Automated Modeling of Complex Systems to Answer Prediction Questions

by

Jeffrey Walter Rickel, B.S., M.S.

Dissertation

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

Doctor of Philosophy

The University of Texas at Austin

May 1995

Automated Modeling of Complex Systems

to Answer Prediction Questions

Approved by
Dissertation Committee:

I dedicate this dissertation to my family: Lynn, Chelsea and Cocoa.

Acknowledgments

During my years at the University of Texas, many people touched my life. This dissertation is a product of their help, encouragement, criticism, and friendship. This acknowledgment section provides a rare opportunity to thank them publicly.

No one influenced my research and my education more than my advisor, Bruce Porter. Bruce improved every aspect of my professional skills, including my writing, my public speaking, and my ability to formulate and solve problems. He has a rare ability to encourage and criticize at the right moments, and he sincerely cares about his students. Above all, Bruce always treated me as a colleague and a friend.

Next to Bruce, no one spent more time helping me than Art Souther, our botany expert. The knowledge base that Art built is a unique, valuable resource. He spent many long, tedious hours preparing the knowledge base for my evaluation, and he spent many more analyzing the models my program built. His knowledge of plant physiology was valuable throughout my research, and his commitment to an accurate, task-independent representation of knowledge prevented me from compromising the generality of my work.

In addition to Bruce and Art, the other members of my committee also provided valuable help. Ben Kuipers was often like a second advisor. He encouraged me, taught me important research strategies, helped advertise my work and abilities to the right people at the right times, and generously shared his many books. Ray Mooney carefully read my dissertation and provided valuable suggestions. He and I also enjoyed many laughs over the years. Gordon Novak's modeling experience was a source of useful observations and entertaining anecdotes. Finally, Brian Falkenhainer's expertise in automated modeling was invaluable; his papers and his many insightful comments guided my research from start to finish.

Many other people both inside and outside our department influenced my research. I benefited from valuable technical discussions with Pandu Nayak, Yumi Iwasaki, and Alon Levy. I could always count on Peter Clark for penetrating questions on my work; his comments on my dissertation improved it immeasurably. My work relied heavily on tools built by others, including the Qualitative Process Compiler developed by Adam Farquhar and Jimi Crawford, the KM system developed by Erik Eilerts, and the many QSIM extensions developed by Dan Clancy. Many people helped improve my papers and presentations, including James Lester, Liane Acker, Bert Kay, Giorgio Brajnik, Ken Murray, and (especially) Rich Mallory. Financial support for my research was provided by a grant from the National Science Foundation (IRI-9120310), a contract from the Air Force Office of Scientific Research (F49620-93-1-0239), and donations from the Digital Equipment Corporation.

Many good friends made my life in Austin enjoyable. I am grateful to all my volleyball teammates over the years, especially Tim Collins, Jack Chen, and Jacob Kornerup. The T-shirts we won together make up half my wardrobe. I was fortunate to share an office with four guys who know how to balance work and laughter: Neelakantan Kartha, Siddarth Subramanian, Rich Mallory, and Tim Collins. Additionally, I enjoyed the company of many other friends inside and outside the department; there are too many of you to mention, but I thank you all.

Closer to home, I am grateful to my parents, Walter and Ellen, for all their love and encouragement. Their love is the rock that supports me: constant, often taken for granted, yet always there when I need it. I also thank my brother, Scott, for his love and friendship. I hope he is as proud of my success as I am of his.

Although I value all my friends and colleagues, the greatest joy in my life comes from my family: Cocoa, Chelsea, and Lynn. I thank Cocoa for her unfailing love and loyalty, and for earning five degrees in the time it took me to complete one. My daughter, Chelsea, has brightened my life beyond words. During my lowest moments, she can make me smile. My deepest gratitude goes to my wife, Lynn. She is my best friend, and always will be. I can never repay her for all her love, support and sacrifices, but I will try.

JEFFREY WALTER RICKEL

The University of Texas at Austin
May 1995

Automated Modeling of Complex Systems to Answer Prediction Questions

Publication No. ____________________

Jeffrey Walter Rickel, Ph.D.
The University of Texas at Austin, 1995

Supervisor: Bruce W. Porter

The ability to answer prediction questions is crucial in science and engineering. A prediction question describes a physical system under hypothetical conditions and asks for the resulting behavior of specified variables. Prediction questions are typically answered by analyzing (e.g., simulating) a mathematical model of the physical system. To provide an adequate answer to a question, a model must be sufficiently accurate. However, the model must also be as simple as possible to ensure tractable analysis and comprehensible results. Ensuring a simple yet adequate model is especially difficult for complex systems, which include many phenomena that can be described at many levels of detail. While tools exist for analysis, modeling is a creative, time-consuming task performed by humans.

We have designed algorithms for automatically constructing models to answer prediction questions, implemented them in a program called TRIPEL, and evaluated them in the domain of plant physiology. Given a prediction question and domain knowledge, TRIPEL builds the simplest differential-equation model that can adequately answer it and automatically passes the model to a simulator to generate the desired predictions. TRIPEL uses knowledge of the time scales on which processes operate to identify and ignore insignificant phenomena and choose quasi-static representations of fast phenomena. It also uses novel criteria and methods to choose a suitable system boundary, separating relevant subsystems from those that can be ignored. Finally, it includes a novel algorithm for efficiently searching through alternative levels of detail in a vast space of possible models. TRIPEL successfully

answered plant physiology questions using a large, multipurpose, botany knowledge base (covering 300 processes and 700 plant properties) independently developed by a domain expert. Because its methods are domain-independent, TRIPEL should be equally useful in many areas of science and engineering.

Contents

Chapter 1

Introduction

1.1 Prediction Questions

Our long-term goal is a computer program that can answer *prediction questions* about physical systems. The following question, from the plant physiology domain, illustrates the general form of a prediction question: "How would decreasing soil moisture affect a plant's transpiration[1] rate?" A prediction question poses a hypothetical *scenario* (e.g., a plant whose soil moisture is decreasing) and asks for the resulting behaviors of specific *variables of interest* (e.g., transpiration rate). Each variable represents a property of the scenario. An answer to a prediction question includes the desired predictions (e.g., "Transpiration will initially remain constant but will decrease as the plant begins to wilt."). Equally important, the answer must explain how domain principles justify the predictions. Therefore, our computer program must solve the following *prediction task*: Given a prediction question and domain knowledge, produce the desired predictions as well as their explanations.

Prediction is important for many tasks in science and engineering. A design engineer must predict how a design will respond to hypothetical conditions (e.g., "How is power consumption affected as the fan speed is increased?"). A diagnostician must predict the consequences of an hypothesized diagnosis and compare them with observed symptoms (e.g., "How would the patient's sodium level be affected if his insulin level were dropping?"). A theorist evaluates theories by using them to make predictions that can be experimentally tested (e.g., "How would the interval between pulses be affected if the density of a pulsar were increasing?"). We are particularly motivated by the use of prediction questions in tutoring, in which a tutoring system teaches domain principles in the context of student questions (e.g., the tutor uses the question about decreasing soil moisture to explain the role of

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

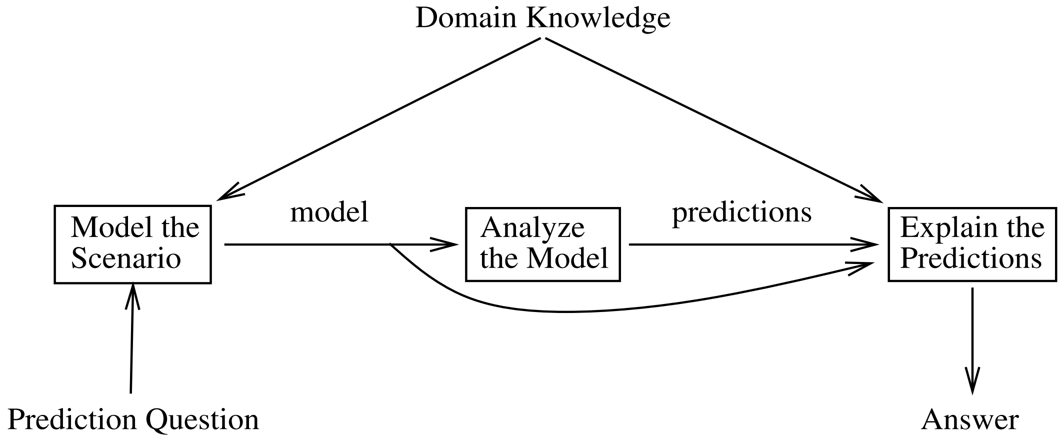


Figure 1.1: The three steps needed to answer a prediction question.

the ABA hormone in regulating plant water loss through transpiration). Answering prediction questions for these tasks requires people with special knowledge, and it can be time consuming and error prone. Therefore, automation would be valuable.

In each of these tasks, a comprehensible explanation of the predictions is crucial. In the tutoring task, the main objective is to teach the domain principles that underlie the predictions. Even when the predictions themselves are the main objective, experience with expert systems shows clearly that humans accept the conclusions of computers much more readily when the computer can justify those conclusions [8]. Therefore, a computer program for answering prediction questions must be capable of making the desired predictions and explaining them well.

1.2 Modeling

Scientists and engineers answer prediction questions by constructing and analyzing a mathematical *model* of the scenario. A mathematical model uses a formal language, such as differential equations, to represent the physical phenomena that govern the scenario. Once the model is constructed, mathematical methods are used to analyze it (e.g., solve the equations for the variables of interest), resulting in the desired predictions. Finally, to answer the original question, scientists and engineers interpret the mathematical predictions in physical terms and show how the physical principles embodied in the model explain the predictions. Thus, as illustrated in Figure 1.1, there are three steps in answering a prediction question: construct a model of the scenario, analyze the model, and explain the results.

To automate this process, we must automate these three steps. There are already a variety of methods that automate the analysis step. For example, to analyze differential equation models, there are programs for solving the equations analytically (e.g., Mathematica [87]) and there are algorithms for simulating such models (e.g., Runge-Kutta [6]). However, the other two steps, modeling and explanation, typically require humans. This dissertation addresses the first of these: automatically constructing models for answering prediction questions.

More specifically, this dissertation provides methods for automatically constructing models consisting of algebraic equations and ordinary differential equations.[2] Each variable in such a model represents a real-valued, continuous, time-varying property of the scenario being modeled. Examples include the amount of water in a plant or its soil, the rate of a process (e.g., transpiration), and the pressure in a plant's cells. Each equation in such a model specifies a variable, or its rate of change, as a function of other variables. These equations represent the physical phenomena that govern the scenario. For example, the rate at which the amount of plant water changes is a function of the rate of water uptake from the soil and the rate of water loss through transpiration. Because many properties of physical systems can be represented with such variables, and because many physical phenomena can be represented with such equations, such models are widely used throughout science and engineering.

In summary, this dissertation focuses on the following *modeling task*:

- Given a prediction question and domain knowledge, produce a model of the scenario, consisting of algebraic equations and ordinary differential equations, that is suitable for answering the question.

1.3 Multiple Models

A modeler must balance two competing goals. First, the model must be sufficiently accurate. If the model omits some relevant phenomenon in the scenario, or fails to represent it in sufficient detail, the predictions or their explanation may be incorrect. For instance, in the example given earlier, transpiration will appear unaffected by decreasing soil moisture unless the model includes the effects of plant water regulation processes. This goal, accuracy, encourages detailed, comprehensive models.

However, a model must also be as simple as possible. If the model includes irrelevant information, it will be more difficult to analyze and explain. For example,

[2] We show later that the methods are useful for building qualitative models [19, 28, 52, 54, 72] as well as quantitative ones. Although the methods are equally applicable to building quantitative models, our empirical evaluation has focused on qualitative models.

a detailed model of the entire plant would include an enormous number of variables, making the model difficult to simulate or solve analytically, and the excess details would obscure the simple reason that transpiration decreases.

For complex systems (such as plants), no single model can satisfy both these goals for a wide variety of questions. Complex systems encompass many phenomena that can be described at many different levels of detail. Hence, if a model is comprehensive enough to provide an accurate answer to a wide variety of questions, it will be unnecessarily detailed for any particular question. Consequently, for complex systems, a modeler must consider multiple models, choosing the simplest adequate model for each question.

This introduces three requirements for any modeling program intended to handle complex systems:

- The modeling program must be able to construct multiple, alternative models that differ in accuracy and simplicity.
- The modeling program must have criteria for determining whether a candidate model is *adequate* (e.g., sufficiently accurate) for answering a given question.
- The modeling program must have criteria for determining whether one candidate model is *simpler* than another.

1.4 Types of Modeling Alternatives

For a given scenario, human modelers are able to construct multiple, alternative models that differ in accuracy and simplicity. A modeling program must consider the same types of modeling alternatives that humans do, for two reasons. First, the experience of scientists and engineers in many different domains has proven their methods useful for representing physical phenomena. Second, to ensure comprehensible explanations, the elements of a model must match the concepts used by scientists and engineers. Therefore, it is important to understand how human modelers tailor the model of a scenario to the question it must answer.

To construct a model, human modelers first determine which physical phenomena in the scenario are relevant. Of the many phenomena governing any complex system (such as a plant), only a handful are relevant to any particular question. For example, of the many processes at work in a plant, the question about decreasing soil moisture only requires modeling the effects of the plant's water regulation processes. The effects of some processes can be ignored because they are *insignificant*. For instance, in the decreasing soil moisture example, metabolic processes and mineral transport processes can be ignored because they do not significantly influence the variable of interest, transpiration rate. The effects of other processes can be treated as *exogenous* (i.e., causally upstream from the input variables in the model). For example, although the processes that regulate soil moisture (e.g., rain and evaporation) do significantly influence the transpiration rate of a plant, they are nonetheless irrelevant to predicting the effects of decreasing soil moisture on transpiration rate. By omitting insignificant and exogenous phenomena from a model, a modeler simplifies it.

For those phenomena the modeler chooses to represent in the model, many levels of detail are possible. For example, water in the plant can be treated as an aggregate, or the water in the roots, stem and leaves can be modeled individually. Similarly, processes can be aggregated. For example, the chemical formula for photosynthesis summarizes the net effects of its many component reactions. For an even simpler level of detail, the dynamics of a process can be summarized by its equilibrium results. 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. If the dynamics of this process are irrelevant, the modeler can simply treat the level of water as an instantaneous function of the level of solutes. These types of alternatives are useful in many areas of science and engineering.

Human modelers have many criteria for choosing among such alternatives. Our objective is not to develop new criteria for humans to use. Rather, our objective is to develop a computer program that can choose among these types of alternatives. Therefore, we must develop a representation for the modeling alternatives and for the knowledge that humans use to choose among them, and we must develop methods that use that knowledge to construct models.

1.5 Compositional Modeling

To answer prediction questions in domains like plant physiology, a large collection of models is needed. A complex system like a plant is governed by a large number of phenomena, and many of these can be described at multiple levels of detail. Additionally, different species of plants are governed by somewhat different phenomena. Any particular question could concern any aspects of any species of plant, so almost any combination of plant physiology phenomena and their levels of detail might be relevant. Thus, the requirements for this modeling task — to answer a wide range of questions about complex systems and to use the simplest possible model to answer each question — make it impractical to construct an adequate library of models ahead of time.

Instead of selecting from a library of complete models, it is more practical for a modeler to construct models from available pieces. The modeler uses domain knowledge to identify the physical phenomena that govern the scenario and to provide multiple levels of detail at which these phenomena can be described. The modeler determines which phenomena are relevant to the question and, for each relevant phenomenon, chooses from among the available levels of detail. Thus, the domain knowledge provides the building blocks for models, and the modeler constructs a model by composing relevant building blocks. This approach is called *compositional modeling* [25].[3]

For instance, in the decreasing soil moisture example, the domain knowledge would provide ways of describing all the phenomena governing the plant and soil, including the effects of metabolic processes, mineral transport processes, plant water regulation processes, and soil moisture regulation processes. The modeler would recognize that only the plant water regulation processes are relevant, and it would construct a model from the available descriptions of these relevant processes.

For complex systems, the domain knowledge needed for compositional modeling is much easier to provide than a library of complete models. The domain knowledge must provide a set of domain phenomena, ways of modeling each separate phenomenon, and ways of identifying these phenomena in scenarios — exactly the sort of information found in textbooks in plant physiology or any other area of science or engineering.

However, compositional modeling raises three new issues for the modeler:

- What should serve as the building blocks for models? Is an equation an appropriate building block, or are smaller or larger building blocks required?

- How should the modeler use the domain knowledge? Should the modeler identify all phenomena in the scenario before selecting relevant ones, or can the two steps be more efficiently interleaved?

- How should the modeler search for the simplest adequate model? Should it build all possible models, prune the inadequate ones, and choose the simplest of those remaining? Or can the modeler interleave construction of candidate models and selection of the simplest adequate one?

The following three sections summarize our approach to these three issues.

[3]We use the term "compositional modeling" to refer to this basic approach, not the particular method developed by Falkenhainer and Forbus.

1.6 The Building Blocks: Influences

In the compositional modeling approach, a model is constructed from building blocks that are provided by domain knowledge. However, the approach does not specify the appropriate size of the building blocks. Since an ordinary differential equation (ODE) model is a set of equations, it might seem natural for the domain knowledge to provide individual equations as building blocks. To understand this issue, we first examine the relationship between the physical phenomena in a scenario and the equations in a model of that scenario.

Each equation in an ODE model represents all the physical phenomena that influence one particular variable in the model (or its rate of change). In a differential equation, the phenomena are typically the effects of processes. For example, the rate at which the amount of sugars in a plant's leaves changes equals the rate of production by photosynthesis minus the rate of consumption by respiration minus the rate of transport to other plant parts. In an algebraic equation, the phenomena are typically the factors that control the rate of a process.[4] For example, the rate of a chemical reaction equals the product of the concentrations of its reactants. Similarly, the rate of acceleration of a body equals the sum of forces on that body divided by its mass (Newton's Law). Thus, each equation in an ODE model is a composition of individual physical phenomena in the scenario, typically either effects of processes or factors controlling their rate.

Before constructing an equation, a modeler must make two types of decisions. First, the modeler must decide which of these phenomena is significant and which can be ignored. In the examples above, photosynthesis might be negligible on a cloudy day, the effect of some reactants on the reaction rate may be negligible if they are available in abundance, and the effect of some forces (e.g., friction) might be negligible. Second, the modeler must choose a suitable level of detail for each significant phenomenon. For example, the modeler could treat photosynthesis as an aggregate process, or it could decompose photosynthesis into its component dark reactions (which produce sugars) and light reactions (which convert light energy to chemical energy).

Because the modeler must reason about individual phenomena, it is not appropriate for the domain knowledge to provide equations as the building blocks for models. Of the phenomena influencing a variable, the modeler might choose any subset as significant, and each significant phenomenon can generally be represented at multiple levels of detail. Therefore, there may be a variety of useful combina-

[4]Often, the algebraic equations in an ODE model are eliminated by substitution into the differential equations. In this discussion, we ignore such algebraic simplifications in order to highlight the different types of phenomena that are represented in an ODE model.

tions of these phenomena and their levels of detail, and hence a variety of possible equations for the variable. Rather than provide each of these equations, the domain knowledge can simply provide the individual pieces from which an equation can be built; each piece is an *influence* representing one phenomenon at one level of detail.

An influence is a causal relation between two variables, as in Qualitative Process Theory [28]. Each variable represents a property of the scenario (e.g., soil moisture or the plant's transpiration rate). Each influence specifies that a variable, or its rate of change, is a function of another variable. For example, each of the following is an influence:

- the rate at which the amount of sugars in a plant's leaves changes is a function of the rate of production by photosynthesis (and perhaps other things).

- the rate of photosynthesis (a chemical reaction) is a function of the amount of carbon dioxide (one of its reactants) in the leaves (and perhaps other things).

- the rate of acceleration of a rocket at lift-off is a function of the force of gravity (and perhaps other things).

Each influence represents *one* of the physical phenomena affecting a variable, as emphasized by the qualification "and perhaps other things." To construct an equation, the modeler uses the domain knowledge to identify all the influences on a given variable, it chooses those influences that represent significant phenomena at an appropriate level of detail, and it composes the chosen influences into an equation.

Additional domain knowledge is needed to compose influences into equations. Forbus [28] developed methods for composing influences into qualitative equations, given the sign of the partial derivative of each influence. Such qualitative models [19, 28, 52, 54, 72] are useful when quantitative details are unavailable (as is often the case in plant physiology) or irrelevant (as is often the case in tutoring). Farquhar [26] extended the methods to construct quantitative equations, given knowledge of whether each influence is an additive term, a multiplicative term, or some other type of term. While some modeling decisions arise at this step — determining the form of the equation — this dissertation focuses on the more important decisions involved in choosing the influences that make up each equation.

The methods described in this dissertation base their modeling decisions on the influences among variables, but not on the quantitative details of the influences. Therefore, the methods are useful in building qualitative models as well as quantitative models. The issues that the methods address are important in constructing both types of models. However, to date, the methods have only been used to construct qualitative models.

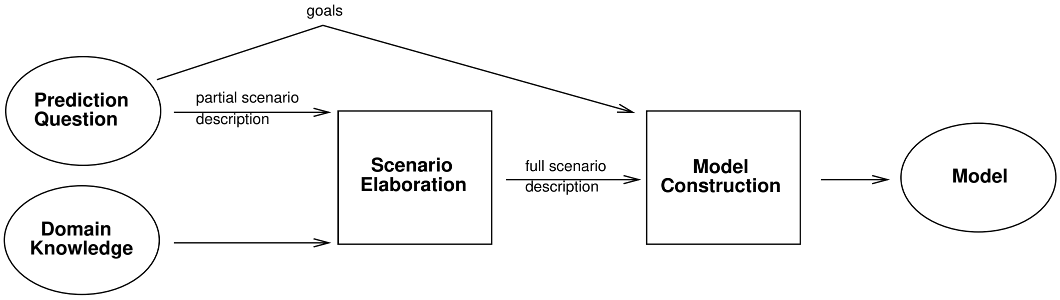


Figure 1.2: The two subtasks of the modeling task: scenario elaboration and model construction.

In summary, the role of the domain knowledge is to provide the influences that could be used to describe the scenario. The modeler constructs a model by selecting those influences that are relevant to answering the given question.

1.7 Demand-Driven Scenario Elaboration

The modeling task can be separated into two subtasks, as illustrated by Figure 1.2. In the *scenario elaboration* task, the modeler uses the domain knowledge to elaborate the scenario description given in the question. Specifically, scenario elaboration infers all the phenomena that govern the scenario (not just the relevant ones) and the levels of detail at which they can be described. As discussed in Section 1.6, this information is represented as influences. For instance, in the decreasing soil moisture example, the modeler would infer influences representing the effects of metabolic processes, plant water regulation processes, soil moisture regulation processes, and many others. As another example, a modeler might use general knowledge of chemical engineering and a specific question about a Dow Chemical factory in Houston to infer the influences representing the factory's many processes.

In the *model construction* task, the modeler uses the goals of the question (e.g., predict the plant's rate of transpiration) to select relevant influences, and it composes the influences into a model of the scenario. For instance, in the decreasing soil moisture example, the modeler would choose influences representing plant water regulation processes at an appropriate level of detail. Thus, scenario elaboration generates the building blocks for models, and model construction selects relevant building blocks.

In principle, scenario elaboration could be run to completion before model construction begins. Conceptually, scenario elaboration would identify all phenomena governing the scenario and all the levels of detail at which they could be described. However, for complex systems (such as a plant), this full description would be enormous, and generating it would take a long time. Not only are there an enormous number of phenomena governing a plant, but scenario elaboration might have to infer all the unstated anatomical details of the plant in order to recognize the governing phenomena. From this extensive description, the modeler will select only a tiny fraction to serve as the model. Moreover, typically only a small fraction of the full description is needed in order to choose that model. Therefore, exhaustively elaborating the scenario before making modeling decisions is both impractical and unnecessary.

To address this issue, our modeling methods use *demand-driven scenario elaboration*. That is, scenario elaboration only generates missing elements of the scenario description as they are needed. The modeler, in constructing its model of the scenario, requests information such as the influences on a specific variable. If the scenario description lacks the requested information, the modeler uses the domain knowledge to infer it, and the information is added to the scenario description. Thus, scenario elaboration and model construction are efficiently interleaved, and scenario elaboration only generates those elements of the scenario description needed for model construction.

1.8 Searching Through Partial Models

Given a question, a modeler must find the simplest model that can adequately answer it. Conceptually, there are three steps: construct candidate models, filter out those that are inadequate, and choose the simplest from the remaining models. However, this generate-and-test approach is impractical. For complex systems, the space of models is too large to generate. A modeler needs an efficient strategy for searching through the space of models while explicitly considering only a fraction of the models.

Our model construction algorithm achieves this goal by searching through the space of *partial models*. Starting from an initially empty model, the algorithm repeatedly identifies relevant aspects of the scenario not yet included in the model. The domain knowledge provides the possible ways of extending the model to remedy each omission. The algorithm extends the model in each possible way, resulting in new partial models, and the process is repeated. Conceptually, this algorithm constructs a tree of partial models; each child is an extension of its parent.

The key to this approach is the ability to eliminate partial models from further consideration. By pruning a partial model at an early stage, the algorithm prunes a large chunk from the space of possible models (i.e., all the model's future descendants in the tree). Our algorithm uses adequacy criteria to recognize partial models that cannot be extended into an adequate model of the scenario, and it prunes them. Furthermore, by extending simpler partial models before more complex ones, the algorithm finds the simplest adequate model without ever trying to remedy the omissions in many of the partial models under construction. Chapter 6 describes the algorithm in detail and proves that it always returns a simplest adequate model for a question when one exists, and the empirical evaluation described in Chapter 9 indicates that it does so efficiently.

1.9 Related Work

This dissertation builds on important previous work in compositional modeling. Our research was particularly influenced by the earlier work of Forbus [28] and Falkenhainer and Forbus [25]. Forbus's Qualitative Process (QP) Theory uses influences as the building blocks for models, and it shows how influences can be generated from domain knowledge. However, it does not provide methods for identifying relevant influences; in QP Theory, the complete scenario description *is* the model. Falkenhainer and Forbus extended the ideas in QP Theory to handle complex scenario descriptions containing multiple levels of detail. Their methods address the task of constructing the simplest adequate model for answering a question. The objectives of our research are also similar to more recent work that has progressed concurrently with our own, especially the work of Nayak [65] and Iwasaki and Levy [43].

Each of these pieces of related work, and many others, addresses a variety of issues. No single description of each previous modeling program can adequately describe the many similarities and differences between that program and ours. Therefore, discussions of these programs are spread throughout the dissertation, organized around individual issues.

In addition to previous work in automated modeling, our research has been guided by the practices of human modelers in biology, ecology, economics, and engineering. While human modelers rarely offer operational advice for automating their task, their textbooks (e.g., [4, 14, 31, 34, 45, 49, 59, 75, 76]) and journal articles (e.g., [32, 38, 39, 40, 50, 69, 78, 79, 82]) often reveal the modeling alternatives they consider and the criteria for their choices.

1.10 Summary of Contributions and Results

Our research provides three types of contributions. First, we formulated declarative criteria that specify when a model is adequate for answering a prediction question and when one model is simpler than another. Second, we designed algorithms for constructing the simplest adequate model for a given question. Finally, we implemented these algorithms in a modeling program called TRIPEL,[5] and we evaluated the program, the algorithms, and the criteria in the plant physiology domain.

The evaluation is especially significant for three reasons. First, the domain knowledge was independently developed by a domain expert. Second, the domain knowledge was designed to support a wide range of tasks besides prediction; it consists of fundamental textbook knowledge. Finally, the domain knowledge is extensive; it describes 700 plant properties and 1500 influences among them. These three factors make the evaluation far more ambitious than any previous evaluation of an automated modeling program.

To evaluate our contributions, we tested TRIPEL on plant physiology questions constructed by the domain expert. For each question, the expert assessed the model TRIPEL constructed. In addition, we evaluated the efficiency with which TRIPEL constructed the models and the importance of key components in TRIPEL. The results indicate that TRIPEL is already an effective modeling program; it typically constructs simple, adequate models, and it does so efficiently. Furthermore, our experiments show that several key components of TRIPEL play an important role in its success. Finally, the results suggest several natural extensions to TRIPEL that would remedy its limitations.

The following list highlights the important features of TRIPEL and the criteria and algorithms that underlie it:

- TRIPEL uses a novel representation to encode the phenomena governing a scenario at multiple levels of detail. The dissertation illustrates how the representation naturally supports a variety of types of abstractions and approximations that human modelers use throughout science and engineering. Our evaluation indicates that the representation is natural and effective for the plant physiology domain.

- TRIPEL uses demand-driven scenario elaboration to control the use of domain knowledge. This often obviates the need to generate many scenario influences.

- In addition to choosing a level of detail that is adequate for answering a given

[5] 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.

question, TRIPEL is sensitive to the user's level of knowledge and the desired level of detail. When considering a decision on level of detail, it checks for information indicating that a candidate level is inappropriate for the user. Such information could be provided on demand from a user model, a pedagogical plan, or the discourse context.

- TRIPEL can exploit a *time scale of interest*. The question can specify a time scale on which the predictions should be made (e.g., seconds, hours, days), and TRIPEL uses this time scale to simplify the model. Specifically, a time scale of interest allows TRIPEL to recognize and eliminate insignificant phenomena and to use simple, quasi-static representations of some phenomena. Furthermore, TRIPEL includes a novel method for determining an appropriate time scale of interest when none is specified. Empirical results show that many irrelevant details are eliminated from models by exploiting the time scale of interest.

- TRIPEL uses novel criteria and methods to choose *exogenous variables*, effectively eliminating exogenous phenomena from the model. Empirical results show that these criteria and methods are effective in eliminating irrelevant phenomena while ensuring an adequate model.

- TRIPEL uses a novel, best-first search algorithm for constructing the simplest model that is adequate for answering a given question, as discussed briefly in Section 1.8 and in more detail in Chapter 6. The algorithm is guaranteed to return a simplest adequate model when one exists, and empirical results indicate that it does so efficiently.

- TRIPEL bases its modeling decisions on the influences among properties of a scenario, but not on the quantitative details of the influences. Therefore, its methods are useful in building qualitative models [19, 28, 52, 54, 72] as well as quantitative models. Qualitative models are useful when quantitative details are unavailable (as is often the case in plant physiology) or irrelevant (as is often the case in tutoring). TRIPEL has been used to construct qualitative models, which it passes automatically to the Qualitative Process Compiler (QPC) [27], a qualitative simulation program. From TRIPEL's model, QPC generates the desired predictions.

1.11 Reader's Guide

The remainder of the dissertation is organized as follows:

- Chapter 2 describes the representation language TRIPEL uses to describe scenarios. A description of the scenario plays a central role in answering a prediction question: the question provides a partial description, the modeler uses domain knowledge to elaborate that description, and the modeler constructs a model from relevant elements of the elaborated description.

- Chapter 3 specifies the elements of a prediction question.

- Chapter 4 discusses demand-driven scenario elaboration, including the types of information it contributes to the scenario description, the types of domain knowledge needed, and the required inference methods. We show that a standard backward-chaining inference engine [12], coupled with the types of knowledge introduced in Qualitative Process Theory [28], is sufficient. The ideas in this chapter are not a primary contribution of our research; the chapter simply shows that the input required for the algorithms in the remaining chapters can be generated efficiently.

- While Chapters 3 and 4 define the inputs for model construction, Chapter 5 defines the output, a simplest adequate model of a scenario. This chapter defines a model and specifies the declarative criteria that determine whether a model is adequate and when one model is simpler than another.

- Chapter 6 presents the algorithm for constructing a simplest adequate model for answering a given prediction question. This algorithm efficiently searches through the space of partial models. After presenting the algorithm, we prove its correctness: the algorithm is guaranteed to return a simplest adequate model, as defined in Chapter 5.

- The algorithm for constructing a simplest adequate model has a subroutine that decides when a variable in a model can be treated as exogenous. Chapter 7 describes the relevant issues and provides an algorithm for making this decision.

- Chapter 8 provides an algorithm for automatically choosing an appropriate time scale of interest for a prediction question. A time scale of interest is an important source of power for a modeler, and the person posing a prediction question cannot always provide it, so this algorithm is an important component of any modeling program for answering prediction questions.

- Chapter 9 discusses an empirical evaluation of TRIPEL in the domain of plant physiology. The chapter discusses the details of the evaluation as well as the results. The results address the quality of the models TRIPEL constructs to answer questions, the efficiency with which it constructs a model for each question, and the importance of several key components of TRIPEL.

