new monotonic constraints without any other changes, so the main challenge is simply to formalize the coherence criteria.

6.2 Simulation

6.2.1 Qualitative Simulation

TRIPEL has been integrated with a qualitative simulation program, which simulates TRIPEL's models to generate predictions. Given a question, TRIPEL constructs a model and passes the model to the Qualitative Process Compiler (QPC) [14]. QPC converts the model to a set of qualitative differential equations, and it simulates the equations, using the QSIM program [27,29], to generate the desired predictions. We chose to use a qualitative simulator rather than a numerical simulator because the BKB does not include quantitative details.

We have run QPC on many of TRIPEL's models, including those models from the evaluation that the expert judged adequate. In all our experiments, when the model is relatively simple (i.e., 15 or fewer variables), QPC predicts a unique behavior, the one predicted by the expert.[14] However, more complicated models result in many possible behaviors; although these models include all relevant influences, the qualitative information provided by the BKB is not sufficient to uniquely determine the behavior.

The ambiguity for the larger models can be eliminated without modifying TRIPEL. One solution is to incorporate quantitative information into the BKB. QPC and QSIM can exploit quantitative information to reduce ambiguity. We have been pursuing an alternative approach: the BKB could be extended to specify those influences that typically dominate other influences, and QPC could be extended to use this information to reduce ambiguity. We believe such information will be easy to obtain and encode, and that it will allow QPC to generate the desired predictions from the plant physiology models TRIPEL constructs, but more work remains.

6.2.2 Numerical Simulation

The algorithms described in this paper should provide a foundation for building numerical models as well as qualitative models. The issues addressed in this paper arise in both cases. However, while TRIPEL has been used to generate qualitative models, it has not been used to generate numerical models.

There are two possible ways to generate numerical equations from influences. First, the domain knowledge can provide a numerical equation for each useful combination of influences on a variable. Forbus and Falkenhainer [17] have successfully used that approach. Second, each influence can specify how it combines with other influences, such as whether it is an additive term, a multiplicative term, or otherwise. After the model is constructed, equations can be generated using these specifications. Farquhar [13] has successfully used this approach for limited types of equations, and it appears feasible for other

[14] There are two extensions to QPC that help it predict a unique behavior. First, while there may be multiple completions of the initial state, we modified QPC to automatically choose the initial state closest to equilibrium (i.e., the state with the most steady variables). This is the most natural interpretation of our prediction questions in most cases. Second, we allowed QPC to use a QSIM extension developed by Clancy and Kuipers [5] that abstracts the behavior of chattering variables. Typically, chatter is irrelevant to answering our questions.

types as well.

Thus, although TRIPEL has not been used to construct numerical models, there are no apparent limitations that prevent such an application. Although we expect that constructing numerical models will raise some additional issues, we believe that TRIPEL will provide an appropriate framework for addressing them.

6.3 Questions that Require Multiple Models

To answer a question, TRIPEL builds only a single scenario model. This approach works in most cases, aided by QPC's ability to change the model somewhat during simulation as its influences become active or inactive (as discussed on page 14). However, some questions require using a combination of models that differ in more fundamental ways (e.g., models with different time scales). Most modeling algorithms, including ours, cannot construct such combinations, although Iwasaki [21] has begun to explore the issues.

6.4 Other Domains

TRIPEL has been designed to apply to a wide variety of domains. We have been careful to avoid representations and methods that lacked such wide applicability. Although we have been influenced by the modeling issues that arise in plant physiology, we have also been guided by the practices of human modelers in ecology, economics, several branches of engineering (chemical, electrical, and mechanical), and other areas of biology. We have also tried to ensure that TRIPEL handles the issues addressed by related modeling programs, or at least that these issues can be addressed as natural extensions of TRIPEL. Nonetheless, our only large-scale application of TRIPEL has been in plant physiology, so our claim that TRIPEL can handle other domains remains untested.

While we believe it can handle many other domains, we expect it to handle some more naturally than others. In particular, its representation is especially suitable for reasoning about pools of substance or energy and the processes that regulate them. Thus, the domains of ecology, human physiology, and chemical engineering seem especially promising as a next step.

