QUALITATIVE REASONING ABOUT PHYSICAL SYSTEMS WITH MULTIPLE PERSPECTIVES

by

LIU, ZHENG-YANG

A DISSERTATION

Presented to the Department of Computer
and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

March 1991

APPROVED: ______________________
Dr. Arthur M. Farley

An Abstract of the Dissertation of

Liu, Zheng-Yang for the degree of Doctor of Philosophy

in the Department of Computer and Information Science

to be taken March 1991

Title: QUALITATIVE REASONING ABOUT PHYSICAL SYSTEMS WITH MULTIPLE PERSPECTIVES

Approved: ______________________________
Dr. Arthur M. Farley

To study commonsense reasoning and to write programs that reason effectively about the physical world, we must understand the nature of perspective-taking in qualitative reasoning — which perspectives to take, how to represent them, and when to shift from one perspective to another.

This dissertation defines a task-driven approach to perspective-taking for automated qualitative reasoning. Central to our approach is the notion that model formulation and selection is an integral part of reasoning about complex physical systems. Given a task, models can be created that reflect different configurations of the system topology, at various structural granularities, and in distinct language forms.

This dissertation describes a computational framework that integrates three perspective-taking dimensions in reasoning about electronic circuits: topological configuration, structural aggregation, and ontological choice. The research shows that by using a task-driven, perspective-taking approach, we can extend the range of automated qualitative causal reasoning about complex physical systems.

VITA

NAME OF AUTHOR: Liu, Zheng-Yang

PLACE OF BIRTH: Hefei, Anhui, China

DATE OF BIRTH: December 26, 1955

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon
University of Maryland, College Park
Hefei Polytechnic University

DEGREES AWARDED:

Doctor of Philosophy, 1991, University of Oregon
Master of Science, 1984, University of Maryland, College Park
Bachelor of Arts, 1981, Hefei Polytechnic University

AREA OF SPECIAL INTEREST:

Computer and Information Science

PROFESSIONAL EXPERIENCE:

Instructor, Computer and Information Science,
University of Oregon, Eugene, OR 1985-87, 1990

Graduate Teaching / Research Fellow, Computer and Information Science,
University of Oregon, Eugene, OR 1985 - 90

Teaching / Research Assistant, Computer Science,
University of Maryland, College Park, MD 1982 - 84

AWARDS AND HONORS:

Tektronix Fellowship Award (Scholarship) (1990, 1987).

Outstanding Service Recognition Award (Certificate),
University of Oregon (1987).

International Student Outstanding Achievement Award (Certificate), University of Oregon (1986).

Outstanding Academic Achievement Award (Certificate), University of Maryland, College Park (1982).

PUBLICATIONS:

[Douglas & Liu 1989]
Douglas, S. A. and Liu, Z. Y., Generating Causal Explanation from a Cardio-Vascular Simulation. In Proceedings of 11th International Joint Conference on Artificial Intelligence, (IJCAI-89), Detroit, Michigan, August, 1989.

[Liu 1984]
Liu, Z. Y., Database Support for Software Engineering. M.S. Thesis, University of Maryland, College Park, 1984.

[Liu 1988]
Liu, Z. Y., Diagnosing and Modeling in Intelligent Teaching Systems. In Expert Systems and Their Applications: Proceedings of the 8th International Conference, Avignon, France, June 1988.

[Liu 1989]
Liu, Z. Y., A Charge-Carrier Ontology for Reasoning about Electronics. In Proceedings of the 3rd International Workshop on Qualitative Physics. XEROX PARC, Palo Alto, California, August 1989.

[Liu 1991]
Liu, Z. Y., Integrating Two Ontologies for Electronics. In B. Faltings and P. Struss, editors, Advances in Qualitative Physics, MIT Press, 1991.

[Liu & Farley 1990]
Liu, Z. Y. and Farley, A. M., Shifting Ontological Perspectives in Reasoning about Physical Systems. In Proceedings of the National Conference of Artificial Intelligence (AAAI-90), Boston, MA, August 1990.

[Liu & Roussopoulos 1984]
Liu, Z. Y. and Roussopoulos, N., A Tutorial To ADMS (Autolog Database Management System), Technical Manual, Department of Computer Science, University of Maryland, College Park, April 1984.

[Yeh & Liu 1984]
Yeh, R. and Liu, Z. Y., Information-Base Support for Software Development, Technical Report, No. 84-2074. Department of Computer Science, University of Maryland, College Park, August 1984.

ACKNOWLEDGEMENTS

I would like to thank my advisor, Professor Art Farley, for his inspiration, guidance, and encouragement. He held numerous discussions with me about various issues addressed in this work, and read and reread the drafts, far beyond the call of duty. I am most grateful to him for his high standards on working ethics, his genuine moral support, and patience throughout this work.

I wish to thank Professors Andrzej Proskurowski, Steve Fickas, Sarah Douglas, and Marjorie Taylor for serving on the committee, reading the drafts, and giving valuable comments. I would also like to thank Professor Ginnie Lo for her advice and generous support when it was needed. This work benefits from discussions with Keith Downing, Pat Hayes, and Dan Weld. I appreciate the help and cheers from fellow graduate students in the department and gracefully acknowledge the assistance from Betty Lockwood, Jan Saunders and Barbara Allen in the office.

I want to thank my wife, Xiao-Mei, for her love, understanding, and sacrifice. I thank my beautiful daughter, Amy, for her good timing to come to this world, for being so considerate that I could concentrate on this work, and for her smiles that make this world meaningful. The patience, understanding and encouragement from my parents is deeply appreciated. Speaking of family, I must thank my host family, Dr. Masoud and Carol Givi, for their love, friendship and help.

Last but not the least, I want to thank the Chao's family for financial sponsorship and for having confidence in me to pursue education in the United States. I am also grateful to Mr. Mi Rui-Lun for his assistance at Hefei Polytechnic University, China. Without their help in the first place, this work would not have been possible.

This work is supported in part by a Tektronix Fellowship Scholarship.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

How to reason effectively about complex physical systems is a fundamental problem in the quest for principles of intelligence. This dissertation investigates one aspect of this problem - reasoning about physical systems with multiple perspectives. The study develops a task-driven, perspective-taking approach to qualitative causal reasoning. This approach is based on two insights regarding modeling in general and research in qualitative physics in particular. First, all model-based reasoning is only as good as the model upon which it is based. Second, no single model is adequate or appropriate for a wide range of tasks. Central to our approach is that model generation and model selection are an integral part of reasoning about complex physical systems.

This chapter sets the stage to address the research problem of this dissertation - how to integrate multiple perspectives to extend the range of automated qualitative reasoning about complex physical systems. At the end of this chapter, we identify specific research issues addressed in this dissertation.

Why Multiple Perspectives?

Physical systems in the real world are infinitely rich and diverse. They share an important property: they can be understood and described from different perspectives. To reason effectively about a physical system in the real world for a particular task requires an appropriate description of the system. For example, we know that human bodies are composed of an enormous number of cells, and everything we do could in principle be described in terms of those cells [Hofstadter 1980]. Most of us, fortunately, do not view ourselves this way. If we did, the reasoning would be grossly inconvenient and inefficient, if not intractable. Yet for some tasks, this description can be by far the most important view, if you are a doctor trying to diagnose some blood disease. Clearly, which perspective to take depends on what task we set out to perform. Very often, multiple perspectives are necessary to carry out a single task.

Figure 1 shows a half-wave rectifier that takes an *ac* input and produces a *dc* output. It is a simple nonlinear circuit. With a sinusoidal input voltage, the circuit gives rise to two different configurations as the diode alternates between forward-bias and reverse-bias. The following explanation is typical of the standard electronics textbook treatment of how the circuit works:

> The capacitor becomes charged up almost to the input peak voltage when the diode is forward biased. When the diode is reverse biased, the capacitor partially discharges through the load. Since the capacitor always has some positive charge, the diode becomes forward biased only near the peaks of input voltage. At this time it passes a current pulse to the capacitor to replace the charge lost to the load. (Bell 1980, p. 45)

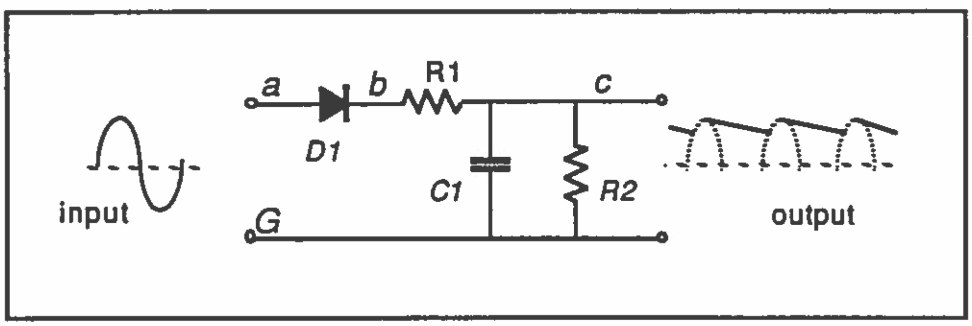


FIGURE 1. Half-Wave Rectifier with Capacitor Smoothing

We can make a number of observations about this example:

- The explanation is purely in qualitative, causal terms. No numerical formulas or calculations are used. In addition, the circuit's overall behavior is inferred from the behaviors of individual components and their interconnections.
- The explanation integrates both macroscopic (device-level) and microscopic (charge-level) descriptions, such as voltage and current or charge.

- The explanation deals with nonlinear behavior of the circuit based on two unstated configurations of the circuit, reflecting distinct device states (as shown in Figure 2).
- The explanation suppresses unnecessary structural and behavioral details. For example, the explanation suppresses mentioning *R1* and *R2* when the diode is forward-biased but still implies their functionality.

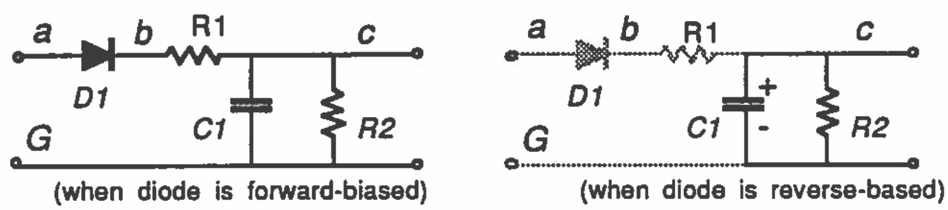


FIGURE 2. Two Configurations of Half-Wave Rectifier

Qualitative and causal reasoning is a central and coherent aspect of human mental life [Forbus & Gentner 1986]. Abstracting from this example of qualitative reasoning, we see three dimensions of "perspective-taking" that are typically involved:

1. Ontological choice - using distinct language forms to describe behavior of a system;
2. Topological configuration - focusing on a subset of the structure of a system;
3. Structural aggregation - abstracting structural elements into composite constructs.

Our ability to reason about the physical world from different perspectives and to switch among them as needs dictate is fundamental to our intelligence and flexibility [Hofstadter 1980, Hobbs 1985, Hayes 1985]. If we are ever to have a machine of even moderate intelligence, we must have a theory of perspective-taking woven into the very foundation of its reasoning processes. To build such a theory, the Artificial Intelligence (AI) field of qualitative physics has recently witnessed a research thrust into reasoning with multiple perspectives [Hayes 1985a, Collins & Forbus 1987, Kuipers 1987, Farley 1988, Addanki, et al. 1989, Weld 1989, Falkenhainer & Forbus 1991]. To date, however, the problem of integrating multiple dimensions of perspective-taking in automated reasoning, crucial for analyzing complex physical systems such as the rectifier above, stands as a challenge. This dissertation develops a computational framework that integrates three perspective-taking dimensions in automated qualitative reasoning: topological configuration, structural aggregation, and ontological choice.

In the rest of this chapter, I will first address the general questions in modeling physical systems from distinct perspectives. I will then discuss the problem of model formulation from appropriate perspectives and shifting perspectives by switching models during reasoning to carry out a specific task. Finally, I will present the thesis statement along with the research issues addressed in this dissertation.

Ontological Choice and Commitment

In order to reason about a physical system, we first have to be able to describe the system in a model using some representational language, as depicted in Figure 3 [Smith, Davis 1989]. Whatever representational formalism we choose to describe the system, a model embodies a set of entities[1], the conceptions that we have about them, and the interrelationships that exist among them. In automated reasoning systems, we say that these three kinds of elements constitute a specific ontological choice for describing the system [Munitz 1974, Hayes 1985a].

The world is initially unlabeled. As Hayes (1985a) points out, labeling into conceptual entities is not made by nature itself, but by us and our modeling language. Although in philosophy there is no universal agreement on how ontology is to be characterized, we

---

[1] An entity is a "thing" that exists and is distinguishable from others in the world, real or imaginary [Munitz 1974]. For example, an entity can be a physical object, such as a car, a book and a radio; or an abstract object, such as the Chinese nation and the American culture; or an imaginary object, such as the king of France in 1990 if there had not been a French Revolution [Hayes 1985a].

espouse Quine's stand that the task of ontology is to establish "what there is" [Munitz 1974]. Thus, suffice it to say that an ontological choice in modeling a physical system consists of a specific language with a particular set of terms, predicates, and axioms for modeling and reasoning [Hayes 1985a, 1985b, Davis & Hamscher 1988].

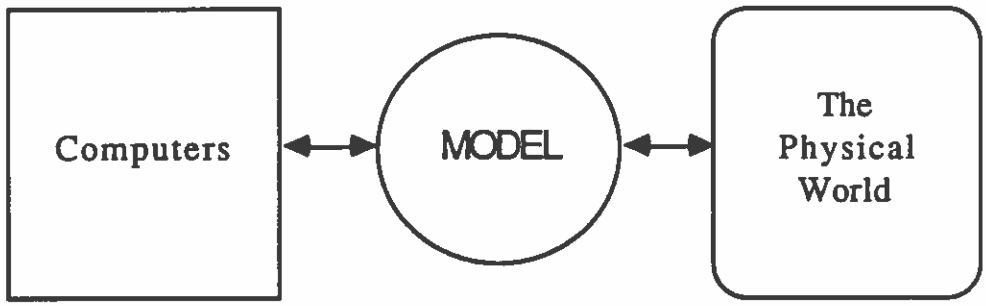


FIGURE 3. Use of Models

A physical system can be described from several, distinct ontological perspectives. For example, Hayes (1985a) identifies two distinct ontologies for reasoning about liquids. He notices that sometimes an engineer thinks of "the liquid in the container" as an object (the contained-stuff ontology) while at other times thinking about a hypothetical collection of molecules traveling together through the system as an object (the piece-of-stuff ontology). Consequently, the two ontologies involve different language forms as terms, predicates, and axioms. When one uses the contained-stuff ontology for liquids, the terms available include volume, pressure and so on. These terms are not applicable to the piece-of-stuff ontology for liquids, which refers to fixed mass, spatiotemporal position, velocity, etc.

Employing a particular language to describe the behavior of a physical system is equivalent to committing to a particular ontological choice. If we require a term as an important means for reasoning, then whatever type of universe is needed to define this term is the universe to which we are committed. For example, as soon as we use the term pressure to describe a liquid, we have committed to the contained-stuff ontology for liquids with pressure defined by the set of predicates and axioms therein. Such an ontological commitment leads to a particular perspective for modeling and subsequent analysis of the real world systems [Hayes 1989].

Topological Configuration

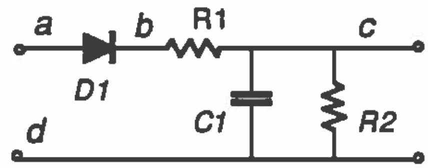
Human engineers, when dealing with the complexity of large scientific and engineering domains, always keep attention directed at the specific task at hand by focusing analysis in the simplest model possible. They eliminate the set of descriptions irrelevant for the given task by considering only those entities that affect their present interests [Hobbs 1985, Addanki, et al. 1989, Weld 1989]. One such technique is to recognize different topological configurations of a complex physical system.

By "configuration" is meant a subset of the system components in the system topology which are on the active flow path for things such as liquid, gas, or electric current in a specific qualitative state of the system. All such components, linear or nonlinear, behave linearly over the duration of the given qualitative state. Figure 4 shows four distinct topological configurations of the half-wave rectifier under different qualitative states.

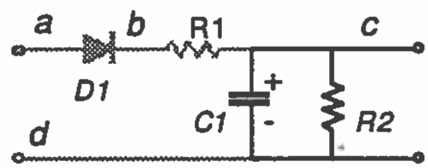
To reason about circuits such as this, engineers formulate local models for each of the configurations and switch between those local models [Keogh & Suntag 1981] when the active configuration changes to a different subset of system components due to the change of the system's qualitative state. When reasoning concerns one of the configurations, they can temporally shut off or "forget" about the models of the other configurations. Such a local model is based on a subset of the topology of the entire system. This facilitates efficient reasoning for problem-solving, especially if the bulk of the analysis stays within only a few local models.

Furthermore, the approach provides a configuration-wise linearization method for analyzing nonlinear physical systems since each such configuration is linear [Rugh 1981]. For a nonlinear circuit, if we can justify why the current state yields the present configuration in terms of the behavior of system components and their internal states, and if we can analyze each such configuration, then we can analyze the whole circuit.

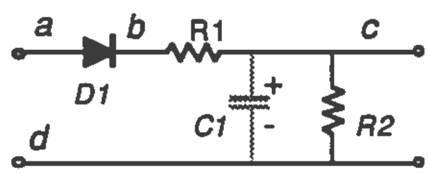
[When diode is forward-biased, and capacitor is charging]



[When diode is reverse-biased, and capacitor is discharging]



[When Voltage$_{a,d}$ = +, but ∂Voltage$_{a,d}$ remains at 0]



[When capacitor is not charged, and Voltage$_{a,d} \leq 0$]

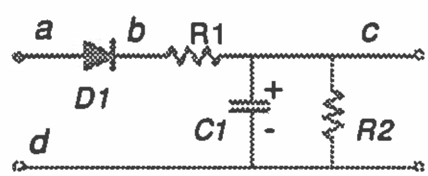


FIGURE 4. Multiple Topological Configurations of a Circuit

Structural Aggregation

Besides topological configuration, structural aggregation is another technique for reducing a complex structural description of a system into a simple one. Its application is easily observable in daily life. We look at a world map and notice that cities are represented as dots. When planning a trip from one city to another, this description is adequate and efficient. Only when we need to know how to get around in a city do we get a street map of the city. We switch between these two maps with ease, as needs dictate.

In circuit analysis, experts conveniently use "black boxes" to suppress unnecessary detail in a model [Sussman & Steele 1980]. An engineer can replace a group of resistors, which occur either in series or parallel, by a single, equivalent resistor in a model to simplify the analytic task at hand. In some cases, the whole circuit can be regarded as a black box in analysis to achieve efficiency and clarity.

Figure 5 shows a black-box description of a circuit by Rusgrove, et al. (1977), where the black box is viewed as a single resistor. It could be that on opening the box one would find that it actually contained a radio transmitter that was converting the *dc* power from the battery into radio-frequency power which in turn was being radiated off into space. The transmitter is certainly not designed as a resistor. But if the task at hand was to find out the total current through the circuit, as measured at ammeter *A*, then the transmitter did act as a resistor. Thus, by aggregating the circuit as a black box and viewing it as a single resistor in this case, the circuit description is properly simplified for the task at hand.

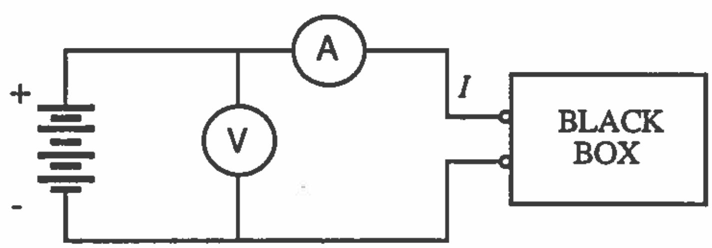


FIGURE 5. A Circuit Viewed as a Black Box

Black-boxing with structural aggregation introduces different structural granularities for modeling and reasoning. Clearly, an advantage of using structural aggregation in automated reasoning is that reasoning becomes efficient. But only part of the motivation here is efficiency. Even if our computers could run our programs infinitely fast to handle complexity, for most purposes of prediction and explanation of the behavior of physical systems, we simply do not need or want information of great detail. When we do need detail, it is typically about a narrow range of behaviors [Falkenhainer & Forbus 1988].

Integrating Multiple Perspectives

We have introduced the problem of perspective-taking in reasoning about physical systems. We have also discussed three dimensions of perspective-taking: ontological choice[2], topological configuration, and structural aggregation. In this work, perspective-taking is defined as generating or selecting models that embody distinct views of a target system for different reasoning tasks. Model generation is defined as representation reformulation of general domain knowledge specializing for a specific task at hand. Using electronic circuits as an example, this dissertation integrates the three dimensions of perspective-taking into a coherent computational framework, ARC[3], for qualitative causal analysis.

Perspective and Model

By "a model of a physical system" is meant a symbolic representation of the system, formulated for computer programs to reason about the system. Each model embodies a specific perspective which can involve choices made on several dimensions. An overall perspective embodied in a model is a position taken in each of the possible dimensions. In this work, a model can be formulated embodying a perspective involving an ontological choice, a topological configuration, and structural granularity.

---

2 In this thesis, we use a technical definition of the term ontology as defined in the field of qualitative physics: an ontology is defined as a language [Hayes 1985a, 1989], a vocabulary [Davis & Hamscher 1988], an organizational structure [Forbus 1988, 1989], with a particular set of terms, predicates, and axioms, for modeling and reasoning about a physical system. The two ontologies used in this work, the device ontology and the charge-carrier ontology, correspond to the macroscopic and microscopic levels of descriptions of circuit behavior in physics [Weidner 1985]. This aspect of the work is in line with the research by Collins & Forbus (1987), Falkenhainer & Forbus (1988), Franz & Weld (1990).

3 ARC — Acronym for *A*utomated *R*easoning about *C*ircuits.

Figure 6 shows eight such models of the rectifier circuit generated by ARC. Each model embodies a distinct perspective of one or several dimensions.

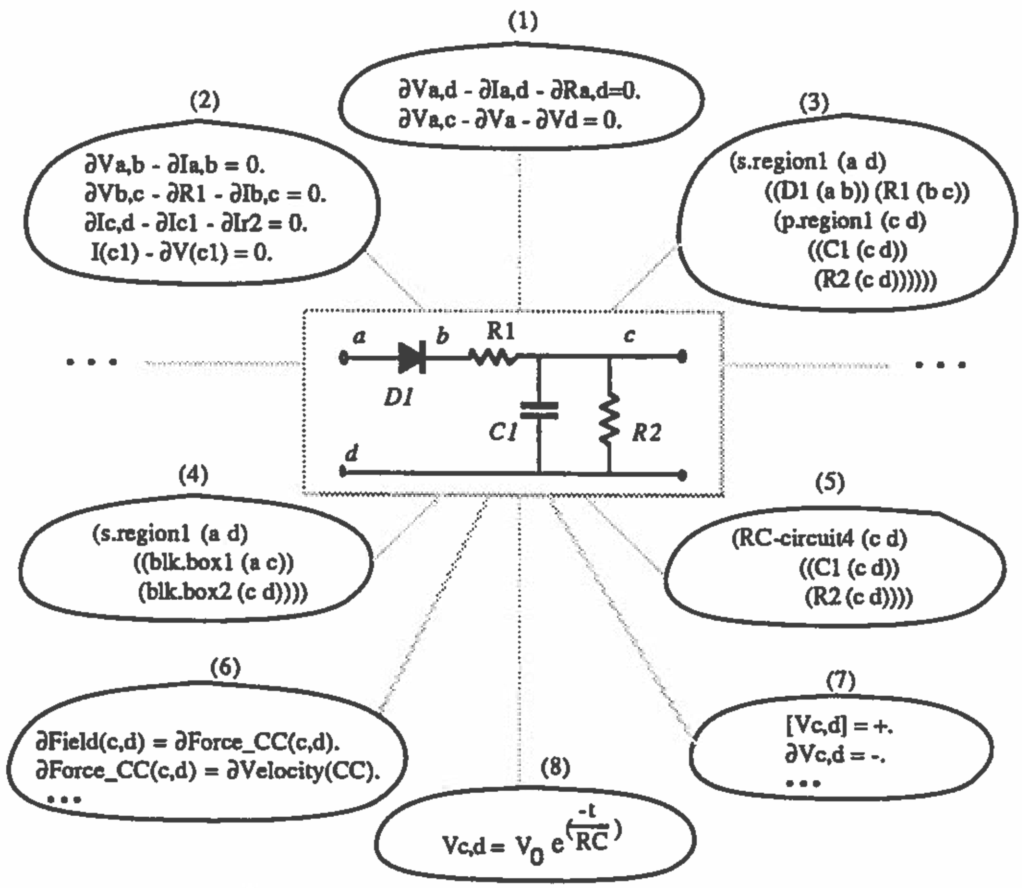


FIGURE 6. Model Formulation from Distinct Perspectives of a Circuit

In general, a model of a circuit can be formulated that embody an N-dimensional choice of perspective. For example, a description of a circuit's active topological configuration in a specific qualitative state stands as a model which embodies a 1D perspective (e.g., models 3 and 5 in Figure 6), revealing the dynamic structure of the circuit topology in the given state. A description of the structural aggregation over the active configuration stands as a model embodying a 2D perspective (e.g., model 4 in Figure 6). This particular perspective (embedded in model 4) reveals not only the dynamic configuration of the circuit, but also a series of two black boxes encapsulating component devices in the configuration. The QDE (qualitative differential equations) description of an active configuration stands as a model embodying a 3D perspective (e.g., models 1, 2, 6, and 7 in Figure 6), with the additional dimension reflecting ontological choice. For example, models 1, 2, and 7 are in the standard device ontology [de Kleer & Brown 1984, Williams 1984], while model 6 is in the charge-carrier (CC) ontology [Liu and Farley 1990]. Model 8 reflects the configuration of the RC circuit and describes how the voltage exponentially decays in the RC configuration.

To generate models for a task, the system topology of the circuit associated in the task is given as input. This is the only circuit-specific information provided for ARC, showing the static structural connectivity of the circuit component devices. It is a direct mapping from the circuit schematics. No assumptions are made about whether it is a linear or nonlinear system or possible current flow paths. Since it is given as input and remains unchanged throughout the reasoning process, we say that the system topology of a circuit stands as the origin of perspective-taking dimensions[4] in ARC. Given a particular task, models embodying appropriate perspectives of a target system can be dynamically generated for carrying out the task.

Shifting Perspectives

Having multiple models embodying distinct perspectives offers little advantage over a single model unless our programs can automatically switch from one to another as necessary. A central theme of this dissertation is that multiple perspectives are often intertwined in effective, efficient reasoning about complex physical systems. It is thus crucial that reasoning with multiple perspectives include mechanisms which enable shifting to a model that embodies an appropriate perspective when the current one is found to be

---

[4] Granted, the system topology of a circuit as a static structural description mapped over from the schematics of the circuit already embodies an ontological perspective. But since it is given as input and remains unchanged in the qualitative simulation, we treat it as primitives in our program.

inappropriate for the task at hand.