- Chapter 10 discusses areas for future work. The chapter discusses limitations of TRIPEL, it shows how TRIPEL could incorporate ideas from related research, and it suggests short-term and long-term extensions. The chapter closes by discussing how TRIPEL could be extended to answer questions other than prediction questions.
- Finally, Chapter 11 summarizes the dissertation.

Although some of TRIPEL's limitations are discussed in earlier chapters, most are postponed until Chapters 9 and 10. This places the limitations in the broader context of TRIPEL's overall performance and the proposed extensions.

1.12 Typographic Conventions

This dissertation uses a few simple typographic conventions to aid the reader. When a new term is introduced informally, it appears in italics (e.g., *widget*). When a formal term is defined, it appears in bold face (e.g., **widget**). Finally, variables, functions and relations used in algorithms appear in sans serif when they are mentioned in prose (e.g., relation).

Chapter 2

Describing Scenarios

A description of the scenario plays an important role in answering a prediction question. The question provides a partial description (e.g., a plant whose soil moisture is decreasing). The modeler uses domain knowledge to elaborate that description (e.g., to specify the influences governing the plant). From the elaborated description, the modeler chooses relevant elements (e.g., the influences representing plant water regulation processes) and composes them into a model of the scenario. The language for describing scenarios is an important part of any program that answers prediction questions.

This chapter presents TRIPEL's language for describing scenarios. The language combines elements from previous scenario description languages, and it also introduces some novel extensions. Subsequent chapters describe how this language is used to pose a prediction question, elaborate a partial scenario description, and construct a scenario model. The concepts introduced here are required for those chapters.

2.1 Scenario Variables: Properties of Entities

Scientists and engineers use *variables* to represent dynamic properties of physical systems. Each variable in an ODE model denotes a real-valued, continuous function of time. Examples include the amount of water in a plant and the rate of transpiration. To provide the elements from which models can be constructed, a scenario description must include such variables.

A variable in a scenario description is a *scenario variable*. To represent its meaning, the scenario description specifies each scenario variable as a *property* of some conceptual *entity* in the scenario. For instance, the following types of properties and entities are useful in many areas of science and engineering:

- A *space* is an entity. A plant is a simple example of a space. However, a space need not be spatially continuous; the collection of leaves of a plant can also be treated as a space. Similarly, the stomates of a plant (i.e., the pores in its leaves) can be treated as a space. Examples of properties that apply to spaces include volume and cross-sectional area (an important property of the stomates, which serve as a conduit for water vapor and other gases).

- A *pool* is an entity. A pool consists of the substance or energy of a particular type in a particular space. Examples of pools include the glucose in a plant, the heat in its leaves, and the water in its roots. Examples of properties that apply to pools include amount and concentration.

- A *process* is an entity. A process is a mechanism of continuous change. Examples include photosynthesis, osmosis, and growth. The state of a process is represented by its rate property.

Entities, properties and variables in TRIPEL's scenario description language are written as ground terms in Predicate Calculus [33]. For example, photosynthesis in a plant, which is an entity, can be written as photosynthesis(plant). The rate of photosynthesis in a plant, which is a scenario variable, can be written as rate(photosynthesis(plant)). Similarly, the amount of water in a plant, another scenario variable, can be written as amount(pool(water, plant)), where pool is a function that maps a type of substance and a space to the corresponding pool. In this representation, a property (e.g., amount) is a function that maps an entity to a real-valued, continuous function of time (i.e., a scenario variable).

2.2 Relations Among Entities: Entity Encapsulation

Often, the entities in a scenario 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 could be treated as an aggregate pool, or the water in the roots, stem and leaves could be treated individually. Analogously, 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 all examples of *entity encapsulation*, which is ubiquitous in science and engineering because it allows scientists and engineers to create abstract descriptions that hide irrelevant details.

A full description of a scenario may include entities at multiple levels of detail, as in the examples above. In order to choose a suitable level of detail and ensure a coherent model, a modeler must understand the relationships among entities in the scenario description. Encapsulation relationships among entities are represented with the **encapsulates** relation. The pair (E1, E2) is an element of this relation if and only if the entity E1 encapsulates the entity E2. 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 (if E1 encapsulates E2 and E2 encapsulates E3 then E1 encapsulates E3).

For example, for pools and processes, the encapsulates relation could be defined as follows:

- A pool encapsulates its *subpools* and *internal transport processes*. A subpool is a pool consisting of a subset of the aggregate pool's contents. The aggregate pool might be decomposed based on taxonomic distinctions in its substance type (e.g., glucose in the plant is a subpool of carbohydrates in the plant) as well as by partonomic distinctions (e.g., glucose in the leaves is a subpool of glucose in the plant). An internal transport process for a pool is a process that transports substance from one subpool to another, with no net loss or gain in the pool. For instance, phloem sap distribution transports sucrose in a plant from the photosynthesizing leaves to fruits and other parts of the plant that cannot produce sugars; this process is an internal transport process of the pool of sucrose in the plant. Thus, an aggregate pool encapsulates the details of its subpools and the processes that shift its contents among them.

- Analogously, a process encapsulates its *subprocesses* and *internal pools*. For example, photosynthesis is actually an aggregate process representing the net effects of two subprocesses, the light reactions and dark reactions, and each of those is an aggregation of many other chemical reactions. A process's internal pools are those pools influenced by its subprocesses but not included in the process's net effects. For example, the net effect of photosynthesis is to convert carbon dioxide, water and light into sugar and oxygen, but, in accomplishing this conversion, it alternately produces and consumes from an internal pool of phosphates in the leaves. Thus, an aggregate process encapsulates the details of its subprocesses and the internal pools they manipulate.

Note that the encapsulates relation represents relationships among alternative levels of description, not spatial relationships. The relation is useful 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).

In summary, when the scenario description includes multiple levels of detail, it must also represent the relationships among levels. Entity encapsulation is one important way of creating multiple levels of detail, and the encapsulates relation defines the relationship among such levels. A modeler needs such information to choose a suitable level of detail for describing the scenario and to ensure a coherent, comprehensible model.

2.3 Behavioral Conditions

A prediction question poses hypothetical conditions and asks for the resulting behavior of certain variables of interest. Often, the hypothetical conditions are stated in terms of scenario variables. There were several examples in the last chapter: "decreasing soil moisture," "increasing fan speed," and "decreasing insulin level." Scenario conditions that are stated in terms of scenario variables are called *behavioral conditions*.

In prediction, there are two important types of behavioral conditions. First, a question may specify the initial state of certain variables. For example, the temperature may be below freezing, or the level of soil moisture may be at the saturation point. Second, a question may specify the behavior of certain variables. "Decreasing soil moisture" is an example. For any scenario variable, the scenario description can specify its initial state, its behavior, or both.

The initial state of a variable is specified as an *(in)equality* (i.e., equality or inequality statement) comparing the variable or its first derivative to another variable or constant. For example, the initial temperature of a plant could be specified precisely as temperature(plant) = 67°F or less precisely as temperature(plant) > 32°F or temperature(plant) > temperature(soil). Its initial rate of change could similarly be specified (using the differential operator D) as D(temperature(plant)) = zero (thermal equilibrium) or D(temperature(plant)) > zero (the plant is warming up).

The behavior of a variable describes its state throughout the scenario. A scenario is not a static situation; it has temporal extent. For example, recall the question "How would decreasing soil moisture affect a plant's transpiration rate?" This question asks for the rate of transpiration over a period of time in which soil moisture is decreasing. The language for specifying the initial state of a variable can also be used to specify its behavior. For example, as a specification of behavior, D(amount(pool(water, soil))) $<$ zero says that soil moisture is decreasing throughout the scenario. Our implementation also allows 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). Although not implemented, the scenario description language could allow behaviors to be specified as arbitrary functions (e.g., a sine wave).

2.4 Structural Conditions

Some scenario conditions cannot be stated in terms of scenario variables. Examples include the existence of individuals like a plant and its soil, the species of plant, partonomic relations (e.g., a plant's parts include its roots, stem and leaves), and some spatial relations (e.g., the plant's roots are surrounded by soil). Any scenario condition that cannot be stated in terms of scenario variables is a *structural condition*. A structural condition is stated as a ground, atomic formula in Predicate Calculus [33]. For example, the formula Surrounded-by(roots,soil) represents the fact that the roots are surrounded by soil.

Our work focuses on ODE models. ODE models predict scenario changes in terms of scenario variables. For this reason, we assume that structural conditions remain constant throughout the scenario (e.g., the roots do not get pulled out of the soil). Structural conditions encode those scenario facts for which the domain knowledge lacks a theory of dynamics.

Our assumption that structural conditions remain constant is a matter of convenience, not necessity. The assumption simplifies presentation and implementation of the key ideas in our research. However, other alternatives are possible. For instance, Qualitative Process Theory [28] allows changes in structural conditions to result from changes in behavioral conditions. Section 10.2.4 shows how TRIPEL could be extended to support such a representation.

2.5 Influences

As discussed in Chapter 1, the phenomena governing a scenario are represented as influences, which serve as the building blocks for models. An **influence** is a causally-directed relation among two scenario 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 derivative) of the influencee is a function of the influencer (and perhaps other variables). 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 to form an equation, the result is a first-order differential equation. A differential influence is written as $\mathsf{V1} \Rightarrow \mathsf{V2}$, where the variable $\mathsf{V1}$ is the influencer and the variable $\mathsf{V2}$ is the influencee.

In contrast, a **functional influence** specifies that the influencee (rather than its first derivative) is a function of the influencer (and perhaps other variables). As with differential influences, there may be multiple functional influences on a variable. When combined to form an equation, they result in an algebraic equation. A functional influence is written as $\mathsf{V1} \rightarrow \mathsf{V2}$, where the variable $\mathsf{V1}$ is the influencer and the variable $\mathsf{V2}$ 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. If the dynamics of this process 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. Quasi-static approximations are important in many branches of science and engineering [14, 46, 77, 78, 80, 82]. In fact, several branches of engineering, notably circuit theory and equilibrium thermodynamics, rest on such approximations [13, 83]. A functional influence that represents a quasi-static approximation is called an *equilibrium influence.*

An equilibrium influence summarizes the net effect of some set of processes on the equilibrium state of the influencee. When the scenario description includes both the equilibrium influence and the underlying processes, the relationship between these two levels of detail must also be represented. To represent this relationship, an equilibrium influence can be associated with an aggregate process that encapsulates the underlying pools and processes that restore equilibrium. By representing the relationship between the two levels of detail, a modeler can choose a suitable level and ensure that the model does not mix the two levels.

2.6 Attributes of Influences

An influence represents some phenomenon at some level of detail. In addition to the influencer, influencee, and type (i.e., differential or functional) of an influence, a modeler must know three other things: when the phenomenon is active, when the phenomenon is significant, and when the influence is a valid approximation of the phenomenon. The following three sections discuss these attributes of influences.

2.6.1 Activity Preconditions

Sometimes one variable influences another only under certain behavioral conditions. For example, the amount of carbon dioxide in the leaves influences the rate of photosynthesis only if the amount of light energy in the leaves is greater than zero. The *activity preconditions* of an influence specify the behavioral conditions under which it is active.

The **activity preconditions** of an influence are a (possibly empty) conjunctive set of behavioral conditions. At a given time in the scenario, an influence is active if and only if each of its activity preconditions is satisfied. (Hence, if it has no activity preconditions, it is always active.)

2.6.2 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.

Processes cause significant change on widely disparate time scales [4, 36, 68, 76, 80]. For example, in a plant, water flows through membranes on a time scale of seconds, solutes flow through membranes on a time scale of minutes, growth requires hours or 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. 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 a *time scale condition.* A **time scale condition** is an (in)equality 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 the time scale condition 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 time scale condition in the influence's significance preconditions.

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. Functional influences, being instantaneous, are significant regardless of the time scale of interest, so their significance preconditions are always empty.

In addition to biological and ecological domains, this type of time scale knowledge appears useful in engineering domains as well. Kokotovic, O'Malley, and Sannuti [50] and Saksena, O'Reilly, and Kokotovic [78] survey hundreds of applications, in many different engineering fields, in which models are simplified using knowledge of the disparate time scales of processes.

2.6.3 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 preconditions, human modelers use many criteria to assess the validity of influences, but the time scale of interest is particularly important. Therefore, the **validity preconditions** of an influence are encoded as a time scale condition. 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 much faster than the time scale of interest [45, 82]. For example, suppose that diffusion of carbon dioxide throughout the leaves achieves a uniform concentration on a time scale of minutes. Then, on a time scale of minutes or longer, the overall concentration of carbon dioxide in the leaves is approximately the same as the concentration in the mesophyll cells. Thus, the influence of carbon dioxide in the leaves on the rate of photosynthesis is a valid approximation to the true influence if the time scale of interest is minutes or longer.

- An equilibrium influence is typically valid only if the underlying processes reach equilibrium on a time scale at least as fast as the time scale of interest. 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 equilibrium influence of solute level on water level is valid on a time scale of minutes or longer.

2.7 Relations Among Influences: Explanation

A full description of a scenario may include multiple levels of detail. Section 2.2 discussed how entities can be described at multiple levels of detail, and it showed how the encapsulates relation represents the relationships among such levels. Similarly, since each influence in a scenario description represents a phenomenon in the scenario, 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 the relationships among all the influences in the scenario description on that variable. Specifically, the modeler must determine which of them represent independent phenomena and which represent different levels of detail for the same phenomenon.

Influences on a given scenario 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.

- 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 these cases, TRIPEL's scenario language allows one influence to be 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. Specifically, 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.

The relationship between an influence and the influences that explain it is represented by the **explanation** relation. The pair (i1,i2) is an element of this relation if and only if influence i2 is an element of the set of influences that explain influence i1. The explanation relation is irreflexive and asymmetric. The transitive closure of the explanation relation is the explanation* relation, which provides an ordering among influences.

The explanation relation represents the relationships among influences of the same type having the same influencee. While there may be similar relationships among influences with different types or influencees, TRIPEL's modeling criteria and algorithms do not require a representation of these relationships. The explanation relation captures those relationships among influences that are relevant to TRIPEL.

2.8 Related Work

Forbus's Qualitative Process (QP) Theory [28] provides the basis for our scenario description language. In QP Theory, scenario variables are called "quantities," and quantities are properties of entities. Differential influences are called "direct influences" and functional influences are called "indirect influences." The activity preconditions of influences are called "quantity conditions." Our structural and behavioral conditions also have close counterparts in QP Theory. However, because QP Theory was not designed to represent modeling alternatives, it does not include a representation for significance preconditions, encapsulation or explanation relationships, or validity preconditions.

Several researchers have addressed the issue of representing modeling alternatives in a compositional modeling framework. For instance, entities in Zeigler's "system entity structure" [89] represent systems, and each entity can be decomposed (possibly in multiple ways, called "aspects") into other entities that represent its components. Each entity has associated variables as well as models that describe the behavior of the variables.

Our scenario description language was most influenced by the compositional modeling framework of Falkenhainer and Forbus [25], which combines and extends ideas from Forbus's QP Theory and Zeigler's system entity structure. Entities represent systems, and each entity can be decomposed into component entities. "Model fragments," the building blocks for models, pertain to entities or specific configurations of multiple entities; model fragments provide individual influences or complete equations for variables that represent properties of the entities. 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 an assumption class represent mutually incompatible modeling alternatives for an entity or phenomenon. Several researchers [43, 65] define interesting variants of this compositional modeling framework, but the basic ideas are widely used.

While our early work adopted the representation introduced by Falkenhainer and Forbus, our representation gradually evolved into its current form for several reasons. First, we chose to use individual influences as model fragments in order to focus the modeler's reasoning at the level of individual phenomena. Influences serve a similar role in a variety of areas of science and engineering [10, 31, 55, 72, 75]. Second, we chose to representationally distinguish activity preconditions, significance preconditions, and validity preconditions, since each plays a distinct role in our modeling algorithms. Third, the explanation relation among influences, which plays an important role in our modeling algorithms, is awkward to represent using assumptions and assumption classes.

Still, on top of these extensions, we could allow influences to be tagged with assumptions, and we could allow assumption classes and rules to specify relationships among these assumptions. However, from carefully studying the types of knowledge that researchers encode using assumptions, we believe that this use of assumptions would not provide any important generality over our current representation. Many such types of knowledge, such as the relationships among alternative levels of detail, can be sufficiently encoded in our representation. Other types of knowledge, such as the conditions under which an assumption class is relevant to a question, are unnecessary because our modeling algorithms have sufficient criteria for making such decisions. Thus, although our modeling algorithms could exploit assumptions and assumption classes with little modification, we chose to avoid these features until their utility is clearly demonstrated.

Throughout areas of science and engineering, time scale is used to determine significance of phenomena and validity of approximations. Yet few researchers in automated modeling have exploited knowledge of time scales. Kuipers [53] shows the utility of decomposing models by time scale, and he provides a simulation algorithm for coupling such models, but a human modeler is required to use their knowledge of time scales to decompose the models. The modeling algorithm developed by Yip [47] can be viewed as removing insignificant influences based on given scale parameters, including the length, time, and velocity scales of interest. The Extended Adiabatic Elimination method of Dieckmann and Williams [20] simplifies a set of differential equations by using quasi-static approximations wherever possible. Iwasaki [42] presents an approach to using time scale that is closest to ours; her modeling algorithm determines the time scale on which each scenario process operates, and it ignores those that are slower than the time scale of interest while treating those that are faster as instantaneous.

2.9 Summary

TRIPEL's scenario description language allows several important types of information to be specified. Scenario variables represent the dynamic properties of entities in the scenario. Behavioral conditions represent the initial state and behavior of variables. Structural conditions represent static properties of the scenario. Influences represent the phenomena that govern the scenario. Activity preconditions specify when these phenomena are active, significance preconditions specify when they are significant, and validity preconditions specify when an influence is a valid approximation of the phenomenon it represents. These types of information are all important in constructing an adequate model of a scenario and in using that model to make predictions.

Additionally, the language allows a scenario to be described at multiple levels of detail. In particular, some entities may encapsulate other entities, summarizing their properties while hiding their details. Similarly, one influence may represent the net effect of several other influences. To ensure a coherent, comprehensible model that captures all relevant aspects of the scenario, a modeler needs to understand the relationships among different levels of detail. For this purpose, the encapsulates relation represents relationships among entities, and the explanation relation represents relationships among influences.

Chapter 3

Causal Prediction Questions

Our research addresses the task of automatically constructing models for answering prediction questions. Informally, a prediction question poses a hypothetical scenario and asks for the resulting behavior of particular variables of interest. More formally, there are two components of a prediction question: the scenario and the goals.

3.1 Scenario

A prediction question provides a partial description of a scenario, expressed in the language introduced in Chapter 2. Specifically, a prediction question specifies structural and behavioral conditions. As illustrated in Figure 3.1, the behavioral conditions specified in a question can include initial conditions, behaviors of selected variables, or both. Additional elements of the scenario description, such as influences and unstated structural conditions, are added during scenario elaboration, which is the subject of Chapter 4.

Behavioral conditions specified in a prediction question are special, because the person posing the question is interested in the effect of these conditions on the variables of interest. Therefore, we will refer to such conditions as *driving conditions*, and the scenario variables appearing in the driving conditions will be called *driving variables*.

3.2 Goals

While the scenario specifies the situation to be modeled, *goals* are needed to determine which aspects of the scenario are relevant. Different types of questions are distinguished by different types of goals. The following sections describe the types of goals that arise in prediction questions.

Structural Conditions	
Plant(plant1)	*hypothetical plant*
Soil(soil1)	*hypothetical soil*
Has-Part(plant1, roots1)	*the plant has roots*
Surrounded-by(roots1, soil1)	*the roots are surrounded by the soil*
Initial conditions	
temperature(plant1) $> 32°$F	*plant temperature is above freezing*
Behaviors	
D(amount(pool(water,soil1))) $<$ zero	*soil moisture is decreasing*

Figure 3.1: An example of a scenario description provided by a question.

3.2.1 Variables of Interest

The primary goal in a prediction question is to predict the behavior of specified variables of interest. This goal provides the necessary focus for modeling, enabling a distinction between relevant and irrelevant aspects of the scenario. For this reason, we require a prediction question to include at least one variable of interest.

Furthermore, our research focuses on *causal* prediction questions, in which the person posing the question wants to know the *causal* effect of the driving conditions on the variables of interest. For example, the question "How would decreasing soil moisture affect a plant's transpiration rate?" asks for the causal effect of decreasing soil moisture on the rate of transpiration. In contrast, consider the question "What is the rate of inflow into a bathtub if the level of water remains constant and the rate of outflow is five gallons per minute?" This question has the basic elements of a prediction question — structural conditions, behavioral conditions, and a variable of interest — but it is not a causal prediction question. The rate of outflow and the level of water do not cause the behavior of the inflow rate (at least not in my tub). Because our research addresses causal prediction questions, we require the question to include at least one driving condition.

There are two reasons for focusing on causal prediction questions. First, many prediction questions in science and engineering are causal ones. Second, causality provides important guidance in modeling, as will be shown later. Throughout this dissertation, "question" and "prediction question" are merely shorthands for "causal prediction question."

3.2.2 Desired Level of Detail

To ensure a comprehensible model, a modeler must choose a level of detail that is suitable for the person posing the question. For example, a model suitable for a first-year botany student will be much simpler than a model suitable for a veteran plant physiologist. The level of detail might also be tailored to fit the discourse context. For example, a tutor might use a student's question to illustrate domain principles that were recently discussed. To guide the choice of a suitable level of detail, a question can include a *desired level of detail.*

The desired level of detail is specified through *glass-box entities* and *black-box entities.* A glass-box entity is an entity (e.g., pool or process) that is too simple. If such an entity, or a simpler, encapsulating entity, is needed in the model, the detailed entities that it encapsulates should be used instead. For example, if photosynthesis is marked as a glass-box entity, the modeler must use its component reactions in its place. In contrast, a black-box entity is an entity whose underlying component entities should not be used because they are too detailed. A black-box entity prevents the modeler from using any entity that it encapsulates. For example, if photosynthesis is marked as a black-box entity, the modeler can include photosynthesis in a model but cannot use any of its component reactions. Together, glass-box and black-box entities prevent the modeler from choosing a level of detail that is inappropriate for the person posing the question.

Although glass-box and black-box entities could be specified explicitly in a question, they might come from a variety of sources. They might come from an overlay user model [35]. For example, if a student understands photosynthesis but not its component reactions, photosynthesis can be marked as a black-box entity. They might also come from a discourse history. For example, if a tutor has recently discussed the component reactions of photosynthesis, photosynthesis might be marked as a glass-box entity. Glass-box and black-box entities provide a simple interface to each of these information sources.

3.2.3 Time Scale of Interest

As discussed in Chapter 2, a time scale of interest provides an important source of power in modeling. It allows the modeler to treat influences that operate on a slower time scale as insignificant. It allows the modeler to model the effects of faster processes using equilibrium influences, based on a quasi-static approximation. It allows the modeler to 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 the question may specify a time scale of interest, often the modeler must determine it automatically. Chapter 8 provides 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, remaining chapters will treat it as part of the question.

3.3 Summary

In summary, our research addresses the task of automatically constructing models for answering causal prediction questions. We define a causal prediction question to consist of the following elements:

- structural conditions
- driving conditions (at least one), consisting of initial conditions, behaviors or both
- variables of interest (at least one)
- desired level of detail, specified as glass-box and black-box entities
- time scale of interest

Chapter 4

Scenario Elaboration

4.1 The Role of Scenario Elaboration

Given a prediction question, a modeler uses domain knowledge to elaborate the scenario description, in order to provide the building blocks for model construction. The prediction question provides a partial description of the scenario, consisting of structural and behavioral conditions. To this description, the domain knowledge adds the influences that represent phenomena in the scenario, along with the relationships among different levels of detail (i.e., the encapsulates and explanation relations). From the resulting scenario description, the modeler constructs a model of the scenario by selecting relevant influences. Thus, the process of scenario elaboration uses the domain knowledge to bridge the gap between the scenario description given in the question and the needs of model construction.

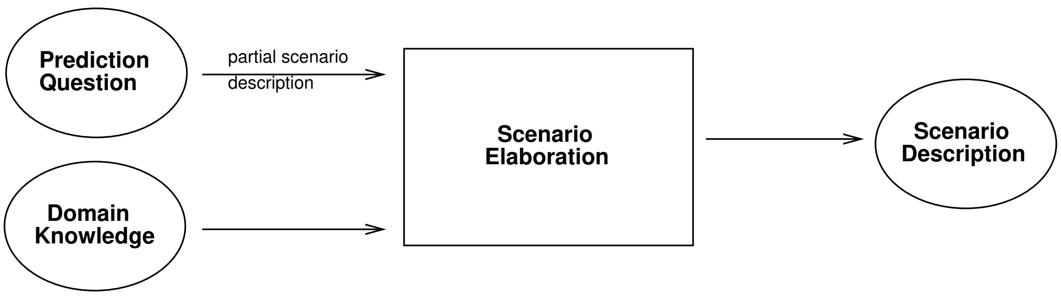


Figure 4.1: The scenario elaboration task.

Define-Influence-Rule
  structural preconditions:
    Plant(?plant)
    Has-Part(?plant, ?roots)
    Roots(?roots)
    Surrounded-by(?roots, ?soil)
    Soil(?soil)
  influence:
    rate(water-uptake(?soil, ?plant) $\Rightarrow$ amount(pool(water, ?plant))
  activity preconditions:
    None
  validity preconditions:
    None
  significance preconditions:
    time-scale-of-interest $\geq$ hours

Figure 4.2: An example influence rule. This rule states "For any plant whose roots are surrounded by soil, the amount of water in the plant is differentially influenced by the rate of water uptake from the soil into the plant. Terms that begin with "?" are universally quantified, logical variables.

4.2 Domain Knowledge

This section describes the types of domain knowledge that are needed to elaborate a scenario description given in a question. The section shows that standard knowledge representation techniques, based on inference rules, can encode the necessary knowledge. However, the specific representation language is not important, since a variety of similar alternatives will serve the same purpose.

4.2.1 Influence Rules

Starting from a partial scenario description given in a question, a modeler must use domain knowledge to identify the influences that represent phenomena in the scenario. To support this task, the domain knowledge includes *influence rules*.

As illustrated by the example in Figure 4.2, an influence rule consists of structural preconditions, which serve as the antecedent of the rule, and an influence and its attributes (activity preconditions, validity preconditions, and significance preconditions), which serve as the consequent of the rule. For example, the rule in

Figure 4.2 states "For any plant whose roots are surrounded by soil, the amount of water in the plant is differentially influenced by the rate of water uptake from the soil into the plant." It also states that this influence is always active (i.e., no activity preconditions), always valid (i.e., no validity preconditions), and significant on a time scale of hours or longer. Structural preconditions consist of a conjunction of structural conditions, and these conditions may contain universally quantified, logical variables (shown as terms beginning with "?" in the example).[1] A logical variable introduced in the structural preconditions may be referenced in the influence and its activity preconditions. Viewed as a logical implication, an influence rule states that the specified influence, with its specified attributes, can be used to describe any scenario in which each structural precondition is satisfied.

4.2.2 Structural Rules

Typically, a prediction question leaves many structural conditions unstated. For example, consider the question "How would decreasing soil moisture affect a plant's transpiration rate?" The question does not mention that the plant has a source of light, that its leaves are surrounded by the atmosphere, or that it has any anatomical parts (e.g., leaves). Yet such structural conditions may serve as structural preconditions for important influences in the scenario. Therefore, in order to infer all the influences for a scenario, the domain knowledge must provide rules for inferring unstated structural conditions. Such rules are called *structural rules*.

The antecedent of a structural rule is a conjunction of structural conditions and the consequent is also a structural condition. Any logical variables appearing in the antecedent may also appear in the consequent, and they are universally quantified throughout the rule. For example, the rule

If Tree(?t) Then Plant(?t)

says "Every tree is also a plant."

Structural rules can also infer the existence of unstated objects. For example, consider the logical implication

If Plant(?p) Then $\exists$?l: Leaves(?l) and Has-Part(?p, ?l)

which says "For every plant, there exists a part of the plant, its leaves." This implication can be encoded as the following logically equivalent structural rules:

[1] Logical variables, which can only appear in the domain knowledge, should not be confused with scenario variables, such as the amount of water in a plant, which represent properties of the scenario.

If Plant(?p) Then Leaves(leaves(?p))
If Plant(?p) Then Has-Part(?p, leaves(?p))

where leaves is a Skolem function [11]. Thus, structural rules can infer structural conditions concerning unstated objects as well as objects mentioned in the question.

Clearly, inferring unstated structural conditions is not a simple matter of deduction. For example, a plant does not necessarily have a source of light, a surrounding atmosphere, and leaves; yet without evidence otherwise, a person answering a prediction question will assume such conditions if they are unstated. Thus, structural rules may be default rules (i.e., plausible, although not certain, inferences). However, the issues that arise when using default rules (e.g., detecting contradictions and retracting assumptions) are orthogonal to the issues addressed in this dissertation, so they are ignored.

4.2.3 Relationships Among Levels of Detail

The domain knowledge also needs rules that allow the modeler to infer the encapsulates and explanation relations. These relations can be inferred with simple rules whose antecedents consist of structural conditions. For example, Figure 4.3 shows a rule for inferring that a pool encapsulates its subpools. This rule could be used to conclude that the pool of sugar in a plant encapsulates the pool of glucose in the plant's leaves. Similarly, Figure 4.4 shows an example rule for inferring that one influence explains another. In our implementation, many elements of the explanation relation are inferred through two general rules:

- The influence of a process (e.g., photosynthesis) on a pool (e.g., water in the leaves) is explained by the influence of its subprocesses (e.g., the light reactions) on that pool.
- The influence of a pool (e.g., carbon dioxide in the leaves) on a process (e.g., photosynthesis) is explained by the influence of its subpools (e.g., carbon dioxide in the mesophyll of the leaves) on that process.

By providing rules for inferring the encapsulates and explanation relations, the domain knowledge can specify relationships among different levels of detail in the scenario description.

4.2.4 Related Work

Our approach of generating influences from rules in the domain knowledge is based on Forbus's Qualitative Process Theory [28]. Other researchers have used similar

If Has-Part(?whole, ?part) and *e.g., a plant has leaves*
Contains(?whole, ?general) and *e.g., the plant contains sugar*
Contains(?part, ?specific) and *e.g., the leaves contain glucose*
Isa(?specific, ?general) *e.g., glucose is a special type of sugar*
Then Encapsulates(pool(?general, ?whole), pool(?specific, ?part))

Figure 4.3: An example rule for inferring the **encapsulates** relation. This rule says that a pool encapsulates its subpools (e.g., the pool of sugar in a plant encapsulates the pool of glucose in the plant's leaves).

If Plant(?p) and Has-Part(?p, ?l) and Leaves(?l)
Then Explanation(rate(photosynthesis(?l)) $\Rightarrow$ amount(pool(water, ?l)),
rate(light-reactions(?l)) $\Rightarrow$ amount(pool(water, ?l)))

Figure 4.4: An example rule for inferring the **explanation** relation. This rule says that the influence of photosynthesis on the water in the leaves is explained by the influence of its component reaction, the light reactions, on water in the leaves.

approaches, and some have explored the issue of matching structural preconditions to structural conditions in the scenario when a strict syntactic match is not possible. For instance, to construct models for solving textbook physics problems, Kook and Novak [51] propose model fragments called "physical models." Physical models can be viewed as inference rules whose antecedent consists of structural preconditions and whose consequent is a single principle or law of physics. However, the structural preconditions are stated in terms of "canonical objects" [67] (e.g., a point mass or an ideal spring), and additional knowledge is used to map scenario entities to appropriate canonical objects. Similarly, applying knowledge in one domain to questions in another domain may require analogical mapping between structural conditions in the scenario and structural preconditions of influence rules [16, 22].

4.3 Demand-Driven Scenario Elaboration

For complex systems such as a plant, exhaustive scenario elaboration is impractical. If all possible structural rules and influence rules were applied to a partial description of a plant, the result would be a full anatomical and physiological description of the plant. This description would be enormous, and the time required to execute all the rules could be prohibitive.

Moreover, exhaustive scenario elaboration is unnecessary. The modeler will select only a tiny fraction of the full description to serve as the model. The modeler may need to examine a larger fraction in order to choose the appropriate model, but that fraction will still be only a small portion of the full description.