7 Conclusions

This paper has described TRIPEL, a compositional modeling program for answering prediction questions about complex systems. Unlike previous modeling programs, TRIPEL constructs models from simple building blocks: individual variables and influences. Although this approach gives TRIPEL considerable flexibility in constructing models, the program must address modeling issues that are solved implicitly in the domain knowledge required by previous programs. TRIPEL addresses these issues with a set of domain-independent, declarative constraints that define an adequate model. In these constraints, variables and influences play a central role in every modeling decision. Based on these constraints, TRIPEL constructs a simplest adequate model for any given prediction question.

We evaluated TRIPEL in the domain of plant physiology using questions and domain knowledge constructed independently by an expert. The evaluation shows that TRIPEL can construct simple, adequate models of a truly complex system. More importantly, the evaluation suggests the most important area for future research: incorporating more sophisticated criteria for determining whether one variable significantly influences another. Because TRIPEL is designed to be extensible, sophistication can be added to this and other areas without requiring other changes to the program.

8 Acknowledgments

The following people provided valuable comments on this work: Giorgio Brajnik, Dan Clancy, Peter Clark, Brian Falkenhainer, Yumi Iwasaki, Bert Kay, Ben Kuipers, James Lester, Alon Levy, Rich Mallory, Ray Mooney, and Gordon Novak. We also thank Pandu Nayak for helpful discussions on his work. Our work benefited from several tools built by others, including the Qualitative Process Compiler developed by Adam Farquhar and Jimi Crawford, the KM system developed by Erik Eilerts, and several QSIM extensions developed by Dan Clancy. We especially appreciate the help of our domain expert, Art Souther.

Support for this research was provided by grants from Digital Equipment Corporation and the National Science Foundation (IRI-9120310) and a contract from the Air Force Office of Scientific Research (F49620-93-1-0239). The research was conducted at the University of Texas at Austin.

A Proofs

A.1 Monotonic Constraints

Lemma 1 *Adequacy constraints 3 and 4 are monotonic constraints.*

Proof If an exogenous variable v in a partial model m violates adequacy constraint 4, there must be an influence path in the system description, leading to v from a driving variable of the question, consisting of influences that are each valid and significant for the given question (by definition of the constraint). Since every extension of m contains v as an exogenous variable (by definition of an extension), every extension violates the constraint as well. Similarly, if v violates adequacy constraint 3, there must be an influence path in the system description, leading to v from another variable v' in m, consisting of influences that are each valid and significant for the given question (by definition of the constraint). Since every extension of m also contains v' and contains v as an exogenous variable (by definition of an extension), every extension violates the constraint as well. □

Lemma 2 *Adequacy constraints 5, 6, 7, and 8 are monotonic constraints.*

Proof Any influence in a partial model is also in each of its extensions (by definition of an extension). Therefore, if an influence in a partial model violates constraint 6, or a pair of influences violates constraint 5 or 8, the constraint will also be violated in every extension. Similarly, if the influences on a dependent variable in a partial model violate constraint 7, the constraint will also be violated in every extension, because an extension cannot change the influences on a partial model's dependent variables (by definition of an extension). □

Lemma 3 *Adequacy constraints 9 and 10 are monotonic constraints.*

Proof As discussed in Section 3.1.4, the entities in a partial model are determined by the model's variables. Therefore, every entity in a partial model is also in each of the model's extensions, since the variables in each extension are a superset of those in the partial model (by definition of an extension). Thus, if a partial model includes entities that violate one of these constraints, every extension will also violate the constraint. □

Lemma 4 *For a given system description and prediction question, let* M *be a scenario model that satisfies adequacy constraints 1 (include variables of* *interest) and 2 (include variables in activity preconditions). If* M *has no free variables and it violates adequacy constraint 11, every extension of* M *also violates the constraint.*

Proof *Assume* that E is an extension of M that satisfies constraint 11. We show by contradiction that such an extension cannot exist.

(i) M violates constraint 11 (given). Therefore, for some variable of interest v, there is no differential influence path in M, leading to it from a driving variable of the question, such that every influence in the path is valid and significant for the given question.

(ii) E satisfies adequacy constraint 11 (by assumption). Therefore, there is a differential influence path in E from a driving variable to v, consisting of influences that are valid and significant for the given question.

(iii) Let i be the last influence in this influence path that is not in M. There must be such an influence because if every influence in the path were in M, all the variables in the path would also be in M (since M satisfies adequacy constraint 2), and hence the influence path would be in M, which contradicts step i.