This research explores a task-driven approach to perspective-taking. For qualitative causal analysis of electronic circuits, we focus on standard perturbation analysis [de Kleer & Brown 1984, Williams 1984]. We introduce a special task-definition language (TDL) to define tasks for qualitative causal reasoning of electronic circuits from structure to behavior. Given a task definition, we show how crucial pieces of information are extracted by ARC to drive perspective-taking with model formulation and selection to carry out the task.

For ontological shift, we use domain-independent ontological-choice rules based on the task at hand to control ontological choice. Ontological shift is carried out by switching QDE models via bridging relations, which associate comparable elements from two related ontologies. In the present implementation of ARC, we use the standard device ontology at the macroscopic level and the CC ontology at the microscopic level to reason about electronics circuits. The bridging relations in this case involve mappings between macroscopic and microscopic concepts of electronics.

In case of topological configuration of a circuit, each model of a configuration involves a specific qualitative state of the circuit as the underlying assumption for the configuration. If the assumption becomes violated as a new qualitative state arises during qualitative simulation with a particular task, ARC identifies and shifts to the new configuration which satisfies the state and where reasoning continues.

When using structural aggregations to simplify the description of a circuit, aggregated constructs can be zoomed in and out following the inherent hierarchical view of the physical organization of the circuit. Furthermore, a task may suggest that black boxes be created to suppress extraneous detail for circuit analysis. Different tasks may lead to generating different black boxes of various granularities of the same circuit. ARC dynamically generates appropriate aggregations as driven by the task at hand.

Thesis Statement and Research Issues

To date, the bulk of work in qualitative physics or model-based reasoning has focused on problems of analysis in the framework of a single model from a single perspective. Most work in qualitative physics assumes that appropriate models are given to the program as input [Falkenhainer & Forbus 1990]. This severely limits the flexibility and range of automated reasoning systems. To develop a computational framework for reasoning with multiple perspectives, this dissertation investigates the following thesis statement:

By using a task-driven, perspective-taking approach, we can extend the range of automated qualitative reasoning about complex physical systems.

To explore a task-driven approach to perspective-taking with ontological choice, topological configuration, and structural aggregation, this dissertation addresses three research issues.

1. Given a perturbation-analysis task, which perspectives does the reasoning program take?
2. How does the program dynamically generate or select models that embody appropriate perspectives?
3. When does the program shift from one perspective to another?

Road Map

Chapter 2 presents an overview of ARC, an implemented computer system that integrates the three dimensions of perspective-taking in reasoning about electronic circuits. Chapter 3 discusses identifying active configurations for automating qualitative reasoning about circuits. Chapter 4 focuses on aggregating parallel and serial constructs within a given configuration to suppress irrelevant details for a given task . Chapter 5 discusses QDE-model generation from required ontological perspectives. In these chapters, we show how shifting perspectives is accomplished by switching models during reasoning as driven by the task at hand. Chapter 6 presents examples and evaluation of ARC. We include an informal experts' evaluation of ARC's performance.

Chapter 7 reviews the literature of the past work related to the current research, discussing the fundamentals of qualitative physics as well as previous approaches to modeling and reasoning with multiple perspectives. Readers who wish to know the background of the current research should read this chapter prior to reading chapters 2, 3, 4, 5 and 6. Chapter 8 concludes the dissertation by discussing the contributions and limitations of this research. Finally, we outline possible extensions as future research problems.

CHAPTER II

AN INTEGRATED FRAMEWORK — ARC

This chapter presents an overview of ARC, an implemented computer system that integrates multiple dimensions of perspective-taking to formulate and select models and reason about electronic circuits. We investigate three such dimensions: topological configuration, structural aggregation, and ontological choice. We begin by presenting the organization and design rationale of ARC. We then introduce a simple task-definition language and describe what information ARC extracts from a task definition for perspective-taking. Finally, we illustrate how ARC selects and integrates perspectives for various tasks with examples from ARC's qualitative simulation. The detailed theories of each specific perspective-taking technique will be discussed in the subsequent chapters.

Organization of ARC

Design Rationale

The virtue of qualitative causal reasoning about a physical system is to reason from structure to behavior [Bobrow 1985, Kuipers 1986, Forbus 1988]. The basic structural description of a circuit is the system topology of the circuit. It shows the static structural connectivity of circuit component devices and remains unchanged throughout the reasoning process. For ARC, it is the only circuit-specific information provided. No assumptions are made about whether the circuit is a linear or nonlinear system or about possible current flow paths. No hints are given about which individual components should constitute certain black boxes. Given a general domain theory of electricity, the system topology of a target circuit, and a task specification, ARC generates one or several models of the circuit and dynamically switch between them during to carry out the task while suppressing unnecessary detail. To this objective, ARC is designed to meet the following requirements:

- Reasoning from structure to behavior, ARC must first identify the active configurations of a target circuit as driven by the task at hand, rather than resorting to a predefined configuration given as input. A configuration provides a dynamic structural description of the target circuit. As we seek to reason about circuits from both the macroscopic and microscopic perspectives, this structural description should be compatible to both the device ontology and the CC ontology.
- After recognizing the active configuration in the simulation, the task at hand may suggest simplifying configuration models by viewing certain sub-circuits as black-boxes to suppress extraneous detail for the task at hand. ARC must be able to generate different structural abstractions, rather than resorting to fixed structural constructs [Sussman & Steele 1980, Davis 1984, Genesereth 1985]. This is a crucial step for ARC to create appropriate structural granularity for the task at hand for efficient qualitative causal reasoning.
- The fact that different tasks may require reasoning with different ontological perspectives and switching among them as needed during reasoning indicates that ARC must be able to reason with independent ontologies, rather than parasitic ones [Collins & Forbus 1987]. To develop a general paradigm for ontological shift between the macroscopic and microscopic levels of behavioral descriptions, ARC must allow bi-directional mapping between the two ontologies, rather than only one-way mapping.

All this requires that ARC's perspective-taking be task-driven. That is, from a given task, ARC extracts necessary information and formulates circuit models appropriate for carrying out the task. ARC generates models on the fly, rather than resorting to canned models. The models are generated in such a way that uninteresting structural and behavioral details of the target circuit are suppressed. In brief, perspective-taking in ARC focuses on formulating or selecting appropriate models that capture the salient features of the target circuit for the task at hand.

ARC Architecture

As an automated framework to reason about electronic circuits, ARC integrates techniques of topological configuration, structural aggregation and ontological shift to carry out tasks for qualitative causal reasoning. Figure 7 depicts the organization of ARC.

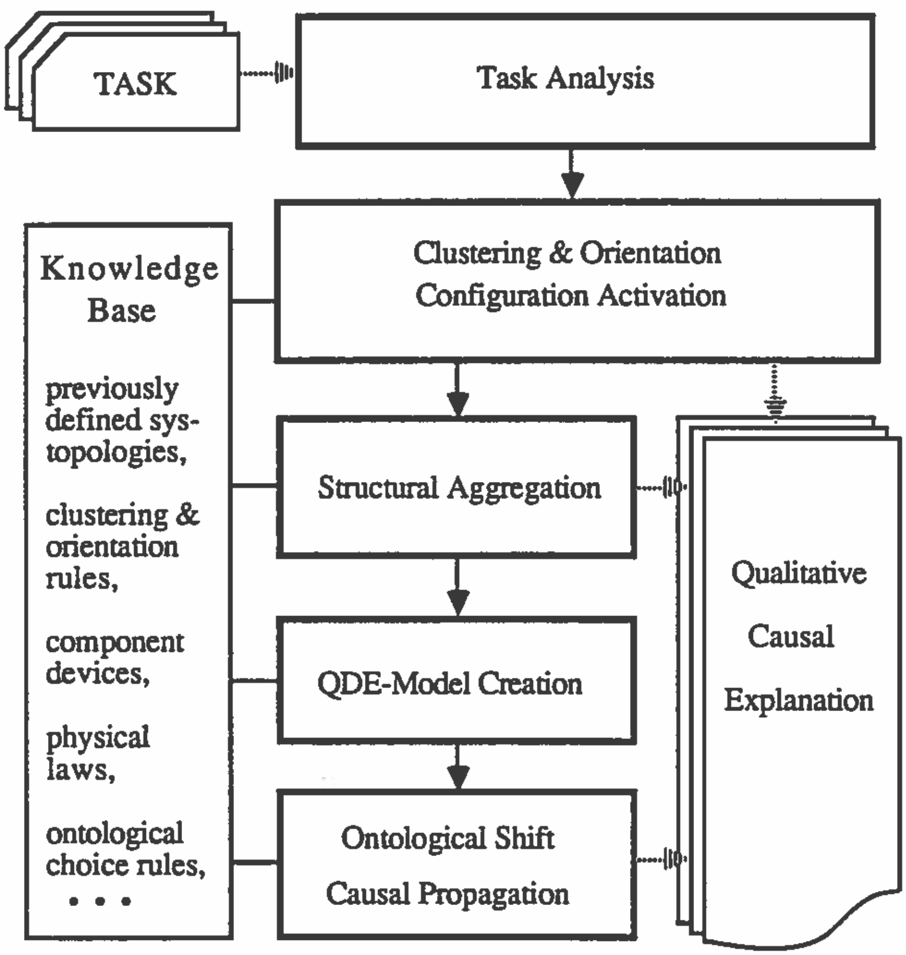


FIGURE 7. ARC Architecture

ARC takes as input a task definition, consisting of the name of a target circuit and specifications of input perturbations and desired output, and produces as output qualitative causal explanations of the circuit behavior. As driven by the task at hand, ARC first recasts the system topology of a target circuit into oriented clusters to reveal a hierarchical view of the structure of the circuit. From these oriented clusters, ARC identifies the active configuration of the circuit in simulation based on the qualitative state of the circuit. Following the configuration process, ARC aggregates certain sub-circuits into black boxes if it can simplify the description of the circuit for the task at hand. Configuration and aggregation both simplify the models of the structure of the target system. Since qualitative causal reasoning proceeds from structure to behavior, how to view the structure of the system directly affects how its behavior is described. ARC then generates QDE (Qualitative Differential Equations) models, from proper ontological perspectives, for causal propagation to explain the circuit's behavior as a result of the input perturbations.

In the following sections, we describe how various perspectives are selected and integrated by ARC to formulate models for carrying out a given task. First, we need to discuss how to define a task and what information ARC extracts from it for selecting appropriate perspectives.

Task Definition

Our interest in qualitative reasoning about a physical system is to be able to generate causal explanations for the behavior of the system resulting from input perturbations to one of the system parameters. This reasoning process follows a specific perturbation-analysis task defined by specifying the input and desired output with respect to the target system under analysis. Three items comprise a task definition:

(1) Name of a target system (Target-System-Spec),

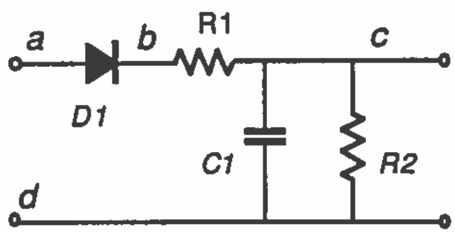
(2) Specification of input perturbations (Input-Spec), and

(3) Specification of output desired (Output-Spec).

System Topology of a Circuit

The name of a target system in a task definition provides ARC with the access to the system topology of the circuit under analysis — individual devices in the circuit and their connectivity. As a pure structural description, it is the only circuit-specific information provided.

To define the system topology of a circuit in ARC, one needs to specify the types of component devices and how the devices are connected, via nodes, into a device network. Figure 8 shows the system-topology specification of the half-wave rectifier circuit .




```
(define-circuit!  Half-Wave-Rectifier1
  ( ; Components:
      (Diode:       D1)
      (Resistor:    R1   R2)
      (Capacitor:   C1) )
  ( ; Connections:
      (a  (terminal1 D1))
      (b  (terminal2 D1)  (terminal1 R2))
      (c  (terminal2 R1)  (terminal1 C1)  (terminal1 R2))
      (d  (terminal2 C1)  (terminal2 R2))))
```

FIGURE 8. ARC's Representation of System Topology of a Circuit

Each connection in the system topology of a circuit is represented as a relation, following the schema:

$$(<node_p> (<terminal_i> <device_a>)\ (<terminal_j> <device_b>)\ ...),$$

which says that $terminal_i$ of $device_a$, $terminal_j$ of $device_b$, etc, are connected to $node_p$. Each device and node in the network is represented as an object that has its own internal state and procedures to interface with the outside world. Given a task, ARC compiles the circuit topology into an object-oriented representation by setting up links and connections as attributes of the objects. Object-oriented representation is chosen because it facilitates modeling of devices with class-wide assumptions [de Kleer & Brown 1984] and provides modularity for qualitative simulation. In addition, ARC maintains a catalog of system-topology specifications of circuits created in the simulation environment, each of which can be referenced by name in a particular task definition.

Input Perturbations

The traditional qualitative simulation paradigm is primarily concerned with how a physical system, in some equilibrium state, responds to a single input perturbation. This has been the practice of circuit analysis in qualitative physics up to date. Conventionally , this is called "small signal analysis" [de Kleer 1984, Williams 1984]. For example, an increase (or decrease) of the voltage at the input node of a circuit is usually expressed as

$$\partial V_{in} = +, \quad \text{or} \quad \partial V_{in} = -.$$

This small-signal-analysis method, however, works under a hidden assumption that the active configuration of the target circuit is pre-defined and the states of the component devices stay within expected landmark values. For automated reasoning about circuits with nonlinear devices, this method no longer suffices because the assumptions simply assume away the problem that a qualitative simulation system such as ARC seeks to solve.

ARC extends the single-perturbation method by allowing a sequence of input perturbations based on a discrete set of time points, rather than on a fixed increment. Such a sequence can be of arbitrary length. In contrast to only a single qualitative value in small signal analysis, each perturbation value in a sequence now consists of its qualitative value and the direction of change (Qval Qdir). For example, one may give a sequence of perturbations to specify a cycle of sinusoidal voltage wave at the input node as shown in Figure 9.

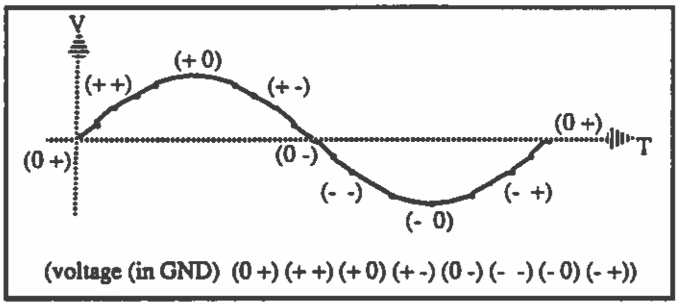


FIGURE 9. Sequence of Perturbations as Input

In this example, each of the perturbing signals in a sequence specifies a qualitative state of the system parameter "voltage$_{in,GND}$" with (Qval, Qdir) [Kuipers 1986], rather than just a single qualitative value. Nodes (in GND) in the input specification serve as the two chosen pole nodes in the circuit[5]. If only one node is specified, ARC will assume "ground" for the other node. A sequence of such perturbations drives the qualitative simulation of the circuit. Each signal's processing is performed in the context created by those signals early in the sequence and early in time. Allowing a sequence of signals improves the range of qualitative reasoning about complex circuits such as the rectifier circuits we saw in the previous chapters. Perturbation for small signal analysis is easily represented as a sequence of length one. For zero-order analysis, we just let Qdir = 0 in (Qval Qdir) of a sequence.

Output Specification

The output ARC produces is a qualitative causal explanation of the target system behavior as a result of the input perturbations. ARC allows the user to indicate specific system variables as desired output. For example, one can specify to analyze voltage, current, or charge flow with respect to the relative nodes in the system topology of a circuit. The nodes specified define the <u>output structural unit</u> consisting of either a single

---

[5] The perturbations to system parameters of a circuit do not have to be at the pole nodes. We can perturb system parameters, such as the resistance of a resistor, to cause changes in a circuit.

device or a group of devices.

An output variable specification is best viewed as placing a "probe" in the circuit. In this respect, the current or charge-flow probe must be placed in series with the output structural unit, and the voltage probe be placed in parallel with the unit where the voltage difference is of interest. If a voltage probe is specified with only one node, ARC automatically assumes "ground" for the other node.

Information Extracted from a Task

Table 1 summarizes the kinds of information ARC extracts from the three items of a task definition. For example, "For the half-wave rectifier with a sinusoidal input voltage, what is the behavior of the output voltage between nodes *c* and *d*?" This task is specified as follows:

- Target System: [Half-wave-rectifier1]
- Input Specification: (voltage (a d) (0 +) (+ +) (+ 0) (+ -) (0 -) (- -) (- 0) (- +))
- Output Specification: (voltage (c d)).

From this task definition, ARC extracts the following pieces of information crucial for selecting perspectives to carry out the task, in addition to the information about the system topology of the target circuit:

> The pole nodes of the circuit are *a* and *d*. The system parameter being perturbed is $voltage_{a,d}$. The input is a sequence of perturbations. The qualitative values and derivatives are given by the sequence of qualitative states specified in the sequence. With this sequence of perturbations, the polarities of the pole nodes change along in the sequence. The output variable of interest is $voltage_{c,d}$. The output structural unit of the output variable is the sub-circuit between nodes *c* and *d*. Both input and output variables are specified in the macroscopic-level device ontology.

TABLE 1. Information Extracted by ARC from a Task Definition

Item in Task	Information Extracted by ARC
Target System Name	System Topology (Circuit Schematics). • component devices & their types • connectivity of devices via nodes
Input Specification	System parameter being perturbed; Pole nodes and polarities Perturbation type • a single quantity perturbation • a sequence of quantity perturbations • a mathematical function[6] Qualitative value of each perturbation Qualitative derivative of each perturbation Input ontology
Output Specification	System variable desired in output Structural unit of the output variable Output ontology

This information extracted from a task provides ARC with guidance for perspective-taking. It enables ARC to formulate appropriate models of the target system to carry out the given task.

Below, we illustrate how ARC dynamically formulates models of the target system to adapt to the task at hand for qualitative causal analysis and produces output explanations.

[6] As an immediate future work, we let ARC accept, as input perturbation, a mathematical description of the behavior of a system parameter, e.g., a sine-wave input voltage.

Task-Driven Perspective-Taking

Configuration

At we emphasized at the outset, structural descriptions are crucial to qualitative causal reasoning since causal reasoning proceeds from structure to behavior [de Kleer & Brown 1984, Forbus 1988]. At the first glance, it seems that the system topology of a circuit, as in Figure 8, already captures the entire structural information of the circuit. Why do we need to identify active configurations from the system topology for qualitative causal analysis? To answer this question, it is important to define what roles configurations play in contributing to a circuit's behavior.

As defined in Chapter 1, an active configuration of a circuit consists of those component devices in the system topology of the circuit which are on the current flow path in a specific qualitative state. All such components, linear or nonlinear, behave linearly over the duration of the qualitative state. As such, by definition, a configuration is a linear circuit.

Since an active configuration represents the flow path in a circuit at a specified time, identifying configurations thus plays the role of zero-order analysis of current flow [White & Fredricksen 1986]. Furthermore, the polarity of a component device's connections in the configuration can be easily determined. Central to the configuration concept is that the impedance of each component device to the current flow follows Ohm's Law. For linear circuits, the process to identify active configurations reveals the directionality of current flow through each component devices. This directionality is important for generating underlying QDEs to describe the behavior of each such device.

For nonlinear circuits, the process to identify active configurations plays an additional role of linearizing the system for analysis. As Bugh (1981) points out, when confronted with a nonlinear-system analysis task, the standard approach is find a way to linearize; that is, to try to eliminate nonlinear aspects of the problem. He notes:

> If one cannot find a way to linearize a nonlinear system engineering problem, the tendency is to try to avoid the situation altogether, presumably in the hope that the problem will go away. Those engineers who forge ahead are often viewed as foolish, or worse. (Bugh 1981, p. 1)

For our purpose, since the behavior of each configuration is linear, if we can justify why the current state of a circuit yields the active configuration in terms of the behavior of circuit devices and their internal states, and if we can analyze each such configuration, then we can analyze the whole circuit.

Figure 10 illustrates ARC's configuration-wise approach capable of reasoning about nonlinear circuits. It identifies the active configuration, as driven by a task, in the system topology of a circuit by examining the state of each component to decide whether it is on the current flow path in the circuit. The subsequent qualitative reasoning to generate the next state of each device is based only on this active configuration. The new states of the component devices, together with the system topology, then determine the next configuration, and so on.

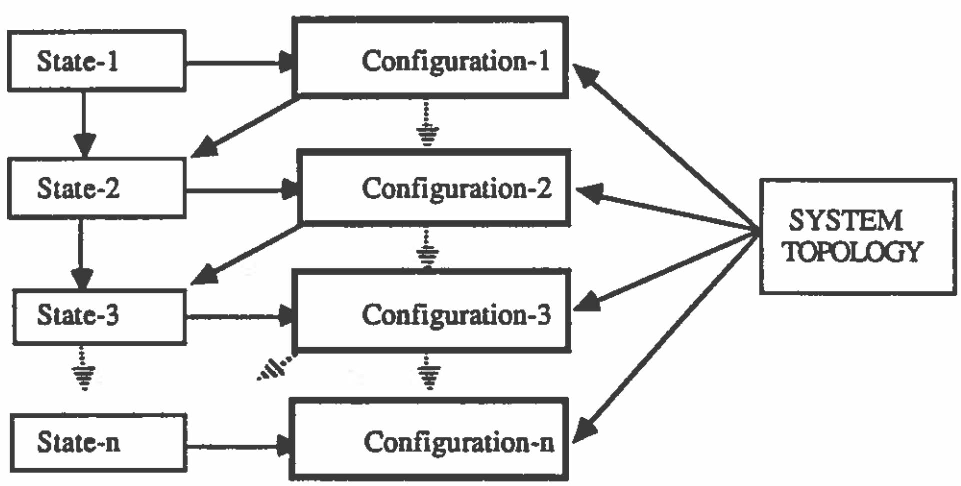


FIGURE 10. Configuration-Wise Approach to Qualitative Simulation

In the standard qualitative simulation framework, the intermediate level of configuration determination is missing. This is, in part, due to the fact that most previous systems represent the structure of a physical system simply by a set of parameters and the constraints that hold among them [Kuipers 1986]. Such a description does not account for the actual physical organization of the system. Hand-coding constraints between system variables in models is the way most previous systems have addressed this important modeling issue. This approach does not solve the problem; it simply avoids the issue.

We have shown in Chapter 1 four different configurations of the half-wave rectifier circuit in different qualitative states (Figure 4) with nodes (*a c*) as the poles. As another example, given two different input perturbations to analyze a ladder circuit in Figure 11, two different configurations arise because of different nodes as poles, though the system topology of the circuit remains the same. Evidently, the information revealed by different configurations, crucial for automated reasoning, is simply not available in the system topology of any circuit.

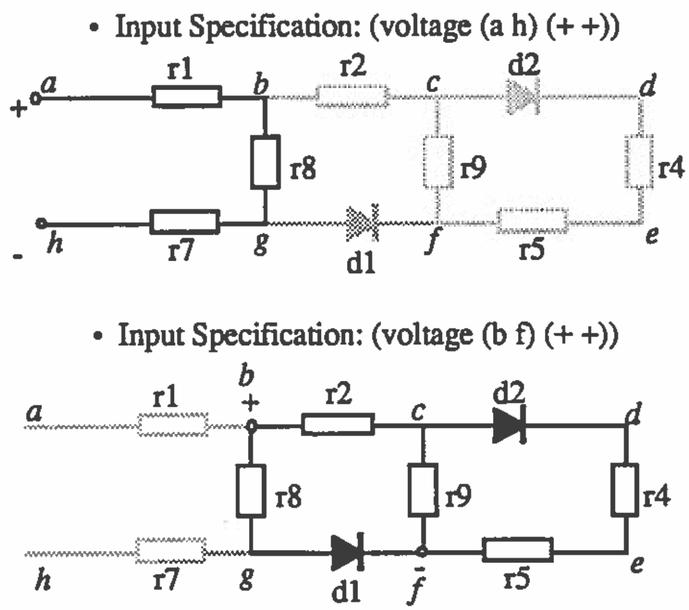


FIGURE 11. Different Configurations of the Same Circuit for Different Tasks

The representation of an active configuration of a circuit stands as a model which embodies a 1D perspective, revealing the dynamic configuration of the system topology in a given qualitative state. ARC creates local models representing these configurations and dynamically switches between them when the active configuration changes to a different set of circuit components during qualitative simulation. Techniques used in ARC to identify configurations help improve the range of automated reasoning about complex physical systems.

In Chapter III, we will describe in detail the approach used by ARC to identifying active configurations as driven by a given task in the qualitative simulation.

Aggregation

A task given to ARC may either ask about the behavior of a sub-circuit or some individual component device in a circuit. The ability to aggregate structural elements into black boxes to suppress unnecessary detail irrelevant to the task at hand not only allows the reasoning process to be more efficient, but also facilitates focusing the resulting explanation to the question asked. Such black boxes play the role of equivalent circuits commonly used in electronic engineering for circuit analysis [Bell 1980], where circuits are equivalent if the corresponding terminals have the same voltage and current in the same direction. A complex circuit can be viewed as a simple circuit for analysis if they two are considered equivalent [Sussman & Steele 1980].

The clues for ARC to aggregate component devices into black boxes come from the output specification in the task. If the task at hand is asking about the behavior of the voltage between any two nodes in the system topology of a circuit under analysis, ARC aggregates the component devices in between the two nodes into a "black-box" and views it as a single structural unit for causal propagation. Furthermore, the rest of the circuit in the active configuration may also be "black-boxed" to further minimize unnecessary detail for the task at hand.

For example, if one is interested in the behavior of voltage between nodes *c* and *d* in the half-wave rectifier, and suppose the active configuration recognized is the one when the diode is forward-biased and the capacitor is charging, then ARC aggregates the structure of the circuit by creating two black boxes, as shown in Figure 12. From this aggregated configuration, QDE models can be generated for qualitative causal reasoning about the behavior of the circuit.