For these reasons, our modeling methods allow *demand-driven scenario elaboration*. During model construction, the modeler requests information from the scenario description through a carefully designed interface. If the scenario description lacks the requested information, the domain knowledge is used to infer it, and the information is added to the scenario description. The interface cleanly separates model construction from scenario elaboration issues, and it allows the domain knowledge to be applied selectively.

4.3.1 Scenario Description Interface

During model construction, a modeler needs to know how scenario variables interact. There are two types of interaction. First, one variable can influence another. Second, one variable can *enable* an influence on another variable; that is, the first variable appears in the activity preconditions of an influence on the second variable. To index into these two types of interactions, the scenario description interface allows the following three requests:

- Given a scenario variable v, return all influences in which v is the influencer (i.e., how does v influence other scenario variables?).

- Given a scenario variable v, return all influences whose activity preconditions reference v (i.e., which influences does v enable?).

- Given a scenario variable v, return all influences in which v is the influencee (i.e., how do other variables influence v?).

To specify the relationships among different levels of detail, the scenario description interface provides the following two functions:

- The function encapsulates? takes two entities, e1 and e2, and returns true if and only if e1 encapsulates e2 (i.e., (e1,e2) is an element of the encapsulates relation).

- The function explanation? takes two influences, i1 and i2, and returns true if and only if i2 explains i1 (i.e., (i1,i2) is an element of the explanation relation).

Thus, given any two entities in the scenario description, the modeler can determine whether one encapsulates the other, and given any two influences in the scenario description, the modeler can determine whether one explains the other.

4.3.2 Implementing the Interface

Demand-driven scenario elaboration is implemented by *backward chaining* [12] through the inference rules provided by the domain knowledge. For example, suppose a prediction question specifies one structural condition, a plant, and the modeler asks for the influences on the amount of water in the plant. Figure 4.5 shows how the influence of transpiration can be found by backward chaining through one influence rule and two structural rules. First, the request is encoded as an influence (in this example, a differential influence) in which the influencer is unknown (i.e., encoded as the logical variable ?influencer). Next, the request is unified with the consequent of an influence rule; this indicates that the influence rule is relevant to the request. The influence rule applies to the scenario if its structural preconditions are satisfied, so backward chaining continues by trying to satisfy each precondition. A structural precondition is established in two ways: by unifying it with an existing structural condition in the scenario description (e.g., Plant(plant1) in the example) or by backward chaining through structural rules (as shown for the second and third structural preconditions in the example). In the example, backward chaining successfully establishes all structural preconditions, resulting in the influence

$$\mathsf{rate(transpiration(plant1, atmosphere(plant1))} \Rightarrow \mathsf{amount(pool(water, plant1))}$$

which is returned as one of the requested influences. The attributes of the influence (its activity, validity, and significance preconditions) are uniquely specified by the ground instance of the influence rule. Thus, backward chaining allows new facts (e.g., influences and structural conditions) to be inferred from the domain knowledge and added to the scenario description as they are needed during model construction.

The interface specified earlier is implemented through backward chaining as follows:

- To find all influences in which a given scenario variable v is the influencer, backward chain on the query v $\Rightarrow$?influencee to find differential influences and on the query v $\rightarrow$?influencee to find functional influences. For these queries, v is fixed and backward chaining establishes bindings for ?influencee.

- To find all influences that a given scenario variable v enables, unify v with scenario variables appearing in the activity preconditions of influence rules, and backward chain on the structural preconditions of these rules.

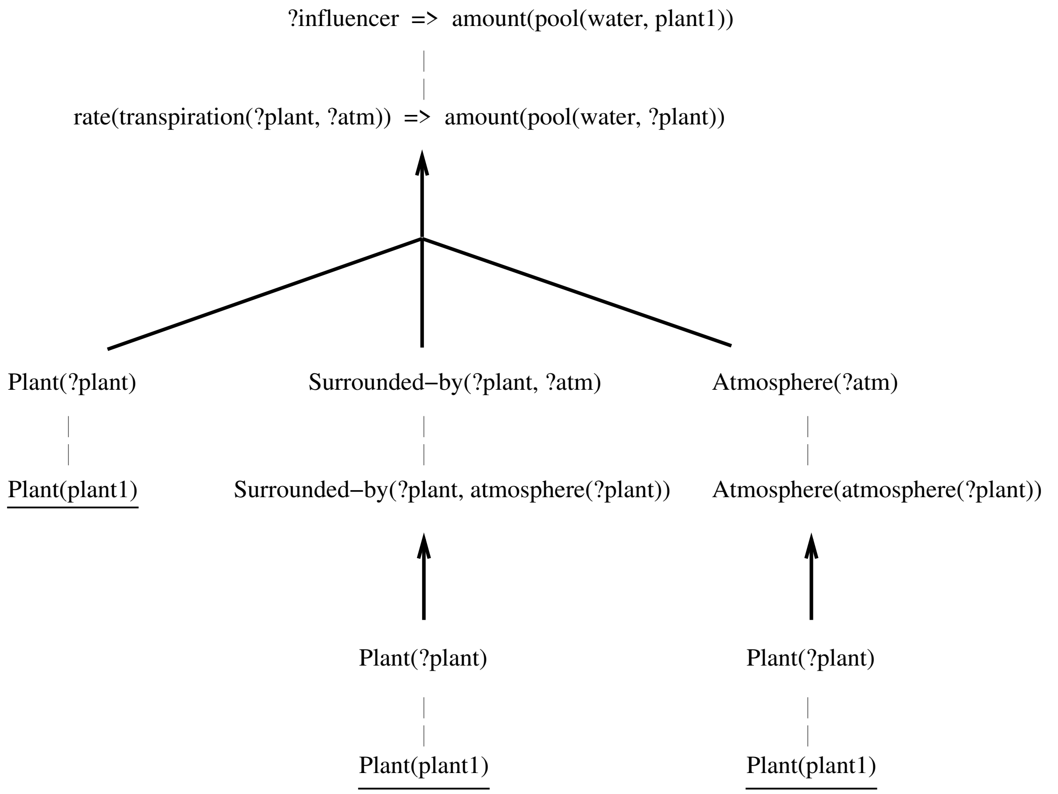


Figure 4.5: Demand-driven scenario elaboration, as implemented by backward chaining. The initial query, shown at the top of the figure, requests a variable that differentially influences the amount of water in the specified plant. Dashed lines represent unification. For example, the initial request for a differential influence is unified with the consequent of the influence rule. Arrows represent inference rules; they point from antecedents to consequents. Underlined facts are those given in the prediction question. This example shows how the influence of transpiration on the amount of water in the plant is found by backward chaining through one influence rule and two structural rules.

- To find all influences in which a given scenario variable v is the influencee, backward chain on the query ?influencer $\Rightarrow$ v to find differential influences and on the query ?influencer $\rightarrow$ v to find functional influences. For these queries, v is fixed and backward chaining establishes bindings for ?influencer.

- To answer requests of the form encapsulates?(e1, e2), backward chain on the query encapsulates(e1, e2), where e1 and e2 are constants (the names of particular entities).

- To answer requests of the form explanation?(i1, i2), backward chain on the query explanation(i1, i2), where i1 and i2 are constants (particular influences).

4.3.3 Related Work

Previous automated modeling programs have used demand-driven scenario elaboration to varying degrees. Some programs (e.g., those of Falkenhainer and Forbus [25] and Lee [56]) exhaustively elaborate the scenario before starting model construction. Other programs (e.g., those of Nayak [66] and Iwasaki and Levy [43]) allow some interleaving of scenario elaboration and model construction. Williams's program [85] automatically generates some equations (the building blocks for his models) via algebraic simplification as they are needed during model construction. Finally, Amsterdam's program [5] uses a model's deficiencies to enable selected rules for adding elements to the model, effectively interleaving some aspects of scenario elaboration with model construction.

4.4 Summary

Scenario elaboration uses the domain knowledge to bridge the gap between the partial scenario description given in a question and the needs of model construction. This chapter specified the scenario description interface with which the needs of model construction are communicated, it discussed the types of domain knowledge that are required, and it presented a simple method, backward chaining, for inferring the requested information on demand. Demand-driven scenario elaboration allows scenario elaboration and model construction to be interleaved, which is far more efficient than performing them serially.

While the notion of demand-driven scenario elaboration is important, the underlying details are not. Backward chaining is a standard method of inference in artificial intelligence, and the types of domain knowledge we propose are similar to those proposed by other researchers. Many variants of backward chaining and of these types of domain knowledge would serve the same purpose. The point of this chapter is that demand-driven scenario elaboration is important and that it can be implemented using standard techniques.

The remainder of the dissertation does not depend on the details of scenario elaboration. The remaining methods only depend on the details of the scenario description language specified in Chapter 2 and the details of the scenario description interface specified in this chapter.

Chapter 5

The Model Construction Task

5.1 Introduction

Given the background provided by previous chapters, we are finally prepared to discuss the main focus of this dissertation: the model construction task. As illustrated in Figure 5.1, this task has two inputs:

- a causal prediction question, which provides variables of interest, driving conditions, a desired level of detail, and a time scale of interest, and
- a scenario description (the output of scenario elaboration), which provides influences and relationships among different levels of detail (i.e., encapsulates and explanation relations).

While these inputs were defined in previous chapters, this chapter defines the output, a simplest adequate scenario model. Section 5.2 defines a scenario model. Section 5.3 specifies the criteria for judging one model as simpler than

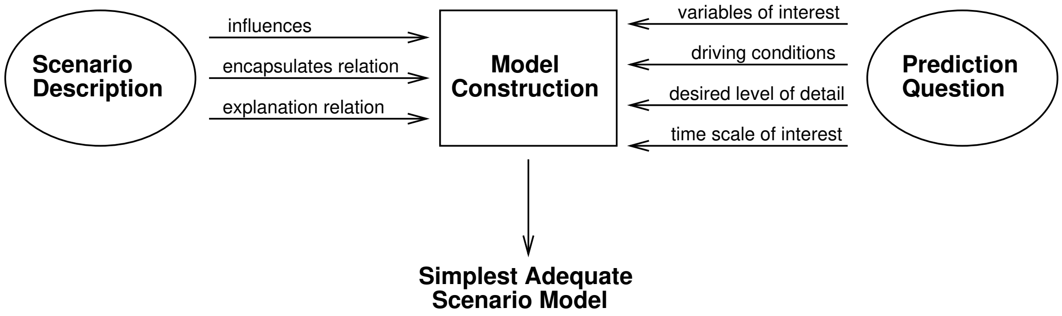


Figure 5.1: The model construction task.

another. Section 5.4 specifies the criteria for determining whether a scenario model is adequate for answering a question. The criteria in this chapter allow a modeler to choose the simplest adequate scenario model from among a set of candidate models for answering a question. The algorithm for generating candidate models will be discussed in Chapter 6.

The criteria in this chapter, especially those that define an adequate model, are an important contribution of our research. These criteria address several issues not addressed in previous research. Furthermore, because the criteria are stated declaratively, they can be evaluated independent of any modeling algorithms that use them, and they serve as the correctness standard for such modeling algorithms. Thus, this chapter serves as an independent contribution of our research as well as a foundation for subsequent chapters.

5.2 Scenario Models

From the elements of a scenario description, a modeler constructs a scenario model. A **scenario model** consists of the following:

- a set of variables (a subset of the scenario variables) partitioned into *exogenous* variables, whose behavior is determined by influences external to the model, and *dependent* variables, whose behavior is determined by the model
- a set of influences (a subset of the scenario influences), 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 5.2 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, where it helps limit the amount of water lost through transpiration.

A scenario model is intended to support analysis (e.g., simulation) regardless of particular behavioral conditions. To make predictions from a particular state of the scenario, the analysis module 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 healthy 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

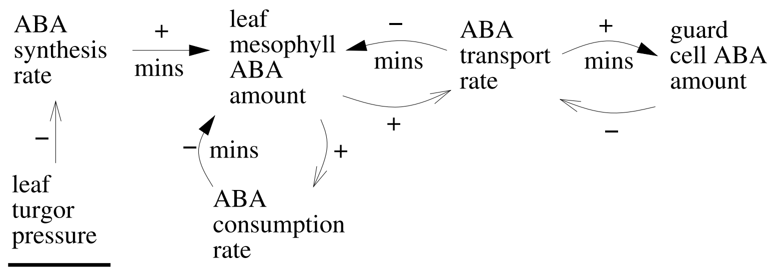


Figure 5.2: A scenario model. This and subsequent figures use the following conventions:

- Arrows with solid tips represent differential influences.
- 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.

models in this way [27, 28, 30]. 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.

5.3 Simplicity

To answer a prediction question, a modeler should construct the simplest adequate scenario model, minimizing irrelevant phenomena and details. If the model includes irrelevant information, it will be more difficult to analyze and explain. Thus, a modeler requires criteria for determining whether one candidate model is simpler than another.

While not infallible, the number of variables in a model is a good heuristic measure of the model's complexity. Simulation complexity tends to increase with the number of variables in the model, 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 M1 and M2, M1 is **simpler** than M2 if and only if M1 has fewer variables than M2.

Human modelers probably use a combination of many criteria to determine the complexity of a model. For example, for some purposes, a large linear model is preferable to a small nonlinear one. For tutoring purposes, one model might be simpler than another if it relies on concepts that are simpler to explain. 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.

Other researchers have proposed different measures of simplicity. Nayak [65] and Iwasaki and Levy [43] 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.[1] 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 great detail); if the first model treats some aspect of the scenario in more detail than the second model, the two models are incomparable under their criterion.

[1] 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.

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.

Falkenhainer and Forbus [25] define one model as simpler than another if the first includes fewer scenario entities. If two models include the same number of entities, one is simpler than the other if its sum of indices of assumption class choices is smaller. An assumption class represents a set of modeling alternatives, and one alternative has a lower index if it is simpler. Our representation does not use assumption classes, so this approach is not feasible.

Amsterdam [5] uses a simplicity criterion very similar to ours. Bond graph elements [48] are the building blocks for his models, and he defines one model as simpler than another if it contains fewer elements.

5.4 Adequate Scenario Model

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

To formalize the criteria, this section provides a set of adequacy constraints. Each constraint is a predicate of three arguments: a scenario description, a causal prediction question, and a scenario model. These constraints are individually necessary and collectively sufficient conditions for adequacy; that is, a scenario model is adequate for a given scenario description and question if and only if every adequacy constraint is satisfied.

In formulating adequacy constraints, we have two objectives. First, they should capture the two intuitive criteria for adequacy. Second, they must be operational; that is, an automated modeling program must be able to efficiently test them to decide whether a given model is adequate. The second objective requires the adequacy constraints to reference only information that is available to a modeler. For example, we cannot require a model's predictions to match the "correct" behavior if the correct behavior is unknown.

Mathematicians, such as systems theorists, have invested considerable work into formalizing the notions of adequate model and valid simplification. However, their criteria suffer from two limitations: they typically apply to restricted classes of systems, and they typically are not operational. For instance, the notion of a valid simplification is usually defined relative to a known base model (i.e., most detailed and accurate model); for complex systems such as a plant, constructing a base model is impractical.

Thus, rather than seek mathematically valid principles, our approach is to formalize the intuitive criteria that human modelers use to achieve sufficiently accurate, coherent models. But formulating operational constraints that capture the two intuitive criteria for adequacy is difficult also. Ultimately, some constraints may prove to be overly restrictive, pruning intuitively adequate models. Equally likely, some scenario models may satisfy all the constraints without being intuitively adequate. Progress in automated modeling requires iteratively formulating and testing adequacy constraints. For the adequacy constraints we propose, this chapter explains why each is intuitively necessary, and Chapter 9 empirically evaluates the constraints in the domain of plant physiology.

5.4.1 Variables in a Model

A model is only adequate if it can make the desired predictions. Clearly, the behavior of a variable of interest cannot be predicted if the variable is not in the model. 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.

During analysis (e.g., simulation), a model must be able to determine which of its influences is active. This requires the ability to evaluate any relevant activity preconditions. The following constraint ensures that the model has 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.

5.4.2 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 scenario description into two parts: the system that is relevant to answering the given question, and its environment (which is irrelevant).

To ensure that a model is adequate and as simple as possible, a suitable system boundary is crucial. Yet despite the importance of this issue, previous automated modeling programs cannot adequately choose exogenous variables for prediction questions. Some programs require exogenous variables to be specified in the question or the domain knowledge, thus shifting responsibility to humans. That approach is impractical when the domain knowledge is extensive. Of the few programs that choose exogenous variables automatically, we explain at the end of this section that their criteria are too weak and can result in either inadequate or unnecessarily complex models.

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 scenario variables can be treated as exogenous, a modeler must be able to determine whether one scenario variable significantly affects another.

The influences in a scenario 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 **scenario influence graph** as follows. The nodes of the graph are the scenario 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 activity preconditions.

An **influence path** is a path of non-zero length in the scenario influence graph. One scenario variable affects another when there is an influence path leading from the first variable to the second.

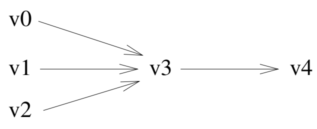
A time scale of interest permits stronger criteria for determining whether one scenario variable affects another. For any particular time scale, only some influences are valid and significant. Thus, one scenario variable **significantly influences** another on a given time scale if and only if there is an influence path leading from

```
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) Scenario influence graph

Figure 5.3: Scenario influence graphs. Part A shows a set of influences, along with their significance preconditions and activity preconditions. Part B shows the corresponding scenario influence graph.

the first variable to the second and every influence in the path is valid and significant on that time scale.

Figure 5.3 illustrates these concepts. Part A shows a set of influences. Part B shows the corresponding scenario influence graph. On a time scale of 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 scenario description, on the time scale of interest, 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 causal prediction question asks for the effects of driving conditions on variables of interest, where the driving conditions are those behavioral conditions specified in the question. 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 scenario description, on the time scale of interest, 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. If the exogenous variables in a scenario model satisfy these two constraints, the model's system boundary is adequate.

To illustrate these criteria for choosing exogenous variables, 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. As will be discussed in Chapter 8, the appropriate time scale of interest for this question is minutes. Part A of Figure 5.4 shows a portion of the elaborated scenario description for the question; 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 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.

Related Work

Despite its importance, no previous work in automated modeling has provided explicit criteria for choosing exogenous variables. Typically, work in automated modeling assumes that either the domain knowledge or the question specifies those variables that can be exogenous. For instance, the modeling algorithms of Williams [85] and Iwasaki and Levy [43] require, as input, the variables that can be exogenous for the question. Although these algorithms can determine which exogenous variables must be included in the scenario model, neither algorithm can determine exogenous

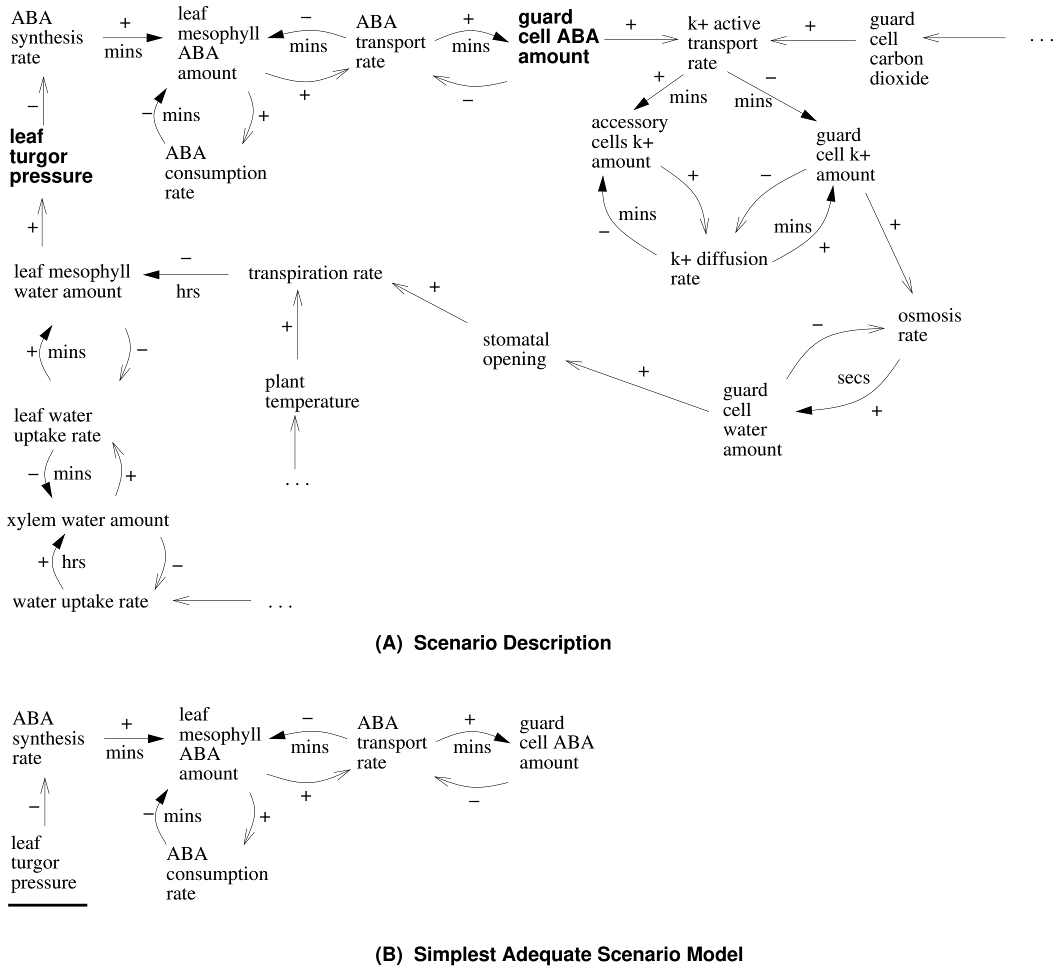


Figure 5.4: **(A)** A portion of the elaborated scenario description for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" The driving variable and variable of interest are shown in bold. Ellipses indicate connections to the remainder of the scenario variables and influences. Alternative levels of detail are not shown. **(B)** The simplest adequate scenario model for answering the question.

variables automatically. For complex systems, this approach is impractical.

The modeling algorithm of Nayak [65] can choose exogenous variables, but it does not have special criteria for doing so. His adequacy criteria are suitable for his modeling task, explaining a specified causal relation, but they are too weak for prediction questions. For instance, his criteria would allow the chosen scenario model to include an exogenous variable that, in the scenario description, is significantly influenced by another variable in the model.

The modeling algorithm of Falkenhainer and Forbus [25] largely determines the system boundary by determining relevant scenario objects. The algorithm requires, as input, a system decomposition. That is, each scenario object is assumed to be a system, and each object can have component objects that represent its subsystems. To determine the objects that are relevant to a question, the algorithm identifies the smallest set of objects that must be modeled to include the immediate influences on the variables of interest; these objects are marked as relevant. Next, to ensure that interactions among these objects are modeled, the algorithm determines a "minimal covering system," the lowest object down the system decomposition that subsumes the relevant objects. That object and its subsystems (down to the level of the initially relevant objects) are relevant. Any variable that is a property of a relevant object, but is only influenced by properties of irrelevant objects, is exogenous.

Their approach has several limitations. While their modeling algorithm requires a system decomposition for the scenario, 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.* [68] 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 [18]. 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.

5.4.3 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, amount(pool(water, guard-cells)) $\Leftarrow$ rate(osmosis(accessory-cells, guard-cells))

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

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

Figure 5.5: Influences on the amount of water in a plant's guard cells.

the modeler must choose a set of influences to represent the phenomena that govern it. The constraints in this section ensure that every dependent variable in a model has an adequate set of influences.

For analysis 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 every 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 5.5 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 are equilibrium influences; 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. In TRIPEL's scenario description language, an influence's validity preconditions (if it has any) are encoded as a time scale condition. Thus, the following constraint ensures that each influence is valid for purposes of answering a given question.

Adequacy constraint 6 (influences valid)
A scenario model is adequate only if each of its influences is valid on the time scale of interest.

For example, the two equilibrium influences in Figure 5.5 are only valid on a time scale of hours, since 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.

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 scenario description, a scenario variable, and a type of influence (i.e., functional or differential), we define a **complete set of influences** as follows:

- The set of most-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 influences that explain it (as specified by the explanation relation) is a complete set.

For example, Figure 5.6 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 most-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 time scale of interest. Therefore, the influences on each dependent variable need only represent all the *significant* phenomena that affect the variable. For a given time scale of interest, 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 on the given time scale. For example, in Figure 5.6, 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 time scale of interest.

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)))

Figure 5.6: Influences on the amount of carbon dioxide in a plant's leaves. The first two are the most-aggregate influences. The influence of photosynthesis is explained by the influence of the dark reactions (and not by any other influences).

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 section, we say that the influences are adequate. Constraints 6 (influences valid) and 7 (influences complete) ensure that the influences provide a sufficiently accurate representation of the governing phenomena, and constraints 5 (influences homogeneous) and 8 (influences not redundant) ensure that the representation is coherent.

Related Work

Most previous work in automated modeling does not enforce explicit constraints like these for the influences on a dependent variable. Typically, each alternative set of influences on a variable resides in a separate model fragment, and each such model fragment has preconditions governing its inclusion in a model. Less typically, each model fragment could include one of the influences, and the assumptions that label these model fragments, along with the compatibility constraints among these assumptions provided by the domain knowledge, are used to find compatible combinations of influences. The person encoding the domain knowledge is responsible for ensuring 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 [24, 86, 47]. This is analogous to identifying an approximately complete set of influences. However, these programs do not consider alternative levels of detail for the elements of the equation.

5.4.4 Entities in a Model

A scenario model is a model of selected entities in the scenario. The entities in a scenario model consist of the following:

- Each variable in a model is a property of an entity. Thus, the entities in a model include all the entities whose properties are represented by the model's variables.

- As discussed in Section 2.5, an equilibrium influence can be associated with an aggregate process that encapsulates the underlying pools and processes that restore equilibrium. Thus, the entities in a model include any process associated with an equilibrium influence in the model.

The entities in a model are important because they indicate the model's view of the scenario. To ensure consistent predictions and a comprehensible explanation, that view must be coherent. That is, although scenario entities can typically be described at multiple levels of detail, a modeler must avoid mixing levels. An entity in a model represents a black box whose internal details are irrelevant to the modeler's objectives. Alternatively, when the internal details are relevant, that entity should be represented in the model in terms of its component entities. For example, a human modeler might treat plant water as an aggregate or might individually model the water in the roots, stems and leaves. Similarly, a human modeler might treat photosynthesis as an aggregate process or might individually model its component reactions. The following constraint ensures that models do not mix levels of detail for the same entity.

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

Besides being coherent, the entities in a scenario model must be consistent with the desired level of detail. As discussed in Section 3.2.2, the desired level of detail is specified as black-box and glass-box entities. A black-box entity prevents the modeler from including too much detail, while a glass-box entity prevents the modeler from using too little detail. The following two constraints ensure that the model is consistent with the desired level of detail.

Adequacy constraint 10 (entities consistent with black-box entities)
A scenario model is adequate only if it does not include an entity that is encapsulated by a black-box entity of the question.

Adequacy constraint 11 (entities consistent with glass-box entities)
A scenario model is adequate only if it does not include a glass-box entity of the question or any entity that encapsulates a glass-box entity.

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 given information will be lost.[2] The following constraint ensures that the model respects the level of aggregation specified in the driving variables.

Adequacy constraint 12 (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.

For example, consider the question "How is the rate of transpiration affected when the amount of water in the leaves decreases?" For this question, the amount of water in the leaves is the driving variable. Therefore, it would be inappropriate for the model to treat plant water as an aggregate (encapsulating water in the leaves) or to include detailed pools of water within the leaves.

If the entities in a model satisfy constraints 9, 10, 11, and 12, we say that the model's level of detail is adequate.

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

Related Work

Most previous compositional modeling programs either include a variant of adequacy constraint 9 (entities coherent) or provide a representation language in which the domain knowledge can provide such a constraint for particular entities. However, the other three constraints are novel.

5.4.5 Influence Paths in a Model

A causal 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, on the time scale of interest, the variables of interest are significantly influenced 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.

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 influencing variable (along with values for other influencing variables) provides the rate of change of the influenced variable; (2) in contrast, a functional influence can cause change only if the influencing variable is changing [28]. 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 growth).

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 13 (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 on the time scale of interest.

Related Work

The requirement that a scenario model relate driving variables to variables of interest is not new. Several people have recognized its importance. However, none of these people require an adequate model to include differential influence paths. Nayak [65] requires an adequate model to provide a causal path linking a single specified driving variable to a single specified variable of interest. Amsterdam [5] 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" [85] 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. While each of these is similar to adequacy constraint 13, our particular formulation is novel. For prediction questions, the requirement of differential influence paths is crucial.

5.5 Other Related Work

Some previous automated modeling programs address tasks in which the correct behavior of the variables of interest is known [3, 84]. In these programs, a model is adequate only if its predictions match the correct behavior (within a specified tolerance). Because a prediction question does not provide the correct behavior, we do not use such an adequacy constraint. However, TRIPEL could be extended to address such questions, in which case this constraint could be added. Section 10.6.2 discusses this issue further.

Some previous automated modeling programs assume that the approximate error introduced by different approximations is known or can be estimated [21, 23, 24, 81]. In these programs, a model is adequate only if the error in its predictions is within a specified tolerance. In the application domain we explored, plant physiology, the approximate error introduced by approximations is not available, so we excluded this constraint. We expect that TRIPEL could be extended to use such knowledge, in which case this constraint could be added.

5.6 Summary

The model construction task takes a causal prediction question and a scenario description as input and returns a simplest adequate scenario model. Intuitively, a scenario model is adequate if it provides a coherent, sufficiently accurate description of the scenario at the desired level of detail. To formalize these criteria, we define a scenario model as adequate for a causal 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 level of detail is adequate (adequacy constraints 9, 10, 11, and 12).
- It relates the driving variables of the question to the variables of interest (adequacy constraint 13).

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.

Chapter 6

The Model Construction Algorithm

6.1 Introduction

Chapter 5 defined TRIPEL's model construction task, shown in Figure 6.1. The current chapter describes TRIPEL's model construction algorithm. The algorithm can be viewed as conducting a search through a space of *partial models* (models under construction); Section 6.2 describes the search space. Sections 6.3, 6.4, and 6.5 describe TRIPEL's model construction algorithm, which efficiently searches this space for the simplest adequate model. Finally, Section 6.6 proves several important properties of the algorithm; most importantly, the algorithm is guaranteed to return a simplest adequate scenario model if one exists.

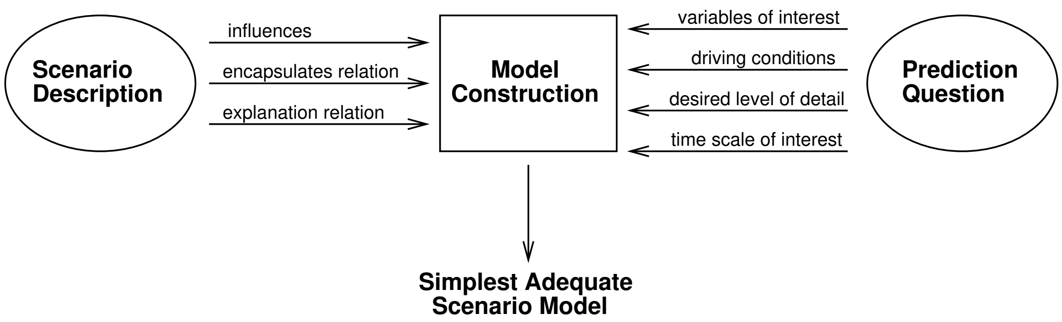


Figure 6.1: The model construction task.

6.2 The Search Space: Partial Models

A naive algorithm might find the simplest adequate model as follows:

1. Generate all possible models of the scenario.

2. Filter out the inadequate models using the adequacy constraints of Chapter 5.

3. Order the remaining adequate models by simplicity.

4. Choose one of the simplest adequate models.

However, for complex systems, there are a vast number of possible models for a scenario, so this generate-and-test algorithm is impractical. Instead, TRIPEL searches the space of *partial* models of the scenario, so it can rule out most models without ever generating or considering 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 scenario variables) partitioned into exogenous variables, dependent variables, and free variables

- a set of influences (a subset of the scenario influences), 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 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 M' is an extension of a partial model M if and only if M' can be constructed from M by making additional modeling decisions. More precisely, M' is an **extension** of M if and only if M and M' are not identical and all of the following conditions are satisfied:

- every variable in M is also in M'

- every exogenous variable in M is an exogenous variable in M'

- every dependent variable in M is a dependent variable in M'

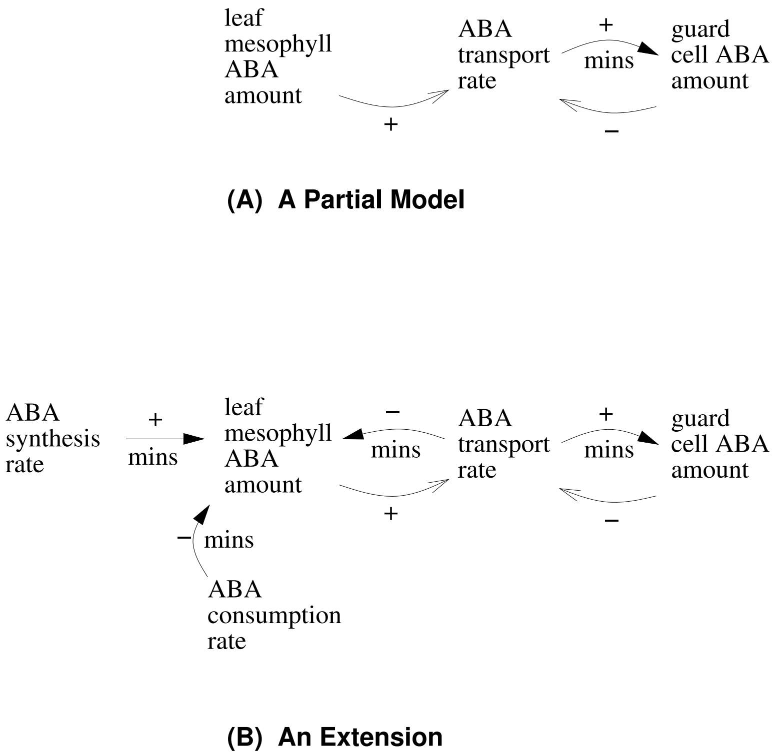


Figure 6.2: The extension relation. Part A shows a partial model in which the variable leaf mesophyll ABA amount is free. Part B shows an extension of that partial model in which the variables ABA synthesis rate and ABA consumption rate are free.

- the set of influences on the dependent variables of M are identical in M and M'

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 6.2 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 (if m1 is an extension of m2 and m2 is an extension of m3 then m1 is an extension of m3). 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 partial model cannot be extended into an adequate scenario model. The adequacy constraints in Chapter 5, although defined in terms of scenario models, can be tested in a partial model 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. The next section shows how TRIPEL exploits monotonic constraints during model construction, and Section 6.5 lists those adequacy constraints that are monotonic.

6.3 The Model Construction Algorithm: Extending Partial Models

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?" For convenience, Figure 6.3 repeats the portion of the scenario description for this question that was shown in Chapter 5. As will be discussed in Chapter 8, the appropriate time scale of interest for this question is minutes.

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

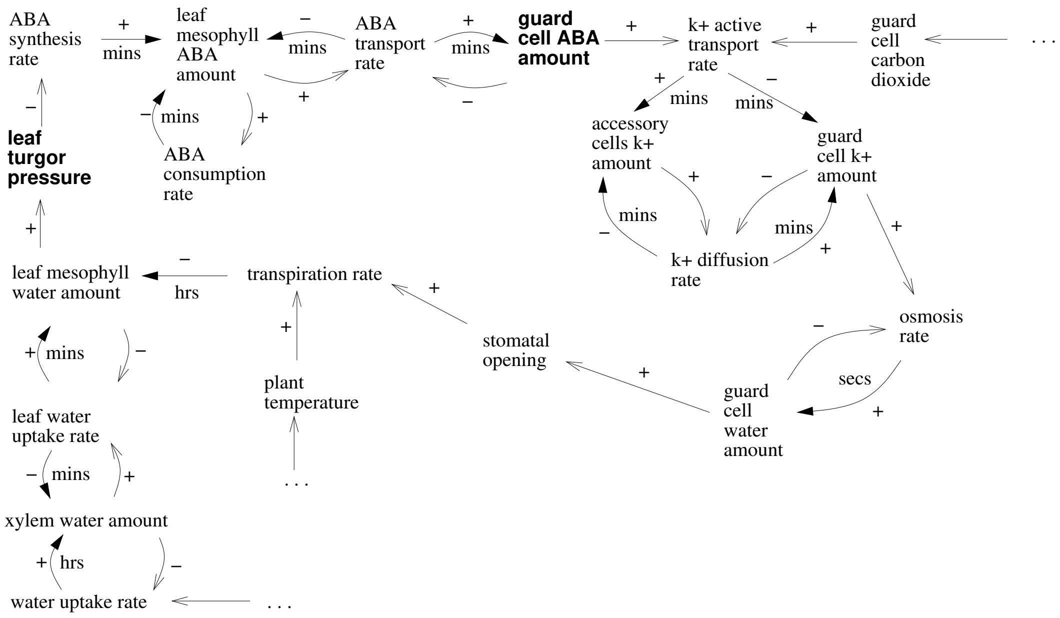


Figure 6.3: A portion of the elaborated scenario description for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" The driving variable and variable of interest are shown in bold. Ellipses indicate connections to the remainder of the scenario variables and influences. Alternative levels of detail are not shown.

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 this partial model is an adequate scenario model, it is returned as the 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 Chapter 7) 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 it 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, 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 6.4) 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).

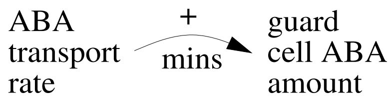


Figure 6.4: The result of applying the function Extend-model to the partial model consisting only of guard cell ABA amount. ABA transport rate is a free variable.

Figure 6.4 shows the partial model returned by Extend-model in our example.

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

The search ends when a simplest adequate scenario model is found or the search agenda becomes empty. If the agenda becomes empty before an adequate scenario model is found, no such model exists, so TRIPEL returns failure.

Figure 6.5 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 6.3).

Figure 6.6 summarizes the model construction algorithm, Find-adequate-model, as well as the successor function Extend-model.

6.4 Influences on Dependent Variables

A modeler must choose an adequate set of influences on each dependent variable in a model. In TRIPEL, this task is performed by the function Dv-models. The task arises in the function Extend-model, which was described in Section 6.3. 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 6.7, the inputs to Dv-models include a scenario description, a question, and the scenario 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 equilibrium influences or differential influences. Also, one adequate set may contain the influences that explain

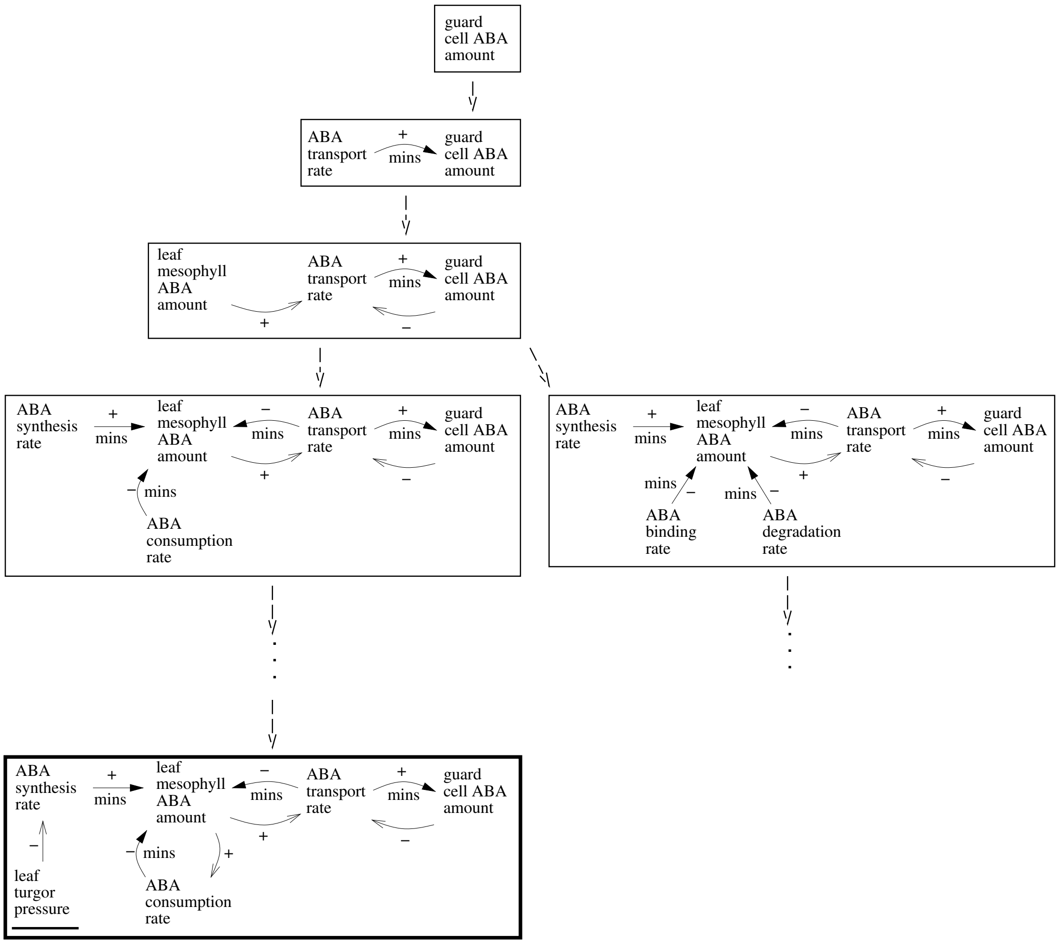


Figure 6.5: The search graph for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" Boxes indicate partial models, and dashed arrows point from a partial model to its successors. The heavy box indicates the simplest adequate scenario model (the goal node returned by the model construction algorithm).