(iv) The influencee of i must be in M. If i is the last influence in the path, its influencee is the variable of interest v. Since M satisfies adequacy constraint 1, v is in M. If i is not the last influence, the next influence in the path is in M (by definition of i), and so i's influencee is in M (since M satisfies adequacy constraint 2).

(v) The influencee of i cannot be an exogenous variable in M. If it were, it would also be exogenous in E (by definition of an extension). But then E could not include any influences on it (by definition of a partial model), and hence i could not be in E.

(vi) The influencee of i cannot be a dependent variable in M. An extension cannot change the influences on a partial model's dependent variables (by definition of an extension), so i could be in E only if it was also in M (which contradicts the definition of i).

(vii) Since the influencee of i cannot be dependent or exogenous in M, and since M has no free variables (given), the influencee of i cannot be a variable in M. This contradicts step iv. That step follows from the assumption that E satisfies adequacy constraint 11. Therefore, that assumption is false.

□

A.2 Proof of Theorem 1

The "only if" follows directly from the definition of the connectivity matrix. To prove the "if," suppose that p is the influence path by which i (the *ith* variable) significantly influences j (the *jth* variable). If p consists only of variables and influences that are potentially relevant, cell (i,j) will contain a 1 (by definition of the connectivity matrix). Otherwise, let e be the last influence in the path that is not potentially relevant. There must be such an influence because if every influence in the path were potentially relevant, all the variables in the path would also be potentially relevant (by definition of the potentially relevant variables and influences).

The influencee of e must be potentially relevant. If e is the last influence in the path, its influencee is j, which is in the connectivity matrix and hence is potentially relevant. Otherwise, if e is not the last influence, the next influence in the path is potentially relevant (by definition of e), so e's influencee is potentially relevant (by definition of the potentially relevant variables and influences). But since e is a valid and significant influence on a potentially relevant variable, it must be potentially relevant (by definition of the potentially relevant variables and influences). This contradicts the definition of e. Therefore, p must consist only of variables and influences that are potentially relevant, and the theorem must hold.

A.3 Proof of Theorem 4

A.3.1 Overview

This section proves that Find-adequate-model is admissible; that is, it is guaranteed to return a simplest adequate scenario model whenever an adequate scenario model exists. To prove this, we view the algorithm as repeatedly eliminating scenario models from consideration until only a simplest adequate scenario model remains. Conceptually, when the algorithm begins, the entire set of legal partial models for the given system description (including all legal scenario models) is under consideration. As earlier, we call the set of partial models under consideration the "consideration set." Each step of the algorithm implicitly eliminates some elements of the consideration set. However, Find-adequate-model never eliminates a scenario model unless either (1) the model is inadequate for the question or (2) if the model is adequate, there is an adequate scenario model still under consideration that is at least as simple. The remainder of this section proves that Find-adequate-model is admissible by proving that it follows this strategy.

A.3.2 Auxiliary Lemmas

The proof of Theorem 4 requires several lemmas. The first two lemmas address the case where the System Boundary Selector says that all remaining variables in a partial model can be exogenous. In this case, Extend-model marks the variables exogenous and returns the resulting scenario model. This effectively eliminates from consideration any extension in which one of these variables is dependent. These two lemmas justify this approach; the first lemma simply establishes one of the antecedents of the second lemma.

Lemma 5 *Every partial model that* Find-adequate-model *passes to* Extend-model *satisfies all adequacy constraints except perhaps adequacy constraint 11 (variables of interest differentially influenced).*

Proof Adequacy constraint 1 (include variables of interest) is satisfied because the partial model is an extension of the initial partial model. Adequacy constraint 2 (include variables in activity preconditions) is satisfied because, whenever Extend-model adds an influence to a partial model, it also adds any variables appearing in the influence's activity preconditions. Adequacy constraints 3 (exogenous variables independent of model) and 4 (exogenous variables independent of question) are satisfied because no model passed to Extend-model has any exogenous variables. Adequacy constraints 5 (influences homogeneous), 6 (influences valid), 7 (influences complete), and 8 (influences not redundant) are satisfied because (1) Dv-models only returns influences that satisfy these constraints and (2) if the influences on a variable in a partial model satisfy these constraints, they will in any extension as well (i.e., the constraints are independent of the rest of the model). Finally, adequacy constraints 9 (entities coherent) and 10 (entities compatible with driving variables) are satisfied because a partial model is only added to the agenda if it satisfies these constraints. □