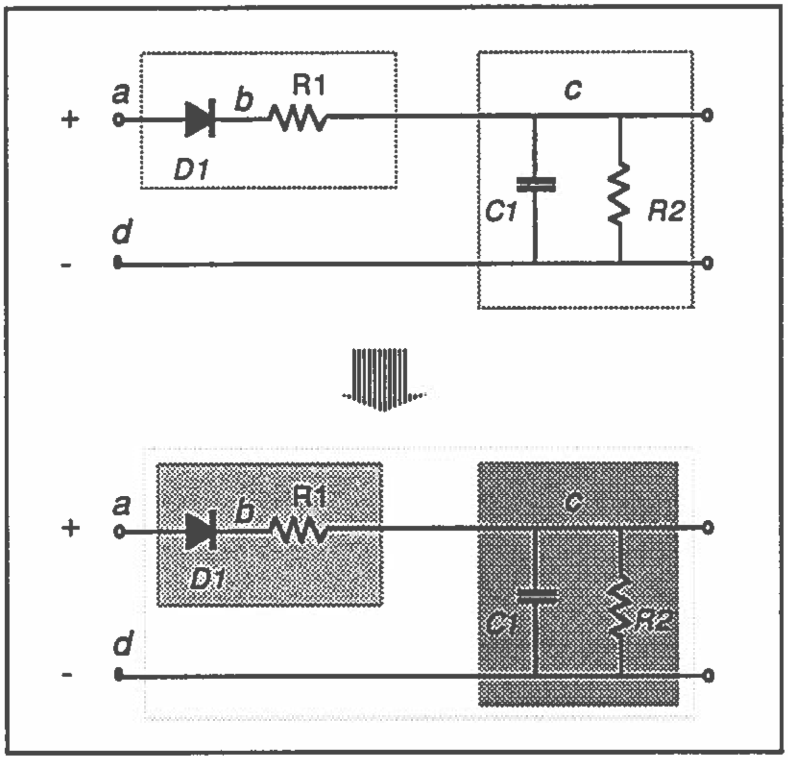


FIGURE 12. Structural Aggregation in Serial Configuration

For example, given the task:

- Target System: [half-wave-rectifier1]
- Input Specification: (voltage (a d) (+ +))
- Output Specification: (current (c d))

ARC generates the following explanations (verbatim script from ARC):

- In the serial region between nodes (a d), since its resistance remains constant, it is clear that voltage@a_d's increase causes current@a_d to increase.
- current@a_d's increase means current@a_c's increase because they two originate from the same source & share the same path in the configuration.

- According to KCL[7] with regard to node c for a serial connection,

  we know that current@a_c's increase leads to the increase of current@c_d.

This example shows that viewing the circuit this way allows ARC to skip the unnecessary behavioral description of resistors *R1*, *R2*, and *C1*.

If the interested behavior specified in a task's output happens to be on one of the branches in a parallel configuration, then this branch is singled out for structural aggregation. In addition, the other branches in the parallel construct is aggregated into a black box because the interested behavior is not affected by the details of the other branches for causal analysis.

In the circuit in Figure 13, if the task asks about the behavior of current through device *w2* when the battery's voltage changes, then *w2* will be the focus of causal reasoning. The rest of the branches are aggregated and viewed as a black box.

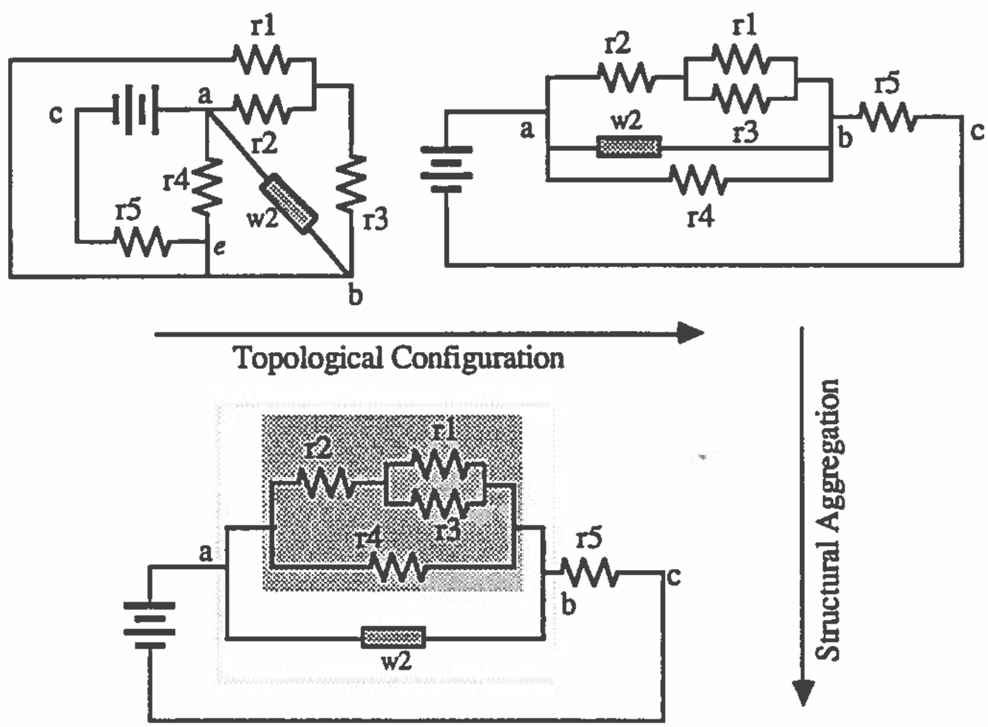


FIGURE 13. Structural Aggregation in Parallel Configuration

[7] KCL — Short hand for "Kirchhoff's Current Law".

For example, given the task:

- Target System: [ps-circuit2]
- Input Specification: (voltage (a c) (+ +))
- Output Specification: (current (a b w2))

ARC reasons as follows:

- In the serial region between nodes (a c), since its resistance remains constant, it is clear that voltage@a_c's increase causes current@a_c to increase.
- current@a_c's increase means current@a_b's increase because they two originate from the same source & share the same path in the configuration.
- In the parallel region between nodes (a b), since its resistance remains constant, we deduce that current@a_b's increase causes voltage@a_b to increase.
- For single device w2 between nodes (a b), voltage@a_b's increase causes current@w2 to increase.

A composite structural unit resulting from structural aggregation can be treated either as a white-box or a black-box. One can view it as a single unit (a black box) but always has the option of de-aggregating it back to its constituents (a white box) when needed.

For example, the above derivation proceeds with the aggregated construct, the parallel region between *a* and *b*, and treats it as a single unit. As such, it does not encounter the irrelevant, transient feedback analysis between the parallel constructs. After the global information is obtained, the derivation then treats it as a white box where resistor *W2* is revealed and concludes that current through *W2* increases. The details of the other resistors need not be mentioned at all for this particular task. The resulting explanation is short and to the point.

Although we did not discuss the configuration in the above example, it is done by ARC prior to creating black boxes. In both of the above examples, the structural and behavioral details within the aggregated structural units are black-boxed because they are considered to be uninteresting to the task at hand. The structural aggregation thus gains efficiency and clarity for the resulting causal explanations.

For the same active configuration, different aggregations can be made for tasks with different output structural units. A description of the structural aggregation over the configuration stands as a model embodying a 2D perspective. This perspective embodies not only the dimension of topological configuration of the circuit, but also the dimension of appropriate structural granularity with chosen black boxes, which encapsulates subsets of individual devices in the configuration. The structural aggregation techniques used by ARC to formulate such models are discussed in detail in Chapter IV.

Macro-Micro Level Shift

Note that the examples above have solely used the vocabulary from the device ontology of electronic circuits involving macroscopic concepts of voltage, current, and resistance. Most previous work of circuit analysis in qualitative physics has focused on the device ontology [de Kleer 1984, de Kleer and Brown 1984, Williams 1984, White & Fredricksen 1986].

While the device-ontology model can handle a wide range of analysis tasks at the macroscopic level, it cannot answer some basic questions that relate structures to behaviors, such as "Why does the current through a resistor increase when the voltage across it increases?" or "Why changing the length of a resistor affect the current through it, even if the voltage across it remain constant?" To answer questions such as these requires an appreciation of the forces that act upon charge carriers inside the device and the effects on charge-carrier movements under the force from externally applied bias voltage. The explanation process of an automated reasoning system for qualitative causal analysis should have an alternative of shifting to this microscopic-level reasoning.

Supplementing the standard device ontology for electronic circuits at the macroscopic level, this research introduces elements of a charge-carrier (CC) ontology for reasoning about circuits at the microscopic level. The CC ontology uses the vocabulary such as field, electric force, and velocity to reason about circuit behaviors. Figure 14 depicts the two ontological choices in ARC. Neither the device ontology nor the CC ontology is parasitic to each other. Tasks can be carried out independently in either ontology.

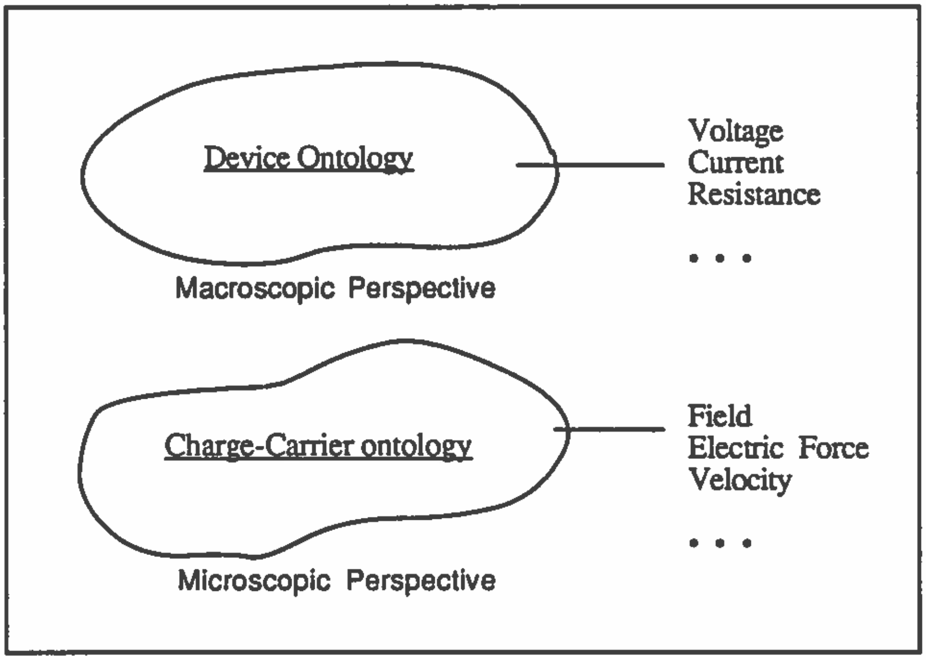


FIGURE 14. Macroscopic and Microscopic Perspectives on Electronics

When a tasks requires both the macroscopic and microscopic levels of reasoning, the ability to switch between the two ontologies to describe the same target system can be critical. ARC uses a set of domain-independent ontological-choice rules to govern the process of ontological shift, via bridging relations. The input and output specifications in a task definition indicate the input and output ontologies, which prompt ARC to select appropriate ontological choice to build QDE models and switch the models, if needed, during reasoning to carry out the given task.

A QDE model embodies a perspective that reflects choices made on all three dimensions of topological configuration, structural aggregation, and ontological choice.

The techniques for generating QDE models and shifting ontological perspectives are discussed in detail in Chapter V.

Below, we show one example to illustrate ontological shift by ARC.

Ontological choice rule: If the input and the output variables are of the same ontology and the analysis requires justification for one of the axioms of the ontology, then shift to a related ontology for reasoning. Otherwise, select the given ontology.

Axioms in any ontology can not be derived in the same ontology. Consider the task: "Explain the behavior of current through a resistor when the voltage across it increases?" This task directly questions the component model of a resistor in the device ontology. Since a component model contains primitive axioms of the ontology, which cannot be derived in the same ontology, this is when an ontological shift must be initiated by following one of the bridging relations to a related ontology which can explain it.

For example, given the following task about the black-box circuit (see Figure 15),

- Target System: [va-circuit3]
- Input Specification: (voltage (p n) (+ +))
- Output Specification: (current (p n)).

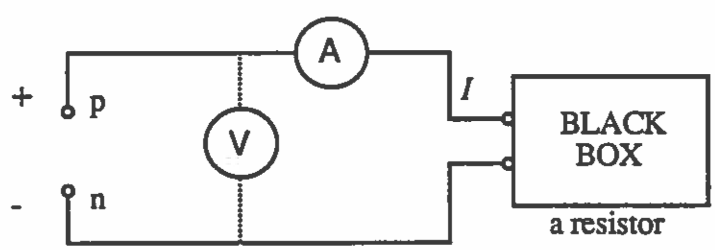


FIGURE 15. Black-Box Circuit

Since this task asks about one of the axioms in the device ontology, ARC shifts between the device ontology and the CC ontology, as illustrated in Figure 16, generating the following explanation:

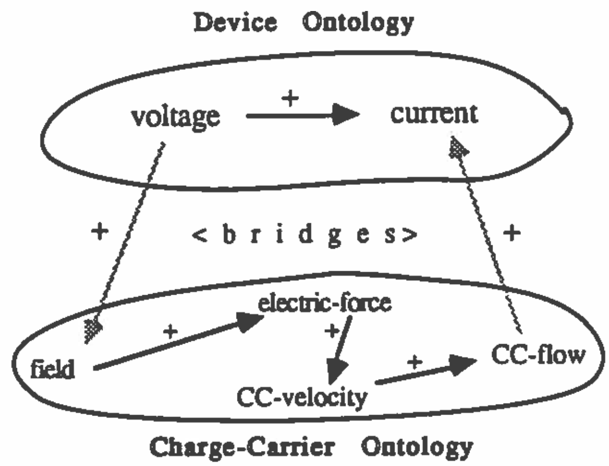


FIGURE 16. Shifting Perspective in the Dimension of Ontological Choice

** Begin with device-ontology **

- For black box blk.box1 between nodes (p n),
  since voltage@n remains constant,
  voltage@p's increase causes voltage@p_n to increase.

** Cross bridge to charge-carrier-ontology **

- In the region between nodes (p n),
  voltage@p_n's increase is equivalent to field@p_n's increase.
- In the electric field between nodes +p and -n,
  field@p_n's increase causes force-on-cc@p_n to increase.
- force-on-cc@p_n's increase causes cc-velocity@p_n to increase.
- cc-velocity@p_n's increase causes cc-flow@p_n to increase.

** Cross bridge back to device-ontology **

- In the region between nodes (p n),
  cc-flow@p_n's increase means current@p_n's increase.

Note that when ontological shifts take place in the above example, the concept of <u>regions</u> is used as a compatible structural description for both the device ontology and the CC ontology. By referring to a common structural view during ontological shift, spatiotemporal continuity of causal reasoning is maintained. We will address the significance of regions as a compatible structural description in the following chapters.

<u>Intertwining Perspectives — An Example</u>

In the above discussion, we have shown the individual cases of perspective-taking. Given a task, ARC integrates three different dimensions of perspective-taking to carry out the task. To wrap up this overview of ARC, the following example shows a complete (verbatim) script generated by ARC to carry out a given task. Reasoning from structure to behavior, it formulates models embodying appropriate perspectives on the fly to carry out the task. The specific data structures in the script will be explained in the following chapters. As the output is largely self-explanatory, we omit further narrations.

```
* ◇◇◇◇◇◇◇◇◇◇◇◇ * ◇◇◇◇◇◇◇◇◇◇◇◇ *
   Analytic Task to be Performed:
   • Target System:        [half-wave-rectifier1]
   • Input Specification:  (voltage (a d) (+ +))
   • Output Specification: (cc-flow (c d))
* ◇◇◇◇◇◇◇◇◇◇◇◇ * ◇◇◇◇◇◇◇◇◇◇◇◇ *
   The topology of the target system: [half-wave-rectifier1] is:
   ((a (terminal1 diode1))
    (b (terminal2 diode1) (terminal1 resistor14))
    (c (terminal2 resistor14)
       (terminal1 capacitor1)
       (terminal1 resistor24))
```

(d (terminal2 capacitor1) (terminal2 resistor24)))

* <><><><><><><><><><><> * <><><><><><><><><><><> *

ARC recasts (clusters & orients) it as follows:

((s.cluster5 ((a end1) (d end2))
    ((diode1 ((a terminal1) (b terminal2)))
     (resistor14 ((b terminal1) (c terminal2)))
     (p.cluster3 ((c terminal1) (d terminal2))
        ((resistor24 ((c terminal1) (d terminal2)))
         (capacitor1 ((c terminal1) (d terminal2))))))))

* <><><><><><><><><><><> * <><><><><><><><><><><> *

•• For the present input signal: (voltage (a d) (+ +)),
[half-wave-rectifier1] has the following configuration:
((s.region5 (a d)
    ((diode1 (a b))
     (resistor14 (b c))
     (p.region3 (c d)
        ((resistor24 (c d)) (capacitor1 (c d)))))))

•• For output (cc-flow (c d)), ARC views the system as follows:
((s.region5 (a d) ((blk.box7 (a c)) (blk.box6 (c d)))))
: blk.box7 = ((s.region8 (a c) ((diode1 (a b)) (resistor14 (b c)))))
: blk.box6 = ((p.region3 (c d) ((resistor24 (c d)) (capacitor1 (c d)))))

* <><><><><><><><><><><> * <><><><><><><><><><><> *

** Begin reasoning in device-ontology **

• In the serial region, s.region5, between nodes (a d),
since voltage@d remains constant,
we deduce that voltage@a's increase causes voltage@a_d to increase.

- Also, in s.region5,

  since resistance@s.region5 remains constant,

  it is clear that voltage@a_d's increase causes current@a_d to increase.

- current@a_d's increase means current@a_c's increase because they two originate from the same source & share the same path in the configuration.

- According to KCL with regard to node c for a serial connection,

  we know that current@a_c's increase leads to the increase of current@c_d.

- For black box blk.box6 between nodes (c d) containing

  ((p.region3 (c d) ((resistor24 (c d)) (capacitor1 (c d)))))

  since resistance@blk.box6 remains constant,

  current@c_d's increase causes voltage@c_d to increase.

  ** Cross a bridge to charge-carrier-ontology **

- In the region between nodes (c d),

  voltage@c_d's increase is equivalent to field@c_d's increase.

- In the electric field between nodes +c and -d,

  field@c_d's increase causes force-on-cc@c_d to increase.

- force-on-cc@c_d's increase causes cc-velocity@c_d to increase.

- cc-velocity@c_d's increase causes cc-flow@c_d to increase.

* ◇◇◇◇◇◇◇◇◇◇◇◇ * ◇◇◇◇◇◇◇◇◇◇◇◇ *

Summary

In this chapter, we have presented an overview of computational framework, ARC, based on the task-driven perspective-taking approach to reasoning about electronic circuits. To carry out a perturbation-analysis task by reasoning from structure to behavior, ARC first identifies the active configuration from the system topology of a circuit. Focusing on the output structural unit specified in the given task, ARC then re-casts the active configuration into composite structural units by structural aggregation that suppresses irrelevant details for the task at hand. Finally, ARC selects proper sets of vocabularies to generate QDE models to describe the circuit's behavior as a result of the input perturbations. We have shown that perspective-taking is task-driven. That is, different models of the same target system can be formulated from appropriate perspectives for different tasks.

In the following chapters, we describe in detail the techniques to automate model formulation and selection with appropriate perspectives of a target system to carry out a given task.

CHAPTER III

IDENTIFYING TOPOLOGICAL CONFIGURATION

This chapter focuses on how to identify active configurations from the system topology of a circuit under analysis. As we pointed out in the last chapter, the system topology of a circuit is a direct mapping from the schematics of the circuit. It gives the information of how the component devices are connected, via nodes, into a device network. It remains unchanged during qualitative causal analysis.

In this chapter, we first discuss the search problem in identifying an active configuration of a circuit in a specific qualitative state and present the clustering technique for revealing the structural hierarchy in the system topology of the given circuit. We then describe the representation of configurations by regions, a chosen structural description for both the macroscopic and the microscopic perspectives of electronic circuits. Finally, we present and analyze the configuration algorithm that searches the oriented clusters to identify the flow path.

A Graph-Search Problem

Identifying configurations in the system topology of a circuit can be viewed as a graph search problem. The process searches for the active component devices on the current flow path in the system topology of a circuit.

Search Complexity Based on Individual Devices

One way to decide whether a device is on the current flow path in a system topology is by checking whether there is a voltage difference across the device in the system topology and, if so, whether the device allows a current flow as implied by the voltage difference[8]. If it does, then the device is part of the configuration and the search records the voltage drop and proceeds to the rest of the devices. The final set of all the active devices thus identified should connect current flow starting from the positive pole (source) all the way to the negative pole (sink) in the system topology of the circuit.

For example, if the voltage difference across a resistor is not zero, then this resistor conducts current and is included in the active configuration. For other types of devices like diodes and capacitors, the search also needs to check if the anode and cathode are connected to the proper nodes for the voltage difference or if the voltage's first-order derivative is zero or not.

The process of identifying an active configuration of a circuit based upon checking individual devices in the system topology according to voltage difference seems fairly straightforward. But the complexity of this search algorithm turns out to be exponential because the search is based on individual component devices and does not see the "big picture" of the circuit.

Consider the task of identifying the active configuration of the ladder circuit (see Figure 17) to find the current flow path.

- Target System: [ladder-circuit4]
- Input Specification: (voltage (a h) (+ +))
- Output Specification: (voltage (b c))

The application of a non-zero voltage between nodes *a* and *h* start the search from node *a*, the positive pole. Initially, all the nodes' voltage is zero. The problem is that when the search comes to a node whose degree is more than two, the node becomes a potential backtracking point. We call such nodes <u>fan-out nodes</u>. In this case, nodes *b, c, f, g* in the

[8] Since there is no difference between using either the device ontology or the charge-carrier ontology for identifying configurations, we will use the device ontology for the sake of discussion in this chapter.

circuit are fan-out nodes.

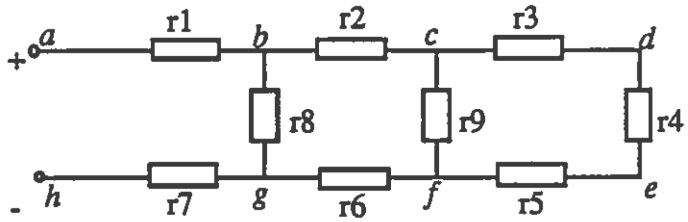


FIGURE 17. A Ladder Circuit

For lack of other knowledge except voltage difference, the choice of which path to take at a fan-out node can only be arbitrary. For example, when search comes from node *a* to node *b*, it may search the path from *b* to *g* by checking *r8* and determines that *r8* is active. Suppose the search takes this path and comes to node *g*. By the same token, the search may choose the path from *g* to *f*, and then from *f* to *c*. At node *c*, it finds that *r2* is active because voltage at node *b* is greater than that of node *c* since voltage at *c* is supposed to have dropped through *r8*, *r6*, and *r9* along the search path.

It is obvious that the path generated is wrong, but the search process does not know it yet. At node *c*, the search proceeds on the path from *c* to *d*, from *d* to *e*, and identifies *r5* to be active. *r5* is active because the voltage at *f* is recorded greater than that at *e*.

Now, a detectable conflict occurs at node *e* because current only flows into *e*, a clear violation of Kirchhoff's Current Law. Something is wrong on the way leading to this conflict. The only places that could contribute to the problem are the fan-out nodes. The search must backtrack to the nearest fan-out node on the search path and try a different choice and proceed from there. If it fails again, backtracks takes place. Obviously, the complexity of the search algorithm based on checking individual component devices is exponential,

$$O(D^n),$$

where D stands for the average degree of fan-out nodes and n for the number of such nodes in the system topology of the circuit under analysis.

The Need for "Big Pictures"

One consequence of the exponential search complexity is that the search for active configurations is slow and inefficient. One may suggest to increase the speed of search by using a fast machine. But this would totally miss the point.

Even if someday a look-ahead program with enough brute force running on a fast machine will overcome the speed problem, it would be a small intellectual gain, compared to the revelation that intelligence depends crucially on the ability to abstract and see the big picture of complex arrays of things. This ability is inherent of expert problem-solving. For example, master chess players not only see individual chess bits on board, but also clusters of bits or patterns for planning winning strategies [Hofstadter 1980]. Likewise, electrical engineers see high-level structural units consisting of a group of devices to simplify circuit analysis [Rusgrove et, al. 1977, Sussman & Steele 1980]. Grouping individual components into high-level descriptions in such a way provides the big picture of the target system for efficient problem-solving.

The problem of the exponential search complexity based on individual component devices is due to representing the system topology as a simple graph, where every device is viewed as on the same level. Such a representation effectively shows the connectivity of the network but totally misses the structural hierarchy whereby the component devices are organized in the actual system.

Seeing the structural hierarchy of a system topology requires expertise. And it is advantageous to the process of topological configuration. Below, we show two techniques used by ARC to reveal the big picture of a circuit under analysis. The first is cluster creation. The second is cluster orientation. We show that this cluster-based reasoning helps solve the inefficiency problem involved in configuration identification.

Clustering to Reveal Structural Hierarchy

Cluster creation in ARC is the process that makes the implicit structural hierarchy in the given system topology of a circuit explicit by aggregating parallel and serial clusters in the device network. Circuits with series-parallel constructs are the norm. For these circuits, the two rules used for identifying parallel-serial constructs are as follows:

Serial Clustering: If there is a degree-two node (except the pole nodes) in the system topology, then the two constructs connected to this node are in series. These two constructs can be merged into one serial cluster.

Parallel Clustering: If two constructs are connected to the same two nodes in the system topology, then the two constructs are in parallel. These two constructs can be merged into one parallel cluster.

When a serial cluster is built, the number of nodes and the number of edges in the graph each decrease by one because the two clusters merge into one. When a parallel cluster is built out of two clusters, the number of nodes remains the same, but the degree of each of the two nodes connected by the two merging constructs decreases by one. As a result of the clustering process, the original graph at the given level is simplified. But the knowledge of the circuit structure has increased because the graph now becomes multi-layered, which shows the implicit structural hierarchy. At each level of clusters, the graph has less nodes and less edges than the original graph of the system topology.

With explicit structural hierarchy revealed, ARC has gained more knowledge to guide the search for configurations. For example, when the search comes to node *b* in the ladder circuit (see Figure 18), it has the information that it has come to a parallel construct, where each branch of the construct can be searched simultaneously. In this case, the parallel construct has two branches: (1) a single resistor *R8*, and (2) a cluster which involves serial and parallel connections. As a result, the backtracking problem is avoided.