```
Find-adequate-model (Q, S)
  agenda ← ∅
  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, Q, S)
              if m' satisfies all monotonic constraints
                then add m' to agenda
  return failure

Extend-model (m, Q, S)
  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 ← ∅
        for each m_v in Dv-models(v, Q, S)
          m' ← extend m with m_v
          add m' to models
        return models
```

Figure 6.6: TRIPEL's model construction algorithm

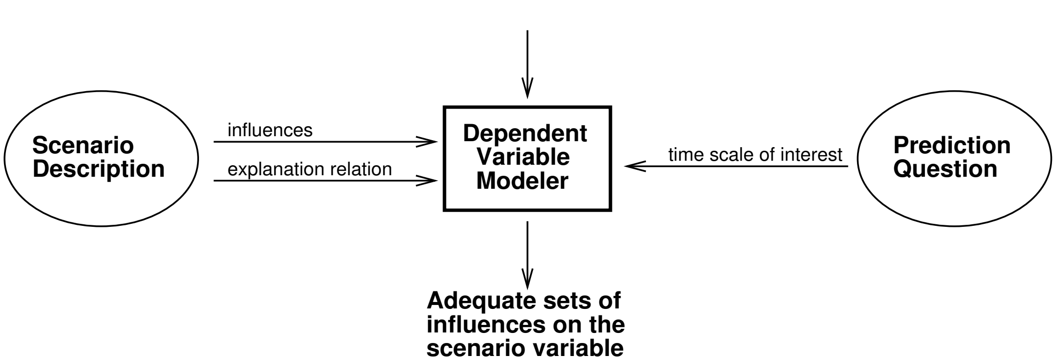


Figure 6.7: The task performed by the function Dv-models.

an influence in another adequate set. 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.

Chapter 5 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).

Adequacy constraint 5, which requires the influences on a dependent variable to have the same type (i.e., differential or functional), allows Dv-models to separately consider sets of functional influences and sets of differential influences. Dv-models separately generates the adequate sets of influences that contain only differential influences and those that contain only functional influences, and 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 scenario description, a prediction question, a dependent variable to be modeled, and the type of influences desired (i.e., functional or differential), Dv-models generates the adequate sets of influences as follows:

1. It generates every complete set of influences (of the specified type) on the dependent variable (i.e., those sets of influences that represent all the phenomena that affect the variable). Section 5.4.3 defines these as follows:
   - The set of most-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 influences that explain it (as specified by the explanation relation) is a complete set.
2. It removes from these sets any influences that are insignificant on the time scale of interest. Each resulting set is approximately complete (as defined in Section 5.4.3), so each satisfies adequacy constraint 7.

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)))

Figure 6.8: Influences on the amount of carbon dioxide in a plant's leaves. The first two are the most-aggregate influences. The influence of photosynthesis is explained by the influence of the dark reactions (and not by any other influences).

3. It discards any set that contains an influence that is invalid on the time scale of interest. Any such set of influences is inadequate because it violates adequacy constraint 6.

4. 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 6.8 (previously shown as Figure 5.6) and assume that seconds is the time scale of interest. The algorithm proceeds as follows:

1. As discussed in Section 5.4.3, there are two complete sets: (1) the first and second influences and (2) the first and third influences.

2. From the first set, remove the insignificant photosynthesis influence. From the second set, remove the insignificant dark reactions influence. 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 3.

3. The set does not include an invalid influence, so it is not discarded.

4. The set is coherent, so it is not discarded. Therefore, it is returned by Dv-models.

As another example, consider the influences shown in Figure 6.9 and assume that hours is the time scale of interest. The algorithm proceeds as follows:

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)))

Figure 6.9: Influences on the cross sectional area of a plant's stomates. The second and third influences are each explained by the first influence.

1. The algorithm generates four complete sets: the second and third influences (the most-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).

2. None of the influences is insignificant, so no set is changed.

3. 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.

4. 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 6.9, but suppose the first influence is not in the scenario description (because the domain knowledge is missing that level of detail). If the time scale of interest is less than hours (e.g., seconds or minutes), no set will survive step 3, 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.

6.5 The Role of Each Adequacy Constraint

Each adequacy constraint from Chapter 5 plays an important role in the model construction algorithm. This section describes the role of each constraint, showing that each is implemented in one of four ways:

- Some monotonic constraints serve as filters. Before a partial model is added to the agenda, these constraints are tested. If any is violated, the partial model is pruned from the search.
- Some non-monotonic constraints are used to extend partial models. These *propagation constraints*, when violated in a partial model, specify the elements that must be added for the constraint to be satisfied (analogous to constraint propagation).
- Some constraints are folded into the function Dv-models.
- Some constraints are folded into the System Boundary Selector.

Adequacy constraint 1 (include variables of interest) is used to construct the initial partial model on the agenda. The initial model includes every variable of interest (as a free variable) because any model without these variables would violate this constraint. Thus, this constraint can be viewed as a propagation constraint applied to the empty model.

Adequacy constraint 2 (include variables in activity preconditions) is used by Extend-model to identify new free variables for a partial model being extended. When an influence is added to a partial model, this constraint requires the model to include any variable in the influence's activity preconditions. If the model lacks one of these variables, the variable is added as a free variable. Since every extension of the partial model must include the influence, any extension that lacks one of these variables will violate this constraint. Thus, this constraint serves as a propagation constraint.

Adequacy constraints 3 (exogenous variables independent of model) and 4 (exogenous variables independent of question) are tested by the System Boundary Selector. They are both monotonic, as shown by the following lemma.

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 scenario description, leading to v from a driving variable of the question, consisting of influences that are each valid and significant on the time scale of interest (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 scenario description, leading to v from another variable v′ in M, consisting of influences that are each valid and significant on the time scale of interest (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. □

Although these two constraints are monotonic, they are not used as filters. Rather, they are used by the System Boundary Selector to decide whether a given free variable can be exogenous. This is more efficient than using them as filters. Chapter 7 explains how the System Boundary Selector uses them.

Adequacy constraints 5 (influences homogeneous), 6 (influences valid), 7 (influences complete), and 8 (influences not redundant) are tested by the function Dv-models. They are all monotonic, as shown by the following lemma.

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). □

Although these four constraints are monotonic, they are not used to filter partial models. Instead, as described in Section 6.4, the function Dv-models uses constraints 5 and 7 to generate potentially adequate sets of influences on a dependent variable, and it uses constraints 6 and 8 to filter these sets.

Adequacy constraints 9 (entities coherent), 10 (entities consistent with black-box entities), 11 (entities consistent with glass-box entities), and 12 (entities compatible with driving variables) are all monotonic, as shown by the following lemma.

Lemma 3 *Adequacy constraints 9, 10, 11 and 12 are monotonic constraints.*

Proof: As discussed in Section 5.4.4 (p. 57), the entities in a partial model are determined by the model's variables and equilibrium influences. Therefore, every entity in a partial model is also in each of the model's extensions, since the variables and influences 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. □

These four constraints are used to filter partial models. They are tested before a new partial model is added to the search agenda.

Adequacy constraint 13 (variables of interest differentially influenced) is monotonic when applied to models that have no free variables, as shown by the following lemma.

Lemma 4 *For a given scenario description and causal 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 *violates adequacy constraint 13, every extension of* M *also violates the constraint.*

Proof: *Assume* that E is an extension of M that satisfies constraint 13. We show by contradiction that such an extension cannot exist.

1. M violates constraint 13 (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 on the time scale of interest.

2. E satisfies adequacy constraint 13 (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 on the time scale of interest.

3. 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 1.

4. 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).

5. 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.

Constraint	Description	Role
1	include variables of interest	propagation
2	include variables in activity preconditions	propagation
3	exogenous variables independent of model	System Boundary Selector
4	exogenous variables independent of question	System Boundary Selector
5	influences homogeneous	Dv-models
6	influences valid	Dv-models
7	influences complete	Dv-models
8	influences not redundant	Dv-models
9	entities coherent	filter
10	entities consistent with black-box entities	filter
11	entities consistent with glass-box entities	filter
12	entities compatible with driving variables	filter
13	variables of interest differentially influenced	filter

Table 6.1: The role of adequacy constraints in model construction.

6. 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).

7. 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 4. That step follows from the assumption that E satisfies adequacy constraint 13. Therefore, that assumption is false.

□

This lemma allows adequacy constraint 13 to be used as a filter. Before a model with no free variables is placed on the search agenda, the constraint is tested. Every partial model to be placed on the agenda, whether the initial model or the result of Extend-model, satisfies adequacy constraints 1 and 2, so the antecedent of the lemma is satisfied. Therefore, if the model violates constraint 13, it is pruned from the search.

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.

Table 6.1 summarizes the information in this section.

6.6 Properties of the Model Construction Algorithm

6.6.1 The Model Construction Algorithm Avoids Redundancy

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 1 (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 [70], and the search graph it constructs is a tree.

6.6.2 The Model Construction Algorithm Always Terminates

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**. Since every partial model is an extension of the empty one (i.e., no variables or influences), the consideration set includes the entire search space when the agenda includes the empty model. Otherwise, it includes only a subset of the search space.

The following theorem ensures that the search will always terminate by showing that the initial consideration set is finite and that each iteration of the while loop decreases the size of the consideration set. Termination is an important property of any algorithm. It is also an important step in proving subsequent theorems.

Theorem 2 *If the scenario description is finite,* Find-adequate-model *will terminate.*

Proof:

1. Every individual step in the algorithm always terminates. Each adequacy constraint can be checked in finite time because (1) there are only a finite number of variables and influences (and hence entities) in a partial model and (2) there are only a finite number of glass-box entities, black-box entities, and driving variables in a question (because the scenario description is finite). The function Dv-models terminates in finite time because there are a finite number of influences on a scenario variable. Finally, as will be shown in Chapter 7, the System Boundary Selector simply checks a finite number of entries in a matrix to decide whether a variable can be exogenous. Thus, Find-adequate-model will terminate if its while loop terminates, which we now show.

2. The search space is finite. Given a scenario description, the search space consists of all partial models of that scenario. A finite scenario description has only a finite set of partial models. To see this, note that a partial model is uniquely determined by four sets: its exogenous variables, dependent variables, free variables, and influences. Each of the first three sets is a subset of the scenario variables, and the influences are a subset of the scenario influences. In a finite scenario description, these sets are finite, so there are only a finite set of unique partial models.

3. Since the search space is finite, the initial consideration set is finite.

4. Every iteration of the while loop prunes the simplest partial model on the agenda from the consideration set. That is, the simplest partial model on the agenda is replaced by, if anything, its successors. A partial model cannot be an extension of any of its successors, and it cannot be an extension of any other partial model on the agenda (from Theorem 1), so it is no longer in the consideration set.

5. The consideration set never increases in size. Every extension of a partial model's successors is also an extension of the partial model.

6. Therefore, since the consideration set is initially finite, and every iteration of the while loop decreases the size of the consideration set, the while loop must eventually terminate.

□

6.6.3 The Model Construction Algorithm is Admissible

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 [70]. 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 the space of all partial models of the scenario (including all 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 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. The following seven subsections list the seven ways that Find-adequate-model prunes models, and they prove that each is justified. Finally, the lemmas from these subsections are combined to prove that Find-adequate-model is admissible.

Pruning Models Without The Variables Of Interest

Find-adequate-model does not start with the empty model on the agenda, so the initial consideration set does not contain all partial models of the scenario. Initially, the agenda contains a partial model consisting of the variables of interest, each a free variable. The following lemma ensures that every adequate scenario model is an extension of this initial model.

Lemma 5 *For a given scenario description and causal prediction question, any scenario model that does not contain the variables of interest is inadequate.*

Proof: Any scenario model that does not contain the variables of interest violates adequacy constraint 1 (include variables of interest), so it is not adequate for the question. □

Pruning Models that Violate Monotonic Constraints

If a partial model violates a monotonic constraint, it is not added to the agenda, thereby pruning it and its extensions. The following lemma ensures that a partial model that violates a monotonic constraint can be discarded.

Lemma 6 *Given a scenario description and a causal prediction question, if a partial model violates a monotonic constraint, it is inadequate for the question, and so is each of its extensions.*

Proof: 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. □

Pruning Models in which a Variable is Dependent

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. The following two lemmas justify this approach; the first lemma simply establishes one of the antecedents of the second lemma.

Lemma 7 *Every partial model that* Find-adequate-model *passes to* Extend-model *satisfies all adequacy constraints except perhaps adequacy constraint 13 (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), 10 (entities consistent with black-box entities), 11 (entities consistent with glass-box entities), and 12 (entities compatible with driving variables) are satisfied because a partial model is only added to the agenda if it satisfies these constraints. □

Lemma 8 *Let* P *be a partial model for a given scenario description. For a given causal prediction question* Q*, suppose* P *satisfies all adequacy constraints except perhaps adequacy constraint 13 (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, none of the other extensions of P is adequate.

1. E must satisfy all adequacy constraints except perhaps adequacy constraint 13 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.

2. Thus, if E is inadequate, it violates adequacy constraint 13. 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.

3. *Assume* there is an extension E' of P that is an adequate scenario model. Then E' satisfies adequacy constraint 13, 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.

4. 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 2.

5. 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).

6. 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.

7. 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.

8. Thus, the influence path implied by the assumption in step 3 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.

□

Pruning Models in which a Variable is 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. This is justified by the following lemma.

Lemma 9 *For a given scenario description and causal prediction question, if a variable* v *in a partial model* M *cannot be exogenous in* M*, (i.e., if it were, it would violate adequacy constraint 3 or adequacy constraint 4), then any extension of* M *in which* v *is exogenous is inadequate for answering the question.*

Proof: This follows directly from the fact that these two constraints are monotonic (Lemma 1). □

Pruning Models Based on 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 the function Dv-models, thereby pruning any extension with a different set of influences on v. The following two lemmas justify this approach. The first lemma ensures that every other set of influences is either inadequate or simply adds some insignificant influences. The second lemma ensures that those sets containing insignificant influences can be discarded.

Lemma 10 *For a given scenario description and causal 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 1 in the function Dv-models generates every complete set of influences, and step 2 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 if we can show that the remaining steps of the algorithm only discard inadequate sets of influences. Step 3 only discards sets that violate adequacy constraint 6, and step 4 only discards sets that violate adequacy constraint 8. Therefore, the lemma holds. □

Lemma 11 *Given a scenario description and a causal prediction question, let* M *be a partial model with a variable* v*. Suppose the influences on* v *in* M *include some that are insignificant on the time scale of interest. 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 partial model A. We show by construction that M' or one of its extensions is also adequate 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 is adequate) and it satisfies all adequacy constraints:

- Constraints 1 and 2 are satisfied because the variables in A' are the same as those in A and the influences in A' are a subset of those in A.

- Constraints 3 and 4 are satisfied because the variables in A' are the same as those in A and the exogenous variables in A' are the same as those in A.

- Constraints 5, 6 and 8 are satisfied because the influences in A' are a subset of those in A.

- Constraint 7 is satisfied because A' contains all influences from A except insignificant ones.

- Constraints 9, 10, 11, and 12 are satisfied because the variables in A' are the same as those in A and the influences in A' are a subset of those in A, so the entities in A' are a subset of the entities in A.

- Constraint 13 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 on the time scale of interest. A' includes all the variables and influences in E except some influences that are insignificant on the time scale of interest. Thus, A' must include every such influence path that A does, and hence A' must satisfy adequacy constraint 13.

□

Pruning Models that Violate Propagation Constraints

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. The following lemma ensures that the pruned extensions are all inadequate.

Lemma 12 *For a given scenario description and causal prediction question, if a partial model* M *includes an influence, any extension of* M *that does not include all the variables appearing in that influence's activity preconditions is not an adequate scenario model.*

Proof: Every extension of M will include the influence (by the definition of an extension). Thus, any extension without all the variables will violate adequacy constraint 2. □

Pruning Models When an Adequate Model is Found

Find-adequate-model returns the first adequate model it finds, effectively pruning the remainder of the consideration set. The following lemma justifies this approach.

Lemma 13 *When* Find-adequate-model *returns an adequate scenario model* M*, no other scenario model in the consideration set is a simplest adequate scenario model unless* M *is also.*

Proof: Find-adequate-model always removes the simplest partial model from the agenda, so no other model on the agenda is simpler than M. Hence, since 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, 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. □

Admissibility Theorem

The following lemma summarizes all the previous lemmas in this section. Recall that the search space initially consists of all partial models (including all scenario models) for the given scenario description.

Lemma 14 *For a given scenario description and causal prediction question,* Find-adequate-model *never prunes a scenario model from the search space 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: Follows directly from Lemmas 5, 6, 7, 8, 9, 10, 11, 12, and 13 as well as the fact that these are the only ways in which Find-adequate-model prunes models from the search space. □

Finally, building on all the previous lemmas, the following theorem proves that the model construction algorithm is admissible.

Theorem 3 (Model construction algorithm is admissible) *Given a finite scenario description and a causal prediction question for which some scenario model is adequate,* Find-adequate-model *will return a simplest adequate scenario model.*

Proof: Lemma 14 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 2 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. □

6.7 Related Work

Falkenhainer and Forbus [25] 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. The 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 can provide 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" [62]).

In their framework, most criteria for model adequacy are specified in the domain knowledge. Their model fragments are analogous to influences, and their assumption classes are similar to TRIPEL's **encapsulates** and **explanation** relations. However, we do not require the domain knowledge to provide relevance conditions or domain-specific constraints among modeling alternatives; our modeling criteria and algorithm obviate the need for that extra "modeling knowledge." Formulating the modeling knowledge so that it ensures an adequate model could be a difficult, time-consuming, error-prone task. In addition, it is not clear how to encode some constraints, such as adequacy constraint 13 (variables of interest differentially influenced), in their language. Removing the need for modeling knowledge has been a driving motivation for our work.

A second approach to model construction is to start with a detailed model and repeatedly simplify it. Williams's method for generating a "critical abstraction" [85] starts with a detailed model of the scenario and simplifies it 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 starts with a detailed model and repeatedly simplifies it by removing insignificant terms in the equations (analogous to eliminating insignificant influences). Nayak's modeling algorithm [65] starts with a detailed model and repeatedly simplifies it 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 a detailed model is impractical. First, it may be difficult to find an initial conservative (i.e., overly complex) model that is sufficiently close to the simplest adequate model. Second, the simplification operators must be fine-grained enough not to skip over the simplest adequate model, but if they are too fine-grained it will take a long time to reach the simplest adequate model.

Nayak [65] proves that his algorithm will reach the simplest adequate model in time polynomial in the size of the scenario description. However, his results do not apply to our task, for several reasons. First, as discussed in Section 5.3, 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 relies on a restrictive assumption: he assumes that every phenomenon in the scenario 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 scenario 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.[1]

TRIPEL's algorithm for model construction is very similar to the one used by Iwasaki and Levy [43]. 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 two major differences between the two algorithms. First, they have no method for automatically choosing exogenous variables. (Chapter 7 describes our method.) Second, like Nayak [65], their simplicity criteria leave many

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

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. In addition to these primary differences, there are other smaller differences:

- They allow the activity preconditions of a model fragment to include predicates as well as behavioral conditions. Correspondingly, 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 scenario description do not have a causal direction. The direction of causality is only assigned after the model is complete, using a causal ordering algorithm [44]. This causes 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 can be given a causal direction before the scenario model is built is an open question [29]. However, our approach has worked well in the plant physiology domain, and we expect similar success in many other domains. Section 10.2.2 discusses this issue further.

- Their algorithm relies on a strong assumption about the domain knowledge (the "library coherence assumption") to guarantee that the equations in an adequate scenario model are complete (i.e., have the same number of equations as dependent variables). In contrast, our modeling algorithm is designed to ensure that.

- Their algorithm is guaranteed to run in time polynomial in the size of the scenario description [58]. However, that result does not apply to our task since it relies on the same assumptions as the similar result of Nayak [65] discussed earlier.

The MECHO program for solving physics problems [61] is also similar to TRIPEL's model constructor. It starts with the unknowns of the problem (analogous to the variables of interest) and searches backward through alternative equations until it finds an adequate model. A model is adequate if it can predict the value of the unknowns from the given information. Beyond these similarities, there are important differences between MECHO and TRIPEL. First, there is no issue of coherence for MECHO to address; alternative equations represent different physics principles, not alternative levels of detail. Second, MECHO does not distinguish between significant and insignificant phenomena, and it does not consider the validity of approximations. Finally, for the textbook physics problems MECHO solves, a much simpler criterion for choosing exogenous variables is sufficient; a variable can be exogenous if and only if its value is given.

Several people have explored an approach to model construction called "discrepancy-driven refinement" [3, 5, 84]. 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; TRIPEL provides a more sophisticated approach to constructing the initial model than these algorithms currently use. Section 10.6.2 discusses this issue further.

Chapter 7

Choosing Exogenous Variables

The exogenous variables of a scenario model constitute its system boundary. The system boundary separates those aspects of the scenario that are modeled from those that are ignored. Consequently, the system boundary must be chosen so that relevant aspects of the scenario are included in the model while irrelevant aspects lie outside the boundary.

Previous chapters introduced the system boundary selection task. Chapter 5 explained the crucial role of exogenous variables in modeling and specified the criteria for choosing them. Chapter 6 explained the role of system boundary selection in TRIPEL's model construction algorithm. The current chapter explains the design of the System Boundary Selector, which makes system boundary decisions when they are required during model construction. Because TRIPEL's criteria for choosing exogenous variables are novel (as explained in Chapter 5), the design of the System Boundary Selector (which implements those criteria) is novel as well.

7.1 The Role of System Boundary Selection

During model construction, system boundary decisions arise in the successor function Extend-model, described in Chapter 6. Given a scenario description, a causal 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

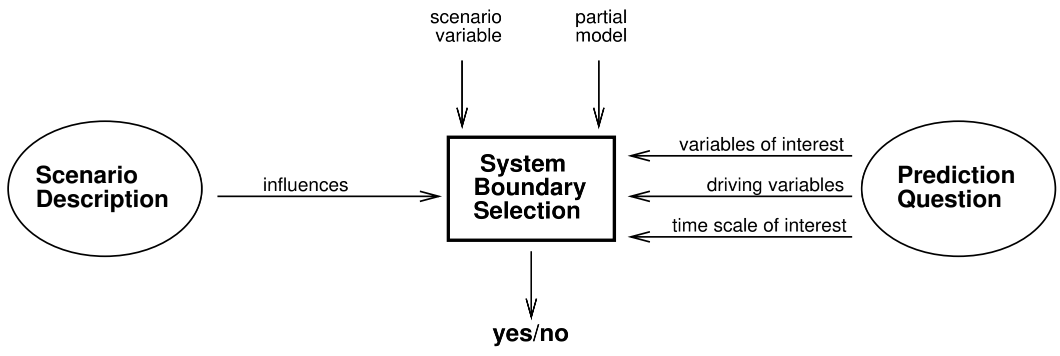


Figure 7.1: The system boundary selection task.

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 6.3 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, as shown by Lemmas 7, 8, and 9.

Figure 7.1 summarizes the system boundary selection task.

7.2 The System Boundary Selector

7.2.1 Overview

The criteria for choosing exogenous variables were specified in Chapter 5. As specified there, an exogenous variable must satisfy two constraints:

- Adequacy constraint 3 — A variable in a scenario model cannot be exogenous if it is significantly influenced in the scenario description, on the time scale of interest, 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 scenario description, on the time scale of interest, 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. Lemma 1 in Chapter 6 shows that 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 5.4.2 that one scenario variable significantly influences another on the time scale of interest if and only if there is an influence path (in the scenario description) leading from the first variable to the second and every influence in the path is valid and significant on that time scale. 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 that previous runs did. 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 7.2.2.