Lemma 6 *Let* P *be a partial model for a given system description, let* Q *be a prediction question, and suppose* P *satisfies all adequacy constraints except perhaps adequacy constraint 11 (variables of interest differentially influenced). Suppose that all free variables in* P *can be treated as exogenous (i.e., they satisfy adequacy constraints 3 and 4). Let* E *be the scenario model that results from making each free variable in* P *an exogenous variable. Then there is an extension of* P *that is a simplest adequate scenario model for* Q *only if* E *is a simplest adequate scenario model for* Q*.*

Proof The extension E has the same number of variables as the partial model P, so E is at least as simple as any other extension of P (by the definition of an extension). Therefore, if E is adequate and some other extension of P is a simplest adequate scenario model, E must be a simplest adequate scenario model as well. We complete the proof by showing that if E is not adequate, no other extension of P is adequate.

(i) E must satisfy all adequacy constraints except perhaps adequacy constraint 11 because (a) P satisfies all these constraints (given), (b) the new exogenous variables satisfy adequacy constraints 3 and 4 (given), and (c) E has the same variables and influences as P.

(ii) Thus, if E is inadequate, it violates adequacy constraint 11. That is, for some variable of interest v, there is no differential influence path in E, leading to it from a driving variable of the question, such that every influence in the path is valid and significant on the time scale of interest.

(iii) *Assume* there is an extension E' of P that is an adequate scenario model. Then E' satisfies adequacy constraint 11, and hence there is a differential influence path in E' from a driving variable to v, consisting of influences that are valid and significant on the time scale of interest.

(iv) Let i be the last influence in this influence path that is not in E. There must be such an influence because if every influence in the path were in E, all the variables in the path would also be in E (since E satisfies adequacy constraint 2), and hence the influence path would be in E, which contradicts step ii.

(v) The influencee of i must be in E. If i is the last influence in the path, its influencee is the variable of interest v. If not, the next influence in the path is in E (by definition of i), and so i's influencee is in E (since E satisfies adequacy constraint 2).

(vi) The influencee of i must be a free variable in P. Otherwise, no extension of P can add an influence on it, and i would have to be in both P and E.

(vii) However, all the free variables in P can be exogenous (given), so there is no influence path from a driving variable to any of these free variables consisting of influences that are valid and significant on the time scale of interest.

(viii) Thus, the influence path implied by the assumption in step iii cannot exist, so E' cannot be an adequate scenario model. Thus, if E is not an adequate scenario model, no other extension of P is an adequate scenario model.

□

The next two lemmas justify the function Dv-models. Given a partial model with a variable v that must be dependent, Extend-model only considers those sets of influences on v returned by Dv-models, thereby implicitly pruning any extension with a different set of influences on v. To justify pruning these extensions, the first lemma ensures that every other set of influences is either inadequate or simply adds some insignificant influences, and the second lemma ensures that those sets containing insignificant influences can be discarded.

Lemma 7 *For a given system description and prediction question, if a set of influences on a variable is not returned by the function* Dv-models*, the set is either inadequate (i.e., violates adequacy constraint 5, 6, 7 or 8) or simply adds insignificant influences to a set that is returned.*

Proof Step i in the function Dv-models generates every complete set of influences, and step ii discards any insignificant influences from these sets. A set of influences will not make it past these steps in two cases: (1) the set is not approximately complete, or (2) the set is identical to one that makes it past these steps except it includes some insignificant influences. In the first case, the set violates adequacy constraint 7 (influences complete). The second case satisfies the lemma because the remaining steps of the algorithm only discard inadequate sets of influences: Step iii only discards sets that violate adequacy constraint 6, and step iv only discards sets that violate adequacy constraint 8. □

Lemma 8 *Given a system description and prediction question, let* M *be a partial model with a variable* v*. Suppose the influences on* v *in* M *include some that are insignificant for the question. Let* M' *be a partial model that is the same as* M *except it does not include the insignificant influences on* v*. Then if* M *or one of its extensions is an adequate scenario model, either* M' *or one of its extensions is also an adequate scenario model and is at least as simple.*