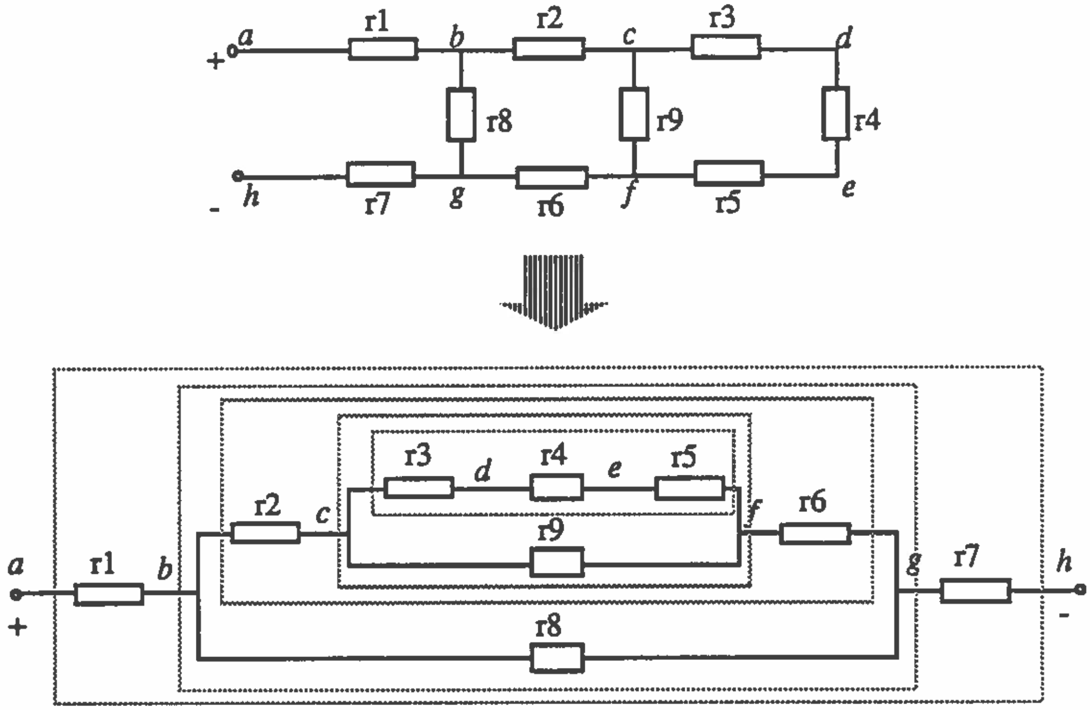


FIGURE 18. Hierarchical View of Ladder Circuit by Clustering

Clustering Procedure

The input to the clustering process is the system topology of a target circuit. The two nodes, as specified in the input of a given task, are marked as the pole nodes. Clustering centers around the two pole nodes. For the same system topology, different pole nodes define different clusters for the circuit. The polarities of the pole nodes are immaterial to the clustering process because the clustering process itself does not identify active configurations but only reformulates the system topology into its hierarchical structure.

In the clustering process, parallel and serial clustering are intertwined. For example, with the ladder circuit, *r3*, *r4*, and *r5* are first chunked in a serial cluster. This serial cluster and *r9* then form a parallel cluster which, in turn, links *r2* and *r6* into a serial cluster, and so on. A high-level description of the clustering algorithm[9] is shown below.

---

[9] The current implementation of this clustering algorithm takes $O(n^3)$. With canonical partial k-tree representation [Arnborg and Proskurowski 1989] for the system topology of a target circuit, this algorithm can be implemented in $O(n)$.

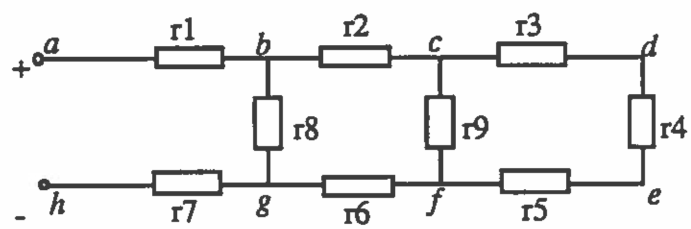
```
Start-clustering (system-topology)
   ST := S-cluster (P-cluster (system-topology));
   if ST = system-topology then
      return(ST)
   else
      Start-clustering(ST);
S-cluster(ST):
  N := Find-a-degree-2-node-in(ST);
  if N = nil then
     return(ST);
  else
     TC := Get-the-two-constructs-connected-to-node(N, ST);
     SC := Make-a-series-cluster(TC);
     ST := Replace-the-node-and-the-2-constructs-with(SC, ST);
     S-cluster(ST);
P-cluster(ST):
  TC := Find-a-pair-constructs-sharing-the-same-two-nodes(ST);
  if TC = nil then
     return(ST);
  else
     PC := Make-a-parallel-cluster(TC);
     ST := Replace-the-2-constructs-with(PC, ST);
     P-cluster(ST);
```

Clusters Representation

Cluster representation is hierarchical in the usual sense: each cluster may have sub-clusters. The data structure for a cluster is recursively defined as follows:

(cluster-name (($node_i$ terminal1) ($node_j$ terminal2)) (cluster-body)),

where (node terminal) pair specifies the connection of the cluster's terminals to nodes in the system topology of a circuit, and cluster-body is a list of constructs, each of which is a cluster. If cluster-body is empty, then the cluster stands as a single device. Cluster-name has a attribute about whether it is a serial or a parallel cluster. For example, given that nodes (*a h*) are the two pole nodes in the ladder circuit, ARC recasts the system topology into a cluster-representation as shown in Figure 19.




```
((a (terminal1 r1))
 (b (terminal2 r1) (terminal1 r2) (terminal1 r8))
 (c (terminal2 r2) (terminal1 r3) (terminal1 r9))
 (d (terminal2 r3) (terminal1 r4))
 (e (terminal2 r4) (terminal1 r5))
 (f (terminal2 r5) (terminal1 r6) (terminal2 r9))
 (g (terminal2 r6) (terminal2 r8) (terminal1 r7))
 (h (terminal2 r7)))
```

⇓

```
((s.cluster10 ((h end1) (a end2))
     ((r7 ((g terminal1) (h terminal2)))
      (r1 ((a terminal1) (b terminal2)))
      (p.cluster8 ((g end1) (b end2))
           ((s.cluster7 ((g end1) (b end2))
                ((r6 ((f terminal1) (g terminal2)))
                 (r2 ((b terminal1) (c terminal2)))
                 (p.cluster5 ((c end1) (f end2))
                      ((s.cluster4 ((c end1) (f end2))
                           ((r3 ((c terminal1) (d terminal2)))
                            (r4 ((d terminal1) (e terminal2)))
                            (r5 ((e terminal1) (f terminal2)))))
                       (r9 ((c terminal1) (f terminal2)))))))
            (r8 ((b terminal1) (g terminal2))))))))
```

FIGURE 19. Recasting System Topology of a Circuit into Hierarchical Clusters

In the next section, we show how the clustering process picks up more information along with merging clusters by assigning each resulting cluster an orientation.

Cluster Orientation Propagation

Orientations as "Road Signs"

In the ladder-circuit example, suppose R6 is replaced by a diode (*d1*) in the ladder circuit as shown in Figure 20. The search may proceed, identifying *r1*, *r2*, *r3*, *r4*, *r5*, and *r9*, till it comes to the diode which it finds shuts off this path. All the efforts spent for *r2*, *r3*, *r4*, *r5*, and *r9* are thus wasted.

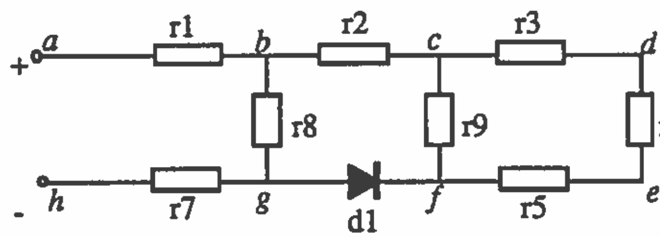


FIGURE 20. Ladder Circuit with Orientation

Wasted search should be avoided. The search should have the information that the cluster on the right-hand side of node *b* is a dead-end path for any signal trying to travel from *b* to *g* through the cluster. This cluster's orientation is due to the orientation of the diode in the cluster.

The orientation of this diode should be propagated during the clustering process when it is merged with other constructs. The orientation then acts like "road signs" for the signals travelling inside the system. With such a road sign for each cluster, the search first checks the orientation of a cluster to avoid blind search. For example, if there had been a sign "Wrong Way" posted for the cluster in the above case, it would have saved exploring the dead-end path. If one changes the polarity by assigning node *h* as the positive node and node *a* as the negative node, then the cluster will be a good flow path. Thus, the orientation of a cluster is described as how the cluster is oriented, with respect to nodes, in the system topology of a circuit.

In general, individual devices[10], such as resistors, diodes, capacitors, etc, have their own primitive orientations. For example, a diode's orientation is from its anode to its cathode, which means that current can only flow from the node connected to the anode to the node connected to the cathode in the circuit. The other direction constitutes an impasse. A resistor's orientation is more flexible because it allows current to flow in either direction. A capacitor's orientation is a bit different. It depends on dV/dt (voltage) applied to the capacitor. Only when dV/dt is not zero is the capacitor considered conducting current. Otherwise, it constitutes an impasse. When individual devices with primitive orientations are merged into a new cluster, the resulting orientation exhibit a variety of possibilities. Table 2 shows the set of orientations examined in the present implementation.

TABLE 2. Types of Orientations for Clusters

Group	Instances
Primitive Orientation	bi-directional, forward, backward, capacitive.
Combined Orientation	bi-directional, forward, backward, capacitive, forward-capacitive, backward-capacitive, impasse.

When building either a series or a parallel cluster, the assignment of the orientation of the cluster depends on its components' orientations with regard to their connectivity in the system topology of a circuit. Note that in the clustering algorithm above, a cluster is built

[10] We assume ideal devices in the discussion.

based on merging a pair of relevant clusters. The pair-wise aggregation into a new cluster allows the new cluster's orientation to be assigned from those of the merging ones in a straightforward manner.

Nondirected Orientation

When building a serial cluster out of two constructs, if one construct's orientation is bi-directional, then the new serial cluster takes the orientation of the other construct. When building a parallel cluster out of two constructs, if one construct's orientation is bi-directional, then the new parallel cluster is bi-directional. A bi-directional orientation, therefore, yields in a serial clustering but dominates in a parallel clustering.

Similarly, when building a series cluster out of two constructs, if one construct's orientation is an impasse, then the new serial cluster is an impasse. When building a parallel cluster out of two constructs, if one construct's orientation is impasse, then the new parallel cluster takes the orientation of the other construct. Contrary to the bi-directional orientation, an impasse orientation dominates in a serial clustering but yields in a parallel clustering.

Different from the above two cases, when a capacitive construct is involved in building a cluster (except in the above two cases), it always adds its capacitiveness to the other construct's orientation in the resulting cluster's orientation like forward-capacitive and backward-capacitive. In this case, the configuration process not only checks the directionality, but also the capacitiveness of the cluster to determine whether it can be part of the current flow path.

Directed Orientations

When both constructs have directed orientations, the resulting cluster's orientation is a bit more involved. We divide the discussion into two parts: serial-cluster orientation, and parallel-cluster orientation. We use the following notations to illustrate the orientation propagation rules, as shown in Figures 21 and 22. The arrow associated with each pair of nodes in the figures indicates a cluster's orientation with respect to its connection to the two nodes. Below each combination is the new cluster with its orientation. An *X* for the new cluster's orientation means it is an impasse. A straight line means it is bi-directional.

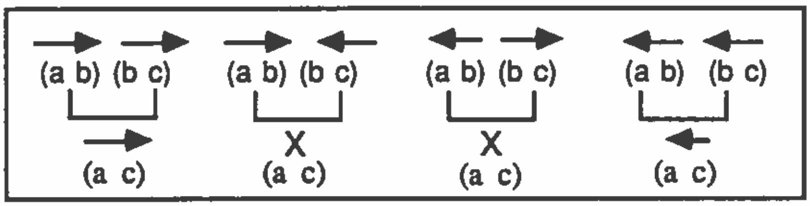


FIGURE 21. Directed Orientation in Serial Clusters

Serial Clustering

If a serial cluster is to be built out of two oriented clusters, then there must be a node of degree two connecting to the two constructs. Suppose the first construct is connected to nodes *a* and *b* and the second construct is connected to nodes *b* and *c*. The resulting orientation of the new serial cluster is depicted in Figure 21. Note that when the two clusters' orientation are opposite to each other, the new cluster orientation is an impasse.

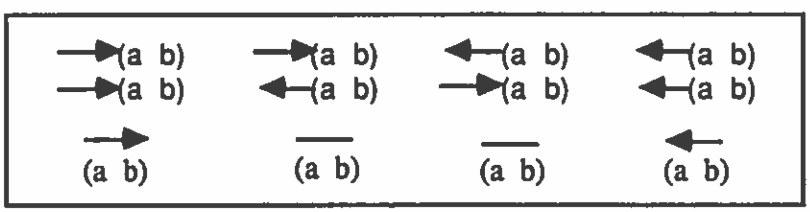


FIGURE 22. Directed Orientation in Parallel Clusters

Parallel Clustering

If a parallel cluster is to be built out of two oriented clusters, then there must be two constructs sharing two nodes. Suppose both construct are connected to nodes *a* and *b*. The resulting orientation of the new parallel cluster is depicted in Figure 22. Note that when the two clusters' orientation are opposite to each other, the new cluster orientation is bi-directional.

The clustering process is based on the pair-wise combination for both parallel and serial clusters. This pair-wise combination helps propagate the orientation to high-level clusters. To show that the final orientation of a cluster is correct regardless of the order of pair-wise combinations, all we need to show is that each pair-wise combination, serial and parallel, is correct. Since each pair-wise combination is correct, resulting in a new cluster, any pair-wise combination with this cluster following the orientation propagation rules is also correct.

The orientation determination during the clustering process does not affect the complexity of the clustering algorithm since it takes constant time. Orientation of a cluster is determined and assigned when two sub-clusters are merged into a new cluster. The outcome of the clustering process is an oriented-cluster representation of the system topology of the target circuit.

Configuration via Cluster Activation

After the system topology is recast into oriented clusters, the search for active configurations is made easy because the original system topology is now structured and oriented. When search comes to any node, it not only uses the knowledge of voltage difference but also the additional knowledge of clusters and their orientations to identify the active configuration of the circuit under analysis.

Region Representation

To represent active configurations, we need to find a common structural description suitable for both the device ontology and the CC ontology reflecting the macroscopic and microscopic-level descriptions of electronic circuits. Notice that the basic structural elements in the device ontology are the component devices and their interconnections in the system topology of a circuit. In contrast, the basic structural elements in the CC ontology are electric fields and poles. In order to relate the two ontological perspectives and to preserve the spatiotemporal continuity of causal propagation during ontological shift, the two ontologies must have a compatible structural view of electronic circuits. ARC uses regions to provides this common structural view as shown in Figure 23.

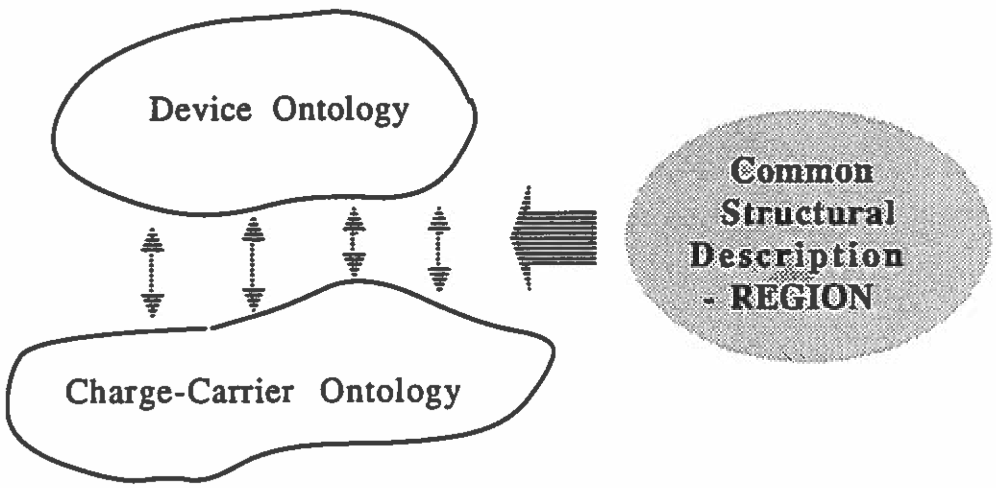


FIGURE 23. Regions as Conceptual Structural Units

Conceptually, a region refers to a structural unit between two chosen poles through which current, or movement of charge carriers is of interest. Regions are hierarchical in the usual sense: a region may consist of any number of sub-regions, connected either in series, or parallel, or mixed ways. We denote a region by this schema:

(region-name (p1, p2) (region-body)),

where p1 and p2 stand for the positive and the negative poles of the region, respectively. Region-body indicates a list of sub-regions. The relevance and transience of polarity is captured in a straightforward manner in region notations. Similar to a cluster's representation, when region-body is empty, the region stands for a single device. Different from clusters which only show the static structural hierarchy of a circuit, regions are used to show the flow path in an active configuration. Region is an important concept in ARC. We will see its use throughout the rest of this dissertation.

Activation Procedure

We now discuss the algorithm for identifying the active configuration from the oriented-cluster representation of a circuit in a specific qualitative state. The process tries to activate the static clusters into active regions for current or charge-carrier movement. The regions retain the same structural hierarchy as clusters if activated. Note that the search for the active configuration proceeds at the cluster level. At any particular level, the lower level details are suppressed. If a cluster is determined not active, then its sub-structure can be safely ignored.

A high-level description of the configuration algorithm based on oriented clusters is shown below. In the algorithm, S-cluster? and P-cluster? are two predicates to test the type of a cluster (serial or parallel). If a cluster is neither of the two types, then it is a single-device cluster. Active? returns true if the cluster allows current flow. The test is done by checking first the orientation of the cluster and then the voltage difference for the possible current flow. If a cluster is determined active, then its sub-clusters are activated by procedures S-activate or P-activate depending on the type of the cluster.

```
Activate (cluster, src, signal)
  if  s-cluster?(cluster) and active?(cluster, src, signal) then
     sub-clusters := get-cluster-body (cluster);
     active-clusters := S-activate (sub-clusters);
     make-s-region (active-clusters);
  else
  if  p-cluster?(cluster) and active?(cluster, src, signal) then
     sub-clusters := get-cluster-body (cluster);
     active-clusters := P-activate (sub-clusters);
```

```
    make-p-region (active-clusters);
  else
  If active?(cluster, src, signal) then
    make-a-single-device-region (cluster);
  else
    return(nil);
```

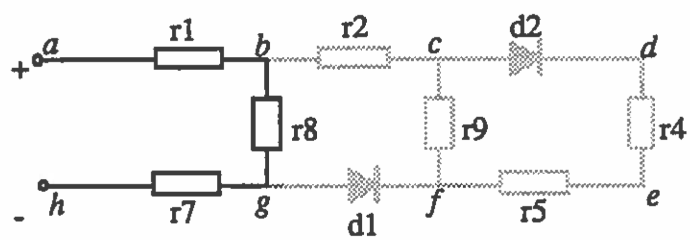
In <u>S-activate</u> and <u>P-activate</u> below, since a cluster body is represented as a list, we use two procedures <u>get-first-element</u> and <u>get-rest-elements</u> to retrieve the head and the tail of the list. To activate each sub-cluster in the list, procedure <u>Activate</u> discussed above is used.

```
S-activate (s-clusters, src, signal)
  If null?(s-clusters) then
    return (nil);
  else
    one-cluster := get-first-element (s-clusters);
    rest-clusters := get-rest-elements (s-clusters);
    one-rgn := Activate (one-cluster, src, signal);
    new-src := get-next-node (one-rgn, src);
    append (one-rgn, S-activate (rest-clusters, new-src, signal));
```

```
P-activate (p-elements, src, signal)
  If null? (p-elements)
    return (nil);
  else
    one-branch := get-first-element (p-elements);
    other-branches := get-rest-elements (p-elements);
    one-rgn := Activate (one-branch);
    append (one-rgn, P-activate (other-branches, src, signal));
```

With the explicit structural hierarchy revealed, the complexity of search for an active configuration of a circuit becomes polynomial. This avoids the exponential search complexity when search is based on individual devices in the system topology of the circuit. Although the configuration process incurs an overhead of building oriented clusters, the clustering process, whose complexity is linear, is done only once for a given pair of pole nodes in the circuit topology.

The output of this search is the active configuration represented as regions. Being a directed subgraph of the system topology, it involves interconnected devices that are on the current flow path in the given qualitative state of the circuit. For example, the configuration of the ladder circuit with nodes *a* and *h* as the positive node and negative node, respectively, is shown in Figure 24.



(s.region5 (a h)
((r1 (a b))
(r8 (b g))
(r7 (g h))))

FIGURE 24. Representation of Active Configuration as Regions

The configuration of this example is a serial region consisting of three sub-regions — *r1*, *r8*, and *r7*. The serial region's polarity (*a h*) shows that current flows from node *a* to node *h*. The three sub-regions' polarities show how the flow path is connected from *a* to *h* — (*a b*) (*b g*) (*g h*). The sub-regions in this case are all single-device regions.

Note also the differences between clusters and regions. Clusters are static, showing all the structural constructs in the system topology. Regions are dynamic, showing only the active flow path with a subset of the component devices in the system topology. A configuration in regions shows explicitly the polarities of each construct while a cluster representation does not. When the external polarities change with the pole nodes, the cluster representation remains the same, but the configuration representation in regions will change.

Configuration Shift in Qualitative Simulation

To advance qualitative simulation techniques which use only a single perturbation for small signal analysis, ARC allows a sequence of perturbations of arbitrary length. The advantage of this is that the processing of those signals early in the sequence and early in time produces a context for the later signal processing. This context involves a history [Hayes 1985] of the system behavior which, together with the incoming signal, directly accounts for what the behavior is going to be next. The critical changes that may occur with a sequence of perturbations is that the active configuration switches to a different subset of individual devices in the system topology of the circuit under analysis.

For linear-resistive systems with a sequence of signals, there is generally only one configuration for a given pair of pole nodes. For systems with nonlinear devices, there may be a set of different configurations for a given pair of pole nodes. If they do not contain any storage devices, such as the full-wave-bridge rectifier in Figure 25, there may be more than one configuration, but always the same cluster representations. Otherwise, there can be several configurations and several cluster representations because of the possible change of actual pole nodes.

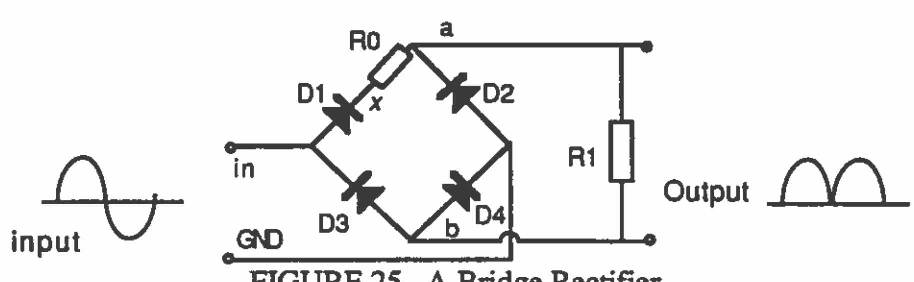


FIGURE 25. A Bridge Rectifier

For example, the charged capacitor in the half-wave rectifier we have seen above becomes the actual power source with pole nodes (*c d*) when the diode is reverse-biased. In this situation, ARC dynamically generates clusters and configurations along with a sequence of input perturbations.

Handling Ambiguous Situations

Qualitative representation of continuous quantities can be ambiguous. For example, Qval(V) = + says $V > 0$. Sometimes, this representation is not adequate. For example, when one gets

$$Qval(V_1) = +, \ \& \ Qval(V_2) = +,$$

it could mean any of these three cases:

$$V_1 < V_2, \quad V_1 = V_2, \quad V_1 > V_2.$$

This clearly indicates a need for quantitative information to disambiguate the situation.

When quantitative information is not available, the traditional approach in qualitative physics is to generate an envisionment [de Kleer 1977], an explicit representation of all possible behaviors. Though a powerful theoretical tool, envisioning solves problems by explicitly generating the entire search space, intentionally ignoring control issues [Forbus 1988]. The disadvantage is that it could generate behavior not realizable in the real situation or not of interest to the particular task at hand.

ARC handles the ambiguity problem of qualitative simulation by turning the control over to the user so that only the behavior interesting to the user is examined. For example, suppose one gives a sequence of signals representing a sinusoidal input to the half-wave rectifier circuit. When the capacitor in the rectifier is charged, it has a positive voltage. When the input's voltage is also positive, ARC asks the user to make a choice as in Figure 26.

The choice made by the user prompts ARC to focus on the behavior of interest to the user. The possible input perturbation (Qval Qdir), whose Qval is positive, can be one of the following:

a. (+ +), b. (+ 0), c. (+ -).

Suppose the present input perturbation is (+ +). Given this, if the user takes choice (1), the situation is that the input's voltage is greater and it is still increasing (Qdir = +). It is obvious that the input prevails. In this case, the capacitor is further charged.

* There is an ambiguous situation here because Capacitor1 is charged & the voltage@a is positive. (Quantitative information is needed to proceed)

In this situation, the following three cases are possible:

(1) voltage@a is greater than the voltage of Capacitor1.
(2) voltage@a is less than the voltage of Capacitor1.
(3) the two quantities are equal.

Your choice (1, 2, or 3):

FIGURE 26. Turning Control of Qualitative Simulation to the User

If the user picks choice (2), then the capacitor prevails and is discharging through the load resistor as an RC circuit. But the input is increasing. It is possible that the input voltage could soon equal the capacitor's voltage or surpass it the next moment. In this case, the circuit may undergo transitions from one configuration to another as shown in Figure 27.

Note the shift of configuration models on the right side of Figure 27. ARC generates these configuration models and dynamically switch among them when the qualitative state of the circuit changes.

If the user chooses choice (3), then the input voltage is equal to the capacitor's voltage. But the input is rising while the capacitor is not. This means the input instantly prevails, and the circuit undergoes configuration transitions. The value at which the two quantities are equal is a landmark value for configurations. Crossing it in either direction causes the configuration to switch.

For the other two cases with perturbations (+ 0) and (+ -), the similar situations occur.