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 on the time scale of interest 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 in Chapter 6. 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."

7.2.2 System Boundary Analysis

Potentially Relevant Variables

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. TRIPEL defines 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 (on the time scale of interest) 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 scenario influence graph. First, each of the variables 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 on 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 will 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?" For convenience, Part A of Figure 7.2 repeats a portion of the scenario 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 remainder of the scenario variables and influences, including the feedback loop through transpiration. 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 Chapter 6).

As illustrated by the example, the search for potentially relevant variables and influences will typically have to traverse only a fraction of the variables and influences of a scenario. In natural systems, like plants, animals, and ecosystems, modularity arises from the widely disparate time scales at which processes cause change [4, 53, 68, 76, 80]. The result is a hierarchy of nearly decomposable subsystems; processes acting *within* a subsystem cause significant change quickly, while processes acting *across* subsystems cause change more slowly [4, 53, 68, 82]. 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.

Computing the Connectivity Matrix

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 scenario influence graph corresponding to the potentially relevant variables and influences. Analogous to the definition in Section 5.4.2, 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 [17].

Properties of the Connectivity Matrix

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.

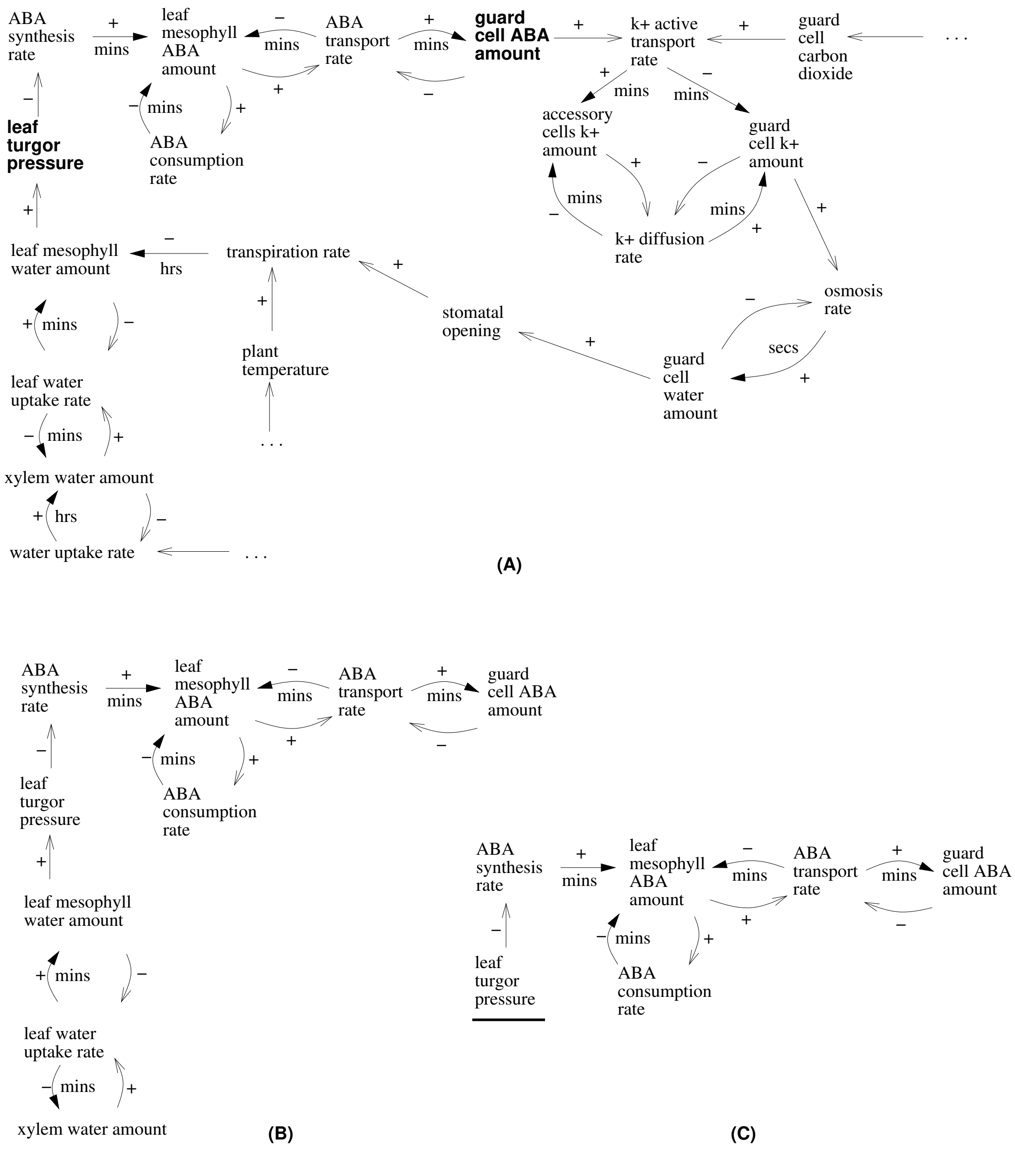


Figure 7.2: **(A)** A portion of the elaborated scenario description for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" The driving variable and variable of interest are shown in bold. Ellipses indicate connections to the remainder of the scenario variables and influences. Alternative levels of detail are not shown. **(B)** The potentially relevant variables for the question. **(C)** The simplest adequate scenario model for the question.

Therefore, the connectivity matrix must include every variable for which a system boundary decision might be required. The following theorem ensures this.

Theorem 4 (Connectivity matrix is complete) *For a given scenario description and causal prediction question, the connectivity matrix constructed by the System Boundary Selector contains every variable for which a system boundary decision might be required by* TRIPEL*'s model construction algorithm.*

Proof: A system boundary decision is only required for variables added to partial models during model construction. The connectivity matrix includes every such variable because the definition of potentially relevant variables and influences mirrors the steps that add variables and influences during model construction:

- The initial partial model for model construction only contains the variables of interest.
- The function Dv-models only adds influences that are valid and significant on the time scale of interest.
- The function Extend-model only adds a variable if it is the influencer of a newly added influence or it appears in the activity preconditions of a newly added influence.

□

The System Boundary Selector uses the connectivity matrix to determine whether one scenario variable significantly influences another. The following theorem ensures that the connectivity matrix can provide this information.

Theorem 5 (Connectivity matrix is correct) *For a given scenario description and causal prediction question, cell* (i,j) *of the connectivity matrix contains a 1 if and only if the ith scenario variable significantly influences the jth variable on the time scale of interest.*

Proof: 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. □

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 (as discussed in Section 7.2.1). 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.

7.3 Summary

In summary, TRIPEL chooses exogenous variables by using a graph connectivity algorithm to test adequacy constraints 3 and 4. For efficiency, TRIPEL computes and caches a Boolean connectivity matrix before it begins model construction. TRIPEL uses the matrix to determine whether a variable in a partial model is significantly influenced by another variable of the model or by a driving variable. The connectivity matrix can be computed efficiently, and it allows TRIPEL to efficiently determine which variables can be exogenous.

The algorithms in this chapter do 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 scenario 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 most important area for future work is improving TRIPEL's ability to recognize insignificant influences, which is discussed in Chapter 10.

Chapter 8

Choosing a Time Scale of Interest

8.1 Motivation

A time scale of interest provides an important focus for modeling. Processes that operate on slower time scales can be ignored, and processes that operate on faster time scales can be modeled as instantaneous (through equilibrium influences). An aggregate pool can be used in place of its subpools when they equilibrate on a faster time scale. These modeling techniques are useful, and often necessary, to achieve conceptual clarity, to enable analytic solutions to a model's equations, and even to enable practical numerical or qualitative simulation. The techniques are widely used in many fields, including economics [82], biology [34, 45, 77, 80, 88], ecology [4, 46, 68, 79] and many areas of engineering [14, 50, 78].

However, a person asking a prediction question cannot be expected to provide the time scale of interest. Typically, the person will not know which influences link the driving conditions to the variables of interest, much less the time scales on which the influences operate. The modeler must choose, as the time scale of interest, a time scale that is adequate for answering the question.

Despite the importance of a time scale of interest, no previous work has provided methods for choosing one. This chapter describes TRIPEL's criteria and algorithm for choosing a time scale of interest when none is specified in the question.

TRIPEL executes the algorithm before performing any other steps. That is, time scale selection precedes system boundary analysis and model construction. However, it is interleaved with scenario elaboration; it requests information from the scenario description just as model construction does.

8.2 Adequate Time Scale

A causal prediction question asks for the causal effect of driving conditions on variables of interest. Section 5.4.5 argued that a scenario model is adequate for answering a causal prediction question only if adequacy constraint 13 is satisfied; that is, for every variable of interest, the model must include a differential influence path leading to it from some driving variable such that every influence in the path is valid and significant on the time scale of interest. Thus, for a given time scale, an adequate model exists only if, for every variable of interest, the scenario description includes such an influence path. This suggests the following criterion for choosing a time scale on which the question can be adequately answered: *A time scale is adequate for answering a causal prediction question only if, for every variable of interest, the scenario description includes a differential influence path leading to it from some driving variable such that every influence in the path is valid and significant on that time scale.* Intuitively, this simply states that the modeler should choose a time scale on which the driving conditions of the question are capable of causing significant change in the variables of interest.

8.3 Finding an Adequate Time Scale

To find a time scale that satisfies this criterion, a modeler must search for the required influence paths. The search for influence paths during system boundary selection is kept manageable by the time scale of interest, but no such focus is available when searching for an adequate time scale of interest. The complete set of influences for a scenario could be enormous, so generating that set and searching through it for influence paths could be prohibitively expensive. Efficient time scale selection requires the ability to generate and search through only a fraction of the influences.

TRIPEL gains efficiency by starting with the fastest possible time scale and testing successively slower time scales until it finds one that is adequate. When TRIPEL tests a time scale, it can ignore all influences that are significant only on slower time scales, so each test operates on a manageable fraction of the scenario influences. By testing faster time scales before slower ones, TRIPEL performs the inexpensive tests before the more expensive ones, because the set of significant influences grows monotonically as TRIPEL considers slower time scales. TRIPEL chooses the first adequate time scale it finds as the time scale of interest.

To determine whether a candidate time scale is adequate, TRIPEL conducts a breadth-first search, starting from the driving variables, for scenario variables that are reachable via significant (on that time scale) influence paths. For each reachable variable, TRIPEL records whether it is reachable via a differential influence path or a functional one. The actual influence paths need not be recorded. The search ends when every variable of interest is reachable by a differential influence path (in which case the time scale is adequate) or when the set of variables reachable at that time scale is exhausted (in which case the time scale is not adequate).

For example, Part A of Figure 7.2 (page 96) shows some of the influences for the question "What happens to the amount of ABA in a plant's guard cells when the turgor pressure in its leaves decreases?" To find an adequate time scale, TRIPEL first tests a time scale of seconds. The figure illustrates that only the ABA synthesis rate is significantly influenced by leaf turgor pressure on this time scale. Next, TRIPEL tests a time scale of minutes. On this time scale, there is a differential influence path from leaf turgor pressure to guard cell ABA amount (along the top of the figure), so this time scale is chosen.

8.4 Practical Time Scale

If a time scale does not satisfy the criterion in Section 8.2, there is no scenario model on that time scale that is adequate for answering the given question. For if the required influence paths do not exist in the scenario description on that time scale, no scenario model will include them, and hence no scenario model will satisfy adequacy constraint 13. Thus, TRIPEL is justified in discarding any candidate time scale that does not satisfy the adequacy criterion.

However, the criterion is not sufficient to ensure that there is an adequate scenario model. While the criterion suggests that the question can be meaningfully answered on any time scale judged adequate, the domain knowledge (and thus the scenario description) may lack certain levels of detail that are required. For example, on that time scale, there may not be a scenario model that satisfies the desired level of detail.

In general, there is probably no way to guarantee that an adequate scenario model exists on a given time scale short of building one. However, trying to build an adequate scenario model is a potentially expensive process. The following simple test can often recognize a time scale that satisfies the adequacy criterion even though no adequate scenario model exists on that time scale.

Although the previous section stated that TRIPEL chooses the fastest adequate time scale as the time scale of interest, it does not. Instead, after finding an adequate time scale, TRIPEL tests that time scale using a slightly stronger version of the adequacy criterion. Specifically, while searching for differential influence paths relating the driving variables to the variables of interest, TRIPEL prunes any path that passes through a scenario variable that could not appear in an adequate scenario model. This includes variables whose entity violates adequacy constraints 10 (entities consistent with black-box entities), 11 (entities consistent with glass-box entities), or 12 (entities compatible with driving variables). It also includes variables whose entity is incompatible with the variables of interest (via constraint 9). If a variable cannot appear in an adequate scenario model, any differential influence path that passes through it cannot be included in an adequate scenario model. TRIPEL tests successively slower time scales, starting with the fastest adequate one, until one of them satisfies the test. The first time scale to satisfy the test is chosen as the time scale of interest.

Since this stronger test is ultimately used to choose the time scale of interest, it would be possible to skip the earlier, weaker test. However, by using both tests as described, TRIPEL provides valuable information. If a time scale passes the first test, it is likely that the question is meaningful on that time scale. If that time scale does not satisfy the stronger test, it is likely that the domain knowledge is simply missing some important level of detail. Thus, to provide feedback on possible inadequacies in the domain knowledge, as well as to indicate that the question is meaningful on a time scale faster than the chosen time scale of interest, TRIPEL uses both tests and warns the user whenever the tests suggest two different time scales.

8.5 Discussion

TRIPEL's method for time scale selection has several limitations. First, it assumes that a single time scale will suffice for answering a question. However, some questions are best answered by combining the results of several models, each with a different time scale [7, 42, 53, 59]. Also, the fastest adequate time scale may not be the best one for answering some questions; there may be more important connections between driving variables and variables of interest on slower time scales.

Another limitation results from using significance preconditions for both model construction and time scale selection. For model construction, an influence's significance preconditions should specify the fastest time scale on which the influence can be significant. This will cause TRIPEL to include the influence in cases where it might be insignificant, but that is better than ignoring it in cases where it might be significant. However, if the domain knowledge uses that convention for significance preconditions, TRIPEL might choose a time scale of interest that is faster than it really should be. Then, when using the fast time scale of interest during model construction, TRIPEL may prune influences that are actually significant on the appropriate time scale of interest. Thus, significance preconditions that are encoded to prevent TRIPEL from omitting significant influences can cause TRIPEL to omit significant influences! This limitation could be remedied with additional knowledge; if TRIPEL knew the fastest time scale on which an influence might be significant *and* the time scale on which the influence is typically significant, it could use the former during model construction and the latter for time scale selection.

Another limitation results from TRIPEL's use of driving variables rather than driving conditions. To illustrate the problem, consider the amount of glucose in a plant's leaves. The amount of glucose is influenced by photosynthesis (which manufactures glucose), and this influence is sometimes significant on a time scale of minutes (e.g., during a sunny day). The amount of glucose is also influenced by respiration (which burns it to release energy), and respiration is significant on a time scale of hours under most conditions. Suppose the question to be answered is "When the rate of respiration exceeds the rate of photosynthesis (e.g., at night), what happens to the amount of glucose?" The variable of interest (amount of glucose) is significantly influenced by some driving variable (rate of photosynthesis) on a time scale of minutes, so TRIPEL chooses that as the time scale of interest. However, on that time scale, respiration is insignificant, so it is not included in the model. Thus, the relevant driving condition cannot even be expressed in terms of the model.

There are two possible remedies for this limitation. The first is simply to require TRIPEL to find a relevant driving condition (rather than driving variable) for each variable of interest, where a driving condition is relevant only if each of its driving variables significantly influences the variable of interest. The other, more complicated, remedy would be for TRIPEL to be more sensitive to the details of the driving conditions. If TRIPEL knew that respiration exceeds photosynthesis only when photosynthesis is operating very slowly, it could choose a more appropriate time scale. I think the first solution could be implemented easily, but the second solution would require more research.

Despite these limitations, TRIPEL's method for selecting a time scale of interest has proven quite successful in practice. The limitations only arise in relatively rare cases. As will be shown in Chapter 9, TRIPEL typically chooses the most appropriate time scale as the time scale of interest, and it does so efficiently.

Chapter 9

Empirical Evaluation

9.1 Introduction

There are two important issues that must be empirically evaluated. The first issue concerns the quality of the models TRIPEL constructs. Chapter 6 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. To address this issue, we tested TRIPEL on plant physiology questions. Section 9.2 describes the knowledge base that provided the plant physiology knowledge, and Section 9.3 describes an assessment of TRIPEL's performance by an expert in plant physiology. In addition, to demonstrate the importance of several elements of TRIPEL, Section 9.4 shows how TRIPEL's performance degrades as these elements are weakened.

The second issue concerns TRIPEL's efficiency. For complex systems, the full scenario description and the space of possible models are very large. TRIPEL will only be practical if it can cope with such complexity. As described in Section 9.2, the plant physiology knowledge base provides an excellent test bed, because it describes many plant phenomena at many levels of detail. Section 9.6 evaluates the efficiency with which TRIPEL constructs models from this knowledge base.

9.2 The Botany Knowledge Base

This dissertation addresses the task of automatically constructing models of complex systems. Specifically, our methods were developed to handle large scenario descriptions that include many phenomena and many levels of detail. Therefore, to empirically evaluate these methods, we require domain knowledge capable of creating such extensive scenario descriptions. For this purpose, we used the Botany

Knowledge Base (BKB) [71].

The BKB is an ideal test bed for evaluating TRIPEL for three reasons. First, its knowledge is extensive. The knowledge base currently contains about 200,000 facts covering plant anatomy, physiology, and development. Its knowledge ranges over many phenomena and levels of detail. 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.

It was not difficult to implement demand-driven scenario elaboration for the BKB. The BKB includes inference methods, specifically rules and taxonomic inheritance, that can be used in a backward-chaining fashion. The types of knowledge required by TRIPEL (described in Chapters 2 and 4) can all be provided by the BKB, either directly or through inference methods. Thus, given a question regarding a plant, scenario elaboration can use the BKB to generate missing elements of the scenario description, including the influences that govern the plant.

However, while the BKB can provide the knowledge TRIPEL needs, we soon discovered a number of errors in its knowledge. The BKB has been used to support a variety of research projects, including machine learning [63, 64], explanation generation [1, 2, 57], and natural language generation [9]. However, none of these projects required the types of knowledge that TRIPEL needs, so the knowledge grew without being tested, and errors accumulated over the years. As the first consumer of these types of knowledge, TRIPEL was the first to uncover the errors.

In order to use the BKB to evaluate TRIPEL, we had to eliminate as many errors as possible. It would be difficult to evaluate TRIPEL if its performance was poor due to faulty knowledge. To uncover errors, we used scenario elaboration to generate the influences governing a prototypical plant, and we showed the results to the expert. He identified erroneous and missing influences, he fixed the BKB, and we repeated the process.

While a considerable number of problems were fixed this way, we eventually reached a point of diminishing returns. It is difficult to anticipate how different pieces of knowledge in the BKB will interact during inference. A change in the BKB would often correct the targeted inference but introduce other unwanted inferences (or eliminate desired inferences). When we reached this point of diminishing returns, we switched to a more predictable method for eliminating the remaining errors. We used the scenario elaboration methods to exhaustively elaborate the description of a prototypical plant, storing this description as facts in the BKB, and the domain expert fixed errors in this description directly. With the help of a variety of simple tools, the expert uncovered and fixed the remaining errors.

The result is a complete scenario description covering a prototypical plant and its environment (soil and atmosphere), all stored in the BKB. This scenario description includes 691 scenario 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. Additionally, the variables, influences, pools and processes cover many different levels of detail for describing a plant. Thus, this description meets the most important requirement for evaluating TRIPEL: it includes many phenomena at many levels of detail.

Although the scenario description is stored explicitly in the BKB, eliminating the problem of errors due to faulty inferences, another problem remains: retrieving knowledge from the BKB, even without inference, is currently slow. Without unbundling TRIPEL's computations from the retrieval of BKB facts, it would be difficult to assess TRIPEL's efficiency, and experiments would be unnecessarily slow. To alleviate this problem, we ran a program to extract all the elements of the scenario description and store them in simple Lisp data structures. This version of the scenario description was used to empirically evaluate TRIPEL, as described in the remaining sections.

9.3 Evaluation by a Plant Physiology Expert

9.3.1 Experiment

To test TRIPEL, we asked the domain expert to construct a set of questions that he thought the BKB could answer. To generate a large number of questions, he constructed a set of question *templates*. For example, Figure 9.1 shows a question template and a question that was generated from it. Because properties of a plant can be described at many levels of detail (e.g., ABA in the plant versus ABA in the leaves), question templates allowed the expert to generate many variations of each basic question.

From the question templates, a random number generator was used to select 16 questions. That is, random numbers were used to select question templates and choose from among the alternatives within each template. TRIPEL was tested on these questions, and the results were informally evaluated by the domain expert. The results showed that TRIPEL performed well, indicating that no changes to TRIPEL or the scenario description were needed.

Question template:

- How does an increase/decrease in *plant/leaf* ABA level affect *plant/leaf* water potential?

Question generated from it:

- How does an increase in leaf ABA level affect leaf water potential?

Figure 9.1: A question template (above) and a question generated from it (below). Items separated by "/" represent alternatives. Italicized choices indicate that the same choice must be made in all places. For instance, in this example, either "plant" or "leaf" must be chosen in both places.

However, to ensure that the results were not affected by the evaluation procedure, we devised the following procedure for evaluating TRIPEL's performance on subsequent questions:

1. The domain expert generates his answer (model and predictions) for a question before looking at TRIPEL's results. This prevents him from being influenced by TRIPEL's choices.

2. The domain expert evaluates TRIPEL's model for the question by comparing it to his own.

3. After the expert has evaluated TRIPEL's performance on all questions, he presents his assessment, and we discuss the knowledge he uses to reach his conclusions.

To formally evaluate TRIPEL, another 15 questions were chosen randomly, and the above procedure was followed. Only these 15 questions were evaluated carefully and in great detail, so they form the basis of our evaluation. However, the informal evaluation suggests that TRIPEL's performance on these 15 is representative of its performance on the entire 31. For this reason, we will sometimes present statistics for all 31 to provide a broader picture.

Of the 15 questions used in the formal evaluation, one had to be thrown out. The expert found the question odd, and he was not sure how to answer it. Therefore, he could not, with confidence, determine which elements of the scenario description were relevant to answering it. In the following sections, all formal evaluation results are based on the remaining 14 questions. Appendix A lists all 31 questions; the first 14 are those used for the formal evaluation. Appendix B shows the models TRIPEL constructed in the formal evaluation.

Instantaneous	Seconds	Minutes	Hours	Days	None
3	2	12	11	1	2

Table 9.1: The distribution of time scales chosen by TRIPEL as the time scale of interest. The top row lists the different time scales it chose, and the bottom row lists the number of times it chose each time scale as the time scale of interest. The last column indicates cases where TRIPEL could not find a practical time scale.

9.3.2 Does TRIPEL choose an appropriate time scale of interest?

Table 9.1 shows the distribution of time scales that TRIPEL chose as the time scale of interest over the entire 31 questions. A time scale of "instantaneous" is chosen when the causal effect is purely functional. For example, the question "How would an increase in carbon dioxide in the leaves affect the rate of photosynthesis?" can be answered on an instantaneous time scale; carbon dioxide is one of the reactants of photosynthesis, so the effect is essentially immediate. The last column in the table indicates cases where TRIPEL found no practical time scale for answering the question. The distribution indicates that TRIPEL made use of a variety of time scales in answering the questions.

Of the 14 questions that were formally evaluated, the expert judged TRIPEL's chosen time scale of interest appropriate in nine. In eight cases, TRIPEL's choice was the same as the expert's. In the other case, TRIPEL's choice was different than the expert's but nonetheless appropriate. TRIPEL's model for this case captured an effect that is significant and more immediate than the one captured by the expert's model. The effect modeled by the expert is less common but more dramatic when it occurs. The expert believes both effects are important and that either one is an adequate answer to the question.

Of the five cases where TRIPEL chose an inappropriate time scale, three were due to errors in the scenario description. In the first case, TRIPEL chose a time scale faster than the expert's because an equilibrium influence in the scenario description was missing a validity precondition. Thus, TRIPEL thought this influence was valid and significant instantaneously, while in actuality it represents an effect that operates on a time scale of minutes. In the second case, TRIPEL chose a time scale slower than the expert's because the effect captured by the expert's model could not be represented using influences from the scenario description. The expert's model crossed levels of detail; he used an "influence" to show that a change in the amount of sucrose in a plant's shoot system suggests that the amount in a specific subpart is likely to be changing similarly. However, the scenario description does not contain any such influence, and in fact TRIPEL could not find any adequate model for this question precisely because the question requires this ability to cross levels of detail. Similarly, in the third case, TRIPEL found no practical time scale because the expert's model crossed levels of detail using an influence that is not in the scenario description.

In the remaining two cases, TRIPEL chose an inappropriate time scale because it erroneously judged an insignificant influence path as significant. In the first case, TRIPEL actually chose the same time scale, based on the same effect, as the expert. However, while the expert acknowledged that this was the most reasonable connection between the driving variable and variable of interest, he said that the connection is probably insignificant. Therefore, the proper answer was that there is no adequate time scale and no adequate model. In the second case, TRIPEL found a long influence path on a time scale of interest faster than the expert's; the expert said the path was in fact insignificant.

These last two cases are the most interesting, because they indicate that TRIPEL's criterion for determining whether an influence path is significant is too simplistic. As defined in Chapter 5, an influence path is significant on a given time scale if each of its influences is valid and significant on that time scale. 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. As we will see, this limitation of TRIPEL also causes problems during model construction.

9.3.3 Does TRIPEL construct adequate models?

Of the 14 questions that were formally evaluated, the expert judged TRIPEL's chosen model adequate in ten. That is, in the expert's judgement, these ten models include all the variables and influences needed to generate the right predictions and explanations. For example, Figure 9.2 shows the adequate 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?" In nine of the ten, the models included all the elements of the expert's model. TRIPEL's model for the remaining one captures an effect that is significant and more immediate than the one captured by the expert's model, as discussed in Section 9.3.2. It may seem strange that TRIPEL constructed an adequate model in one case where it chose an inappropriate time scale. However, this was the case where a relevant influence was missing its validity preconditions, as discussed in Section 9.3.2. TRIPEL constructed the right model, but it erroneously thought the model operates on a faster time scale than it does.