Proof If M or one if its extensions is an adequate scenario model, call that model A. We show by construction that M' or one of its extensions is also an adequate scenario model and is at least as simple. Construct A' from A by simply removing the insignificant influences on v. If A = M, then A' = M'. Otherwise, A' is an extension of M'. A' is at least as simple as A because it has the same variables. Furthermore, A' is an adequate scenario model because it contains no free variables (since A has none) and it satisfies all adequacy constraints:

- Constraint 7 is satisfied because A' contains all influences from A except insignificant ones.
- Constraint 11 is satisfied for the following reasons. A is adequate, so it satisfies this constraint. Therefore, for every variable of interest, there is an influence path in A leading from a driving variable to the variable of interest, and every influence in the path is valid and significant for the question. A' includes all the variables and influences in A except some influences that are insignificant, so A' must include every such influence path that A does, and hence A' must satisfy adequacy constraint 11.

- All the other constraints are satisfied because the exogenous variables in A and A' are the same, the dependent variables in A and A' are the same, and the influences in A' are a subset of those in A.

□

A.3.3 Key Lemma

We can now prove the key lemma in the proof of Theorem 4. When Find-adequate-model is invoked, we define the consideration set to be the entire set of legal partial models (including all legal scenario models) for the given system description. As Find-adequate-model repeatedly eliminates elements of the consideration set, this lemma shows that it always retains a simplest adequate scenario model if one exists.

Lemma 9 *For a given system description and prediction question,* Find-adequate-model *never prunes a scenario model from the consideration set unless either (1) the model is inadequate or (2) there is an adequate scenario model still in the consideration set that is at least as simple.*

Proof There are only seven ways in which Find-adequate-model prunes elements of the consideration set, and each satisfies the lemma:

(i) **(Initializing the Agenda)** Initially, the agenda contains a partial model consisting of the variables of interest, each a free variable. At that point, the consideration set has been reduced to that partial model and all of its extensions, implicitly eliminating those scenario models that do not contain the variables of interest. However, none of the eliminated models is adequate, because each violates adequacy constraint 1 (include variables of interest).

(ii) **(Monotonic Constraints)** If a partial model violates a monotonic constraint, it is not added to the agenda, thereby pruning it and its extensions from the consideration set. The partial model itself is inadequate because it violates the constraint. By definition, a monotonic constraint, when violated for a partial model, is violated for any extension of that model. Thus, each extension is inadequate for the question as well.

(iii) **(Free Variable Cannot Be Exogenous)** When the System Boundary Selector says that a variable in a partial model must be dependent, Extend-model effectively prunes any extension in which the variable is exogenous. By definition of the System Boundary Selector, the partial model would violate adequacy constraint 3 or adequacy constraint 4 if the variable were exogenous. Since these two constraints are monotonic (Lemma 1), any extension of the partial model in which the variable is exogenous will also be inadequate.

(iv) **(All Free Variables Can Be Exogenous)** When the System Boundary Selector says that all remaining variables in a partial model can be exogenous, Extend-model marks the variables exogenous and returns the resulting scenario model. This effectively prunes any extension in which one of these variables is dependent. If any of the pruned extensions is an adequate scenario model, Lemmas 5 and 6 ensure that the scenario model returned by Extend-model is also, and it is at least as simple.

(v) **(Influences on a Dependent Variable)** Given a partial model with a variable v that must be dependent, Extend-model only considers those sets of influences on v returned by Dv-models, thereby implicitly pruning any extension with a different set of influences on v. Lemmas 7 and 8 ensure that these extensions can be pruned without violating the current lemma.

(vi) **(Variables in Activity Preconditions)** For each partial model to be returned, Extend-model adds variables that are required by adequacy constraint 2 (include variables in activity preconditions). This effectively prunes those extensions without the variables. However, all the influences in the partial model will also be in each extension (by definition of an extension). Thus, if an extension lacks some variable appearing in the activity preconditions of those influences, the extension will violate adequacy constraint 2. Therefore, no such extension can be an adequate scenario model.