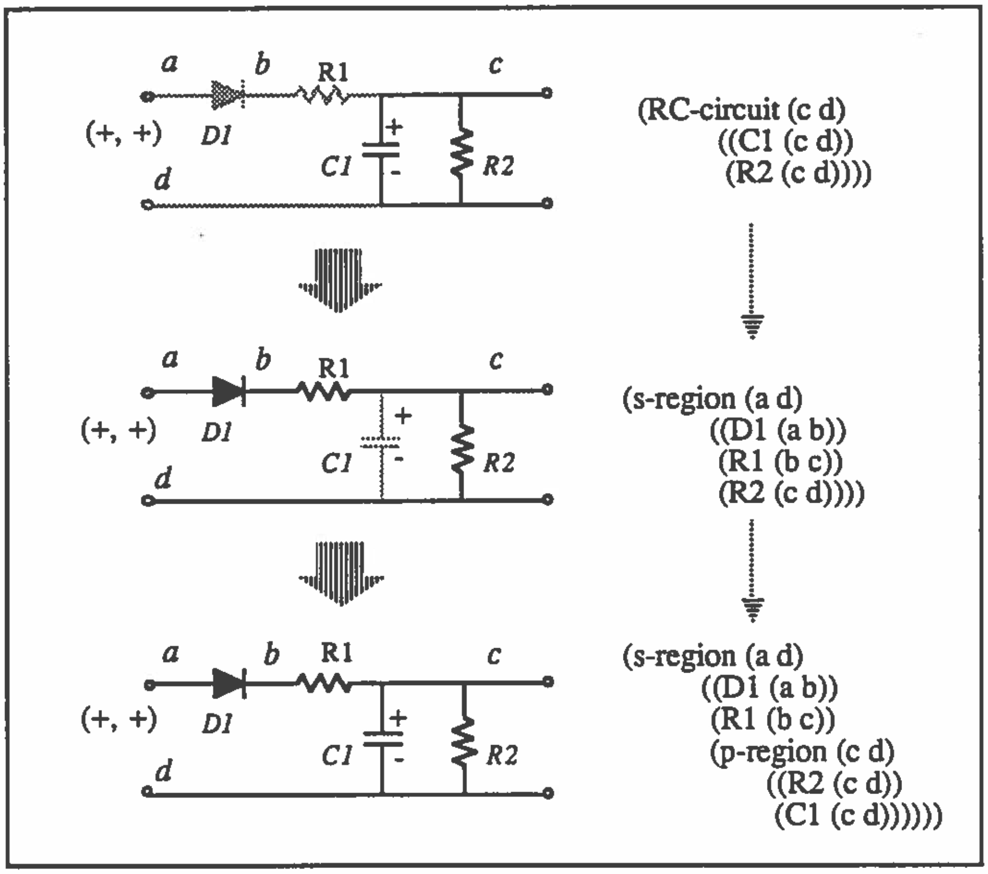


FIGURE 27. Shifting Perspective in the Dimension of Topological Configuration

Summary

In this chapter, we have discussed how to identify active configurations of a circuit under analysis in a given qualitative state. We started by pointing out that the structure of a physical system has two distinct aspects — the static system topology, and the dynamic configurations. An active configuration of a circuit lets our program focus on a subset of the system topology consisting of only the active devices on the flow path. To identify an active configuration of a circuit, we presented techniques to recast the system topology of the circuit into oriented clusters to reveal an hierarchical view of the circuit structure. We then presented a configuration activation algorithm based on oriented clusters. We introduced regions as a common structural view between both the device ontology and the CC ontology of electronic circuits. Finally, we discussed how ARC handles ambiguous situations in identifying the active configurations with a sequence of perturbations.

Several benefits are obtained by identifying active configurations:

- It analyzes why current does or does not exist in certain parts of a circuit using only qualitative knowledge, providing a zero-order explanation.
- It simplifies subsequent analysis by determining the flow path and the polarities of component devices' connections in the system topology.
- A nonlinear circuit is re-cast as a set of configurations to which linear analysis methods can be applied. This extends the range of existing qualitative causal reasoning techniques.

In the next chapter, we discuss how to aggregate component devices in an active configuration into black boxes in order to further simplify the models of a circuit for a given task.

CHAPTER IV

GENERATING STRUCTURAL AGGREGATION

In the previous chapter, we have discussed the process of dynamic topological configuration, which shows the active flow path in a given situation. The process starts by aggregating individual devices in the system topology into oriented clusters and then searches the oriented clusters for an active configuration. There, structural aggregation into clusters helps improve efficiency in identifying active configurations because at any given level, the lower-level detail can be ignored. This chapter investigates a different aspect of structural aggregation: creation of black boxes that suppress extraneous details for a given task at hand.

Structural Aggregation as Encapsulation

Suppressing Irrelevant Details

To date, qualitative modeling techniques have developed around various means for abstracting the value spaces of parameters that are used to represent system qualitative states and for simplifying the constraints that hold among those parameter values [Farley 1988]. Few approaches have attempted to automate abstracting complex structural descriptions of physical systems to simplify their models for the task at hand. As a result, uninteresting or unnecessary behavioral detail has to be considered, producing more harm than benefit.

The standard technique in AI to reduce complexity is abstraction. Abstraction in modeling is generally regarded as a way to ignore details [Smith & Davis 1989]. But doing it incorrectly can prevent consideration of important behaviors. For many tasks, the mapping from a structural description to structural abstraction is the critical step in selecting the appropriate structural granularity and suppressing extraneous details.

In ARC, structural aggregation is not just to ignore all the details. Sometimes we need details. We only want to ignore those details which are unnecessary for the task at hand. By focusing on the details required by the task and encapsulating the rest, ARC achieves efficiency and flexibility in qualitative causal analysis.

Black-Boxing around Output Structural Unit

For task-driven perspective-taking, the information for creating black boxes comes from the output specification in a given task. For example, when a task asks about the behavior of voltage between two nodes in a circuit as a result of given input perturbations, the individual devices in between those two nodes comprise the output structural unit.

Structural aggregation centers around the output structural unit as specified in a task. Specifically, the individual devices in the output structural unit are aggregated into a black box and viewed as an individual entity for analysis. Any other individual devices that do not affect the causal relationship between the input perturbations and the desired output are also boxed. For different output structural units as specified in different tasks, different aggregations will be generated.

The procedure that creates black boxes takes two items as input: (1) the active configuration in regions, and (2) the two nodes that delimit the output structural unit. The outcome of this procedure is an equivalent circuit to the active configuration, with black boxes replacing the output structural unit and other uninteresting structural details. The use of black boxes in circuit analysis follows the equivalent circuit concept in electronic engineering [Bell 1980].

Take the circuit in Figure 28 for example. If one is interested in analyzing the behavior of current flow from *b* and *f* as a result of a perturbation to the voltage of the battery, as specified in the following task,

- Target System: [p-circuit4]
- Input Specification: (voltage (a g) (+ -))
- Output Specification: (current (b f))

then the individual devices, delimited by nodes *b* and *f*, *R4* & *R5*, constitute the output structural unit.

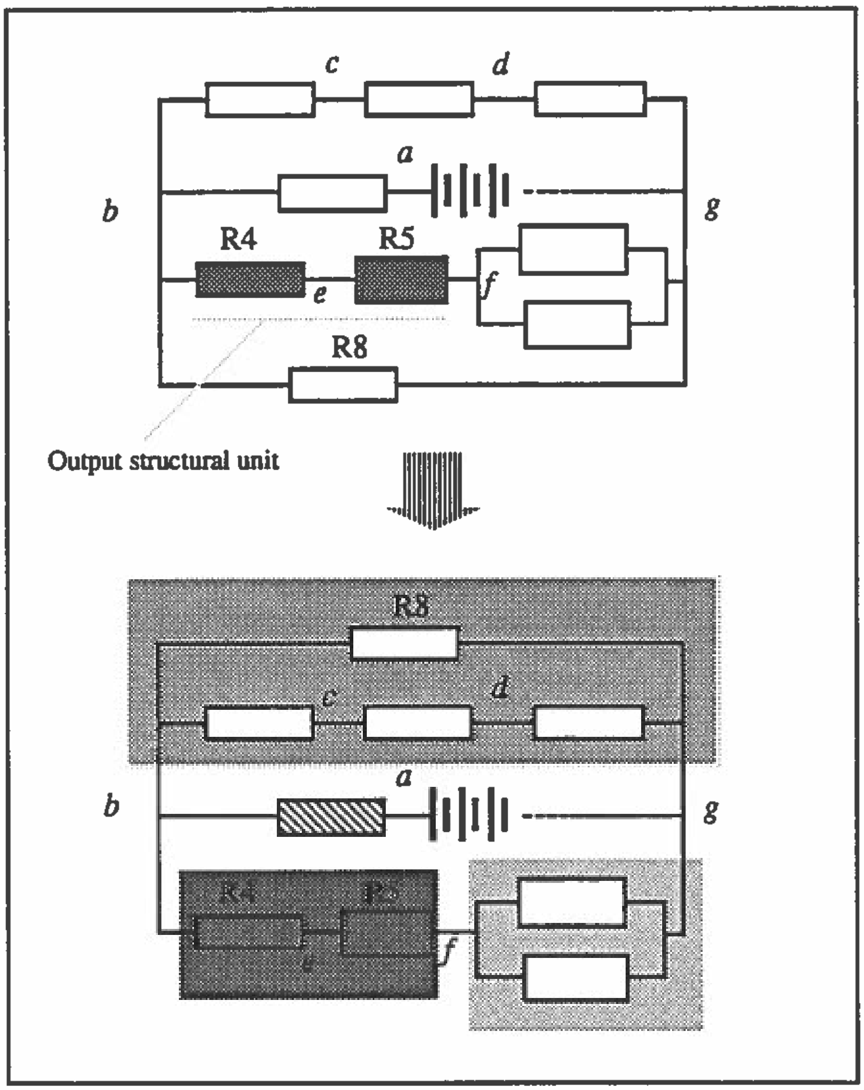


FIGURE 28. Creation of Black Boxes to Suppress Extraneous Detail

R4 and *R5* then will be aggregated as a black box. Furthermore, the other component devices irrelevant to this task are also aggregated as black boxes, as shown in the figure.

The resulting aggregated structural description of the circuit with the black boxes is equivalent to but much simpler than the original one given. The simplification is achieved without loosing important detail for the given task. As a result, qualitative causal reasoning based on the simplified structural description of the circuit becomes more efficient.

Below, we present the algorithm for creating black boxes to simplify models of a circuit structure for a given task. Note that structural aggregations at this stage are performed over the active configuration represented as regions. It is thus important to familiarize ourselves with the data structure of region before we dive into the discussion of the algorithm.

Procedures on Regions

To review, a region in an active configuration is a conceptual structural unit with two poles. Its represented is:

(region-name (p1, p2) (region-body)),

where region-name gives the name of the region. (p1 p2) specifies the positive and the negative poles of the region, respectively. Region-body indicates a list of sub-regions of the region. Note the apparent recursive definition of a region. If region-body of a region is empty, the region stands for a single device.

There are three types of regions in an active configuration — serial regions, parallel regions, and single-device regions. A region's type is recorded as a property of the region-name in the configuration process discussed in the previous chapter. Three predicates used to check the type of a region are S-region?(rgn), P-region?(rgn), and Single-device?(rgn), each of which returns true if rgn is of the indicated type. Otherwise, it returns false.

Procedures Get-region-poles(rgn) and Get-region-body(rgn) get the poles and region-body of rgn, respectively. Region-body can be a list of regions if rgn is either a parallel or a serial region. Since region-body is represented as a list, we use two procedures get-first-element and get-rest-elements to retrieve the head and the tail of the list.

We are now ready to discuss the aggregation procedures.

Aggregation Procedures

Construct Boxes

We start with the procedure used to aggregate a set of individual structural units into one black box. Its implementation is straightforward, as shown below. Make-a-black-box in line 8 generates an object with two attributes:

- type — black box;
- contents — the structural units encapsulated.

The contents of a black box can be retrieved if there is such a need for viewing it as a white box.

```
1   Box! (structural-units)
2     if structural-units = nil then
3       return(nil);
4     else
5     if single-device?(structural-units) then
6       return(structural-units);
7     else
8       black-box := Make-a-black-box(structural-units);
9       return(black-box);
```

The challenge of structural aggregation for a given task is to search for the set of individual devices in the output structural unit that are to be boxed in a given configuration of regions, and to determine other regions uninteresting to the task at hand which are also boxed.

To see if the set of individual devices in the output structural unit delimited by the two given nodes is inside a given region, the following procedure, Aggregate-black-box, is used. Figure 29 depicts this algorithm. It takes as input the region of an active-configuration and two nodes delimiting the output structural unit as specified in a task. It first checks to see if the region is the very construct to be boxed by comparing the two nodes with the region poles (lines 2 & 3). If it is, then it boxes this region (line 4) and finishes the aggregation process. Otherwise, it checks if it is a serial region (line 7) or a parallel region (line 15) and continue search inside the region.

```
1   Aggregate-black-box (region, 2nodes)
2     poles := get-region-poles (region);
3     if poles = 2nodes then
4       Box! (region);
5     else
6       sub-regions := get-region-body (region);
7       if s-region?(region) then
8         if nodes-at-same-level?(sub-regions, 2nodes) then
9           Box-same-level-rgns ('(), sub-regions, 2nodes);
10        else
11          if nodes-inside?(sub-regions, 2nodes) then
12            Search-each-region ('(), sub-regions, 2nodes);
13          else
14            return(nil);
15      if p-region?(region) then
16        if nodes-inside?(sub-regions, 2nodes) then
17          Search-parallel-branches ('(), sub-regions, 2nodes);
18        else
19          return(nil);
20     else
21       return(nil);
```

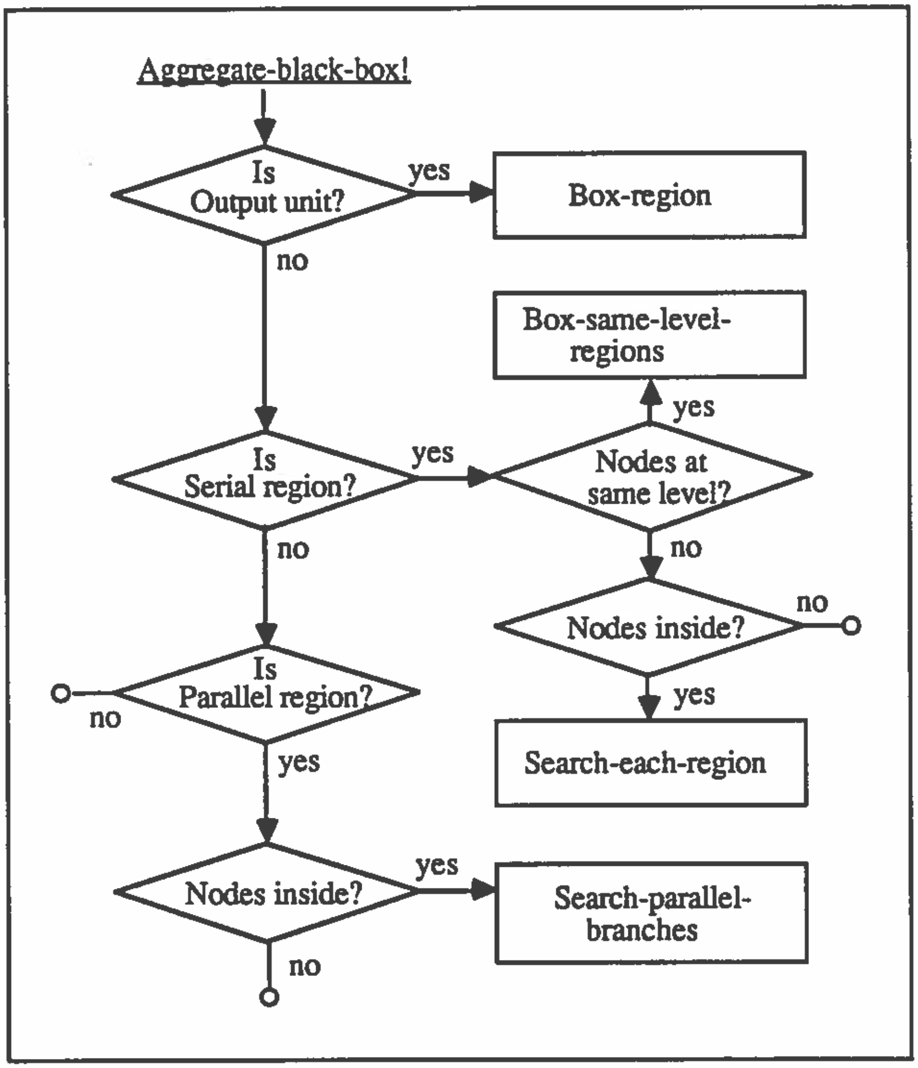


FIGURE 29. Aggregation Process to Create Black Boxes

There are two predicates, used in lines 8, 11 & 16, to determine whether two nodes are within a region before aggregation takes place. Nodes-at-same-level? returns true if the two nodes are on the same level as the sub-regions of a serial region. Nodes-inside? returns true if the two nodes are inside one of the sub-regions of the given region. For example, take the region as shown in Figure 30 for example. Nodes *a, b, e* and *f* are on the same level as the sub-regions of this region. Nodes *c, d* and *g* are not on the same level as the sub-regions, but are inside one of the sub-regions. The complexity of searching for the two nodes in a region is linear to the number of sub-regions visited.

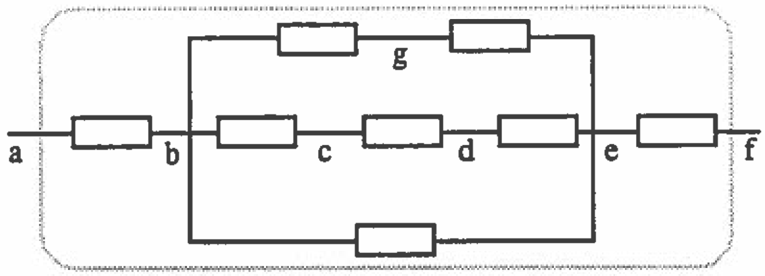


FIGURE 30. Levels of Nodes in Structural Hierarchy

Searching Same-Level Regions

If the two nodes of the output structural unit are at the same level as the sub-regions of a serial region (line 8), then procedure Box-same-level-rgns, described below, is used to search for the output structural unit. Figure 31 illustrates the ideas involved in procedure Box-same-level-rgns. Figure 32 shows the algorithm of this procedure.

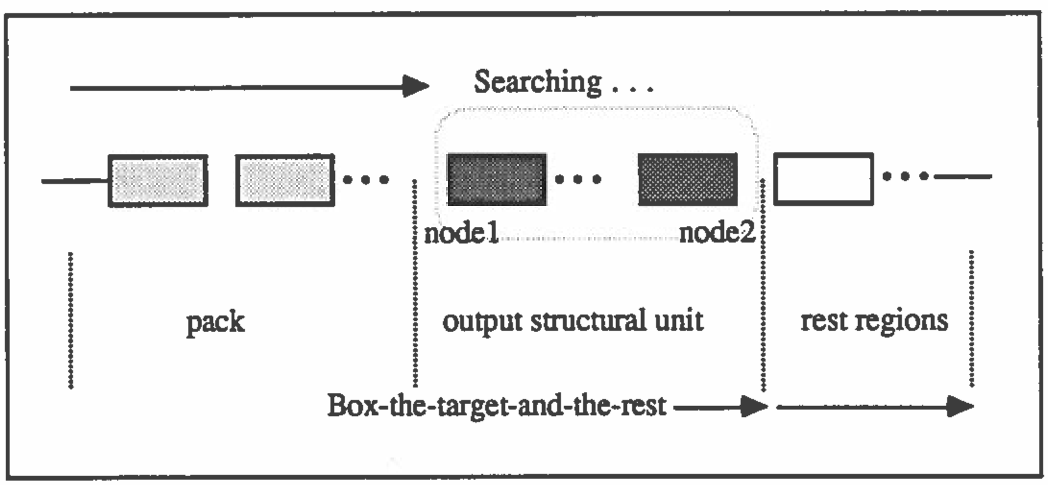


FIGURE 31. Output Structural Unit in Serial Region

The procedure Box-same-level-rgns scans the sub-regions of the serial region in sequence, packing along the way those sub-regions which does not contain any of the two nodes. When scanning the list of sub-regions, it checks if a region's positive pole matches one of the two nodes (line 5). If one does, then this region is the first of the list of regions to be boxed. When the positive node is found, it first boxes those regions already scanned and packed (line 7). The pack could be empty if the output structural unit begins with the first sub-region in the given serial region. It then searches the rest of the list (line 8) by procedure Box-the-target-and-the-rest. This procedure completes the output structural unit and box it. It then box the left over regions beyond the output structural unit.

```
1   Box-same-level-rgns (pack, sub-regions, 2nodes)
2     one-region := get-first-element (sub-regions);
3     rest-regions := get-rest-elements (sub-regions);
4     poles := get-region-poles (one-region);
5     if positive-pole-match-a-node?(poles, 2nodes) then
6       node- := get-negative-node (poles, 2nodes);
7       append (Box! (pack),
8               Box-the-target-and-the-rest ('(), rest-regions, node-));
9     else
10      pack := append (one-region, pack);
11      Box-same-level-rgns (pack, rest-regions, 2nodes);
```

```
1   Box-the-target-and-the-rest (result, sub-regions, target-negative-pole)
2     one-region := get-first-element (sub-regions);
3     rest-regions := get-rest-elements (sub-regions);
4     result := append (one-region, result);
5     rgn-neg-pole := get-negative-pole (one-region);
6     if rgn-neg-pole = target-negative-pole then
7       append (Box! (result), Box! (rest-regions));
8     else
9       Box-the-target-and-the-rest (result, rest-regions, target-negative-pole);
```

In Box-the-target-and-the-rest, since it already knows that the first region in the list is to be in the black box, it focuses searching for the negative pole of the output structural unit, appending each sub-region on the way as part of the resulting black box (line 4).

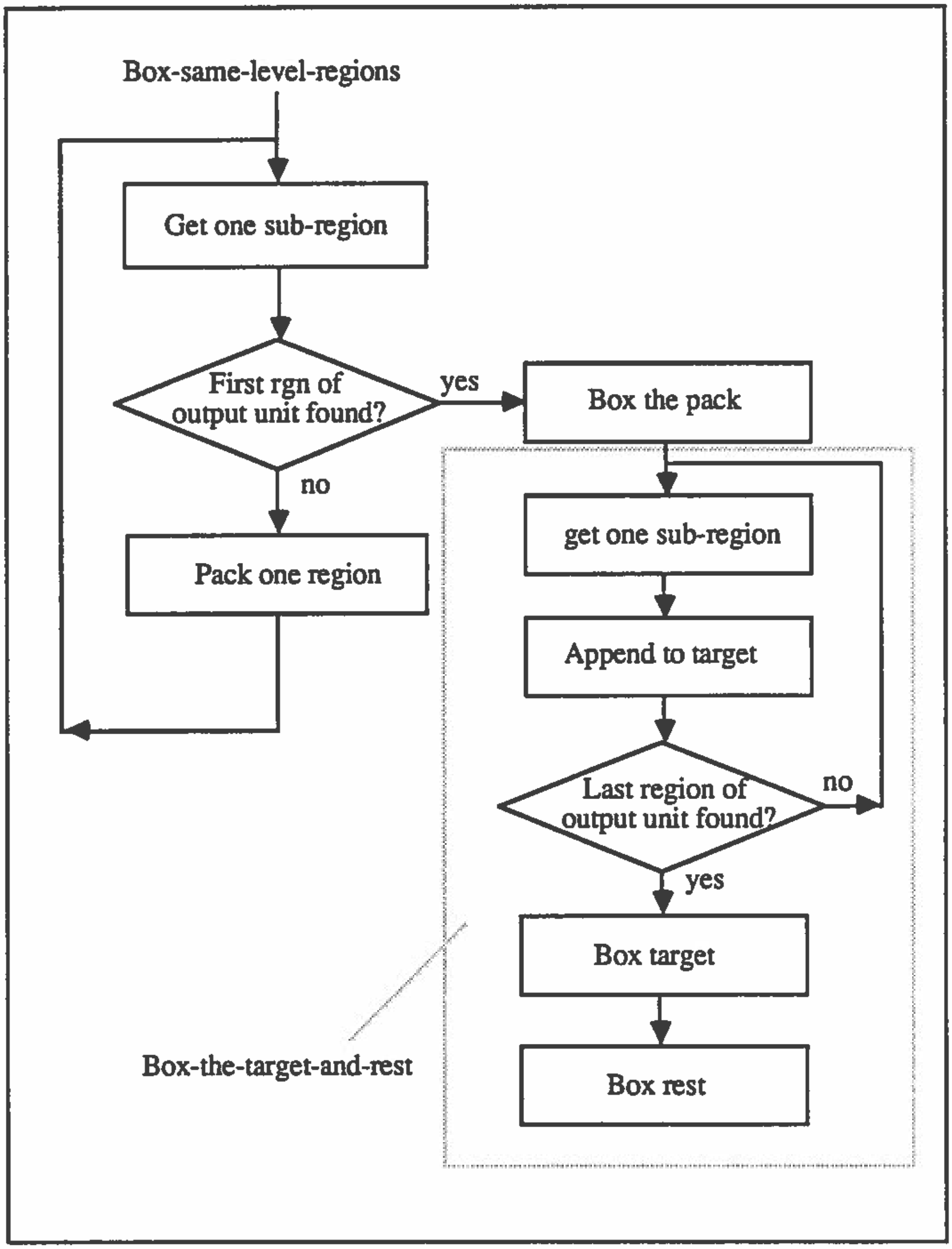


FIGURE 32. Black Boxing in Serial Region

When the negative node is found, it makes two boxes. The first box contains all the regions appended up to the point where the negative node is found. This is the target output structural unit. The second box contains the rest of the sub-regions in the list. The second box is empty if the last sub-region in the list contains the negative node. Note that each sub-region in the serial region is treated as an individual unit, whether it is a single device, or a parallel region. The complexity of these two procedures is *O(n)*, where *n* is the number of sub-regions in the serial region.

Searching Parallel Regions

If the output structural unit is determined to be inside a parallel region, then procedure Search-parallel-branches, described below, is used to search each of the parallel branches for the target output structural unit. Note that parallel branches are of two kinds: (1) a single device, or (2) a serial region. For a single device, since its poles are exactly the same as those of the parallel region, the device should have been found by procedure Aggregate-black-box discussed above.

Thus, for each parallel branch, the procedure checks to see if it is a serial region (line 5). Figures 33 depicts the ideas involved in the search.

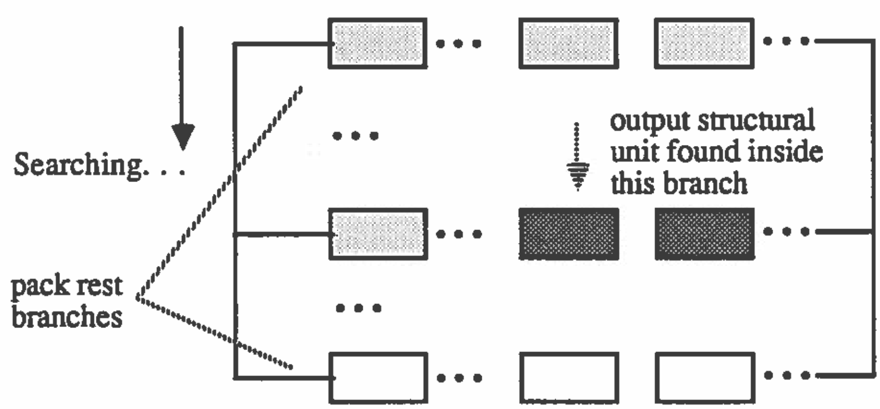


FIGURE 33. Output Structural Unit in One Branch of Parallel Region

If the branch is a serial region, there are three cases to consider:

(1) the two nodes are at the same level as the sub-regions of this serial region;

(2) the two nodes are not at the same level, but are inside one of the sub-regions of this serial region;

(3) the two nodes are not inside this serial region.

The first case says that the output structural unit is in this branch. It calls procedure Box-same-level-rgns, as described above, to do the job and packs the rest of the parallel branches into a black box (line 9).

The second case is similar to the first one, except the nodes are not at the same level as the sub-regions of the serial branch region but are inside one of the sub-regions. It therefore calls procedure Search-each-region (line 13), to be described in the following section. As in case one, it packs the rest of the branches into a black box (line 14).