For two of the remaining four questions, TRIPEL could not find any adequate

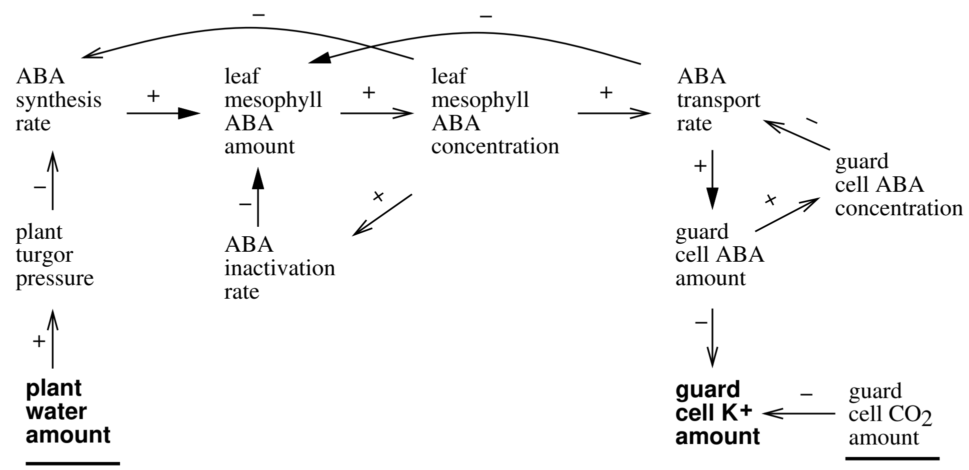


Figure 9.2: The model TRIPEL constructed to answer the question "How does a decreasing amount of water in a plant affect the amount of K^+ in its guard cells?" (The figures in this chapter use the conventions introduced in Chapter 5.)

model. In both these cases, the expert's model crosses levels of detail using an influence not in the scenario description, as discussed in Section 9.3.2.

The remaining two failures, which are the most interesting, were already analyzed in Section 9.3.2. Specifically, they are the two cases in which TRIPEL found a connection between the driving variable and variable of interest that is in fact insignificant. As discussed in that section, the problem is that TRIPEL's criterion for determining whether an influence path is significant is too simplistic.

9.3.4 Why does TRIPEL sometimes fail to find an adequate model?

TRIPEL determined that there is no adequate model for eight of the 31 questions. (For two questions, it could not even find a practical time scale of interest.) In each case, the problem lies with the scenario description. Conceptually, each such question requires a model of several aspects of the plant, and the available levels of detail for these aspects are not compatible. For example, the scenario description cannot provide an adequate model for the question "What is the effect of an increasing amount of CO_2 in the symplast of a plant's leaves on the rate of photosynthesis in the leaves?" The problem is that the scenario description does not relate properties of the leaves' symplast to properties of the leaves as a whole. In each such case, a close variant of the question can be successfully answered. For example, the scenario description can provide an adequate model for the question "What is the effect of an increasing amount of CO_2 in the symplast of a plant's leaves on the rate of photosynthesis there?" In this question, both the driving variable and variable of interest concern the symplast. There were more questions without adequate models in the first set of questions (six) than the second set (two) because generation of the first set erroneously ignored coherence constraints in the question templates (e.g., the italicized elements in Figure 9.1).

9.3.5 Do the models TRIPEL constructs include irrelevant elements?

Across the 23 questions for which TRIPEL found an adequate model, the size of the simplest adequate model it found varied considerably. Table 9.2 shows the size of the simplest adequate model it found for each of these questions. The table shows that, on average, there are about 2 influences on each variable in these models. Almost all variables in these models represent rates of processes and amounts and concentrations of pools; the table shows how many pools and processes are represented in each model.

For the ten (out of 14) questions for which the expert judged TRIPEL's model adequate, Table 9.3 compares the size of the model with its number of irrelevant elements (according to the expert). These numbers are somewhat misleading, though; one error in TRIPEL's judgement typically forces it to include many irrelevant elements. For this reason, the types of errors it made are more interesting. Most of the irrelevant elements in these models (as well as the models that were only informally evaluated) were included because TRIPEL erroneously thought an influence or influence path was significant. 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 scenario description. For each time scale in the scenario description (e.g., minutes or hours), the expert considers a variety of more specific time scales (e.g., a few minutes versus many minutes). When the expert chooses "few minutes" as the time scale of interest, he ignores processes operating on a time scale of many minutes. Because the scenario description does not distinguish these two time scales, TRIPEL treats the slower processes as significant.

- As discussed in Section 9.3.2, TRIPEL's criterion for determining whether an influence path is significant is too simplistic. Therefore, TRIPEL sometimes includes a feedback loop that the expert can tell is insignificant.

- TRIPEL does not consider behavioral conditions when assessing an influence's significance, because it attempts to build models that will cover any behavioral conditions that might arise. In contrast, the expert sometimes determines

#	TSOI	Variables	Influences	Pools	Processes
16	instantaneous	3	2	2	0
1	instantaneous	6	5	4	1
2	hours	6	7	3	2
3	instantaneous	7	6	5	1
17	minutes	7	7	3	2
4	minutes	8	10	3	2
18	seconds	10	14	2	2
5	minutes	11	14	5	3
19	seconds	12	16	3	2
6	hours	16	25	6	6
20	hours	16	25	6	6
21	hours	18	27	6	6
7	minutes	19	28	7	7
8	minutes	25	41	9	9
22	hours	26	41	8	10
23	hours	37	66	14	14
9	minutes	41	70	14	15
24	hours	59	111	18	25
25	hours	74	131	23	33
10	minutes	78	145	23	32
11	minutes	82	147	23	31
15	minutes	82	148	23	31
12	days	93	173	27	39

Table 9.2: The size of the simplest adequate model TRIPEL found for those questions where it found an adequate model. Each row represents a question. The first column shows the question number. The second column shows the time scale of interest (TSOI) that TRIPEL chose. The remaining columns show the number of variables, influences, pools and processes in the simplest adequate model TRIPEL found. The rows are ordered by the size of the models.

#	TRIPEL's Model	Irrelevant Elements
1	6, 5, 4, 1	none
2	6, 7, 3, 2	none
3	7, 6, 5, 1	none
4	8, 10, 3, 2	none
5	11, 14, 5, 3	none
6	16, 25, 6, 6	5, 8, 2, 1
7	19, 28, 7, 7	6, 9, 2, 2
8	25, 41, 9, 9	none
9	41, 70, 14, 15	29, 55, 10, 10
11	82, 147, 23, 31	64, 121, 17, 28

Table 9.3: The number of irrelevant elements in the models TRIPEL constructed. Each row represents a question. The first column shows the question number. The second column shows the number of variables, influences, pools and processes in the simplest adequate model found by TRIPEL. The third column shows the number of these variables, influences, pools and processes that are not relevant to answering the question. The rows are ordered by the size of the models.

> that an influence is insignificant because it is significant only under behavioral 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.

In summary, most of TRIPEL's models do not include irrelevant elements, and most of the irrelevant elements that it includes could be eliminated by extensions to its criteria for identifying significant influences and influence paths.

9.4 Ablation Experiments

The preceding sections suggest that TRIPEL is relatively successful at building simple, adequate models from a large scenario description. However, while those sections point out specific weaknesses, they do not identify the keys to TRIPEL's successes. This section quantifies the importance of two key elements of TRIPEL: its system boundary criteria and its use of a time scale of interest.

9.4.1 Weakening the System Boundary Criteria

TRIPEL's criteria for choosing exogenous variables are an important source of its power. A model's exogenous variables constitute its system boundary, and everything outside the boundary is deemed irrelevant. Suitable criteria for choosing exogenous variables help a modeler include relevant phenomena while omitting irrelevant ones. In contrast, if the modeler's criteria are flawed, the model may include irrelevant phenomena or, worse yet, exclude relevant phenomena. Chapter 5 proposed the following two criteria for choosing exogenous variables, which are used by TRIPEL (adequacy constraints 3 and 4):

- A scenario model is adequate only if none of its exogenous variables is significantly influenced in the scenario description, on the time scale of interest, by another variable in the model.
- A scenario model is adequate only if none of its exogenous variables is significantly influenced in the scenario description, on the time scale of interest, by a driving variable (other than itself if it is a driving variable).

To evaluate the validity of these criteria, as well as determine their role in TRIPEL's success, we replaced the criteria with the following alternative: *A scenario model is adequate only if none of its exogenous variables is significantly influenced in the scenario description, on the time scale of interest, by any other variable in the scenario description (regardless of whether that other variable is in the model or is a driving variable).* Intuitively, a variable can be exogenous under this criterion only if it is regulated on a time scale slower than the time scale of interest. For example, on a time scale of minutes, growth processes are insignificant, so the size of a plant can be exogenous.

This criterion provides a good comparison for several reasons. First, it is intuitively reasonable. Second, it is more conservative than TRIPEL's criteria; any variable that can be exogenous under this criterion can also be exogenous under TRIPEL's criteria. Finally, this criterion is simpler conceptually as well as simpler to implement.

We modified TRIPEL to use this conservative criterion for choosing exogenous variables, and we tested the resulting program on all 31 questions. The conservative program found an adequate model for exactly those questions that TRIPEL did. For those questions, Table 9.4 compares the sizes of the models found by TRIPEL (shown earlier in Table 9.2) with the sizes of the models found by the conservative program.

This table illustrates two points. First, the conservative criterion typically results in a significantly larger model. Second, the model found using the conservative criterion typically includes all the variables and influences in TRIPEL's model.

#	TSOI	TRIPEL	Conservative	Variables/Influences in Common
16	instantaneous	3/2	3/2	3/2
1	instantaneous	6/5	11/11	6/5
2	hours	6/7	13/15	6/7
3	instantaneous	7/6	10/9	7/6
17	minutes	7/7	8/8	7/7
4	minutes	8/10	9/11	8/10
18	seconds	10/14	14/18	10/14
5	minutes	11/14	55/84	10/13
19	seconds	12/16	14/18	12/16
6	hours	16/25	35/50	16/25
20	hours	16/25	35/50	16/25
21	hours	18/27	35/50	18/27
7	minutes	19/28	34/50	19/28
8	minutes	25/41	43/67	25/41
22	hours	26/41	39/59	26/39
23	hours	37/66	63/103	37/66
9	minutes	41/70	67/109	41/70
24	hours	59/111	85/148	59/111
25	hours	74/131	98/168	74/131
10	minutes	78/145	120/207	78/145
11	minutes	82/147	105/183	81/140
15	minutes	82/148	105/183	81/144
12	days	93/173	106/190	92/172

Table 9.4: The sizes of the models constructed by TRIPEL and by a variant that uses a more conservative criterion for choosing exogenous variables. Each row represents a question. The first column shows the question number. The second column shows the time scale of interest that TRIPEL chose for the question. The third column shows the number of variables and influences in the simplest adequate model that TRIPEL found. The fourth column shows the number of variables and influences in the simplest adequate model found by the conservative variant. The last column shows the number of variables and influences that the two models have in common. The rows are ordered by the size of the models.

(The conservative model sometimes excludes a few variables and influences that are in TRIPEL's model because the conservative model includes alternatives in their place.)

For every question evaluated by the domain expert, the extra elements in the conservative models are all irrelevant (according to the expert). This suggests that TRIPEL's criteria for choosing exogenous variables are effective in retaining relevant phenomena while excluding irrelevant phenomena. The difference in size between TRIPEL's models and the conservative models shows the extra power TRIPEL's criteria provide for achieving simple models.

9.4.2 The Importance of a Time Scale of Interest

As discussed throughout this dissertation, a time scale of interest is an important source of TRIPEL's power. A time scale of interest allows TRIPEL to treat influences that operate on a slower time scale as insignificant. It allows TRIPEL to model the effects of faster processes using equilibrium influences, based on a quasi-static approximation. It allows TRIPEL to treat separate pools as a single aggregate when they equilibrate on a faster time scale. If TRIPEL did not use a time scale of interest, the simplest adequate model for a question would be much more complex than it currently is.

To test this claim, we ran TRIPEL on each of the 31 questions without using a time scale of interest. Without a time scale of interest, TRIPEL cannot recognize insignificant influences, and it cannot use influences whose validity depends on the time scale of interest. For those questions where TRIPEL originally found no adequate model, the modified version of TRIPEL also found no adequate model (because these questions expose gaps in the scenario description). For those questions on which TRIPEL did originally find an adequate model, Table 9.5 compares the size of that model against the size of the model constructed when TRIPEL does not use a time scale of interest.

This table illustrates two points. First, when TRIPEL does not exploit a time scale of interest, it constructs a model that is significantly larger, and the effect is even more dramatic than it was using the conservative criterion for choosing exogenous variables. Second, TRIPEL often cannot construct an adequate model even though it could when using a time scale of interest. Without using a time scale of interest, TRIPEL is forced to model more phenomena, and it is more likely to need two phenomena for which the scenario description does not include 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 scenario description.

For every question evaluated by the domain expert, the models built without

#	TSOI	TRIPEL	No TSOI
16	instantaneous	3/2	97/181
1	instantaneous	6/5	116/218
2	hours	6/7	13/18
3	instantaneous	7/6	116/218
17	minutes	7/7	64/114
4	minutes	8/10	116/218
18	seconds	10/14	na
5	minutes	11/14	na
19	seconds	12/16	na
6	hours	16/25	64/114
20	hours	16/25	64/114
21	hours	18/27	64/114
7	minutes	19/28	99/185
8	minutes	25/41	65/115
22	hours	26/41	116/218
23	hours	37/66	64/114
9	minutes	41/70	na
24	hours	59/111	na
25	hours	74/131	116/218
10	minutes	78/145	139/269
11	minutes	82/147	168/307
15	minutes	82/148	168/306
12	days	93/173	97/181

Table 9.5: The sizes of the models constructed by TRIPEL and by a variant that does not exploit a time scale of interest. Each row represents a question. The first column shows the question number. The second column shows the time scale of interest that TRIPEL chose for the question. The third column shows the number of variables and influences in the simplest adequate model that TRIPEL found. The fourth column shows the number of variables and influences in the simplest adequate model found by the variant; "na" means that no adequate model was found. The rows are ordered by the size of the models.

using a time scale of interest are strictly inferior to the models TRIPEL originally constructed. That is, in every case, they add irrelevant elements without adding any missing relevant elements (according to the expert).

9.4.3 Combining the Ablations

To see the result of combining the two variants of TRIPEL — ignoring the time scale of interest and using the conservative criterion for choosing exogenous variables — we created a third variant of TRIPEL and tested it on all 31 questions. Table 9.6 combines the results of this new variant with the results shown in Tables 9.4 and 9.5. The last two columns show the new results: the simplest and most-detailed adequate models constructed by the new variant for each question. These columns are interesting for two reasons. First, the size of the simplest models it found shows how models degrade when the previous two variants are combined. Second, the most-detailed models constructed by the new variant are in fact the most-detailed adequate models (from the scenario description) for the questions, so the last column provides a useful comparison with the size of the models TRIPEL originally constructed.[1]

9.5 Simulation Experiments

So far, the dissertation has focused entirely on modeling. However, as stated in Chapter 1, our long-term goal is a computer program that can answer prediction questions, not just a program that builds models. A program for answering prediction questions must pass the model constructed by TRIPEL to an analysis program that can generate the predictions. The issues that TRIPEL addresses are important regardless of the method of analysis. This section describes the result of integrating TRIPEL with one particular analysis program, the Qualitative Process Compiler [27].

Integrating TRIPEL with an analysis program serves two purposes. First, it provides further evaluation of the models that TRIPEL constructs. At this early stage of research in automated modeling, an expert's evaluation is crucial, because an expert is able to assess whether a model will make the right predictions *and* whether it makes them for the right reasons. However, humans are prone to errors in generating predictions from formal models, so a formal analysis program is valuable in catching such errors. Second, although TRIPEL supports many meth-

[1] Although most of the detailed models are quite large, notice that the detailed model for one question is small. That question specifically asks about a process, plant water distribution, that treats the entire plant as a conduit for water transport from the soil to the atmosphere. Thus, every adequate model for answering the question abstracts all the internal details of the plant.

#	TSOI	TRIPEL	Conservative	No TSOI	Combined (simplest)	Combined (detailed)
16	instant.	3/2	3/2	97/181	106/193	195/354
1	instant.	6/5	11/11	116/218	129/234	159/296
2	hours	6/7	13/15	13/18	14/19	14/19
3	instant.	7/6	10/9	116/218	129/234	156/290
17	minutes	7/7	8/8	64/114	71/122	72/124
4	minutes	8/10	9/11	116/218	129/234	137/247
18	seconds	10/14	14/18	na	na	na
5	minutes	11/14	55/84	na	na	na
19	seconds	12/16	14/18	na	na	na
6	hours	16/25	35/50	64/114	71/122	76/132
20	hours	16/25	35/50	64/114	71/122	76/132
21	hours	18/27	35/50	64/114	71/122	76/132
7	minutes	19/28	34/50	99/185	106/193	195/354
8	minutes	25/41	43/67	65/115	71/122	75/130
22	hours	26/41	39/59	116/218	129/234	160/298
23	hours	37/66	63/103	64/114	71/122	72/124
9	minutes	41/70	67/109	na	na	na
24	hours	59/111	85/148	na	na	na
25	hours	74/131	98/168	116/218	129/234	159/296
10	minutes	78/145	120/207	139/269	152/285	160/298
11	minutes	82/147	105/183	168/307	177/319	195/354
15	minutes	82/148	105/183	168/306	177/318	195/354
12	days	93/173	106/190	97/181	106/193	195/354

Table 9.6: A summary of the ablation experiments. Each row represents a question. The first column shows the question number. The second column shows the time scale of interest that TRIPEL chose for the question. The other columns show the sizes (number of variables and influences) of the models constructed by TRIPEL, the conservative variant, the variant that ignores the time scale of interest, and the variant that combines these two ablations (the simplest and most-detailed models it constructed). The rows are ordered by the size of the models.

ods of analysis in principle, each method may require some additional issues to be addressed. Integration of TRIPEL with an analysis program will expose these issues.

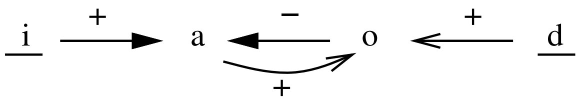
Models constructed from the BKB cannot be numerically simulated because it contains few quantitative details. It is designed to cover a wide variety of plant species; while different species are governed by similar phenomena, they differ greatly in their quantitative details. Furthermore, in its current stage of development, the BKB is primarily intended for tutoring. For that task, quantitative details are typically irrelevant. Thus, generating predictions using models constructed from the BKB requires *qualitative simulation.*

Qualitative Simulation

To perform qualitative simulations using the models TRIPEL constructs, I integrated TRIPEL with the Qualitative Process Compiler (QPC) [27], a qualitative simulation program. After TRIPEL constructs a scenario model, it passes the model to QPC. Starting from the initial state of the specified scenario, QPC identifies the active influences in the scenario model (i.e., those influences whose activity preconditions are satisfied), and it uses them to generate a set of *qualitative differential equations* (QDEs). QPC simulates the equations using the QSIM program [52, 54]. As the state of the scenario changes during simulation, the set of active influences in the scenario model may change, so QPC repeatedly formulates a set of QDEs and simulates them until simulation is complete.

Qualitative differential equations are an abstraction of ordinary differential equations. Figure 9.3 shows a simple model consisting of influences (Part A) and the corresponding QDEs (Part B). This figure illustrates two points. First, differential influences are combined in a stronger way than functional influences: each differential influence is assumed to be an additive term (although it is possible to specify otherwise). Second, the information about the function f lacks quantitative details.

The output of QPC is a qualitative behavior for the scenario. For example, Figure 9.4 shows the qualitative behavior of one variable, the amount of water in a bathtub, as the water in the tub drains out. The figure illustrates two important aspects of qualitative predictions. First, the magnitude of a variable is represented only by its relationship to fixed "landmarks"; in this example, the landmarks are the amount corresponding to a full tub and the amount corresponding to an empty tub. The dotted lines indicate the transition from one qualitative state to another, not the quantitative trajectory. Second, a variable's rate of change is represented as the sign of its first derivative: zero (steady, shown as $\circ$), negative (decreasing, shown as $\downarrow$), and positive (increasing, typically shown as $\uparrow$, but not shown in the



(A)

$$\frac{da}{dt} = i - o$$
$$o = f(d,a), \qquad \frac{\partial f}{\partial d} > 0,\ \frac{\partial f}{\partial a} > 0$$

(B)

Figure 9.3: **(A)** A set of influences representing a simple model of a bathtub. The variable a represents the amount of water in the tub, i is the rate of inflow from the faucet, o is the rate of outflow through the drain, and d is the size of the drain. The variables i and d are exogenous. **(B)** The corresponding qualitative differential equations.

example).

For some domains and tasks, qualitative predictions are appropriate. For plant physiology, different species differ significantly in their quantitative details. For this reason, plant physiology books often describe only the qualitative behavior of plants and the mechanisms responsible for that behavior. Thus, the combination of TRIPEL and QPC could provide a valuable foundation for a tutoring system in plant physiology or similar domains.

Unlike numerical simulations, QPC does not necessarily predict a unique behavior for the physical system described by a scenario model. Often, due to lack of quantitative details, there are multiple behaviors for the system that are consistent with the QDEs. This is because a QDE model is an abstraction of many different ODE models, each with different quantitative details. Thus, each qualitative behavior of a QDE model is an abstraction of the quantitative behaviors resulting from some of the underlying ODE models.

Integrating TRIPEL and QPC

Integrating TRIPEL with QPC required solutions to several problems:

- For each variable in a scenario model, TRIPEL must specify the range of the variable and any important landmarks. To address this issue, I manually added, for each variable in the elaborated scenario description, its range for

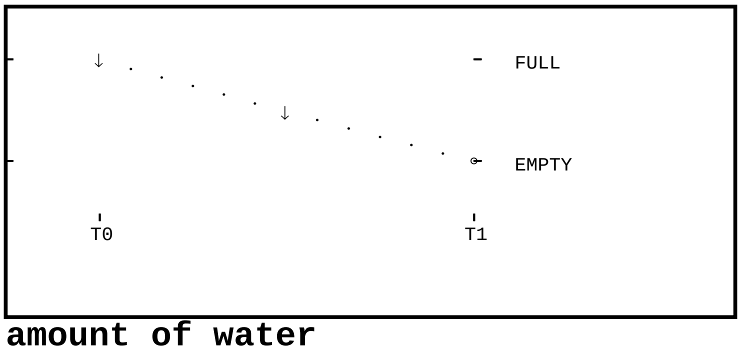


Figure 9.4: The qualitative behavior of the amount of water in a bathtub as the water drains out. This behavior includes three qualitative states: the amount is full and decreasing at the initial time instant `t0`, it is between full and empty and is decreasing in the time interval between `t0` and `t1`, and it is empty and steady at the final time instant `t2`.

a prototypical plant under typical daytime conditions. For example, amount variables are all positive (e.g., a plant without any water would not look much like a plant). When TRIPEL passes a scenario model to QPC it also passes range constraints for every variable in the model.

- TRIPEL must specify an initial state for variables whose initial state is not specified in the question and cannot be determined using the model. To address this issue, I modified QPC as follows. After QPC generates all the initial states of the scenario that are consistent with the model and question, it calls a subroutine that selects the initial state that is closest to equilibrium. That is, it selects the initial state with the greatest number of steady variables.

- TRIPEL must specify the qualitative behavior for exogenous variables whose behavior is not specified in the question. Currently, QPC assumes that exogenous variables are constant unless otherwise specified. Such an assumption is typically appropriate, so TRIPEL simply allows QPC to make such assumptions.

- Finally, qualitative simulation often produces multiple behaviors whose difference is irrelevant to answering the given question. It is typically difficult to identify the differences among multiple behaviors to determine whether those differences are relevant. To address this issue, I allowed QPC to use a tool developed by Dan Clancy [15]. This tool collapses certain differences among behaviors and represents the ambiguity as part of a single behavior (e.g., the symbol ↕ indicates that a variable may be increasing or decreasing). In our experience using QPC to answer plant physiology questions, this tool has been invaluable in collapsing irrelevant distinctions.

These solutions have worked well in practice. The domain expert typically assumes that the initial state of a plant is as close to equilibrium as possible, he typically assumes that exogenous variables are constant unless otherwise specified, and the range constraints include the range of interest for all the questions we have simulated. However, although these solutions sufficed for the simulation experiments to be described shortly, they are not intended as general solutions. Chapter 10 describes some limitations of these solutions and areas for future work.

Experiment and Results

For each of the ten models that the expert judged adequate, TRIPEL passed the model to QPC for simulation. Those five models with the fewest variables (corresponding to question 1–5) all generated a single, unique behavior. Four of these behaviors (for questions 1, 2, 3, and 5) matched the expert's predictions exactly. (The expert described all his predictions in qualitative terms.)

From the model for question 4, QPC predicted a different behavior than the expert. The question asks "What happens to turgor pressure in a plant's leaves as root water absorption decreases?" Figure 9.5 shows the model that TRIPEL constructed to answer the question. The problem with the model is that it does not include the influence of transpiration on leaf water amount. TRIPEL omitted this influence because transpiration operates on a time scale of hours, while TRIPEL chose minutes as the time scale of interest. Without transpiration, there is an inflow of water into the leaves but no outflow, so the amount of leaf water increases. In contrast, the expert assumed that the plant starts out with transpiration and water absorption balanced, so a decrease in water absorption causes leaf water to decrease. TRIPEL's model is inadequate for generating that prediction, but the expert overlooked the inadequacy because transpiration plays an indirect role in answering the question.

TRIPEL's error in this question is a result of the time scales in the scenario description. As discussed in Section 9.3.5, the expert uses a finer gradation of time scales than those in the scenario description. To the expert, root water absorption operates on a time scale of many minutes, while transpiration operates on a time scale of one to two hours. Therefore, to the expert, these two processes are closely comparable, and they cannot be separated under most conditions. Yet the scenario description tells TRIPEL that the processes operate on time scales of minutes and hours, respectively, so TRIPEL erroneously separates them.

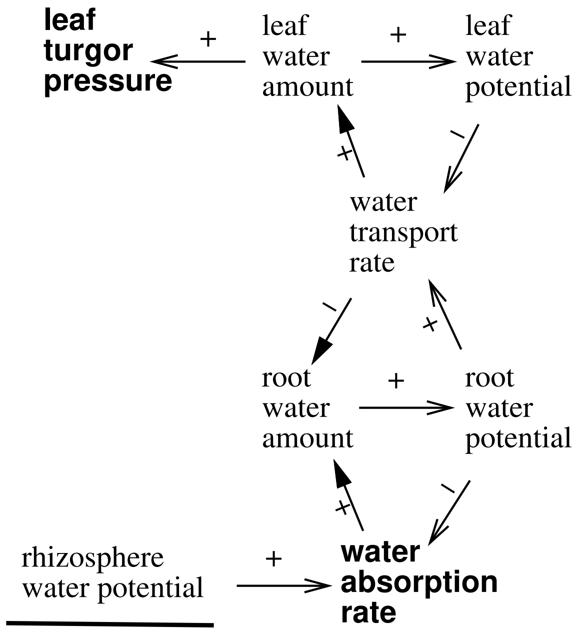


Figure 9.5: The model TRIPEL constructed to answer the question "What happens to turgor pressure in a plant's leaves as root water absorption decreases?"

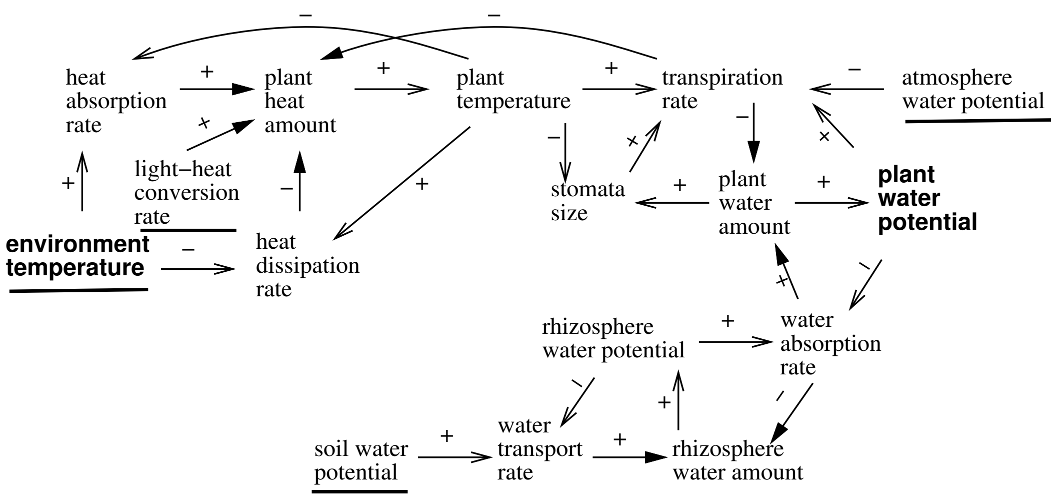


Figure 9.6: The model TRIPEL constructed to answer the question "What happens to a plant's water potential as the temperature of the environment decreases?"

While QPC generated a single, unique behavior for the five models with the fewest variables, it generated a number of possible behaviors for the next simplest model (for question 6). However, the ambiguity is due only to the influences on one variable, transpiration rate. Figure 9.6 shows the model for question 6. Given qualitative information alone, QPC cannot tell whether a change in plant temperature will cause a corresponding change in transpiration (through the immediate influence) or an opposite change (via stomata size). However, if I assert to QPC that the influence of plant temperature on transpiration dominates the influence of stomata size (i.e., that transpiration behaves as if it is only a function of plant temperature while plant temperature is changing), QPC generates a single, unique behavior for this question, the one predicted by the expert. The influence of temperature dominates the influence of stomata size because temperature is a driving force for transpiration, while stomata size merely controls the resistance. This example demonstrates three things: the influences in the model are adequate for answering the question, qualitative information alone is insufficient to generate the required prediction, and knowledge of dominant influences eliminates the ambiguity.

For the remaining four models (questions 7–10), QPC also generated a number of possible behaviors. However, unlike the model for question 6, there are many sources of ambiguity in these models. As with question 6, the problem does not lie with TRIPEL's choice of influences. The problem is that the qualitative information provided by the BKB is not sufficient to uniquely determine the behavior of the variables of interest.

The ambiguity in the predictions could be reduced in several ways. One solution would be to incorporate some quantitative information into the BKB. QPC can incorporate varying amounts of quantitative information into its qualitative simulation, and the extra information typically reduces ambiguity. Alternatively, the BKB could provide information as to which influences tend to dominate other influences. QPC could use this information to reduce ambiguity, as it did for question 6 (described earlier). Along with a colleague, I have begun exploring the second approach as an extension to QPC, but more work remains.