(vii) **(Returning the First Adequate Model)** Find-adequate-model returns the first adequate scenario model M that it finds, effectively pruning the remainder of the consideration set. Since it always removes the simplest partial model from the agenda, no other model on the agenda can be simpler than M. The definition of an extension ensures that M is as simple as any of its extensions and that every model on the agenda is as simple as any of their extensions, so no model in the consideration set is simpler than M. Thus, since M is an adequate scenario model, no other scenario model in the consideration set can be a simplest adequate model unless M is also.

□

A.3.4 Main Proof

Finally, building on the previous lemmas, we can prove Theorem 4: *Given a system description and a prediction question for which some scenario model is adequate,* Find-adequate-model *will return a simplest adequate scenario model.*

Proof (of Theorem 4) Lemma 9 ensures that Find-adequate-model never prunes an adequate scenario model unless another adequate scenario model, at least as simple, remains in the consideration set. If there is an adequate scenario model, then the lemma ensures that the consideration set cannot become empty. Furthermore, if there is an adequate scenario model and the consideration set is reduced to a single adequate scenario model, that model must be a simplest adequate scenario model.

Theorem 3 ensures that Find-adequate-model eventually terminates. Upon termination, either the agenda (and hence consideration set) is empty or the consideration set consists of a single adequate scenario model (which is returned). If there is an adequate scenario model for the question, the previous paragraph ensures that the first case cannot arise, and it ensures that the model in the second case must be a simplest adequate scenario model. □

B Evaluation Details

This appendix lists all the plant physiology questions, constructed by the expert, on which TRIPEL was formally evaluated (as described in Section 5). Most of the models that TRIPEL constructed for these questions can be found elsewhere [44].

(i) How would an increasing amount of CO_2 in a plant's leaves affect the rate of photosynthesis in the leaves?
(ii) How does increasing soil water potential affect a plant's water distribution rate?
(iii) How does a decreasing amount of water in a plant affect the amount of K^+ in its guard cells?
(iv) What happens to a plant's water potential as the temperature of the environment decreases?
(v) How would an increasing rate of solar irradiation to a plant's leaves affect the temperature of the leaves?
(vi) What happens to turgor pressure in a plant's leaves as root water absorption decreases?
(vii) How would a decreasing amount of water in the earth's atmosphere affect a plant's photosynthesis rate?
(viii) How does an increasing level of ABA in a plant's leaves affect transpiration from the leaves?
(ix) How does increasing water potential in a plant's leaves affect the rate of K^+ efflux from the guard cells in the leaves?
(x) How does an increasing amount of ABA in the guard cells of a plant's leaves affect osmosis to the leaves' accessory cells from the leaves' guard cells?

(xi) How does an increasing rate of diffusion of heat from the stems of a plant to the atmosphere surrounding the stems affect the water potential of the symplast in the stems?

(xii) How does a decreasing rate of evaporation from a plant's leaves affect the amount of CO_2 in the atmosphere surrounding the leaves?

(xiii) How does a decreasing rate of photosynthesis in a plant's shoot system affect the pressure potential in the phloem of its leaves?

(xiv) As the amount of water in a plant's cell walls increases, what happens to the plant's turgor pressure?

References

[1] S. Addanki, R. Cremonini, and J.S. Penberthy. Graphs of models. *Artificial Intelligence*, 51:145–177, 1991.

[2] T.F.H. Allen and T.B. Starr. *Hierarchy.* University of Chicago Press, Chicago, 1982.

[3] Jonathan Amsterdam. *Automated Qualitative Modeling of Dynamic Physical Systems.* PhD thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1993. Technical Report 1412.

[4] Catherine A. Catino. *Automated Modeling of Chemical Plants with Application to Hazard and Operability Studies.* PhD thesis, University of Pennsylvania, 1993.

[5] D.J. Clancy and B.J. Kuipers. Behavior abstraction for tractable simulation. In *The Seventh International Workshop on Qualitative Reasoning about Physical Systems*, pages 57–64, Orcas Island, Washington, 1993.

[6] John W. Collins and Kenneth D. Forbus. Reasoning about fluids via molecular collections. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 590–595, Los Altos, CA, 1987. Morgan Kaufmann.

[7] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms.* McGraw-Hill, New York, 1989.

[8] R. Davis. Diagnostic reasoning based on structure and behavior. *Artificial Intelligence*, 24:347–410, 1984.