The third case says that the output structural unit is not in this branch. The procedure simply packs this branch (line 16) and proceeds to searching the rest of the branches (line 17). Figure 34 shows the algorithm of this procedure.

```
1   Search-parallel-branches (pack, parallel-regions, 2nodes)
2     one-region := get-first-element (parallel-regions);
3     rest-regions := get-rest-elements (parallel-regions);
4     sub-regions := get-region-body (one-region);
5     if s-region?(one-region) then
6       if nodes-at-same-level?(sub-regions, 2nodes) then
7         other-branches := append(pack, rest-regions);
8         append (Box-same-level-rgns ('(), sub-regions, 2nodes),
9                 Box! (other-branches) );
10      else
11      if nodes-inside?(sub-regions, 2nodes) then
12        other-branches := append (pack, rest-regions);
13        append (Search-each-region ('(), sub-regions, 2nodes),
```

```
                Box! (other-branches) );
    else
        pack := append (one-region, pack);
        Search-parallel-branches (pack, rest-regions, 2nodes);
else
    pack := append (one-region, pack);
    Search-parallel-branches (pack, rest-regions, 2nodes);
```

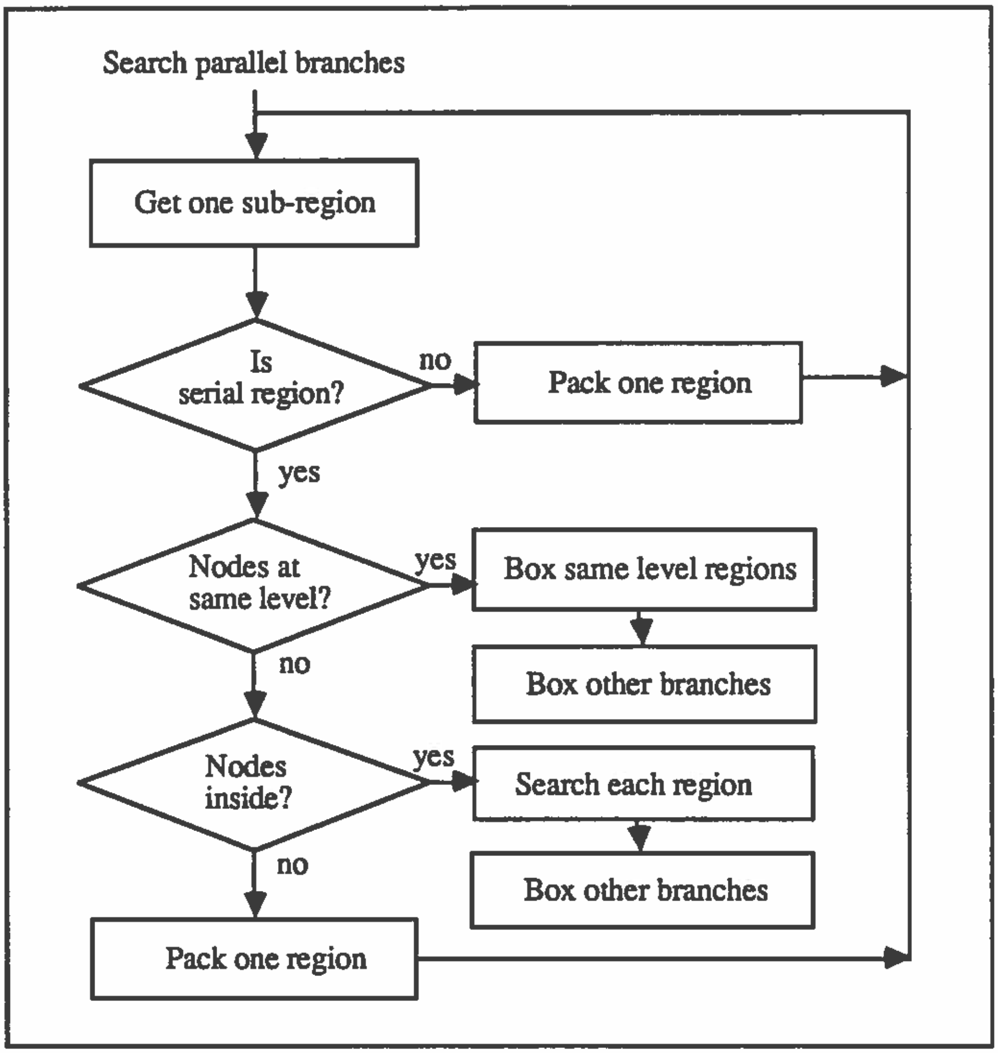


FIGURE 34. Black Boxing in Parallel Region

Finally, we describe procedure Search-each-region for searching each of the sub-regions in a serial region. This procedure should not be confused with Box-same-level-rgns discussed above. Box-same-level-rgns is called when ARC finds out that the two nodes of the output structural unit are on the same level as the sub-regions of a serial region. If the two nodes are determined to be inside one of the sub-regions, procedure Search-each-region is used.

```
1   Search-each-region (pack, sub-regions, 2nodes)
2     if sub-regions = nil then
3       return(nil);
4     else
5       one-region := get-first-element (sub-regions);
6       rest-regions := get-rest-elements (sub-regions);
7       output-unit-found := Aggregate-black-box (one-region, 2nodes);
8       if output-unit-found ≠ nil then
9         append (Box! (pack), output-unit-found, Box! (rest-regions));
10      else
11        pack := append (one-region, pack);
12        Search-each-region (pack, rest-regions, 2nodes));
```

Note that each sub-region is either a parallel region with its own sub-regions or a single device. Procedure Aggregate-black-box , as described above, is called at line 7. It may return a region with black boxes embedded. For example, take the circuit shown in Figure 35. If the output structural unit is between nodes *r* and *w*, the final outcome returned at line 9 is as shown.

Aggregation Shift for Different Tasks

We have described that, in order to create appropriate structural granularity in an active configuration of a circuit for a given task, the structural aggregation focuses search on the primary set of individual devices in the output structural unit as delimited by the two nodes in the output specification of the task. The algorithm is linear in time complexity with the size of the series-parallel structure representing the active configuration.

For the same circuit, different tasks may suggest aggregating different subsets of structural units and generate different black boxes to suppress extraneous detail of the circuit for efficient qualitative reasoning.

For example, suppose the active configuration of the rectifier circuit identified is the one when the diode is forward-biased and the capacitor is charging. Figure 36 shows how black-boxing is done in terms of two located output structural units for two different tasks.

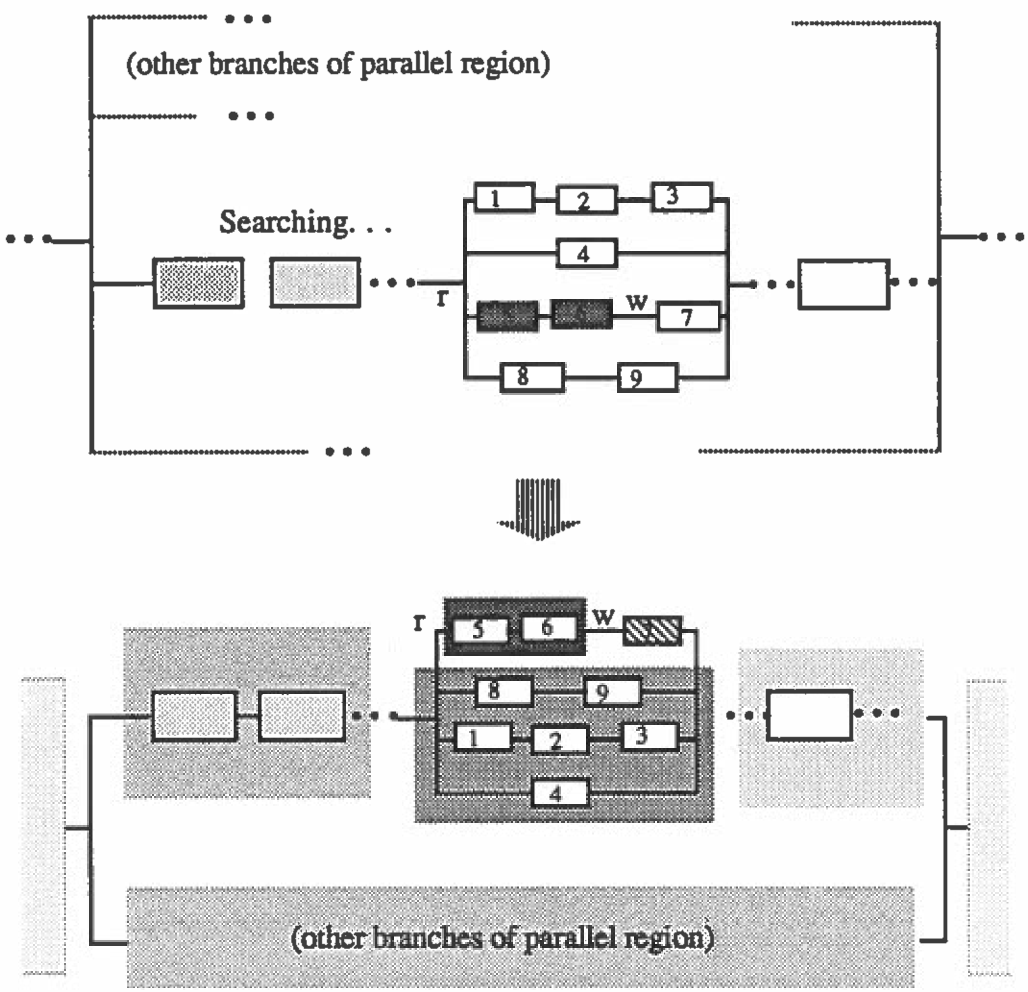


FIGURE 35. Multi-Level Black-Boxing to Minimize Unnecessary Detail

(1) If the output of a task indicates the behavior of voltage between nodes (*c d*), i.e.,

- Target System: [Half-wave-rectifier1]
- Input Specification: (voltage (a d) (0 +) (+ +) (+ 0) (+ -) (0 -) (- -) (- 0) (- +))
- Output Specification: (voltage (*c d*)),

then ARC aggregates the structure of the circuit and creates two black boxes, as shown in Figure 36-(1).

(2) If the output of a task indicates the behavior of the diode in this configuration, i.e.,

- Target System: [Half-wave-rectifier1]
- Input Specification: (voltage (a d) (0 +) (+ +) (+ 0) (+ -) (0 -) (- -) (- 0) (- +))
- Output Specification: (voltage (*a b*)),

then R1, R2, and C1 are grouped as a black box, as shown in Figure 36-(2).

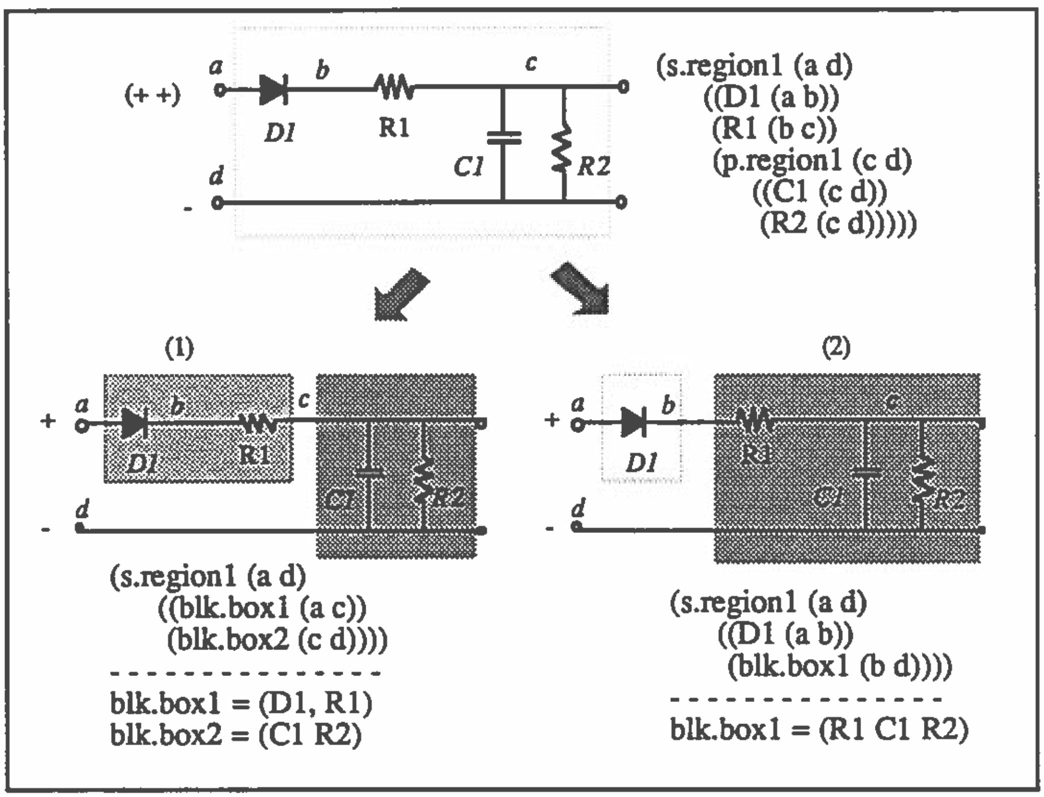


FIGURE 36. Shifting Perspective in the Dimension of Structural Aggregation

The model of an aggregated configuration of a circuit embodies a 2D perspective. This perspective reveals not only the dynamic configuration of the circuit, but also the chosen structural granularity with appropriate black boxes encapsulating individual devices in the configuration.

In Figure 36, structural aggregation in the given active configuration of the rectifier plays the role of creating equivalent circuits commonly used in electronic engineering. Sussman and Steele (1980) have introduced a constraint language to declare slices to aggregate individual devices. But the responsibility to create instances of slices in their system rests with the user. ARC extends the notion of slices by automatically creating appropriate structural granularities as driven by the task at hand.

Summary

In this chapter, we have described an algorithm for aggregating individual devices in an active configuration of a circuit so as to simplify the circuit models for the task at hand. The simplification of the circuit model is based on the concept of equivalent circuits from electronic engineering. Given two equivalent circuits, one simpler than the other, it is clearly to our advantage to use the simpler one for efficiency and clarity during causal analysis. More importantly, the significance of task-driven aggregation is that the resulting model of the target system successfully ignores those extraneous details for the task at hand.

Structural aggregation for a particular task centers around the output structural unit as specified in a given task, and leads to a distinct structural granularity of the circuit under analysis. The structural granularity of the same circuit may shift when there is a different task. We have shown that the choice of the appropriate structural granularity is task-driven, by describing how black boxes are created of the same circuit configuration for different tasks.

The search for the output structural unit is based on the two nodes given by the output specification in a task. For parallel and serial configurations, the two nodes are expected to be on the same level in the configurations. For two nodes not on the same level, it makes no sense to analyze the behavior of current, or charge flow, between the two nodes. We have, therefore, assumed that a "current probe" is in series with the output structural unit, by Kirchhoff's Current Law. For voltage, we assume that a "voltage probe" is in parallel[11] with the output structural unit, by Kirchhoff's Voltage Law.

In the next chapter, we describe how ARC generates QDE models from a given configuration, which may contain black boxes, and how ARC controls ontological shift between the macroscopic and microscopic levels of circuit behavior descriptions.

---

[11] However, a voltage probe whose terminals are connected to any two nodes not on the same levels does not violate KVL because one can always find a loop with the probe. In other words, there is always a voltage difference between any two nodes in a circuit. To handle such a voltage probe placed on different levels generally requires quantitative knowledge. Integrating qualitative and quantitative reasoning is one of our immediate future work.

CHAPTER V

DESCRIBING MACRO AND MICRO LEVEL BEHAVIORS

Topological configuration and structural aggregation we have examined in the previous two chapters are both operations performed on models of the structure of a target system. As we seeks to reason with both the device ontology and charge-carrier (CC) ontology of electronic circuits at the macroscopic and microscopic levels, this chapter discusses three issues: (i) how to generate QDE models embodying a specific ontological choice from an aggregated configuration, (ii) how to propagate changes in a QDE model, and (iii) how to shift ontological choice during causal analysis when a task suggests reasoning from more than one ontological perspective.

A QDE model generated from an active configuration of a target system embodies a perspective reflecting choices on three dimensions: a specific topological configuration, an appropriate structural granularities, and an ontological choice. We will describe how to propagate changes using QDE models for causal analysis as a result of a perturbation to one of the circuit parameters.

QDE Model Generation

Extensions to Confluence Theory

In order to create QDE models for causal analysis, we need a means that is sensitive to the causal mechanisms, the directionality from causes to effects, and the sequential dependency of causal chains [Forbus & Gentner 1986a]. Since the structure of a circuit is normally described as how component devices are connected, via nodes, into a device network, we adopt de Kleer & Brown's (1984) component-based confluence theory to model each individual structural unit and its interconnections with other units in an active configuration.

The compatibility and continuity conditions, from system dynamics [Shearer 1971], specific to the electronic circuits are the network laws such as Kirchhoff's Voltage Law (KVL), Kirchhoff's Current Law (KCL), and Ohm's Law. These physical laws are applicable to individual components and aggregated constructs uniformly. For example, the voltage across each of the parallel elements is equal according to KVL. The current is equal through each of the serial elements according to KCL. Since a configuration is a linear circuit, the impedance (resistance or reactance) of each of the constituent regions in an active configuration can be formulated according to Ohm's Law. In short, qualitative descriptions of a region's behavior can be systematically formulated in ARC.

One belief in qualitative physics is that if we choose a component-based approach to modeling, we often miss crucial global relationship [Forbus 1988]. But if we hand-code any global relationships into a model, we violate the important "no-function-in-structure" (NFIS) principle [de Kleer & Brown 1984].

This research shows that, with automated modeling, global relationships can be modeled without violating the NFIS principle. For example, given any region with sub-regions, the relationship between the region's total current and total voltage drop can be described in a QDE model. Such descriptions provide global relationships for the individual components within the region and help solve severe complexity problems for certain tasks if qualitative reasoning only uses local information. At first glance, these global features may seem to violate the NFIS principle, for it goes beyond the original confluence theory. But, since we view the aggregated units as single elements, those global features are in fact "local" to the aggregate units. Since the QDEs describing these structural units are generated dynamically via topological configuration and structural aggregation and not hand-coded into a QDE model, the principle of NFIS principle associated with the confluence theory remains intact.

Exogenous and Endogenous Variables

Note that confluences as constraint equations do not distinguish causes from effects. In particular, de Kleer and Brown's confluence theory does not have any principled way of distinguishing between possible causal factors for causal propagation. For example, if one needs to represent the relationship "the change of a resistor's resistance (∂R) can cause the current (∂I) through it to change, but not vice versa," the formalism of confluence $\partial V - \partial R - \partial I = 0$ alone would not suffice.

We approach this problem in ARC by classifying the qualitative variables in a QDE model into two types, (1) system parameters (exogenous variables) and (2) system values (endogenous variables). The system parameters in a QDE model correspond to those target-system features which can be directly manipulated to cause changes in the system. They always stand for the root causes whenever the system behavior changes. By contrast, the system values are those whose changes are caused by either a system parameter or other system values. From a pragmatic standpoint of troubleshooting, for example, faults would fall under the system parameters and symptoms under the system values. This strong typing is useful in ensuring directionality of causal propagation in analyzing complex behavior.

From a selected ontological choice based on the task at hand, the process to generate QDE models of a circuit uses the region descriptions of the active configuration. Figure 37 shows the process of QDE model generation from regions, a common structural view of both the device ontology and the CC ontology. Accordingly, we divide the discussion of QDE model generation into two parts: (1) QDE models in the device ontology, (2) QDE models in the CC ontology.

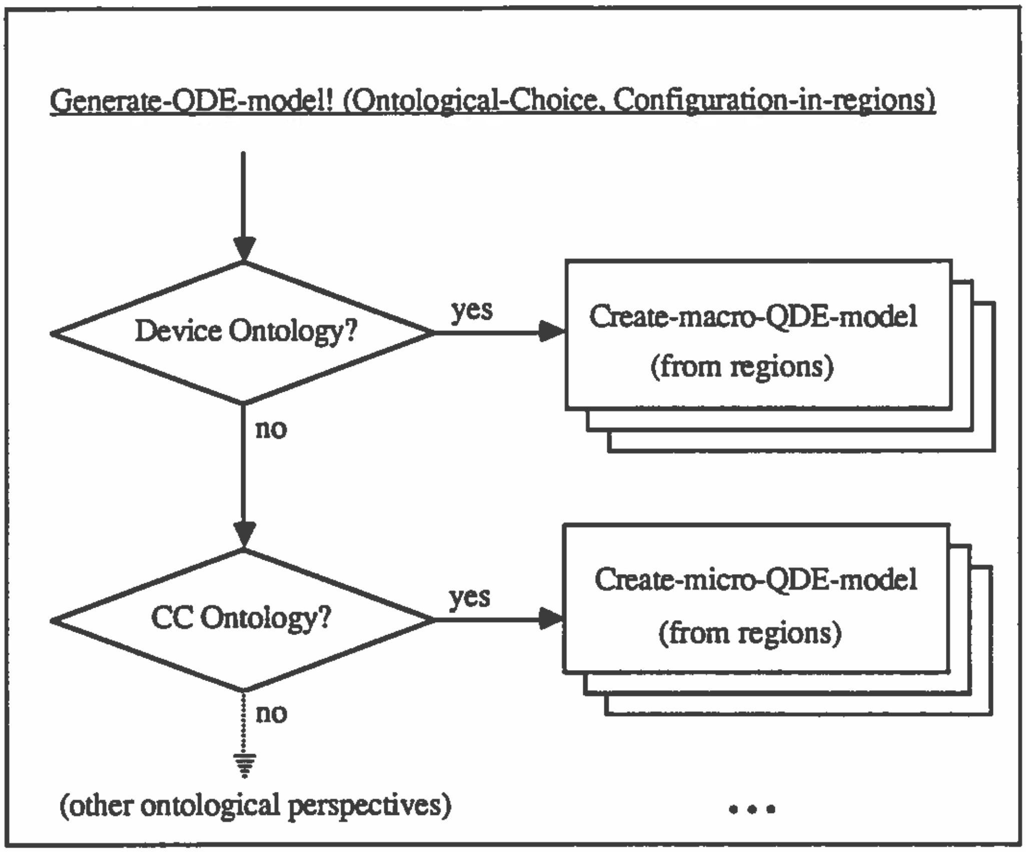


FIGURE 37. QDE Model Generation Process

Creation of QDE Models at Macroscopic Level

There are three types of regions in a given configuration: serial regions, parallel regions, and regions of individual devices, including black boxes. Construction of QDEs for a parallel or serial region involves three steps.

- First, the region is treated as an individual structural unit. As such, the descriptions of Ohm's Law and voltage compatibility for the unit can be created.
- Second, the interrelationships among the sub-regions in the region are described. For example, resistance of the region is summarized of its sub-regions. The continuity condition describing KCL for each connection node in a serial region is described.

- Finally, each sub-region itself is described. Since each of the sub-regions is either a single device, or a parallel or serial region, it recursively uses the same procedure as for the given region to create its own QDEs.

Black-boxes are treated as individual devices. The hiding of irrelevant structural detail inside a black box ultimately leads to suppressing the behavioral detail of those structural units. When generating variable names, each resistor's resistance and each capacitor's capacitance are marked as system parameters. The voltage at the pole nodes are also marked as system parameters. We assume that these system parameters can be directly manipulated or perturbed to causal changes in a circuit. Variable names that are not so marked are classified as system values by default. A high-level description of generating a QDE model in the device ontology is shown below.

```
Create-macro-QDE-model! (regions)
   if regions = nil then
     return(nil);
   else
     one-region := get-first-element (regions);
     rest-regions := get-rest-elements (regions);
     case type-of(one-region)
        s-region:
          qdes := Build-serial-region-QDEs (one-region);
        p-region:
          qdes := Build-parallel-region-QDEs (one-region);
        single-device:
          qdes := Build-single-device-QDEs (one-region);
        end-case
     append (qdes, Create-macro-QDE-model! (rest-regions) );
```

The actual generation of QDEs is done by procedures at lines 9, 11, and 13. The QDEs thus formulated are appended to those to be generated for the rest of the regions (line 15). The complete set of QDEs generated from a given active configuration constitutes one QDE model of the target circuit under analysis.

QDEs for Serial Regions

To create QDEs for a serial region, procedure Build-series-region-qdes is used. As discussed above, there are three steps involved in creating QDEs for a region with sub-regions. (1) Describe the region as an individual unit, (2) describe the relationships among its sub-regions, and (3) describe each of the sub-regions itself. The procedure is shown below.

```
1   Build-serial-region-QDEs (s-region)
2    Create-proper-names-for-s-region-QDEs(s-region);
3    append ( Ohm-law-QDE (rgn-voltage, rgn-current, rgn-resistance),
4              voltage-compatibility-QDE (rgn-voltage, src-voltage, sink-voltage),
5              resistance-compatibility-QDEs (rgn-resistance, sub-resistances),
6              s-region-kcl-QDEs (rgn-current, sub-regions),
7              Create-macro-QDE-model! (sub-regions) );
```

Lines 2 generates variable names used in the QDEs to be created. These names include region-voltage, region-current, region-resistance, source-voltage, sink-voltage, and sub-resistances.

Ohm's Law (line 3) for a region, whose pole nodes are (a b), is described as

$$\partial V_{a,b} - \partial I_{a,b} - \partial R_{a,b} = 0.$$

The voltage compatibility condition (line 4) for the region is described as

$$\partial V_{a,b} - \partial V_a + \partial V_b = 0.$$

These two descriptions are created by viewing the region as an individual structural unit.

Resistance-compatibility condition (line 5) describes the relationship that the total resistance of the region equals the sum of the resistances of its sub-regions, as

$$\partial R_{total} - \partial R_1 - \partial R_2 - \ldots - \partial R_n = 0.$$

Procedure S-region-kcl (line 6) generates descriptions of Kirchhoff's Current Law (KCL) for each node connecting the sub-regions of the serial region. KCL says the sum of currents at each node is zero. In other words, the current flowing into a node equals the current flowing out of the node.

$$\partial I_{a,b} - \partial I_{b,c} = 0,$$

$$\partial I_{b,c} - \partial I_{c,d} = 0,$$

$$\bullet \bullet \bullet.$$

Reflecting the law of conservation for flow, this is the continuity condition for a serial region.

The QDEs for the sub-regions are generated at line 7, which recursively calls Create-macro-QDE-model! (sub-regions).