In summary, the simulation experiments show that those models judged adequate by the domain expert result in a unique, correct qualitative prediction in about half the cases. In one other case, simulation showed an inadequacy in a model that the expert overlooked. Finally, for the remaining models, QPC generates many possible behaviors, showing that qualitative information alone is insufficient for predicting the behavior of larger models.

9.6 Efficiency

For each of the domain expert's questions, we evaluated the efficiency with which TRIPEL constructs the simplest adequate model. We separately evaluated the three primary steps that TRIPEL performs: time scale selection, system boundary analysis, and model construction. Note that the timing data reported in this section does not include the time required for scenario elaboration, since scenario elaboration was run to completion before TRIPEL was run, as described in Section 9.2. The timing data pertains to Harlequin Lispworks 3.2 Common Lisp running on a DEC 3000/500 workstation.

9.6.1 Time Scale Selection

Of the three steps we evaluated, time scale selection is by far the most efficient. The time required for time scale selection is a small fraction of the time required for system boundary analysis and model construction. Across all the domain expert's questions, Table 9.7 shows the number of seconds TRIPEL took, in the *worst case*, to choose a particular time scale as the time scale of interest.

It takes longer to choose a slower time scale of interest for two reasons. First, the set of significant influences grows monotonically as TRIPEL considers slower time scales. Thus, there are fewer influences to search through on a time scale of seconds than there are on a time scale of hours. Second, TRIPEL starts with the fastest possible time scale and tests successively slower time scales until it finds one that is

Instantaneous	Seconds	Minutes	Hours	Days	None
.003	.01	.39	.82	.95	1.9

Table 9.7: The time required by TRIPEL to choose a time scale of interest. For each time scale, the table shows the number of seconds required for time scale selection, in the worst case, when that time scale was chosen as the time scale of interest. The last column indicates the number of seconds required during time scale selection, in the worst case, to determine that no practical time scale exists for the given question. The times do not include the time required for scenario elaboration.

adequate. For example, to choose hours as the time scale of interest, TRIPEL must test four time scales: instantaneous, seconds, minutes and hours. Thus, the time required to choose a slow time scale of interest includes the time required to test faster time scales.

9.6.2 System Boundary Analysis

Recall from Chapter 7 that 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. The Floyd-Warshall algorithm requires $\Theta(n^3)$ time, where n is the number of potentially relevant variables [17]. (Of course, this complexity analysis ignores the cost of scenario elaboration.)

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 was nearly independent of the question; it depended primarily on the time scale of interest. When the time scale of interest was instantaneous or seconds, there were one or two dozen potentially relevant variables, and system boundary analysis finished in less than one second. However, when the time scale of interest was minutes, there were always about 450 potentially relevant variables, and there were always about 650 on a time scale of hours. Since the entire scenario description includes 691 variables, these numbers represent a significant fraction.

There is a high cost to identifying this many potentially relevant variables. First, it will cause demand-driven scenario elaboration to generate a significant fraction of the complete scenario description, which could be very costly. Second, it 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 Sections 9.3.2 and 9.3.5. 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 from their global properties, such as length, spatial distance covered, and cumulative delays along the path. Thus, the same problem that causes TRIPEL to err in time scale selection and to include irrelevant elements in models causes inefficiency during system boundary analysis.

There is a simple solution to the problem for some cases. Sometimes, a wide variety of questions can be answered from the same basic scenario description. That is, the complete scenario description for each question includes the same influences; each question is distinguished simply 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; from domain knowledge of chemical engineering, the complete set of influences could be generated and used to answer a wide variety of questions. While exhaustive scenario elaboration may be expensive (e.g., for the BKB it takes about 24 hours), it may be worthwhile in such cases. Given a complete scenario description, TRIPEL can generate a complete connectivity matrix (i.e., including all scenario variables) for each possible time scale. For the scenario description generated from the BKB, this also takes about a day. 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, the long-term solution is clear. 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.

9.6.3 Model Construction

After time scale selection and system boundary analysis, TRIPEL executes its model construction algorithm, Find-adequate-model. For the expert's questions, the time required for model construction is quite reasonable, as shown by Table 9.8. This table shows the time required for model construction on 29 of the expert's questions; in the remaining two questions, TRIPEL finds no practical time scale, so it never performs model construction. For most questions, model construction takes less than one minute, often less than one second. Of the questions where it found an adequate model, the longest it took was about three minutes. The longest it took to recognize that no adequate model exists for a question was less than eight minutes.

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 scenario 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.

TRIPEL performs a best-first search for the simplest adequate model. When TRIPEL finds an adequate model, it is the simplest, so the search terminates. Thus, all the partial models that TRIPEL generates fall in one of three classes: the simplest adequate model, models that were pruned by monotonic constraints, and models left on the agenda at termination.

For each of the 29 questions that required model construction, the "Best-first Search" column in Table 9.9 shows how many partial models were pruned by monotonic constraints and how many were left on the agenda. The numbers indicate that TRIPEL only generates a manageable number of partial models, especially compared to the size of the search space. Even when there is no adequate scenario model, TRIPEL can recognize this fact by explicitly generating only a small fraction of the search space.

To determine how much of TRIPEL's efficiency is due to its best-first search strategy, we modified TRIPEL to perform an exhaustive search. Thus, model construction terminates only when the search agenda is empty. With this search strategy, all the partial models that TRIPEL generates fall in one of two classes: models pruned by monotonic constraints, and adequate models that TRIPEL finds. For each of the 29 questions, the "Exhaustive Search" column in Table 9.9 shows how many partial models were pruned by monotonic constraints and how many adequate scenario models were found. In many cases, the exhaustive search strategy caused TRIPEL to consider many more partial models than it did using a best-first search, indicating the importance of its best-first strategy.

#	TSOI	Model Size (variables)	Time (seconds)
26	hours	na	.03
27	minutes	na	4
28	minutes	na	2
29	hours	na	19
13	hours	na	202
30	minutes	na	449
16	instantaneous	3	.04
1	instantaneous	6	.01
2	hours	6	.04
17	minutes	7	.03
3	instantaneous	7	.06
4	minutes	8	.09
18	seconds	10	.03
5	minutes	11	.1
19	seconds	12	.04
6	hours	16	.2
20	hours	16	.2
21	hours	18	2
7	minutes	19	.6
8	minutes	25	.8
22	hours	26	5
23	hours	37	5
9	minutes	41	2
24	hours	59	69
25	hours	74	51
10	minutes	78	192
11	minutes	82	80
15	minutes	82	92
12	days	93	130

Table 9.8: The time required for model construction. Each row represents a question. The first column shows the question number. The second column shows the time scale of interest TRIPEL chose. The third column shows the number of variables in the model TRIPEL constructed (or "na" if TRIPEL found no adequate model). The fourth column shows the amount of time (in seconds) that TRIPEL spent during model construction (i.e., the amount of time to execute the function **Find-adequate-model**). The times do not include the time required for scenario elaboration. The rows are ordered by the size of the models.

#	TSOI	Simplest Adequate Model	Best-first Search	Exhaustive Search
26	hours	na	2, 0	2 (na)
28	minutes	na	61, 0	61 (na)
27	minutes	na	186, 0	186 (na)
29	hours	na	306, 0	306 (na)
13	hours	na	751, 0	751 (na)
30	minutes	na	1536, 0	1536 (na)
16	instantaneous	3	11, 4	13 (3)
1	instantaneous	6	1, 0	1 (1)
2	hours	6	4, 0	4 (1)
17	minutes	7	4, 0	4 (1)
3	instantaneous	7	13, 2	13 (3)
4	minutes	8	8, 3	16 (2)
18	seconds	10	1, 0	1 (1)
5	minutes	11	13, 3	71 (10)
19	seconds	12	1, 0	1 (1)
6	hours	16	11, 9	269 (37)
20	hours	16	11, 9	269 (37)
21	hours	18	25, 36	2359 (101)
7	minutes	19	14, 18	7947 (732)
8	minutes	25	10, 9	101 (9)
22	hours	26	76, 58	10624 (816)
23	hours	37	98, 20	147 (21)
9	minutes	41	45, 14	652 (120)
24	hours	59	234, 67	956 (64)
25	hours	74	188, 72	13272 (1008)
10	minutes	78	828, 196	4638 (36)
15	minutes	82	684, 41	3726 (324)
11	minutes	82	740, 121	6132 (1296)
12	days	93	476, 69	14157 (1560)

Table 9.9: The number of partial models pruned during model construction. Each row represents a question. The first column shows the question number. The second column shows the time scale of interest TRIPEL chose. The third column shows the number of variables in the simplest adequate model TRIPEL constructed. The fourth column shows how many partial models TRIPEL pruned during model construction: the first number shows how many were pruned by monotonic constraints, while the second number shows how many were still on the agenda when TRIPEL found the simplest adequate model. The last column shows the results when TRIPEL uses an exhaustive search strategy: the first number is the number of models pruned with monotonic constraints, and the second number (in parentheses) is the number of adequate models it found.

9.7 Summary

To evaluate our criteria and algorithms for automatically constructing models, we evaluated TRIPEL in the plant physiology domain. The plant physiology knowledge was provided by the Botany Knowledge Base (BKB). The BKB is an ideal test bed for TRIPEL because it is a large, multi-purpose knowledge base that was independently developed by a domain expert. It describes many phenomena at many levels of detail, so constructing simple, adequate models from it is a difficult task. Using the BKB, TRIPEL constructed models to answer plant physiology questions that were generated by the domain expert. For these questions, we evaluated the quality of the models TRIPEL constructed, the efficiency with which it constructed them, and the importance of several of TRIPEL's key components.

The evaluation supports the following conclusions:

- TRIPEL is already an effective modeling program. From a large knowledge base, it typically generates simple, adequate models. The knowledge it requires is available as fundamental plant physiology knowledge that is natural for a domain expert to encode.

- TRIPEL's criteria for selecting exogenous variables and its ability to choose and exploit a time scale of interest, both important contributions of our research, play an important role in TRIPEL's ability to construct simple yet adequate models.

- TRIPEL's algorithms for time scale selection and model construction are very efficient. Despite the enormous number of possible models, TRIPEL finds a simplest adequate model quickly by generating and considering only a small fraction of the possible models.

- Several extensions could significantly improve TRIPEL's performance. Most importantly, its criterion for determining whether an influence path is significant is too simplistic. 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. This limitation causes three problems: TRIPEL sometimes chooses a time scale of interest that is too fast, it sometimes includes irrelevant elements in models, and it identifies too many variables and influences as potentially relevant during system boundary analysis. Section 10.1.2 discusses the necessary extension.

Chapter 10

Future Work

TRIPEL provides an excellent foundation for future work. It was designed to be extensible so that progress in particular areas can be easily incorporated. This chapter discusses a variety of ways that TRIPEL could be extended. The chapter shows how TRIPEL could incorporate ideas from related research, it suggests several simple, short-term extensions, and it discusses important areas where significant research will be required.

While the subsections are largely independent and can be read in any order, they are grouped into sections to highlight the type of issue they address. Section 10.1 discusses extensions to TRIPEL's criteria for determining whether an influence or influence path is significant and whether a model is coherent. Section 10.2 discusses extensions to TRIPEL's language for describing scenarios and to the types of domain knowledge TRIPEL requires. Section 10.3 discusses future work in scenario elaboration. Section 10.4 discusses future work in using models to generate predictions. Finally, Section 10.5 briefly discusses testing TRIPEL in domains other than plant physiology, and Section 10.6 discusses using TRIPEL to answer questions other than causal prediction questions.

10.1 Modeling Criteria

10.1.1 Significant Influence

The ability to recognize insignificant influences is an important source of power for any modeler, including TRIPEL. Currently, TRIPEL uses a time scale of interest to determine whether an influence is significant. However, TRIPEL's algorithms do not depend on this particular criterion in any fundamental way. In principle, TRIPEL could be extended to include other criteria for recognizing insignificant influences.

The evaluation suggests that these 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:

- Some influences are significant only under certain behavioral conditions. 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). Similarly, according to relativity, the velocity of a body influences its mass, but that influence is insignificant if the body is moving much slower than the speed of light.

- Some influences are insignificant because they are dominated by other influences. For example, there are three influences on the amount of water in a plant's apoplast:[1] uptake from the soil, evaporation from the leaves (transpiration), and osmosis into the symplast. However, the effect of osmosis is typically overshadowed by the effects of uptake and transpiration, so it is an insignificant influence on the amount of apoplast water.

Ultimately, TRIPEL should take into account the time scale of interest, desired accuracy, behavioral conditions, 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 some formal (albeit heuristic) methods for recognizing insignificant terms (i.e., influences) in equations [49, 59]. 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 [47] has designed an automated modeling program that uses these methods. Yip's program does not address many of the issues addressed by TRIPEL, so a combination of the two programs would be worth exploring.

[1] Roughly speaking, the apoplast of a plant is its network of dead parts. In contrast, the symplast of a plant is its network of living parts. For example, cell walls are part of the apoplast, while the contents of cells are part of the symplast.

10.1.2 Significant Influence Path

As discussed in Chapter 9, TRIPEL's criterion for determining whether an influence path is significant is too simplistic. 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. This limitation causes three important problems: TRIPEL sometimes chooses a time scale of interest that is too fast, it sometimes includes irrelevant elements in models, and it identifies too many variables and influences as potentially relevant during system boundary analysis.

TRIPEL can easily be extended to use more sophisticated criteria in assessing the significance of an influence path. The graph algorithms that TRIPEL uses do not record each path from one variable to another. However, they can record the length of the shortest path from one variable to another, so TRIPEL could use that information to assess whether one variable significantly influences another. The algorithms could also record the minimum spatial distance covered by the influence paths from one variable to another. Therefore, TRIPEL could be extended to consider these factors in assessing the significance of influence paths. Determining how these factors should be used in the assessment is an important area for future work.

10.1.3 Mixing Levels of Detail

TRIPEL is careful to avoid mixing levels of detail. Adequacy constraint 9 (entities coherent) ensures that TRIPEL's models never include two entities that are related by the encapsulates relation. For example, TRIPEL would never include, in the same model, both the pool of water in a plant and the pool of water in its leaves. Our experience suggests that this constraint is very useful; by ensuring a consistent level of detail, it results in coherent, comprehensible models. A survey of the modeling literature for biology and ecology suggests that human modelers obey this constraint when constructing formal models.

However, the evaluation described in Chapter 9 suggests that human modelers sometimes mix levels of detail. The domain expert sometimes included in his models an influence that bridges two levels of detail. For example, consider the question "How would decreasing solar irradiation to a plant's leaves affect the plant's carbon dioxide absorption?" To answer this question, the domain expert constructed a model at the leaves level. This model shows how solar irradiation of the leaves affects their absorption of carbon dioxide. In addition, since the question asks about carbon dioxide absorption at the plant level, the expert added the following influence to the model: carbon dioxide absorption into the plant is a function of carbon dioxide absorption into the leaves. This influence bridges the gap between the variable of interest (at the plant level) and the rest of the model (at the leaves level).

Such mixing of levels of detail certainly does not make the model incoherent or incomprehensible. Therefore, TRIPEL should be extended to allow cases like this. However, future work is required to determine when it is acceptable to mix levels of detail. Such an extension would not have helped TRIPEL during our evaluation, because the domain knowledge does not provide influences that cross levels of detail. Therefore, to allow TRIPEL to use such influences, they must be included in the domain knowledge.

10.2 Domain Knowledge and Scenario Descriptions

10.2.1 Multiple Decompositions

TRIPEL uses the encapsulates relation to determine whether entities in a scenario model are mutually coherent. For example, the pool of water in a plant encapsulates the pool of water in its leaves, so a scenario model should include one or the other but not both. However, when an entity can be decomposed in multiple ways, the encapsulates relation may be insufficient for recognizing incoherent combinations. 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.

It would not be difficult to devise a representation for multiple decompositions of entities and influences. Some previous work addresses this issue; for instance, Zeigler [89] developed a representation that allows an entity to be decomposed in multiple ways. Given a method for recognizing that two entities or two influences in a model come from incompatible decompositions, monotonic constraints could be implemented to prune such models. As stated in Section 6.5, TRIPEL can incorporate new monotonic constraints without changes in its model construction algorithm.

10.2.2 Causality

As proposed by Forbus in his Qualitative Process Theory [28], influences in TRIPEL have a fixed causal direction (i.e., a designated influencer and influencee). Assigning causality to influences has proven easy and appropriate in the plant physiology domain, and we expect similar success in many other domains.

However, there is some debate as to when influences can be assigned a causal direction [29]. In contrast to Forbus's approach, some researchers believe that influences cannot be given a causal direction until after a model is complete [44]. Given a model with non-causal equations, these researchers use a "causal ordering" algorithm to assign a causal direction to the equations (and hence the individual influences that make up the equations) [44, 65]. Thus, modeling algorithms that follow this approach cannot exploit causality until a model is already constructed [43, 65].

TRIPEL exploits the causal direction of influences in several ways. During time scale selection, it uses the directions to find causal paths from driving variables to variables of interest. During model construction, it uses the directions to choose adequate sets of influences on free variables. The System Boundary Selector uses the causal direction of influences to determine whether a variable can be exogenous. Finally, testing adequacy constraint 13 (variables of interest differentially influenced) requires a causal direction for influences. Of all these ways that TRIPEL exploits the causal direction of influences, only the last could be done if causal directions were not specified until after a model is complete. The others exploit causality at an earlier stage.

Waiting until a model is complete to assign causal directions is overly cautious; most influences can be given a fixed causal direction before a model is constructed. Causal ordering algorithms for mixed sets of differential and algebraic equations [41, 44, 65] almost always orient differential influences the way they are causally oriented in TRIPEL. (Iwasaki and Simon [44] discuss some rare exceptions.) Therefore, the only real restriction is TRIPEL's assumption that functional influences have a fixed causal direction. However, Iwasaki [41] shows that equilibrium influences should be causally oriented based on their underlying dynamic details. If this is so, the person encoding the domain knowledge can often provide a causal direction for equilibrium influences. Thus, most influences can be given a causal direction in the domain knowledge.

Even if influences cannot be given a causal direction in the domain knowledge, I believe they can, and should, be given a causal direction during scenario elaboration. A physical understanding of the scenario should precede modeling decisions, and causality is an integral component of physical understanding.

Nevertheless, TRIPEL could be extended to allow non-causal influences. TRIPEL would have to treat each non-causal influence as if it could be causally directed either way. In general, this will cause TRIPEL to include more irrelevant elements in models, because it will have to include all variables that *might* significantly influence the variables of interest (depending on the unknown causal directions). The number of irrelevant elements in models is likely to depend on the fraction of influences without a causal direction. Once the model is complete and a causal ordering algorithm assigns causality to the influences, TRIPEL could further simplify the model.

10.2.3 Model Fragments

In compositional modeling, models are constructed from building blocks provided by the domain knowledge. The building blocks are often called "model fragments" [25]. In TRIPEL, the building blocks are influences. However, some other researchers allow a model fragment to contain multiple influences on a variable, or even a complete equation.

It would be simple to extend TRIPEL to handle such model fragments. If a model fragment provides multiple influences on a variable, this would simply constrain Dv-models. Specifically, Dv-models should only consider combinations of influences that include all or none of the influences in each model fragment. Even simpler, if a model fragment contains a complete equation (i.e., all relevant influences) for a variable, Dv-models is not needed at all.

10.2.4 Dynamic Structural Conditions

TRIPEL assumes that structural conditions are constant throughout the scenario. This assumption simplifies presentation and implementation of the key ideas in TRIPEL. However, it is not an important assumption.

Qualitative Process (QP) Theory [28] provides a reasonable alternative. In TRIPEL, a structural condition can be inferred from a structural rule whose antecedent is a conjunction of structural conditions. In contrast, QP theory allows the antecedents of structural rules to include behavioral conditions. Thus, structural conditions can change during a scenario as a consequence of changes in behavioral conditions. Let's call these structural conditions "derivable" to distinguish them from "primitive" structural conditions, for which the domain knowledge has no way of predicting change.

The primary advantage of derivable structural conditions is representational convenience and comprehensibility. Although they add no expressive power to the underlying behavioral conditions and primitive structural conditions, they provide a higher level of description that is valuable in building, maintaining, and explaining a knowledge base.

TRIPEL could be extended to handle derivable structural conditions. During scenario elaboration, TRIPEL generates the influences in a scenario by backward chaining on influence rules and structural rules. If the antecedents of structural rules can include behavioral conditions, this process would be identical except for one change: rather than backward chaining on behavioral conditions, TRIPEL would collect the behavioral conditions encountered during backward chaining and add them to the activity preconditions of the resulting influence. In effect, this process converts derivable structural conditions into their underlying primitive structural conditions (which must be established during backward chaining) and behavioral conditions (which become activity preconditions of influences).

Alternatively, TRIPEL could retain derivable structural conditions for use during explanation. To accomplish this, TRIPEL could allow the activity preconditions of influences to include derivable structural conditions, and it could include in the model those structural rules that can conclude these conditions. This is the approach taken by Iwasaki and Levy [43], and it would be a natural and useful extension to TRIPEL.

10.2.5 Inferring Time Scale of Significance

The time scale on which a differential influence is significant bundles two pieces of knowledge: the rates at which the influencing process operates, and the level of change in the influenced variable that is considered significant. Sometimes, the time scale can be encoded directly in the domain knowledge (i.e., stored in an influence rule). Other times, it may be more practical to infer the time scale from these two pieces of knowledge. The latter approach is especially useful when the level of significant change depends on the question. Iwasaki [42] has explored this approach. It does not matter to TRIPEL whether significance preconditions come from influence rules or are inferred from other information.

10.2.6 Building Large Knowledge Bases

Our evaluation showed clearly that building large knowledge bases is a difficult, error-prone task. The Botany Knowledge Base (BKB) is very large, and it was constructed over many years. This poses two problems. First, it is difficult to ensure that knowledge is represented consistently throughout. Changes in terminology and representation conventions over time cause the same things to be represented in different ways, and relationships among the alternative representations are often left out. Second, individual pieces of knowledge often interact in unexpected ways during inference even though they seem reasonable in isolation.

To address these problems, a variety of tools are needed. During our debugging of the BKB, we developed a number of simple tools to catch problems. These tools incorporated a variety of "consistency principles" that allow one piece of knowledge to suggest errors in another. For example, if a scenario variable has both differential and functional influences on it, the functional influences are probably equilibrium influences, and hence they should have validity preconditions. Such simple tools proved invaluable in identifying problems in the BKB. However, more fundamentally, the knowledge base must take a more active role in its development. It must relate each new piece of knowledge to existing knowledge, including identifying inferential consequences of new knowledge that conflict with existing knowledge. There has been some important work in this area [64], but much work remains.

The evaluation described in Chapter 9 suggests that TRIPEL itself might be a valuable tool for debugging a knowledge base. When TRIPEL cannot find an adequate model for answering a question, there are gaps in the domain knowledge. TRIPEL could be extended to pinpoint its reasons for failure and suggest the types of knowledge that are missing.

10.3 Scenario Elaboration: Elaborating Behavioral Conditions

Analysis (e.g., simulation) requires methods for elaborating the behavioral conditions specified in the question. The question may not provide initial values for all variables in the model, and it may not provide behaviors for all exogenous variables, but this information is typically required for analysis.

There are two considerations when elaborating behavioral conditions. First, some conditions are more likely than others, and humans often assume these likely conditions without stating them explicitly. Second, the question may be more meaningful under some conditions than others. That is, behavioral conditions should be chosen so that the driving variables significantly affect the variables of interest.

There may be a conflict between these two considerations; the conditions under which the question is most meaningful may be atypical. For example, consider the question "How does decreasing soil moisture affect the level of oxygen in a plant's roots?" This question is best answered assuming that the soil is initially saturated with water, because the roots are starved for oxygen under such conditions. Under more typical conditions, decreasing soil moisture would have little effect on the oxygen in the roots. Thus, just as TRIPEL chooses a time scale of interest on which the driving conditions significantly affect the variables of interest, a modeler must choose behavioral conditions using the same consideration.