[9] Thomas Ellman, John Keane, and Mark Schwabacher. Intelligent model selection for hillclimbing search in computer-aided design. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 594–599, Menlo Park, CA, 1993. AAAI Press.

[10] Brian Falkenhainer. Modeling without amnesia: Making experience-sanctioned approximations. In *The Sixth International Workshop on Qualitative Reasoning*

about Physical Systems, pages 44–55, Edinburgh, Scotland, 1992. Heriot-Watt University.

[11] Brian Falkenhainer. Ideal physical systems. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 600–605, Menlo Park, CA, 1993. AAAI Press.

[12] Brian Falkenhainer and Kenneth D. Forbus. Compositional modeling: Finding the right model for the job. *Artificial Intelligence*, 51:95–143, 1991.

[13] Adam Farquhar. *Automated Modeling of Physical Systems in the Presence of Incomplete Knowledge.* PhD thesis, Department of Computer Science, University of Texas at Austin, 1993. Technical Report AI93-207.

[14] Adam Farquhar. A qualitative physics compiler. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1168–1174, Menlo Park, CA, 1994. AAAI Press.

[15] Kenneth D. Forbus. Qualitative process theory. *Artificial Intelligence*, 24:85–168, 1984.

[16] Kenneth D. Forbus. Qualitative physics: Past, present, and future. In Daniel S. Weld and Johan de Kleer, editors, *Readings in Qualitative Reasoning about Physical Systems*, pages 11–39. Morgan Kaufmann, San Mateo, CA, 1990.

[17] Kenneth D. Forbus and Brian Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 380–387, Menlo Park, CA, 1990. AAAI Press.

[18] Jay W. Forrester. *Principles of Systems.* Wright-Allen Press, Cambridge, MA, 1968.

[19] H.J. Gold. *Mathematical Modeling of Biological Systems.* John Wiley and Sons, New York, 1977.

[20] Arthur C. Guyton. *Textbook of Medical Physiology.* W.B. Saunders, Philadelphia, 1981.

[21] Yumi Iwasaki. Reasoning with multiple abstraction models. In Boi Faltings and Peter Struss, editors, *Recent Advances in Qualitative Physics*, pages 67–82. MIT Press, Cambridge, 1992.

[22] Yumi Iwasaki and Inderpal Bhandari. Formal basis for commonsense abstraction of dynamic systems. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 307–312, San Mateo, CA, 1988. Morgan Kaufmann.

[23] Yumi Iwasaki and Alon Y. Levy. Automated model selection for simulation. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1183–1190, Menlo Park, CA, 1994. AAAI Press.

[24] Yumi Iwasaki and Herbert A. Simon. Causality and model abstraction. *Artificial Intelligence*, 67(1):143–194, May 1994.

[25] Stephen J. Kline. *Similitude and Approximation Theory*. McGraw-Hill, New York, 1965.

[26] P.V. Kokotovic, R.E. O'Malley, Jr., and P. Sannuti. Singular perturbations and order reduction in control theory — an overview. *Automatica*, 12:123–132, 1976.

[27] Benjamin Kuipers. Qualitative simulation. *Artificial Intelligence*, 29:289–338, 1986.

[28] Benjamin Kuipers. Abstraction by time scale in qualitative simulation. In *Proceedings of the Sixth National Conference on Artificial Intelligence (AAAI-87)*, pages 621–625, Los Altos, CA, 1987. Morgan Kaufmann.

[29] Benjamin Kuipers. *Qualitative Reasoning: Modeling and Simulation with Incomplete Knowledge*. MIT Press, Cambridge, MA, 1994.

[30] S.A. Lapp and G.J. Powers. Computer-aided synthesis of fault trees. *IEEE Transactions on Reliability*, April 1977.

[31] James Lester. *Generating Natural Language Explanations from Large-Scale Knowledge Bases*. PhD thesis, Department of Computer Science, University of Texas at Austin, 1994.

[32] James Lester and Bruce Porter. Developing and empirically evaluating robust explanation generators: The knight experiments. *Computational Linguistics*. Forthcoming.

[33] James Lester and Bruce Porter. Scaling up explanation generation: Large-scale knowledge bases and empirical studies. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 416–423, Menlo Park, CA, 1996. AAAI Press/MIT Press.