QDEs for Parallel Regions

QDEs for a parallel region are generated in a similar way as for a serial region, except the descriptions for compatibility and continuity condition for the region are different. The procedure is shown below.

```
1  Build-parallel-region-qdes (p-region)
2    Create-proper-names-for-p-region-QDEs(p-region);
3    append ( Ohm-law-QDE (rgn-voltage, rgn-current, rgn-resistance),
4              voltage-compatibility-QDE (rgn-voltage, src-voltage, sink-voltage),
5              resistance-compatibility-QDEs (rgn-resistance, sub-resistances),
6              p-region-kcl-QDEs (rgn-current, sub-regions),
7              Create-macro-QDE-model! (sub-regions) );
```

Lines 2 creates variable names used in QDEs describing the parallel region and its sub-regions. When the parallel region is treated as an individual structural unit, the QDEs created are the same as for a serial region (lines 3 & 4). When it comes to the sub-regions of the parallel region, things begin to differ. Qualitatively, however, the relationship between the total resistance of the region and the resistances of the sub-regions is the same as that for a serial region (line 5), as

$$\partial R_{total} - \partial R_1 - \partial R_2 - \ldots - \partial R_n = 0,$$

although their quantitative descriptions are different. This shows the abstract nature of qualitative representation.

KCL for a parallel region (line 6) says the total current through the region equals the sum of all the sub-currents through the parallel branches, as

$$\partial I_{total} - \partial I_1 - \partial I_2 - \ldots - \partial I_n = 0.$$

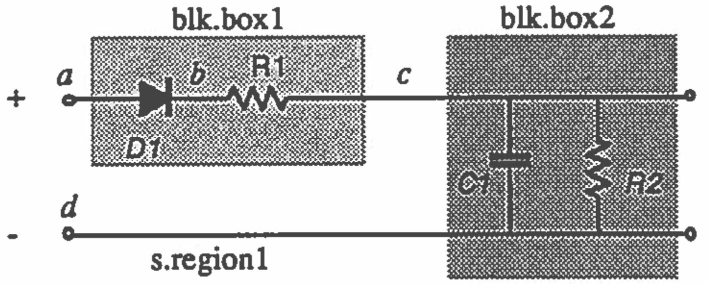
The QDEs for the sub-regions are generated at line 7, which recursively calls Create-macro-QDE-model! (sub-regions). The descriptions of KVL for the parallel branches are created when each of them is treated as a single region, for which the same voltage name is given, since all branches of a parallel region have the same poles.

QDEs for Single Device

Writing QDEs for a single device only concerns the Ohm's Law and voltage compatibility of the device. To complete our description, the procedure is shown here.

```
Build-single-device-qdes (region)
  Create-proper-names-for-single-device-QDEs (region);
  append (Ohm-law-QDE (voltage, current, resistance),
           voltage-compatibility-QDE (rgn-voltage, src-voltage, sink-voltage));
```

The complexity of the algorithm Create-macro-QDE-model! for generating a QDE model is $O(c)$, where c stands for the total number of regions in a given active configuration. When there are a few structural units as a result of structural aggregation which creates black boxes, the algorithm is fairly efficient. Figure 38 shows the actual QDE model generated by ARC for the half-wave rectifier in the aggregated configuration where the diode is forward-biased.



$$\partial V@a\_d - \partial I@a\_d - \partial R@s.region1 = 0.$$
$$\partial V@a\_d - \partial V@a + \partial V@d = 0.$$
$$\partial R@s.region1 - \partial R@blk.box1 - \partial R@blk.box2 = 0.$$
$$\partial I@a\_d - \partial I@a\_c = 0.$$
$$\partial I@a\_c - \partial I@c\_d = 0.$$
$$\partial V@a\_c - \partial I@a\_c - \partial R@blk.box1 = 0.$$
$$\partial V@a\_c - \partial V@a + \partial V@c = 0.$$
$$\partial V@c\_d - \partial I@c\_d - \partial R@blk.box2 = 0.$$
$$\partial V@c\_d - \partial V@c + \partial V@d = 0.$$

FIGURE 38. A Device-Ontology QDE Model Generated from Active Configuration

Creation of QDE Models at Microscopic Level

This section introduces elements of a Charge-Carrier (CC) ontology[12] at the microscopic level to supplement the device ontology at the macroscopic level for reasoning about electronic circuits. At the microscopic level, the basic function of electronic devices is to control the movement of electric charge carriers, such as free electrons (or holes) [Bell 1980].

---

[12] In [Liu 1991], we show a more complete description of the CC ontology. In this chapter, we only outline elements of the CC ontology. We focus on the problem of shifting between the device ontology and the CC ontology during causal reasoning.

The primitives for the CC ontology include concepts such as field, force, velocity, and charge-flow. The central notion in the CC ontology is the charge-carrier collection. Considering individual charge carriers would be prohibitive and unnecessary since all positive or negative charge carriers act alike. Considering an anonymous collection of charge carriers as one individual greatly reduces the complexity of modeling their behavior. Thus, a CC collection is similar in spirit to Collins and Forbus's (1987) molecular collection (MC).

As we have discussed above, the representation of regions is used as a common structural description for both the device ontology and the CC ontology. When considered as occupying a cylindrical piece of space, a region has two features that capture its physical shape: length, $L(p,n)$, and cross-sectional area, $A(p,n)$. Likewise, a pole, p, when considered as a two-dimensional surface, has two features: surface area, S_p, and unit charge, Q_p.

The QDEs of a region at the microscopic level are created by viewing the region as an electric field. We describe the relationships as shown in Figure 39 for our discussion.

- A field between two poles has intensity (E), which is directly proportional to the net charge (Q) at the poles and inversely proportional to the distance (L) between the poles;
- The electric force (F) on the CC collection in a field is directly proportional to the field's intensity (E);
- The velocity (V) at which the CC collection moves in a field is directly proportional to the electric force (F) on the charge carriers;
- The flow of charge (C) as a result of charge-carriers' movement is directly proportional to the charge carriers' velocity under the electric force in the electric field (V) and to the cross-sectional area (A) of the region.

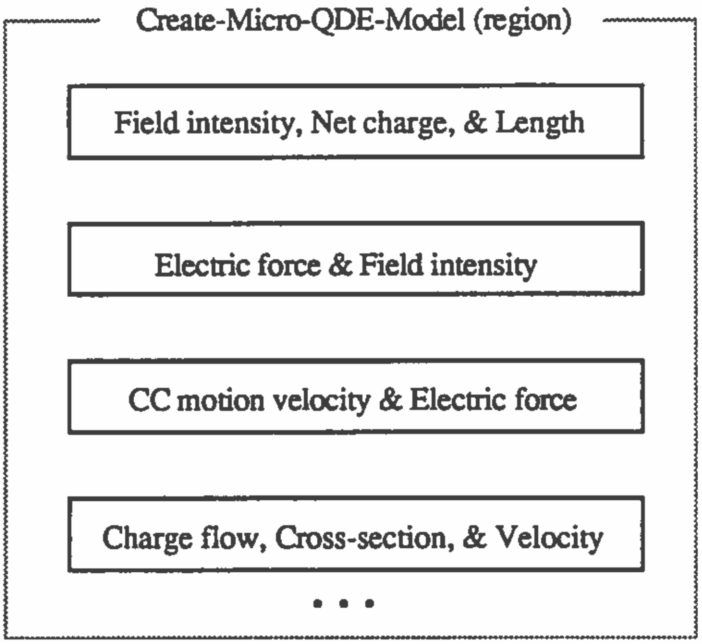


FIGURE 39. Elements of CC Ontology

These relations are represented as the axioms of the CC ontology. The variable names for the distance (*L*) and cross-sectional area (*A*) are marked as system parameters in QDE models. Given a particular region with poles (*p n*), ARC generates a QDE model of the region as shown in Figure 40.

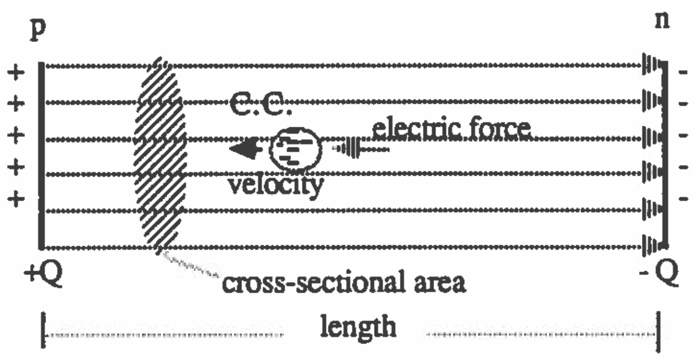
The CC ontology is at a microscopic level when compared to the device ontology. For circuit analysis, the CC ontology supplements, but is not "parasitic" to, the device ontology. For example, give the task "What is the behavior of the CC flow if the electric field increases?",

- Target System: [bridge-rectifier1]
- Input Specification: (field (in gd) (+ +))
- Output Specification: (cc-flow (in gd))

ARC generates the following explanations:

- In the electric field between nodes +in and -gd,
  field@in_gd's increase causes force-on-cc@in_gd to increase.
- force-on-cc@in_gd's increase causes cc-velocity@in_gd to increase.

• cc-velocity@in_gd's increase causes cc-flow@in_gd to increase.



$$\partial E(p,n) = \partial Qp - \partial L(p,n).$$

$$\partial F(p,n) = \partial E(p,n).$$

$$\partial v(p,n) = \partial F(p,n).$$

$$\partial C(p,n) = \partial A(p,n) + \partial v(p,n).$$

FIGURE 40. A CC-ontology QDE Model Generated from Region of Configuration

A QDE model generated by ARC is used to derive the behavior of a target system, reflecting a chosen perspective of three dimensions of appropriate topological configuration, structural aggregation, and ontological choice. One may suggest that the CC-ontology axioms could be simply lumped into the device-ontology QDE models for modeling and reasoning. However, there is ample evidence that models embodying a jumble of interrelated ontologies produce more harm than good [Winograd & Flores 1987]. Not only would the mix leads to inefficiency when using the QDE model for causal analysis, but it can pose problems for controlling causal propagation at a certain level since there were no distinction of macro and micro levels anymore. An ontologically consistent model of a physical system is the basis for the kind of simplicity and understandability that makes our analytic program robust and usable.

QUALEX — Causal Inference Engine

This section describes a causal inference engine, QUALEX, which takes a perturbation to one of the system parameters in a QDE model and outputs a description of how and why the behavior of the target system should change as a result of the perturbation.

Qualitative Value Theorem

We begin with the qualitative value theorem used in QUALEX for deriving qualitative values in a QDE model. First, we define the role played by a qualitative variable in a QDE.

Definition: Each qualitative variable in a QDE plays a role in balancing the constraint equation. Such a role is computed as the product of its qualitative value and its sign in the equation.

To find the qualitative value of a variable in a QDE, we introduce the following qualitative value theorem:

QV Theorem: Suppose a QDE has N variables in the form

$$\partial X_1 \pm \partial X_2 \pm \ldots \pm \partial X_n = 0.$$

In order to find the qualitative value of variable ∂X_i in the confluence, the other variables $(\partial X_1, \ldots, \partial X_{i-1}, \partial X_{i+1}, \ldots, \partial X_n)$ must all play an identical role, $\underline{R}$, if their qualitative values are not zero. If such an $\underline{R}$ exists, then the qualitative value of ∂X_i equals the product of $sign(\partial X_i) * \underline{R} * (-1)$.

In the expression "$sign(\partial X_i) * \underline{R} * (-1)$", $sign(\partial X_i)$ stands for the sign preceding ∂X_i in the QDE, and $\underline{R} * (-1)$ stands for ∂X_i's role which is always opposite of $\underline{R}$ so as to keep the balance of the constraint equation. Therefore, the product of the sign and the role give the qualitative value of variable ∂X_i.

QUALEX Algorithm

QUALEX, as a causal inference engine for intrastate causal analysis, takes three items as input: (1) a QDE model of a target system, (2) a perturbation to one of the system parameters as the cause, and (3) a target variable whose behavior we are interested in. The process derives causal chains from the cause to the target variable. Both the perturbation (the cause) and the target are specified in a given task. If the target is not specifically given, than the process derives causal chains to all the system variables whose values change as a result of the perturbation. The QUALEX algorithm is shown below. Figure 41 depicts the causal reasoning process.

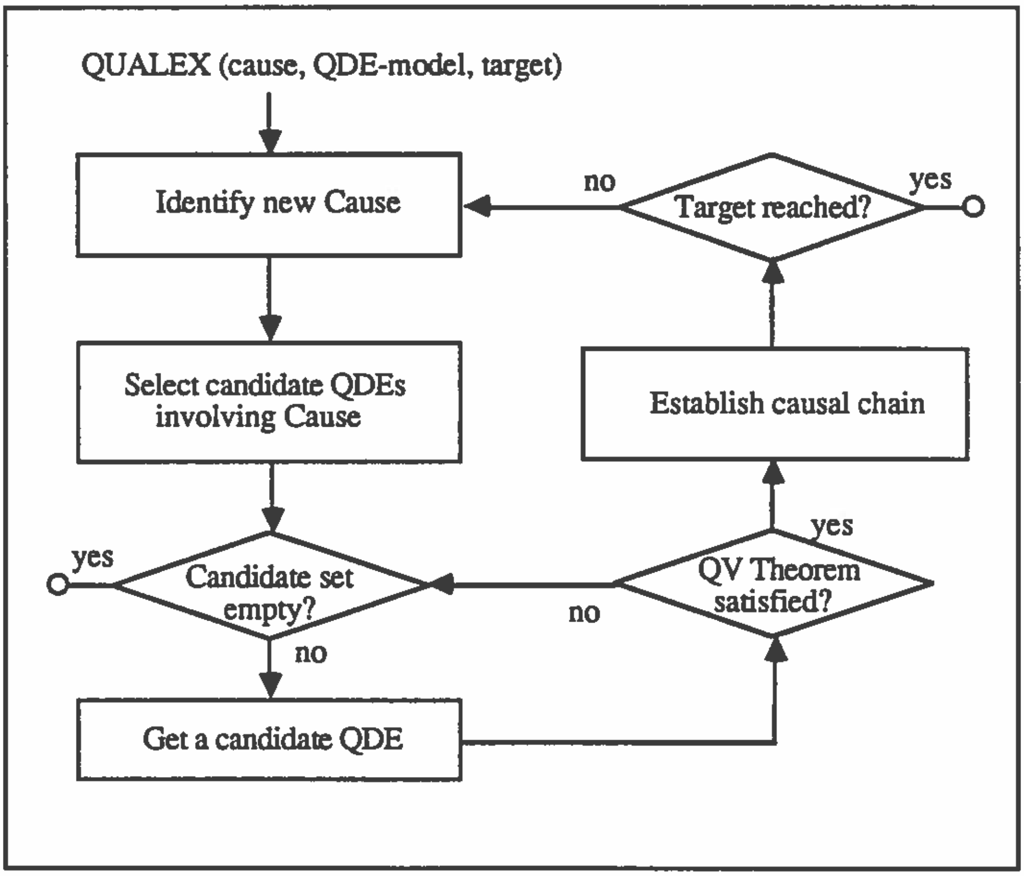


FIGURE 41. Causal Inference Engine — QUALEX

```
1  QUALEX (QDE-model, cause, target-var)
2     candidate-set := get-qdes-involving-cause (cause, QDE-model);
3     if candidate-set ≠ nil then
4     for-each Qde in candidate-set do
5        common-role := QV-theorem-satisfied? (Qde);
6        if common-role then
7           sys-var := get-unknown-var (Qde);
8           qual-val := common-role * sign(sys-var) * (-1);
9           assign! (sys-var, qual-val);
10          establish-causal-chain (cause, sys-var, QDE);
11          if sys-var ≠ target-var then
12             QUALEX (QDE-model, sys-var, target-var);
```

Given an input, the derivation process of QUALEX first finds a set of candidate QDEs which involve the perturbed variable (line 2). If none is found, then the process finishes. Otherwise, the process goes through each QDE in the set (line 4) to see if the given perturbation can lead to any change to any of the variables in the QDE. It does this by checking if the QDE satisfies the QV-theorem (line 5). The satisfaction of the QV-theorem tells us two things. First, there is a single unknown variable in the given QDE. Second, a common role is found for all the known variables in the QDE. According to the QV theorem, the new qualitative value for this unknown is computed and assigned (lines 8 & 9). The process then makes a link from the cause to this system variable. If their qualitative values are equal, then it is a positive link. Otherwise, it is a negative link. The qualitative value change of the system variable then becomes a new cause to propagate further changes to the rest of the variables in the QDE model.

The complexity of the QUALEX algorithm is O(nv), where n stands for the number of QDEs in a given model and v stands for the average number of variables in a single QDE.

Note that a single-perturbation assumption is used to reduce the complexity of multiple causalities. This suggests that in order to derive a causal chain, one can hold all but one system parameter constant so as to create a situation ideal for studying the causal impact of this individual cause on all the system variables. The advantage of this is obvious: if a variable, X, remains constant, then $\partial X = 0$. The zero effects can be easily handled during qualitative causal reasoning.

The key notions used in the algorithm are the roles of qualitative variables in the model and the QV theorem described above. These notions are formed on the basis of mathematical deductive reasoning about the constraint equations. Thus, the correctness of the algorithm for generating the causal chains is entirely dependent on the validity of the QDE model. For the model to be valid, it is essential that it is both correct and complete: correct in the sense that it does not violate the underlying physical principles of the domain, and complete in the sense that none of the descriptions and the compatibility and continuity conditions of the structural components in a given configuration is missing. This inference engine is sound because it is based on the mathematics and logic deduction. It does not produce incorrect solutions if it cannot provide a solution.

Ontological Shift

Up to this point, we have shown how QDE models are generated in a given ontological choice, and how they are used for causal reasoning. We now discuss how to control ontological shift by switching QDE models as driven by the task at hand.

Bridging Device Ontology and CC Ontology

The CC ontology is designed with the motivation to supplement the device ontology so that causal reasoning about circuits can shift between the two ontologies. We introduce the notion of bridging relations to enable ontological shift. Such ontological bridges associate comparable terms from the two related ontologies of a target system.

Ideally, it would be nice if we could define one-to-one mappings for all the terms in one ontology to those in the other. But in general, we cannot expect to find such mappings. A superficial answer is that there may be different numbers of terms in the two ontologies. But a deeper examination reveals that the two ontologies may carve the world in different granularities. For example, the basic structural elements in the device ontology are the component devices and their interconnections in the system topology of a circuit. In contrast, the basic structural elements in the CC ontology are electric fields and spatiotemporal location of charge carriers. When we talk about the net force on a charge-carrier collection inside the depletion region of a diode, for example, there is simply no comparable term in the device ontology to match because the smallest structural unit in the device ontology is the whole device itself.

This observation requires that the bridging relations be defined in such a way that the spatiotemporal continuity of causal propagation is maintained during ontological shift. Thus, the bridging relations must be defined over compatible structural elements of the same target system. We call this the structural compatibility principle. For example, voltage should not be mapped directly to the force on a charge-carrier collection, but it can be mapped to the corresponding electric field. Ontological shift in causal propagation follows such bridges to ensure coherent and robust causal explanation.

The "regions" we discussed above offer this common structural view for certain terms in the two ontologies. To revisit the concept, a region refers to a spatial unit between two chosen poles where current or charge-carrier movement is of interest. As such, bridging relations can be defined over regions. With regions, macroscopic and microscopic concepts of electronics can be related to each other. Table 3 below shows the mapping, based on regions, between the macroscopic and microscopic behavioral descriptions of electronic circuits used in ARC.

Crossing Ontological Bridge

These mappings provide a simple means for bridging the two ontological perspectives during causal reasoning while maintaining the spatiotemporal continuity of causal propagation. In order to facilitate both-way mapping between the two ontologies, bridging relations are represented as constraint equations rather than some one-way conversion rules. As such, "crossing a bridge" is done by giving a perturbation in one end of a

TABLE 3. Mapping Between Macroscopic and Microscopic Descriptions

Macroscopic	bridge	Microscopic (CC)
voltage	<=>	intensity of field in region;
resistance	<=>	resistivity and physical shape of region;
current	<=>	charge-carrier movement in region;
capacitance	<=>	pole size and distance in region.

bridging relation and deriving its effect in the other end. The perturbation is given in one ontology and the effect is received in a different ontology. For example, given bridging relation $\partial voltage_{p,n} = \partial field(p,n)$, when $\partial voltage_{p,n}$ is perturbed and changes, $\partial field(p,n)$ changes correspondingly. When such a perturbation comes from one session of QUALEX running a QDE model of a target circuit in one ontology and the effect is sent as a perturbation to another session of QUALEX running a QDE model of the same circuit but in a different ontology, we say that causal reasoning has been bridged through from one ontology to another.

Ontological bridges provides a means for ontological shift between two related ontologies, but raises the issue of how to control their application. Since we view an ontology as defined by a language with a set of terms, predicates and axioms, we propose a set of ontological-choice rules to govern the process of ontological shift, by noting what terms, predicates and axioms are involved in a particular task. Below, we show that selection of ontological choice is task-driven, i.e., the decision as to which ontology to use and when to shift ontological perspective depends on the specific task at hand.

Control of Ontological Shift

For qualitative causal reasoning in ARC, the choice of which ontology to select and when to shift from one to another depends on the task at hand, as for topological configuration and structural aggregation described in the previous chapters.

In a given task, the input ontology and output ontology could be same or different. For example, "Given a sinusoidal voltage as input, explain the behavior of voltage at output," or "Perturb the voltage here and explain the behavior of the movement of charge carriers there as a result." The input ontology and output ontology provide a basis for ARC to select the appropriate ontological choice to carry out the task at hand. We introduce the following domain-independent rules to govern ontological choice.

Rule 1: If the input and the output variables are of the same ontology and the analysis does not require justification of any of the axioms in the ontology, then select the given ontology.

This rule is straightforward. In fact, this is the implicit rule used in most previous work in qualitative physics, with only a single model of a single ontology.

Rule 2: If the input and the output variables are of the same ontology and the analysis requires justification for one of the axioms of the ontology, then shift to a related ontology for explanation.

This rule reflects the crucial concept of ontology in qualitative physics. To reiterate, an ontological choice for a problem domain for an automated reasoning program consists of a distinct representational language of terms, predicates, and axioms for describing and reasoning about the domain [Hayes 1985]. In the language, axioms are primitive entities, based on which other things such as theorems can be derived. But an axiom cannot be derived or explained in the same language of which it is a part. When a task asks to explain something that reflects one of the axioms in an ontology, it is time to shift to a related ontology and then shift back to complete the reasoning.

Rule 3: If the input and the output variables are from different ontologies, proceed with the input ontology until causal propagation comes to the region of the output variable and then shift to the output ontology to complete the reasoning.

This rule is used for a task which explicitly "asks for" ontological shift since the input and output are stated in two different ontologies. Since we have identified a common structural description with regions for the two ontologies, ARC begins reasoning in the input ontology until it comes to the region of the output variable, where it crosses into the output ontology following one of the mutual bridges and completes the reasoning task.

QDE Model Switching

Since each QDE model carries a single ontological choice, ontological shift is accomplished by switching QDE models during reasoning. This section describes the procedure used by ARC for selecting and shifting ontological choices. The process for selecting ontological choices is shown in Figure 42.

Bridging relations and ontological-choice rules provide a means for how and when to shift ontological choice, respectively, to carry out a given task. To sum up, after topological configuration and structural aggregation, ARC generates QDE models from appropriate ontological choices and then calls QUALEX for causal propagation. The ontological choice for QDE-model generation is driven by the ontological choice rules based on the task at hand. When ontological shift is deemed necessary, ARC selects one of the bridging relations between the two ontologies and performs ontological shift by crossing the bridge from one QDE model to another to carry out the given task.

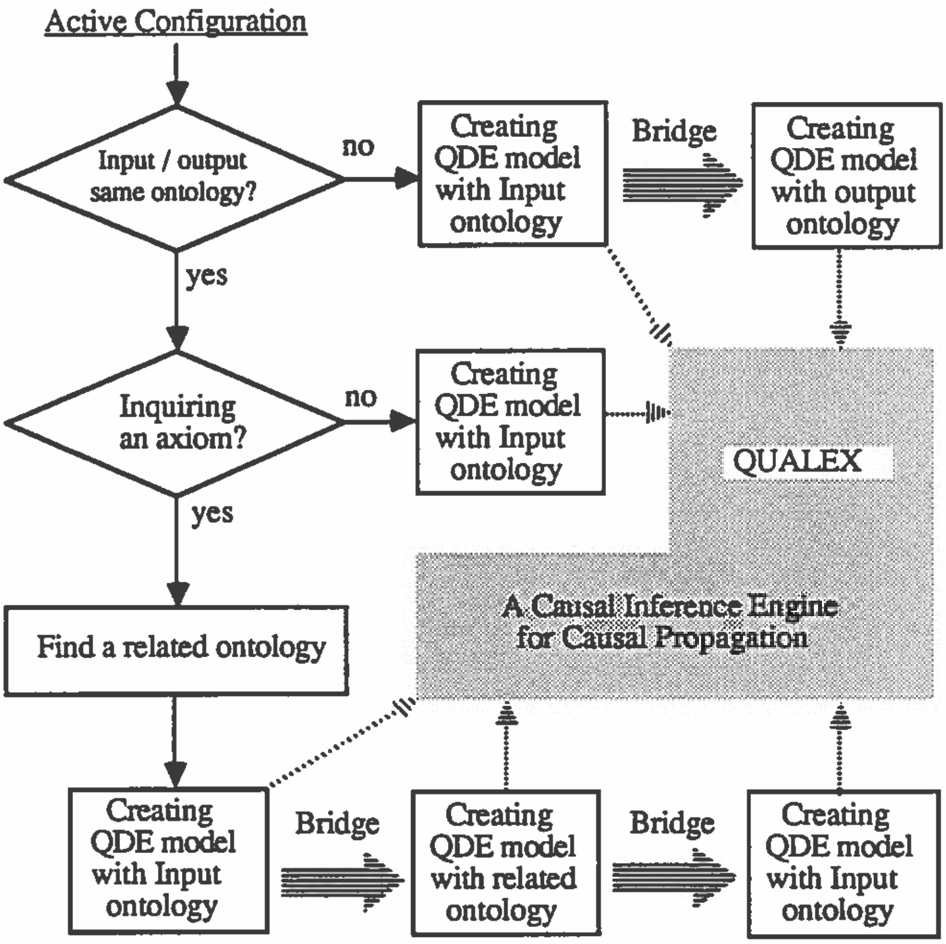


FIGURE 42. Ontological Shift via Switching QDE Models

For example, "Given the half-wave rectifier, if the input voltage between nodes *a* and *d* increases, what happens to the charge carrier flow from nodes *c* to *d*?"

- Target System: [half-wave-rectifier1]
- Input Specification: (voltage (a d) (+ +))
- Output Specification: (cc-flow (c d))

In this task, the input and output are stated in different ontologies. According to Ontological Choice Rule #3, ARC generates two QDE models of this circuit, one in the input device ontology and the other in the output CC ontology. Switching QDE models during reasoning, ARC generates the following explanations:

** Begin reasoning in device-ontology **

- In the serial region, s.region5, between nodes (a d),
  since voltage@d remains constant,
  we deduce that voltage@a's increase causes voltage@a_d to increase.
- Also, in s.region5,
  since resistance@s.region5 remains constant,
  it is clear that voltage@a_d's increase causes current@a_d to increase.
- current@a_d's increase means current@a_c's increase because they two originate from the same source & share the same path in the configuration.
- According to KCL with regard to node c for a serial connection,
  we know that current@a_c's increase leads to the increase of current@c_d.
- For black box blk.box6 between nodes (c d) containing
  ((p.region3 (c d) ((resistor24 (c d)) (capacitor1 (c d)))))
  since resistance@blk.box6 remains constant,
  current@c_d's increase causes voltage@c_d to increase.