10.4 Numerical Models

The work described in this dissertation should provide a foundation for building numerical ODE models as well as qualitative models that lack numerical details. The issues addressed in the dissertation 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 [30] 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 [26] has successfully used this approach for limited types of equations, and it appears feasible for other types as well. Thus, there are no apparent limitations that prevent TRIPEL from constructing numerical models, but no such application has been attempted.

10.5 Other Domains

Although TRIPEL has only been tested in the domain of plant physiology, it was designed to handle many domains within science and engineering. To test the generality of its modeling criteria and the knowledge it requires, it should be evaluated in new domains. Because its representation is particularly suitable for reasoning about pools and processes, it should be especially effective in the domains of ecology, human physiology, and chemical engineering.

10.6 Other Types of Questions

This dissertation has focused on one type of question: causal prediction questions. However, while TRIPEL was specifically designed for causal prediction questions, many of the issues TRIPEL addresses arise in other types of questions as well. In this section, we discuss how TRIPEL could be extended to handle two other types of questions: non-causal prediction questions and explanation questions.

10.6.1 Non-Causal Prediction Questions

In a causal prediction question, the person posing the question wants to know the causal effect of driving conditions on variables of interest. In contrast, consider the question "What is the rate of inflow into a bathtub if the level of water remains constant and the rate of outflow is five gallons per minute?" This question has the basic elements of a prediction question — structural conditions, behavioral conditions, and a variable of interest — but it is not a causal prediction question. The rate of outflow and the level of water do not cause the behavior of the inflow rate. Nevertheless, the given information is sufficient to make the desired prediction.

TRIPEL exploits causal prediction questions in several ways. To choose a time scale of interest, it looks for causal influence paths from driving variables to variables of interest. To construct models, it begins with a partial model consisting only of the variables of interest, and it repeatedly extends models to include variables that causally influence free variables. The System Boundary Selector searches for causal paths from driving variables to variables in a model. Finally, adequacy constraint 13 (variables of interest differentially influenced) requires causal paths from driving variables to variables of interest. Each of these steps is predicated on the question being a causal prediction question.

However, as the bathtub example illustrates, these steps must be modified to handle non-causal prediction questions. In general, it is possible to draw inferences from influences in either direction, regardless of causality. Therefore, to answer non-causal prediction questions, TRIPEL must treat the influence graph as an undirected graph, the "interaction graph." To choose a time scale of interest, TRIPEL would look for an "interaction path" (path in the interaction graph) relating the driving variables and variables of interest. When extending models, TRIPEL would add any variables that "interact" with (i.e., influence or are influenced by) free variables. The System Boundary Selector must ensure that a variable is not exogenous if it significantly interacts with (via an interaction path) a driving variable or another variable in the model. Finally, adequacy constraint 13 would only require interaction paths, rather than influence paths, relating driving variables and variables of interest.

In order to handle non-causal prediction questions, an earlier version of TRIPEL took this approach [73]. The approach showed great promise, but it was less effective than the current version of TRIPEL in two areas. First, it tended to include more irrelevant elements in models, because more variables are related by interaction paths than by causal influence paths. Second, the program sometimes built models around interaction paths that were not the most important ones. The most important area for future research on this approach is the ability to distinguish important interaction paths from unimportant ones.

10.6.2 Explanation Questions

Other than prediction questions, *explanation questions* are the most important type of question in science and engineering. An explanation question is identical to a prediction question except it specifies the behavior of the variables of interest, and the goal is to construct a model that will predict the specified behaviors from the other structural and behavioral conditions given in the question.

Explanation questions arise in many tasks. Monitoring and diagnosing the behavior of physical systems requires constructing a model that explains observations so as to recognize when faults arise and pinpoint their origin. Theory formation requires constructing a model that explains observations in order to identify the underlying causal mechanisms. In tutoring, a student might be told that a physical system behaves in a certain way (e.g., "When a plant begins to wilt, ABA builds up in its guard cells") and the student may ask why.

There has been a lot of important work on answering explanation questions [3, 5, 37, 84]. The most common technique is called "discrepancy-driven refinement." In this technique, the modeler constructs an initial model of the scenario and compares its predictions against the known behavior of the variables of interest. Discrepancies suggest particular changes to the model that will reduce or eliminate them. This process is repeated until the predictions of the model are sufficiently close to the behaviors to be explained.

The most important role of TRIPEL in answering explanation questions is in constructing the initial model. Work in discrepancy-driven refinement has concentrated on techniques for revising models to eliminate discrepancies, not on constructing the initial model. An initial model constructed by TRIPEL would be more likely to contain all and only the relevant aspects of the scenario. Furthermore, by recording the insignificant influences that were pruned, and why, TRIPEL could provide important guidance for model revision.

While the theory formation task can be viewed as answering explanation questions, it introduces one additional requirement: the domain knowledge is not sufficient to construct an adequate model. In theory formation, the objective is to extend the domain knowledge so that it can provide an explanation. However, this task also fits into the framework described in this dissertation. The new requirement can be satisfied by adding additional sophistication to the scenario elaboration module. For example, rather than simply instantiating general principles, scenario elaboration might generate influences by analogy to other, more familiar domains [22].

10.7 Summary

In summary, the methods in this dissertation provide an important foundation for further research in automated modeling. Many valuable improvements can be incorporated into TRIPEL as modular extensions. Nevertheless, experience with other types of questions and other domains is needed to determine the generality of our methods.

Chapter 11

Conclusion

This dissertation addresses the task of automatically constructing models to answer causal prediction questions. Such questions pose a hypothetical scenario and ask for the causal effect of driving conditions on variables of interest. Given a question and domain knowledge, the objective is to construct the simplest model of the scenario that is adequate for answering the question. The dissertation focuses on building models that consist of algebraic equations and ordinary differential equations (or qualitative counterparts). Such questions and models are ubiquitous in science and engineering.

Influences are the building blocks for models. Each influence represents some phenomenon in the scenario at some level of detail. Influences are appropriate building blocks for models because they allow a modeler to select relevant phenomena and choose a relevant level of detail for each. The influences in a model are combined to form its equations.

The process of scenario elaboration generates missing elements of the scenario description, including influences, from a question and domain knowledge. In addition to the influences that govern the scenario, a complete scenario description includes scenario variables, which represent properties of entities in the scenario, behavioral conditions, which represent the initial state and behavior of scenario variables, structural conditions, which represent static properties of the scenario, and attributes of influences, including activity preconditions, significance preconditions, and validity preconditions. The encapsulates and explanation relations represent relationships among different levels of detail for describing the scenario. Because the complete scenario description for a complex system may be extensive, elements are only generated as needed during model construction, using a method called demand-driven scenario elaboration.

To operationalize the notion of a simple, adequate scenario model, this dissertation proposes a novel definition of simplicity and a novel set of adequacy constraints. The constraints specify variables that must be included in an adequate model, they specify when a variable can be treated as exogenous, they define an adequate set of influences on a dependent variable, they ensure that an adequate model is coherent and that it includes an appropriate level of detail, and they ensure that the model relates driving variables to variables of interest.

The dissertation provides a model construction algorithm for efficiently constructing the simplest adequate model for a given causal prediction question. For complex systems, the space of possible models is enormous, but the algorithm searches this space efficiently by searching the space of partial models. By pruning a partial model at an early stage, the algorithm prunes a large chunk from the space of possible models. Because the model construction algorithm prunes models judiciously, it is guaranteed to return a simplest adequate model if one exists. In addition to providing the basic model construction algorithm, the dissertation also provides novel algorithms for choosing the influences on dependent variables and for choosing exogenous variables.

A time scale of interest provides an important focus for modeling. The dissertation shows how a time scale of interest allows insignificant phenomena and invalid levels of detail to be recognized. Because it is typically difficult for a person posing a question to specify a time scale of interest, and because a suitable time scale of interest is crucial for eliminating irrelevant details, the dissertation provides an algorithm for choosing a suitable time scale of interest for a causal prediction question automatically.

Many of the claims in this dissertation are empirical. To evaluate the criteria and algorithms presented in this dissertation, I implemented them in a program called TRIPEL and evaluated the program in the plant physiology domain. Using a large, multipurpose knowledge base independently developed by a domain expert, TRIPEL constructed models to answer questions that were also constructed by the expert. According to the expert, TRIPEL typically generates simple, adequate models. The use of a time scale of interest and TRIPEL's novel criteria for choosing exogenous variables play an important role in its success. The evaluation also suggests the most important area for future work: TRIPEL's criterion for determining whether an influence path is significant is too simplistic. The evaluation suggests extensions to this criterion that should allow TRIPEL to consistently construct simple, adequate models efficiently.

There are many important areas for future work. Automated modeling programs will require more sophisticated criteria for recognizing insignificant influences and influence paths. Representing domain knowledge is a difficult, error-prone task;

developing knowledge bases with extensive coverage in areas of science and engineering will require new tools and techniques. Finally, this dissertation only focuses on one type of question, causal prediction questions, and it only describes an empirical evaluation in one domain, plant physiology; further progress in automated modeling will require a similar study of other types of questions and other domains.

Appendix A

Plant Physiology Questions

This appendix lists all the plant physiology questions, constructed by the expert, on which TRIPEL was tested (as described in Chapter 9). The first 14 formed the basis of the formal evaluation.

1. How would an increasing amount of CO_2 in a plant's leaves affect the rate of photosynthesis in the leaves?

2. How does increasing soil water potential affect a plant's water distribution rate?

3. How does an increasing level of ABA in a plant's leaves affect transpiration from the leaves?

4. What happens to turgor pressure in a plant's leaves as root water absorption decreases?

5. How does a decreasing amount of water in a plant affect the amount of K^+ in its guard cells?

6. What happens to a plant's water potential as the temperature of the environment decreases?

7. How would an increasing rate of solar irradiation to a plant's leaves affect the temperature of the leaves?

8. How would a decreasing amount of water in the earth's atmosphere affect a plant's photosynthesis rate?

9. How does increasing water potential in a plant's leaves affect the rate of K^+ efflux from the guard cells in the leaves?

10. 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?

11. 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?

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

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

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

15. How does an increasing amount of water in the accessory cells of a plant's leaves affect the rate of K^+ influx to the guard cells in the leaves?

16. How does an increasing amount of CO_2 in a plant's guard cells affect the size of the stomates in its leaves?

17. How does increasing water potential in a plant affect the plant's ABA amount?

18. How does decreasing water potential in a plant's accessory cells affect the amount of water in its guard cells?

19. How does an increasing amount of K^+ in a plant's accessory cells affect the amount of water in its guard cells?

20. How does decreasing soil water potential affect a plant's transpiration rate?

21. What happens to the size of a plant's stomates as soil moisture increases?

22. What happens to the water potential of a plant's leaves as the soil dries out?

23. How does a rising level of ABA in a plant affect the plant's water potential?

24. What happens to a plant's apoplast water potential as the temperature of the environment decreases?

25. How does decreasing photosynthesis in a plant's leaves affect the amount of glucose in its root system?

26. How does increasing temperature in the atmosphere surrounding a plant affect the plant's photosynthesis rate?

27. How does increasing transpiration from a plant's shoot system affect the plant's carbon dioxide absorption?

28. How would decreasing solar irradiation to a plant's leaves affect the plant's carbon dioxide absorption?

29. What is the effect on plant temperature of an increasing diffusion of heat from the atmosphere surrounding the shoot system to the earth's atmosphere?

30. What is the effect of an increasing amount of CO_2 in the symplast of a plant's leaves on the rate of photosynthesis in the leaves?

31. How does a decreasing amount of water in a plant's leaves affect osmosis from the guard cells to the accessory cells?

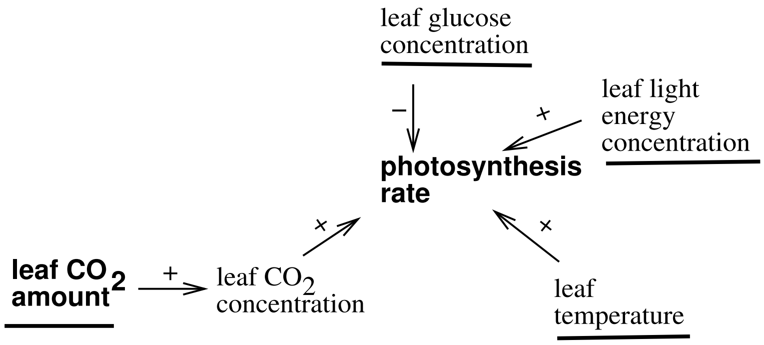
Appendix B

The Models TRIPEL Constructed

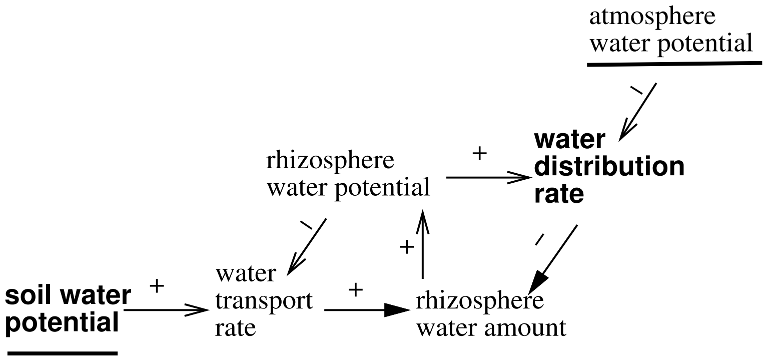
This appendix shows the models that TRIPEL constructed for the formal evaluation questions, as discussed in Chapter 9. The section numbers indicate the question numbers. The conventions are the same as in earlier figures:

- Arrows with solid tips represent differential influences.
- Arrows without solid tips represent functional influences.
- Exogenous variables 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.
- Activity preconditions of influences are not shown.
- Driving variables and variables of interest are shown in bold.

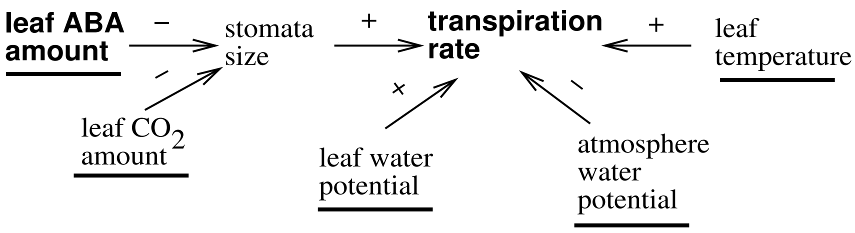
B.1 How would an increasing amount of CO_2 in a plant's leaves affect the rate of photosynthesis in the leaves?



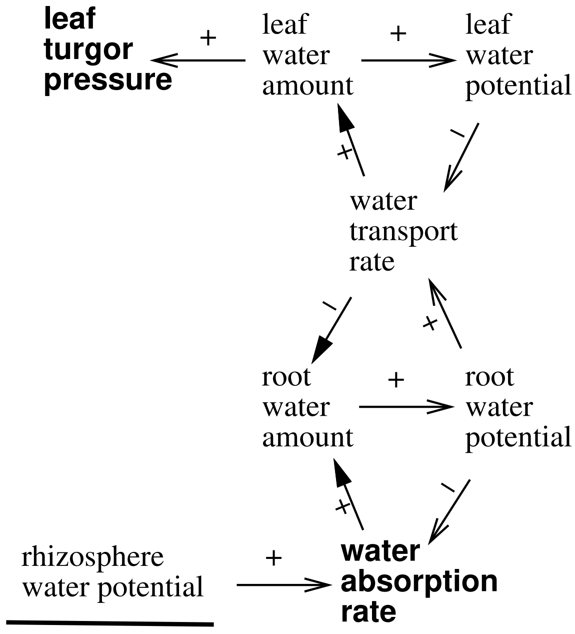
B.2 How does increasing soil water potential affect a plant's water distribution rate?



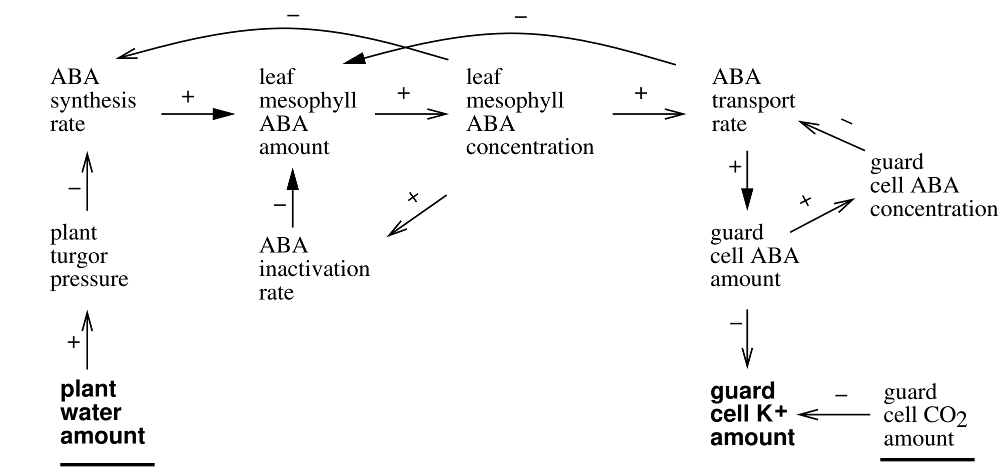
B.3 How does an increasing level of ABA in a plant's leaves affect transpiration from the leaves?



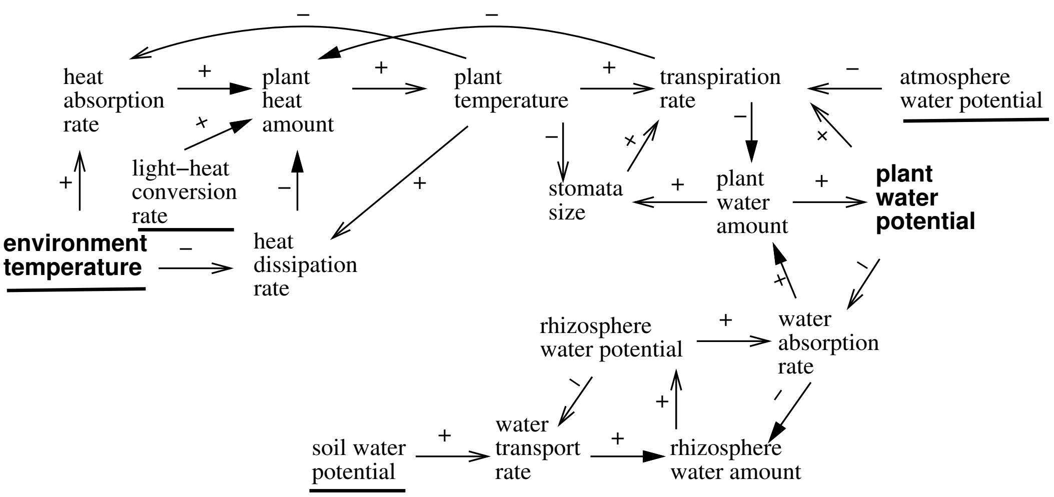
B.4 What happens to turgor pressure in a plant's leaves as root water absorption decreases?



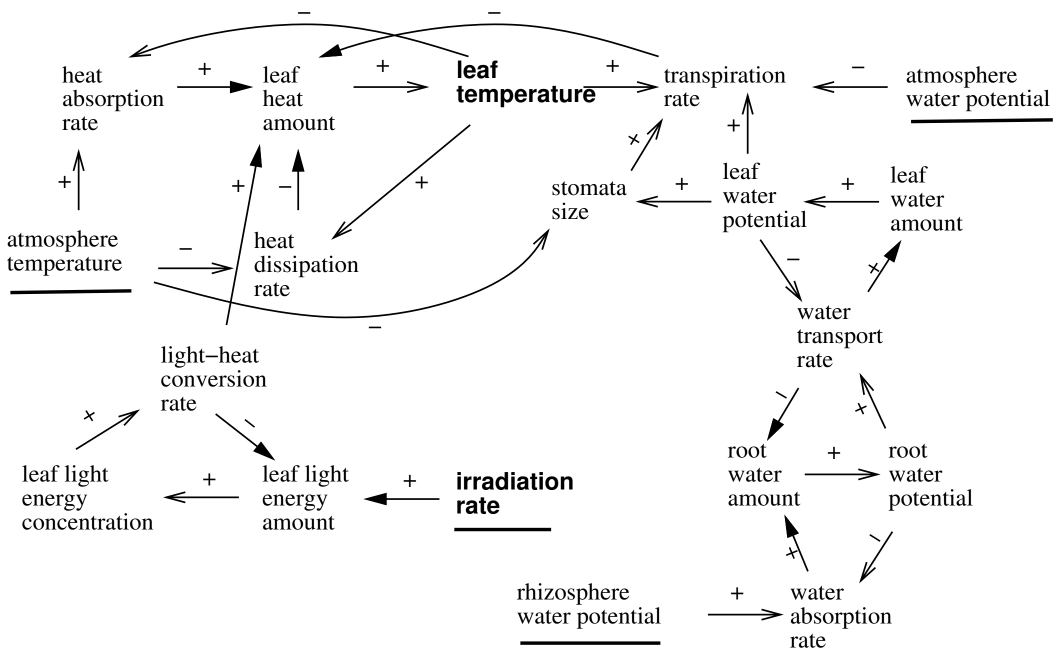
B.5 How does a decreasing amount of water in a plant affect the amount of K^+ in its guard cells?



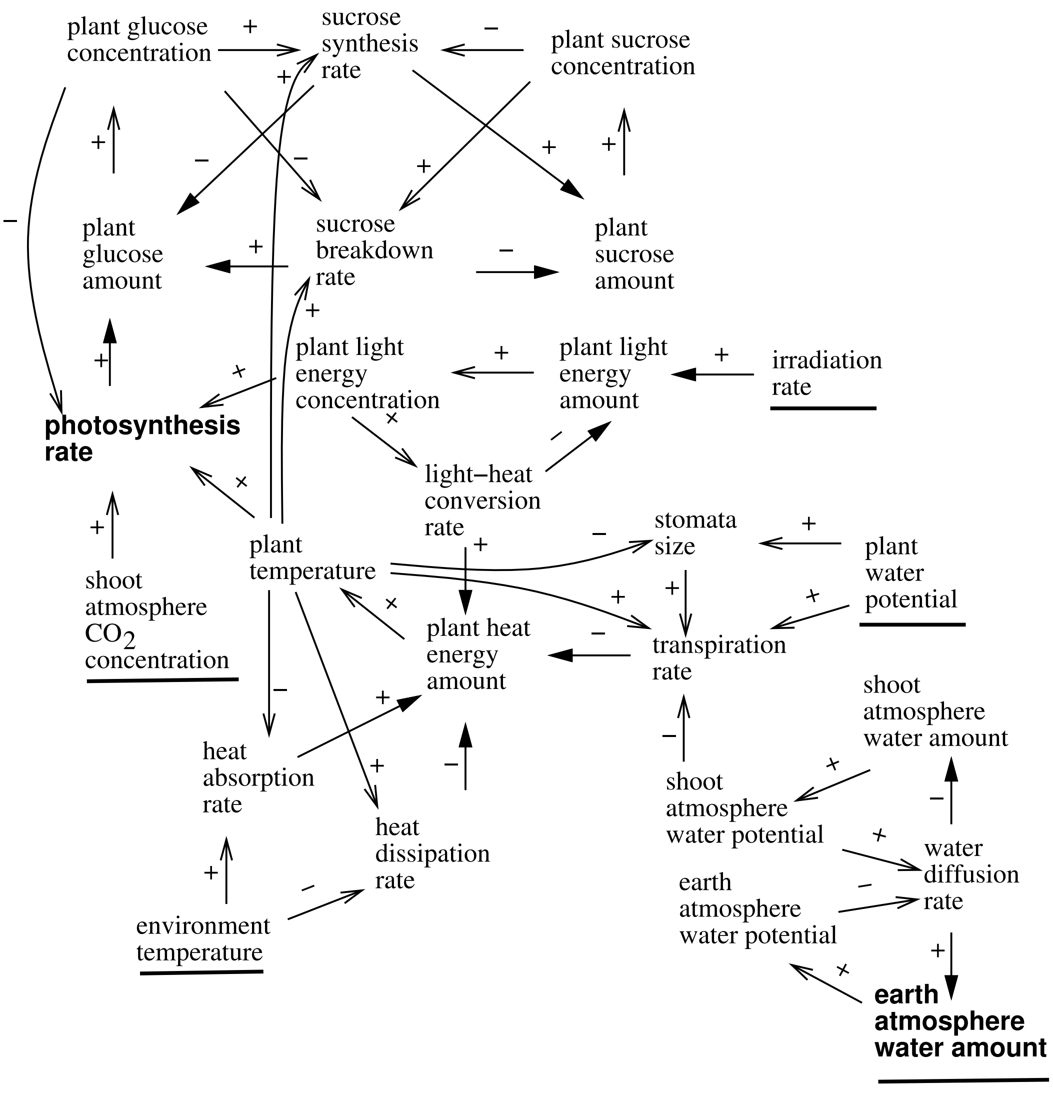
B.6 What happens to a plant's water potential as the temperature of the environment decreases?



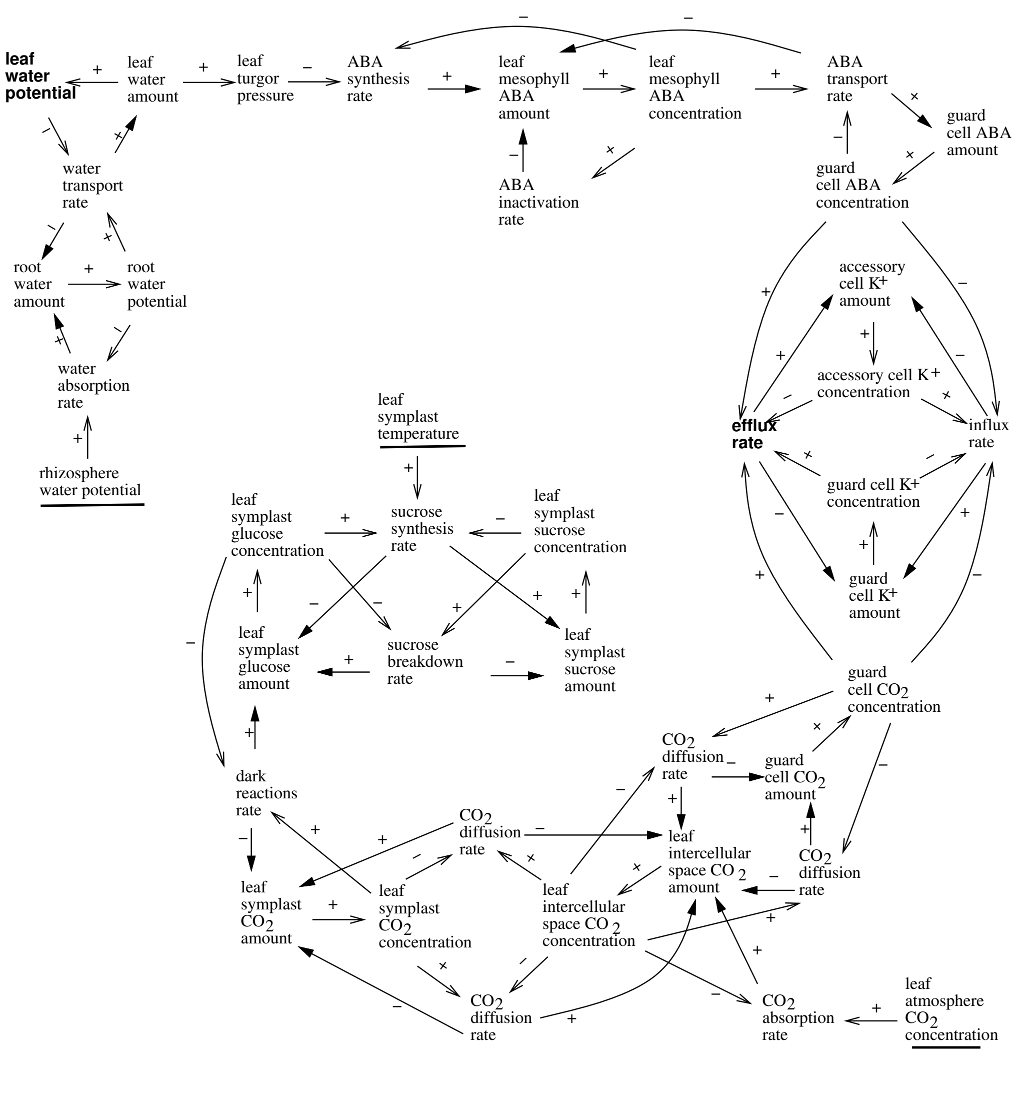
B.7 How would an increasing rate of solar irradiation to a plant's leaves affect the temperature of the leaves?



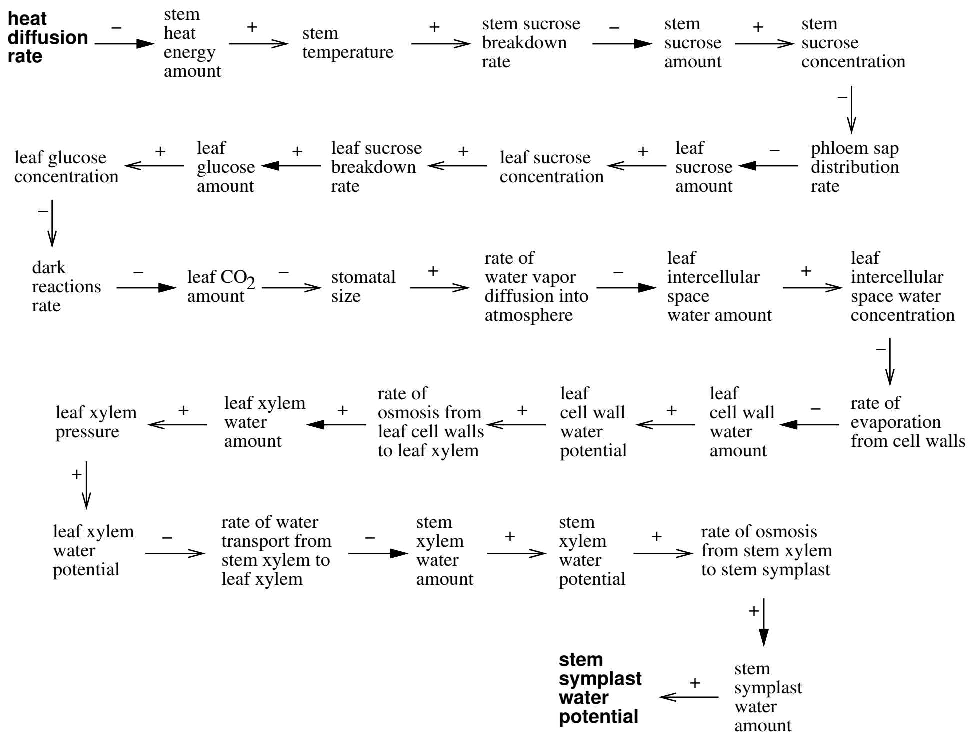
B.8 How would a decreasing amount of water in the earth's atmosphere affect a plant's photosynthesis rate?



B.9 How does increasing water potential in a plant's leaves affect the rate of K^+ efflux from the guard cells in the leaves?

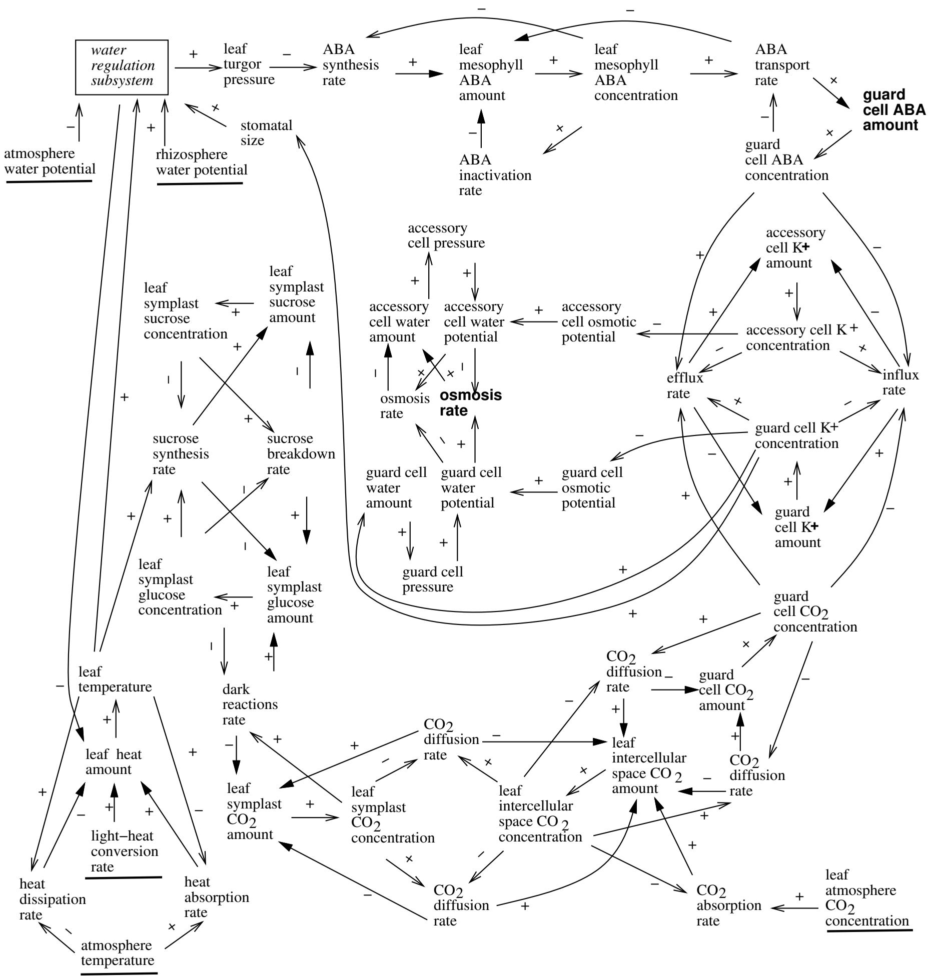


B.10 How does an increasing rate of diffusion of heat from the stems of a plant to the atmosphere surrounding the stems affect the water potential of the symplast in the stems?



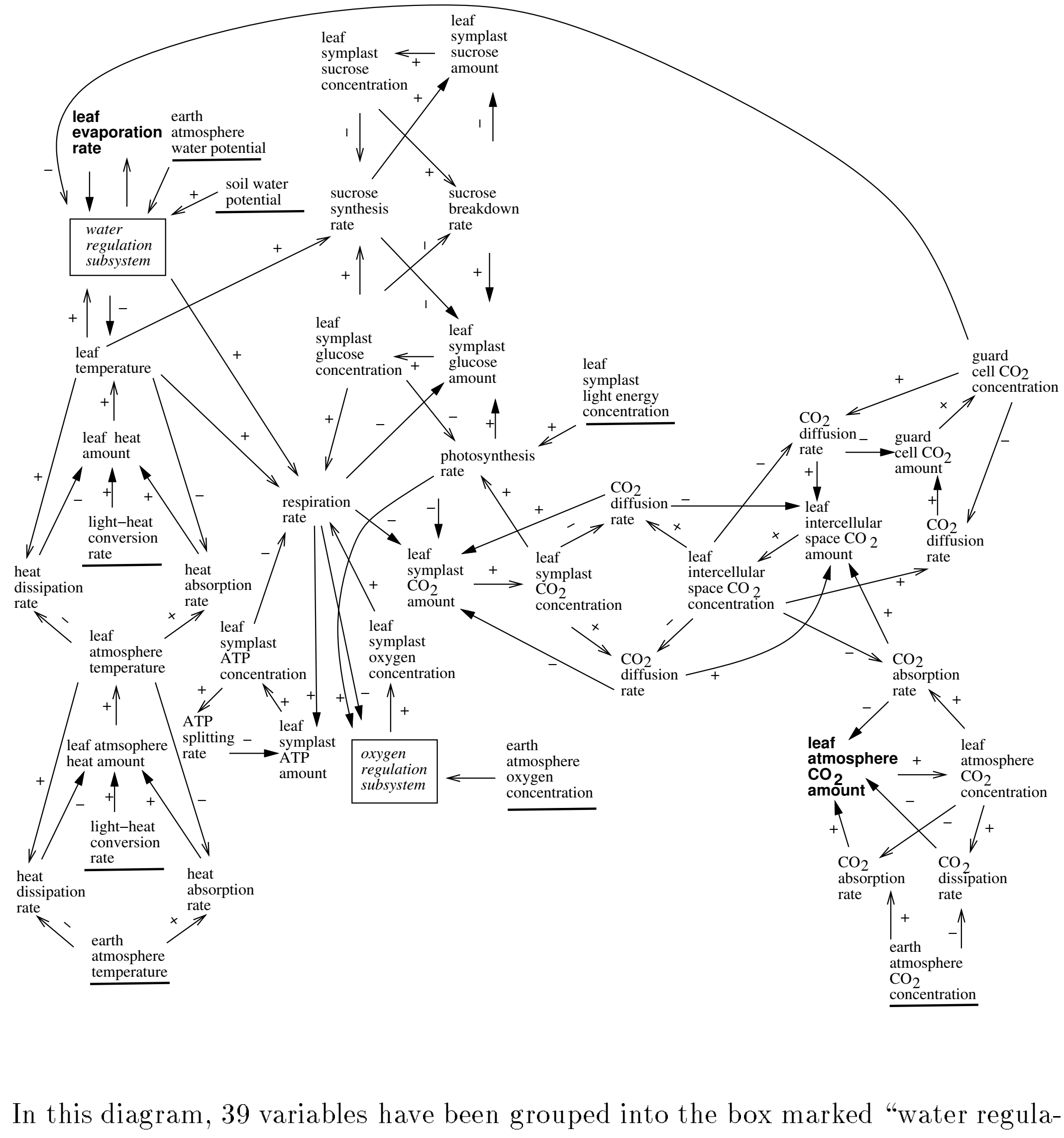
For this question, TRIPEL found a significant (by its criteria) influence path from the driving variable to the variable of interest on a time scale of minutes, and it constructed a large, complex model around the path. By the expert's criteria, this path is insignificant. According to the expert, there is a short, significant influence path from the driving variable to the variable of interest on a time scale of hours. Therefore, since the most interesting aspect of TRIPEL's model is the influence path it found, we show the path rather than the entire model.

B.11 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?



In this diagram, 28 variables have been grouped into the box marked "water regulation subsystem" to save space. These variables model the transport of water among the following pools: water in the rhizosphere, root cell walls, root xylem, root symplast, leaf cell walls, leaf mesophyll symplast, and leaf intercellular space.

B.12 How does a decreasing rate of evaporation from a plant's leaves affect the amount of CO_2 in the atmosphere surrounding the leaves?



In this diagram, 39 variables have been grouped into the box marked "water regulation subsystem" to save space. These variables model the transport of water among the following pools: water in the soil, rhizosphere, root cell walls, root xylem, root symplast, leaf xylem, leaf cell walls, leaf symplast, leaf intercellular space, and leaf atmosphere.

Similarly, ten variables have been grouped into the box marked “oxygen regulation subsystem.” These ten variables model the transport of oxygen among the following pools: oxygen in the leaf symplast, leaf intercellular space, and leaf atmosphere.

Bibliography

[1] L. Acker, J. Lester, A. Souther, and B. Porter. Generating coherent explanations to answer students' questions. In H. Burns and J. Parlett, editors, *Intelligent Tutoring Systems: Evolutions in Design.* Lawrence Erlbaum Associates, Hillsdale, NJ, 1990.

[2] Liane Acker and Bruce Porter. Extracting viewpoints from knowledge bases. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 547–552, Menlo Park, CA, 1994. AAAI Press.

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

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

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

[6] William E. Boyce and Richard C. DiPrima. *Elementary Differential Equations and Boundary Value Problems.* John Wiley and Sons, New York, third edition, 1976.

[7] J.U. Brackbill and B.I. Cohen, editors. *Multiple Time Scales.* Academic Press, New York, 1985.

[8] Bruce G. Buchanon and Edward H. Shortliffe, editors. *Rule-Based Expert Systems: The Mycin Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA, 1984.

[9] Charles B. Callaway. Tools for lexical augmentation and verification in large scale knowledge bases. Master's thesis, Artificial Intelligence Laboratory, University of Texas at Austin, 1995.

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

[11] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, Boston, MA, 1987.

[12] Eugene Charniak and Drew McDermott. *Introduction to Artificial Intelligence.* Addison-Wesley, Reading, MA, 1984.

[13] David K. Cheng. *Field and Wave Electromagnetics.* Addison Wesley, Reading, MA, 1983.

[14] Joe H. Chow, editor. *Time-Scale Modeling of Dynamic Networks with Applications to Power Systems.* Number 46 in Lecture Notes in Control and Information Sciences. Springer-Verlag, New York, 1982.

[15] 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.

[16] A. Collins and D. Gentner. How people construct mental models. In D. Holland and N. Quinn, editors, *Cultural Models in Thought and Language*, pages 243–265. Cambridge University Press, Cambridge, UK, 1987.

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

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

[19] J. de Kleer and J.S. Brown. A qualitative physics based on confluences. *Artificial Intelligence*, 24:7–83, 1984.

[20] Ulf Dieckmann and Colin P. Williams. Model approximation via dimension reduction. In T. Ellman, editor, *Working Notes of the AAAI Workshop on Approximation and Abstraction of Computational Theories*, pages 146–153, San Jose, CA, 1992.

[21] 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.

[22] Brian Falkenhainer. A unified approach to explanation and theory formation. In J. Shrager and P. Langley, editors, *Computational Models of Scientific Discovery and Theory Formation*. Morgan Kaufmann, San Mateo, CA, 1990.

[23] 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.

[24] 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.

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

[26] Adam Farquhar. *Automated Modeling of Physical Systems in the Presence of Incomplete Knowledge*. PhD thesis, Artificial Intelligence Laboratory, University of Texas, 1993. Technical Report AI93-207.

[27] 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.

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

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

[30] 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.

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

[32] R.H. Gardner, W.G. Cale, and R.V. O'Neill. Robust analysis of aggregation error. *Ecology*, 63(6):1771–1779, 1982.

[33] Michael R. Genesereth and Nils J. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Kaufmann, Los Altos, CA, 1986.

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

[35] Ira P. Goldstein. Overlays: A theory of modelling for computer-aided instruction. Artificial Intelligence Laboratory Memo 495, Massachusetts Institute of Technology, Cambridge, MA, 1977.

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

[37] Walter Hamscher, Luca Console, and Johan de Kleer, editors. *Readings in Model-Based Diagnosis.* Morgan Kaufmann, San Mateo, CA, 1992.

[38] Y. Ijiri. Fundamental queries in aggregation theory. *Journal of the American Statistical Association*, 66(336):766–782, 1971.

[39] Y. Iwasa, V. Andreasen, and S. Levin. Aggregation in model ecosystems. I. perfect aggregation. *Ecological Modelling*, 37:287–302, 1987.

[40] Y. Iwasa, S.A. Levin, and V. Andreasen. Aggregation in model ecosystems. II. approximate aggregation. *IMA Journal of Mathematics Applied in Medicine and Biology*, 6:1–23, 1989.

[41] Y. Iwasaki. Causal ordering in a mixed structure. In *Proceedings of the Seventh National Conference on Artificial Intelligence (AAAI-88)*, pages 313–318, San Mateo, CA, 1988. Morgan Kaufmann.

[42] 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.

[43] 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.

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

[45] J.A. Jacquez. *Compartmental Analysis in Biology and Medicine.* University of Michigan Press, Ann Arbor, MI, 1985.

[46] S.E. Jorgenson. *Fundamentals of Ecological Modelling.* Elsevier, New York, 1988.

[47] 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.

[48] D.C. Karnopp, D.L. Margolis, and R.C. Rosenberg. *System Dynamics: A Unified Approach.* Wiley-Interscience, New York, 1990.

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

[50] 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.

[51] H.J. Kook and G.S. Novak, Jr. Representation of models for expert problem solving in physics. *IEEE Transactions on Knowledge and Data Engineering*, 3(1):48–54, 1991.

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

[53] 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.

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

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

[56] Xiang-Seng Lee. *Temporal and Spatial Analysis in Knowledge-Based Physics Problem Solving.* PhD thesis, Artificial Intelligence Laboratory, University of Texas, Austin, TX, December 1992. Technical Report AI93-205.

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

[58] 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.

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

[60] Zheng-Yang Liu and Arthur M. Farley. Shifting ontological perspectives in reasoning about physical systems. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI-90)*, pages 395–400, Menlo Park, 1990. AAAI Press.

[61] G.F. Luger. Mathematical model building in the solution of mechanics problems: Human protocols and the MECHO trace. *Cognitive Science*, 5:55–77, 1981.

[62] 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.

[63] Kenneth S. Murray. KI: An experiment in automating knowledge integration. Technical report, Artificial Intelligence Laboratory, University of Texas at Austin, 1988. Technical Report AI88-90.

[64] Kenneth S. Murray and Bruce W. Porter. Controlling search for the consequences of new information during knowledge integration. In *Proceedings of the Machine Learning Workshop*, pages 290–295, Palo Alto, CA, 1989. Morgan Kaufmann.

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

[66] P.P. Nayak. *Automated Modeling of Physical Systems.* PhD thesis, Department of Computer Science, Stanford University, September 1992. Report No. STAN-CS-92-1443.

[67] G.S. Novak, Jr. Representations of knowledge in a program for solving physics problems. In *Proceedings of IJCAI-77*, pages 286–291, Cambridge, MA, 1977.

[68] 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.

[69] R.V. O'Neill and B. Rust. Aggregation error in ecological models. *Ecological Modelling*, 7:91–105, 1979.

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

[71] 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.

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

[73] Jeff Rickel and Bruce Porter. Automated modeling for answering prediction questions: Exploiting interaction paths. In *The Sixth International Workshop on Qualitative Reasoning about Physical Systems*, pages 82–95, Edinburgh, Scotland, 1992. Heriot-Watt University.

[74] 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.

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

[76] 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.

[77] S.I. Rubinow. *Introduction to Mathematical Biology.* John Wiley and Sons, New York, 1975.

[78] 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.

[79] W.M. Schaffer. Ecological abstraction: The consequences of reduced dimensionality in ecological models. *Ecological Monographs*, 51(4):383–401, 1981.

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

[81] 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.

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

[83] K. Wark. *Thermodynamics.* McGraw-Hill, New York, 1983.

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

[85] B.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.

[86] 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.

[87] Stephen Wolfram. *Mathematica: A System for Doing Mathematics by Computer.* Addison-Wesley, 1988.

[88] Bernard P. Zeigler. Simplification of biochemical reaction systems. In L.A. Segal, editor, *Mathematical Models in Molecular and Cellular Biology.* Cambridge University Press, Cambridge, 1980.

[89] Bernard P. Zeigler. *Multifacetted Modelling and Discrete Event Simulation.* Academic Press, New York, 1984.

Index

Vita

Jeffrey Walter Rickel, the son of Walter and Ellen Rickel, was born in Madison, Wisconsin, on March 11, 1963. After graduating from Berkner High School, Dallas, Texas, in 1981, he entered Texas A&M University on a National Merit Scholarship. He received the degree of Bachelor of Science in Computer Science from Texas A&M in May 1985, graduating summa cum laude. As a senior, he was awarded a Senior Engineering Achievement award, for which a chevron bearing his name hangs at the entrance to the Zachry Engineering building.

Jeff worked at Texas Instruments Incorporated in Dallas, Texas, developing software each summer from 1980 through 1984. In May 1985, he joined Texas Instruments as a Software Design Engineer in the Industrial Automation branch, where he was responsible for the design and development of artificial intelligence systems. While working full-time, he took graduate classes at the University of Texas at Dallas, where he received the degree of Master of Science in Computer Science in December 1987. In August 1988, he took a leave of absence from Texas Instruments to enter the Ph.D. program in Computer Science at the University of Texas.

Permanent Address: 11303 Prairie Dog Trail
Austin, Texas 78750

This dissertation was typeset with LaTeX 2_ε[1] by the author.

[1] LaTeX 2_ε is an extension of LaTeX. LaTeX is a collection of macros for TeX. TeX is a trademark of the American Mathematical Society. The macros used in formatting this dissertation were written by Dinesh Das, Department of Computer Sciences, The University of Texas at Austin.