[34] Alon Y. Levy. *Irrelevance Reasoning in Knowledge Based Systems*. PhD thesis, Department of Computer Science, Stanford University, July 1993. Report No. STAN-CS-93-1482.

[35] C.C. Lin and L.A. Segal. *Mathematics Applied to Deterministic Problems in the Natural Sciences*. Macmillan, New York, 1974.

[36] Sui-ky Ringo Ling. Using a domain theory to guide automated modeling of complex physical phenomena. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1766–1772, San Mateo, CA, 1995. Morgan Kaufmann.

[37] Sanjay Mittal and Brian Falkenhainer. Dynamic constraint satisfaction problems. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 25–32, Menlo Park, 1990. AAAI Press.

[38] P. Pandurang Nayak. Causal approximations. *Artificial Intelligence*, 70:277–334, 1994.

[39] P. Pandurang Nayak and Leo Joskowicz. Efficient compositional modeling for generating causal explanations. *Artificial Intelligence*, 83:193–227, 1996.

[40] R.V. O'Neill, D.L. DeAngelis, J.B. Waide, and T.F.H. Allen. *A Hierarchical Concept of Ecosystems.* Princeton University Press, Princeton, NJ, 1986.

[41] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Reading, MA, 1984.

[42] B. Porter, J. Lester, K. Murray, K. Pittman, A. Souther, L. Acker, and T. Jones. AI research in the context of a multifunctional knowledge base: The botany knowledge base project. Technical Report AI88-88, University of Texas at Austin, 1988.

[43] C.J. Puccia and R. Levins. *Qualitative Modeling of Complex Systems.* Harvard University Press, Cambridge, MA, 1985.

[44] Jeff Rickel. *Automated Modeling of Complex Systems to Answer Prediction Questions.* PhD thesis, Department of Computer Science, University of Texas at Austin, May 1995. Technical Report AI95-234.

[45] Jeff Rickel and Bruce Porter. Automated modeling for answering prediction questions: Selecting the time scale and system boundary. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1191–1198, Menlo Park, CA, 1994. AAAI Press.

[46] Nancy Roberts, David Andersen, Ralph Deal, Michael Garet, and William Shaffer. *Introduction to Computer Simulation.* Addison-Wesley, Reading, MA, 1983.

[47] T. Rosswall, R.G. Woodmansee, and P.G. Risser, editors. *Scales and Global Change: Spatial and Temporal Variability in Biospheric Processes.* John Wiley and Sons, New York, 1988.

[48] V.R. Saksena, J. O'Reilly, and P.V. Kokotovic. Singular perturbations and time-scale methods in control theory: Survey 1976–1983. *Automatica*, 20(3):273–293, 1984.

[49] L.A. Segal, editor. *Mathematical Models in Molecular and Cellular Biology*, chapter 3. Cambridge University Press, Cambridge, 1980.

[50] Mark Shirley and Brian Falkenhainer. Explicit reasoning about accuracy for approximating physical systems. In T. Ellman, R. Keller, and J. Mostow, editors, *Working Notes of the AAAI Workshop on Automatic Generation of Approximations and Abstractions*, pages 153–162, 1990.

[51] H.A. Simon and A. Ando. Aggregation of variables in dynamic systems. *Econometrica*, 29:111–138, 1961.

[52] Daniel S. Weld. Reasoning about model accuracy. *Artificial Intelligence*, 56:255–300, 1992.

[53] Daniel S. Weld and Johan de Kleer, editors. *Readings in Qualitative Reasoning about Physical Systems*. Morgan Kaufmann, San Mateo, CA, 1990.

[54] Brian C. Williams. Critical abstraction: Generating simplest models for causal explanation. In *The Fifth International Workshop on Qualitative Reasoning about Physical Systems*, pages 77–92, Austin, TX, 1991. University of Texas at Austin.

[55] Brian C. Williams and Olivier Raiman. Decompositional modeling through caricatural reasoning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1199–1204, Menlo Park, CA, 1994. AAAI Press.

[56] Kenneth Man-kam Yip. Model simplification by asymptotic order of magnitude reasoning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 634–641, Menlo Park, CA, 1993. AAAI Press.

[57] Kenneth Man-kam Yip. Model simplification by asymptotic order of magnitude reasoning. *Artificial Intelligence*, 80:309–348, 1996.