** Cross a bridge to charge-carrier-ontology **

- In the region between nodes (c d),
  voltage@c_d's increase is equivalent to field@c_d's increase.
- In the electric field between nodes +c and -d,
  field@c_d's increase causes force-on-cc@c_d to increase.
- force-on-cc@c_d's increase causes cc-velocity@c_d to increase.
- cc-velocity@c_d's increase causes cc-flow@c_d to increase.

Summary

In this chapter, we have described the approach to reasoning about electronic circuits with both the device ontology at the macroscopic level and the CC ontology at the microscopic level. We started by describing how to generate QDE models in a selected ontological choice from the active configuration of a circuit under analysis. We then described QUALEX, a causal inference engine that takes a QDE model and a perturbation and derives causal chains underlying the circuit behavior as a result of the input perturbation. A QDE model reflects only a single ontological choice of the target system. When a task requires more than one ontological choices, multiple QDE models are generated. For ontological shift, we then addressed issue of bridging relations defined over regions, and presented the process of ontological shift via model switching based on a set of domain-independent ontological choice rules. As a result, ARC can answer more questions with both the macroscopic and microscopic descriptions of electronic circuits than with just a single one.

CHAPTER VI

EXAMPLES AND EVALUATION

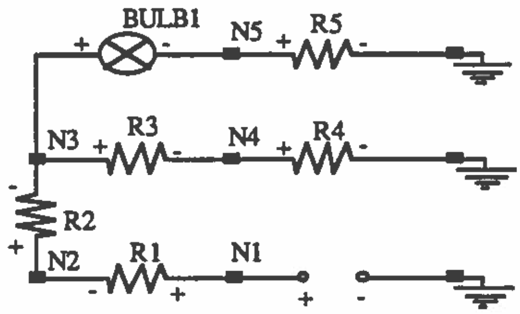
In the previous chapters, we have described how ARC integrates three dimensions of perspective-taking to formulate and select models for different tasks. This chapter presents examples used to evaluate ARC. The sources of the examples are: (1) from related AI literature concerning the domain of electronic circuits, and (2) from standard physics and electronic engineering textbooks.

At the end of this chapter, we discuss the experience of experts and practitioners in the field of electrical engineering who were asked to review ARC's performance on standard problems, to pose new questions, and to evaluate its performance.

Zero-Order and First-Order Reasoning

We begin by considering the work by White and Fredricksen (1986), who investigate teaching students qualitative reasoning skills for reasoning about electrical circuits. They point out that this approach represents a radical departure from how physical theories have been taught in classrooms. Traditionally, only quantitative analysis is taught and students are left to develop their own qualitative methods, which they rarely do until long after they become skilled at quantitative analysis.

White and Fredricksen note that, even for the simplest circuit, there are different kinds of questions that one can ask about its behavior. These questions require different kinds of reasoning. For example, given the circuit in Figure 43, one could start by asking, "Is the light in this circuit on or off?" This type of question can be answered by a simple form of qualitative reasoning, which they call "zero order" reasoning because it involves existence, not change. One could also ask "What happens to the light as we increase the resistance of resistor R5? Does the light get brighter or dimmer? Why?" Answering these questions requires a more sophisticated form of qualitative reasoning, which they call "first order" reasoning because it involves the first-order derivatives.




```
(define-circuit! WF-circuit   ;; system topology
  ;; Components =
       ((RESISTOR: R1 R2 R3 R4 R5 BULB1))
  ;; Connections =
       ((N1 (terminal1 R1))
        (N2 (terminal2 R1) (terminal1 R2))
        (N3 (terminal2 R2) (terminal1 R3) (terminal1 BULB1))
        (N4 (terminal2 R3) (terminal1 R4))
        (N5 (terminal2 BULB1) (terminal1 R5))
        (GD (terminal2 R5) (terminal2 R4))))
```

FIGURE 43. A Linear Resistive Circuit

White and Fredricksen's work focuses mainly on pedagogical issues. They argue that only when zero-order concepts are mastered should first-order qualitative models of circuit behavior be taught. Finally, only after students can reason about and understand circuit behavior in qualitative terms, should quantitative reasoning be introduced. As such, their work focused mainly on the zero-order models of circuit behaviors.

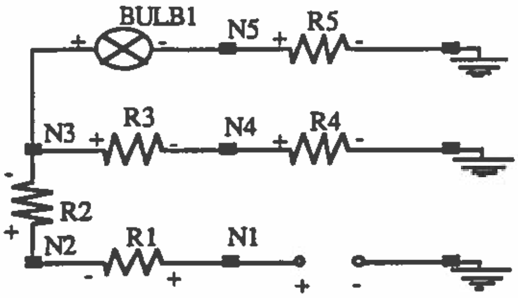
ARC can answer both zero-order and first-order questions. Given a task, ARC first recasts the system topology of the circuit in terms of oriented parallel-serial clusters, and then generates the active configuration from the clusters.

To find if the light bulb is on or off, we check if the light bulb is part of the active configuration in the given qualitative state of the circuit. The light is on if it is. For this zero-order reasoning question, we give a task where *Qval* = +, but *Qdir* = *0*. The active configuration generated by ARC shows that BULB1 is in the configuration, which means that the light (BULB1) is On. Figure 44 shows the clusters and the active configuration of this circuit for the given task.

In White and Fredricksen's (1986) system for zero-order reasoning, the polarities of the terminals of every device in the circuit are first determined. In order to explain why the light bulb is on, their system traces a path from the positive terminal of the light bulb to the positive pole of the battery. The existence of such a path explains why the light bulb is on. This differs from the topological configuration in ARC, which identifies not just a single path between a device and the power source, but the complete current flow path of a circuit in a given qualitative state. When the state changes, the circuit may switch to a different configuration to reveal the new current flow path.

The clustering process in ARC reveals a hierarchical view of the system topology of a circuit. As pointed out by Palies, et al. (1986), the process of recognizing how devices are grouped in series or parallel is a crucial step in circuit analysis. Failure to recognize parallel and serial connections is one of the most common problems in circuit analysis for students [Palies, et al. 1986].

For the "first-order" analysis of the current through the light bulb as a result of the change of the resistance in *R5*, we give a task definition whose input specification indicates positive change (increase) in resistance of *R5* with steady voltage between nodes *N1* and *GD* as the power source of the circuit. The output specification indicates interest in current through the light bulb between nodes *N3* and *N5*. Figure 45 shows the first-order derivation of why the light becomes dimmer as a result of the increase in resistance of *R5*.



Task: • Target System: [wf-circuit]
• Input Specification: (voltage (n1 gd) (+ 0))
• Output Specification: (current (n3 n5))

Clusters:

```
((s.cluster7 ((n1 end1) (gd end2))
      ((r1 ((n1 terminal1) (n2 terminal2)))
       (r2 ((n2 terminal1) (n3 terminal2)))
       (p.cluster6 ((n3 end1) (gd end2))
            ((s.cluster5 ((n3 end1) (gd end2))
                   ((bulb1 ((n3 terminal1) (n5 terminal2)))
                    (r5 ((n5 terminal1) (gd terminal2)))))
             (s.cluster4 ((n3 end1) (gd end2))
                   ((r3 ((n3 terminal1) (n4 terminal2)))
                    (r4 ((n4 terminal1) (gd terminal2))))))))))
```

Active Configuration:

```
((s.region73 (n1 gd)
      ((blk.box9 (n1 n3))
       (p.region72 (n3 gd)
            ((s.region71 (n3 gd) ((bulb1 (n3 n5)) (r5 (n5 gd))))
             (blk.box8 (n3 GD)))))))
: blk.box9 = ((s.region5 (n1 n3) ((r1 (n1 n2)) (r2 (n2 n3)))))
: blk.box8 = ((s.region4 (n3 gd) ((r3 (n3 n4)) (r4 (n4 gd))))
```

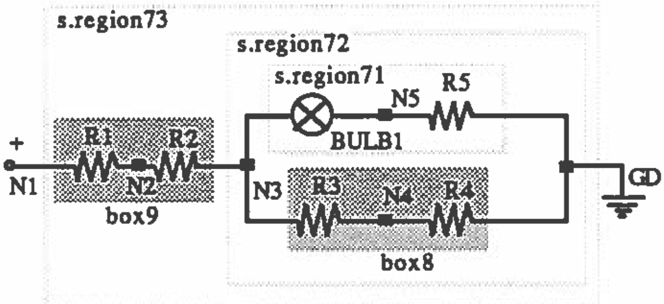
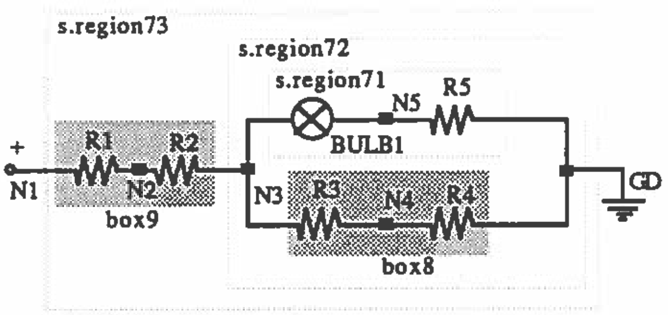


FIGURE 44. Zero-Order Reasoning by Identifying Active Configuration

When ARC propagates changes in the circuit for the given task, it uses global knowledge of circuit structure to disambiguate the local, uncertain situation.



Task:
- Target System: [wf-circuit]
- Input Specification: ((voltage (n1 gd) (+ 0)) (resistance r5 +))
- Output Specification: (current (n3 n5))

• • •

- In the serial region, s.region71, between nodes (n3 gd), since resistance@bulb1 remains constant, we deduce that resistance@r5's increase causes resistance@s.region71 to increase.
- In the parallel region, p.region72, between nodes (n3 gd), since resistance@blk.box8 remains constant, we deduce that resistance@s.region71's increase causes resistance@p.region72 to increase.
- In the serial region, s.region73, between nodes (n1 gd), since resistance@blk.box9 remains constant,we deduce that resistance@p.region72's increase causes resistance@s.region73 to increase.
- Also, in s.region73, since voltage@n1_gd remains constant, it is clear that resistance@s.region73's increase causes current@n1_gd to decrease.
- current@n1_gd's decrease means current@n1_n3's decrease because they originate from the same source & share the same path in the configuration.
- According to KCL with regard to node n3 for a serial connection, we know that current@n1_n3's decrease leads to the decrease of current@n3_gd.
- For black box blk.box9 between nodes (n1 n3), current@n1_n3's decrease causes voltage@n1_n3 to decrease.
- since voltage@n1 remains constant, voltage@n1_n3's decrease means that voltage@n3 increases.
- In the parallel region, p.region72, between nodes (n3 gd), since voltage@gd remains constant, we deduce that voltage@n3's increase causes voltage@n3_gd to increase.
- For black box blk.box8 between nodes (n3 gd) containing ((s.region4 (n3 gd) ((r3 (n3 n4)) (r4 (n4 gd))))), since resistance@blk.box8 remains constant, voltage@n3_gd's increase causes current@blk.box8 to increase.
- In the parallel region, p.region72, between nodes (n3 gd), we deduce that current@blk.box8's increase causes current@s.region71 to decrease.
- current@s.region71's decrease means current@n3_n5's decrease because they two originate from the same source & share the same path in the configuration.

FIGURE 45. First-order Reasoning from Structure to Behavior

Note that when the voltage at node *N3* increases, the voltage difference between *N3* and *GD* increases. The current through the serial branch with *R3* and *R4* increases, since the resistance of *R3* and *R4* remains constant. But the resistance in the serial region with the light bulb between *N3* and *GD* has increased. Thus, the qualitative value about the change in current through the light bulb becomes ambiguous. This is because, for the serial region 71, the QDE of Ohm's Law is: $\partial V_{n3,gd} - \partial R_{n3,gd} - \partial I_{n3,gd} = 0$. When both $\partial V_{n3,gd} = +$ and $\partial R_{n3,gd} = +$, $\partial I_{n3,gd} = ?$.

ARC gets around this problem by utilizing the global knowledge derived. It knows that, as a result of the increase in resistance of *R5*, the total current decreases, which means that the current through the parallel region also decreases. For the parallel region of two branches, since total current decreases but the current through one of the two branches increases, ARC concludes that the current through the branch with the light bulb decreases. This style of first-order reasoning differs from previous qualitative reasoning techniques [de Kleer 1984, Williams 1984], which use only local knowledge for circuit analysis. Those systems using purely local knowledge cannot handle this ambiguous situation.

Slices as Structural Aggregation

Sussman and Steele (1980) introduce a method to express slices as structural abstraction for circuit analysis. Slices are defined using a language of hierarchical constraint networks to represent the multiple viewpoints in the synthesis and analysis of electrical networks. For example, the idea that two resistors in series are equivalent to a single resistor is expressed as follows (Sussman & Steele 1980, p. 20):

```
(constraint series-resistors
   ((r1 resistor)
    (r2 resistor)
    (req resistor)
    (a adder))
    (== (>> t1 r1)  (>> t1 req))
    (== (>> t2 r2)  (>> t2 req))
    (== (>> a1 a)  (>> resistance r1))
    (== (>> a2 a)  (>> resistance r2))
    (== (>> sum a) (>> resistance req)))
```

Figure 46 shows their graphic representation of this declaration. Sussman and Steele note the importance of the definition of such a slice as a means to introduce an alternative perspective of a circuit.

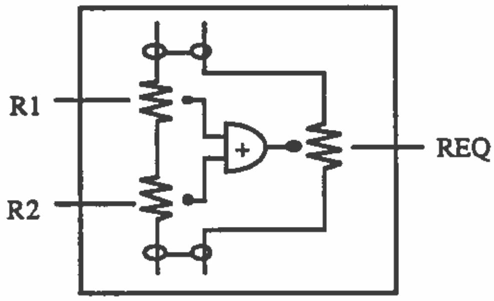


FIGURE 46. A Slice Encapsulating Two Resistors in Series

ARC shares the same motivation: to create multiple views or slices for different tasks. The question is: "Who creates such slices that show various structural granularities?" In Sussman and Steele's system, it is done by the user. If one wants to define a serial resistor, one creates an instance of the constraint and then specifies explicitly the two component-resistors. For example, to construct an instance of the serial resistor of R43 and R44, the user gives the following declarations (Sussman & Steele 1980, p. 20):

```
==> (create sr1 series-resistors)
SR1
==> (== (>> r1 sr1) r43)
IDENTITY
==> (== (>> r2 sr1) r44)
IDENTIFY
```

They pointed out that such points of view may not fit neatly into a single hierarchy. For example, a flat three-resistor circuit can be hierarchically decomposed in at least three different ways: (i) as three separate resistors; (ii) as the serial combination of the upper two, in series with the lowest; (iii) as the uppermost, in series with the serial combination of the lower two. They call this organization an almost-hierarchy.

ARC handles the issue of "almost-hierarchy" by focusing on a given task. Slices in ARC, represented as regions of various granularities, are created as driven by the task at hand. Figure 47 show the task-driven aggregation of the simple circuit of three resistors in series.


```
(define-circuit! SS-circuit
 ;; Components =
          ((RESISTOR: R1 R2 R3))
 ;; Connections =
          ((a (terminal1 R1))
           (b (terminal2 R1) (terminal1 R2))
           (c (terminal2 R2) (terminal1 R3))
           (d (terminal2 R3))))
```

For output (current (a b)), ARC views the system as follows:

```
  ((s.region4 (a d) ((r1 (a b)) (blk.box5 (b d))))),
 : blk.box5 =
    ((s.region7 (b d) ((r2 (b c)) (r3 (c d)))))
```

For output (current (a c)), ARC views the system as follows:

```
  ((s.region4 (a d) ((blk.box6 (a c)) (r3 (c d))))),
 : blk.box6 =
    ((s.region8 (a c) ((r1 (a b)) (r2 (b c)))))
```

For output (current (a d)), ARC views the system as follows:

```
  ((blk.box9 (a d))),
 : blk.box9 =
    ((s.region4 (a d) ((r1 (a b)) (r2 (b c)) (r3 (c d)))))
```

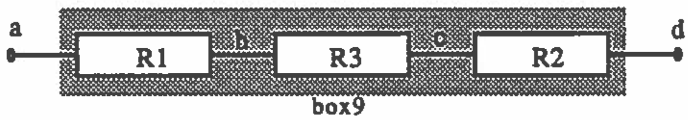
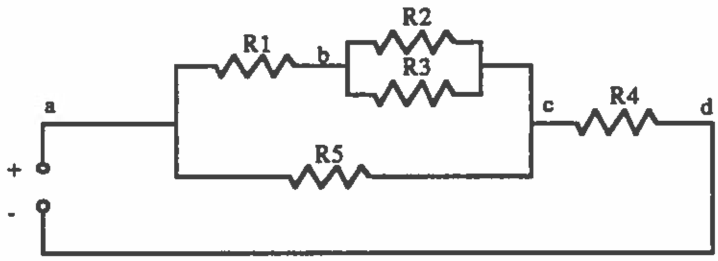


FIGURE 47. Structural Aggregation as Driven by Different Tasks

The aggregation is based on the output structural unit specified in each task. The user only specifies the task and ARC generates various structural units of appropriate granularities for the task. In this respect, there is no need to pre-define an almost-hierarchy in ARC.

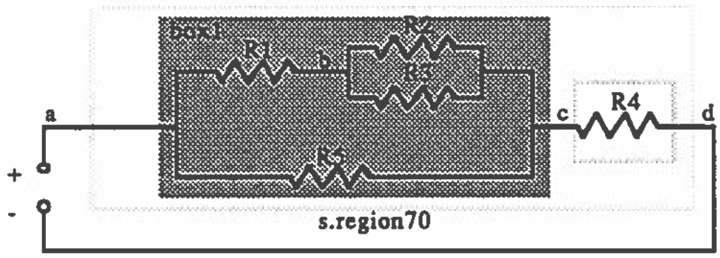
Eliminating Unnecessary Feedback Analysis

Structural aggregates are useful for suppressing irrelevant details during circuit analysis. ARC focuses on qualitative causal reasoning using qualitative constraints to avoid extraneous causal propagation. Figure 48 shows an example of suppressing detail using a circuit from [Weidner 1985], a standard physics textbook.




```
(define-circuit! weidner-circuit3
 ;; Components =
        ((RESISTOR: R1 R2 R3 R4 R5))
 ;; Connections =
        ((a (terminal1 R1) (terminal1 R5))
         (b (terminal2 R1) (terminal1 R2) (terminal1 R3))
         (c (terminal2 R2) (terminal2 R3) (terminal2 r5) (terminal1 R4))
         (d (terminal2 R4))))
```

Task:
- Target System: [weidner-circuit3]
- Input Specification: (voltage (a d) (+ +))
- Output Specification: (current (c d))



Derivation:

- In the serial region, s.region70, between nodes (a d), since resistance@s.region70 remains constant, it is clear that voltage@a_d's increase causes current@a_d to increase.
- current@a_d's increase means current@a_c's increase because they two originate from the same source & share the same path in the configuration.
- According to KCL with regard to node c for a serial connection, we know that current@a_c's increase leads to the increase of current@c_d.

FIGURE 48. Causal Derivation with Uninteresting Detail Suppressed

Intuitively, we know that when the $\partial V_{a,d}$ increases, the current through *R4* increases. We skip the detail of *R1, R2, R3* and *R5* and treat them as a black box. The derivation in Figure 48 utilizes the aggregated structural description of the circuit to suppress unnecessary detail. For output *(current (c d))*, ARC views the circuit as a serial region joining a black-box between nodes *(a c)* and *R4* between nodes *(c d)*. The uninteresting details about individual devices *(R1, R2, R3* and *R5)* in the black box are suppressed. The resulting derivation is short and to the point.

What if ARC could not aggregate individual devices for circuits such as this? The answer is simple: causal reasoning would have to consider the behavior of each individual device in the circuit, whether it is relevant to the given task or not. In particular, causal reasoning would have to deal with primitive-level negative feedback for all the parallel branches [de Kleer 1984, Williams 1984]. This has been a problem in AI thus far. Figure 49 shows a simple circuit of two resistors in parallel, a current divider, which Williams analyzes as follows: "An increase in I_{in} produces an increase in I_1, causing V_{in} to rise. The rise in voltage is applied across *R2*, increasing I_2 and, hence, reducing the effect of the initial current increase on I_1" [Williams 1984]. Imagine answering the question for the above circuit using this approach. Williams point out that most circuit analyses ignore this kind of feedback analysis.

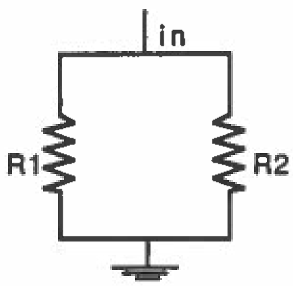


FIGURE 49. A Simple Parallel Circuit and Negative Feedback

ARC suppresses primitive-level feedback such as those in the parallel constructs by creating a hierarchical view of the system topology of a circuit. However, since ARC uses the QV theorem to propagate changes in a QDE model, each qualitative variable in the QDE model is assigned a qualitative value only once during the derivation. Because ARC does not generate a full envisionment, necessary feedback analysis cannot be handled by ARC. The difficulties encountered by feedback analysis have been discussed by de Kleer (1984) and Williams (1984).

For structural aggregation, ARC extends the notion of slices by not only providing a method to represent the various slices, but also automating the structural aggregation process. But ARC's structural aggregation currently is limited to parallel and serial constructs only.

Circuit Analysis Involving Shifting Configurations

ARC is designed to take a sequence of input perturbations of arbitrary length. This enables it to perform large-signal analysis, e.g., a sine-wave voltage input. Most previous work [de Kleer 1984, Williams 1984, White & Fredricksen 1986] only considered small signal analysis, e.g., a single perturbation. This does not accommodate automated reasoning about nonlinear circuits, because nonlinear circuits generally manifest several topological configurations to achieve its function when driven by alternating voltages. Figure 50 shows both a positive and negative clipping circuits from [Bell 1980], a standard electronic engineering textbook. A clipper is essentially a rectifier that connects a diode in series with the load resistor. Figure 50 shows the operation of the negative clipping circuit. Bell writes:

> The function of a clipper circuit is to clip off an unwanted portion of a waveform. The negative clipper passes the positive half-cycle of the input and removes the negative half-cycle. The positive clipper passes the negative half-cycle and clips of the positive portion. (Bell 1980, p. 56)

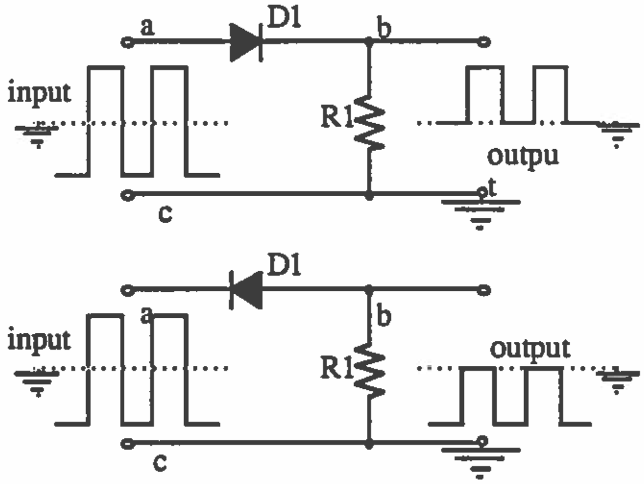
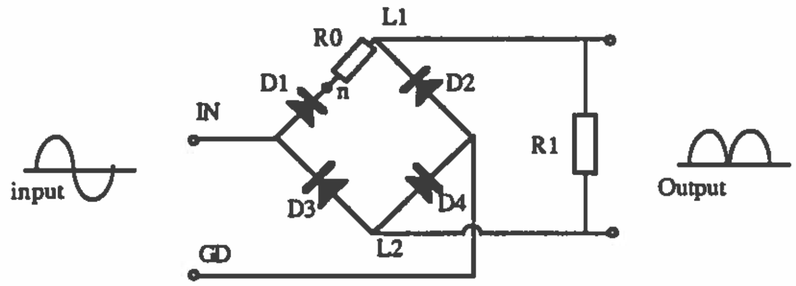


FIGURE 50. Negative and Positive Clippers

```
(define-circuit! negative-clipping-circuit
  ;; Components =
      ((RESISTOR: R1) (DIODE: D1))
  ;; Connections =
      ((a (terminal1 D1))
       (b (terminal2 D1) (terminal1 R1))
       (c (terminal2 R1))))

Task: • Target System:        [negative-clipping-circuit]
      • Input Specification:  (voltage (a c) (+ 0) (0 0) (- 0) (0 0) • • •))
      • Output Specification: (voltage (b c))

      • When the input is: (voltage (a c) (+ 0)), the configuration is:
            ((s.region71 (a c) ((d1 (a b)) (r1 (b c)))))

      • When the input is: (voltage (a c) (0 0)), the configuration is empty.
      • When the input is: (voltage (a c) (- 0)), the configuration is empty.
      • • •
      • When the input is: (voltage (a c) (+ 0)), the configuration becomes:
            ((s.region71 (a c) ((d1 (a b)) (r1 (b c)))))
      • • •
```

FIGURE 51. Configurations of Negative Clipper

When the input signal rises to be positive, the active configuration includes the output structural unit *(R1 (b c))*, which means that the positive portion of the signal generates a current through the output structural unit *(R1)* from *b* to *c*. When the input signal becomes zero or negative, the active configuration becomes empty. Figure 52 below shows a full-wave bridge rectifier that uses both the positive and negative portions of a sinusoidal input.



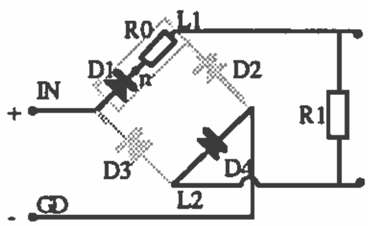

```
(define-circuit! bridge-rectifier
  ;; Components =
     ((RESISTOR: R0 R1)    (DIODE:    D1 D2 D3 D4))
  ;; Connections =
     ((IN  (terminal1 D1) (terminal2 D3))
      (GD  (terminal1 D2) (terminal2 D4))
      (n   (terminal2 D1) (terminal1 R0))
      (L1  (terminal2 R0) (terminal2 D2) (terminal1 R1))
      (L2  (terminal1 D3) (terminal1 D4) (terminal2 R1))))
```

Task:

- Target System: [bridge-rectifier]
- Input Specification: (voltage (in gd) (+ +) (+ 0) (+ -) (0 -) (- -) (- 0) (- +) (0 +) • • •)
- Output Specification: (voltage (l1 l2))

- When the input is: (voltage (in gd) (+ +)), the configuration is:
  ((s.region77 (in gd) ((blk.box39 (in l1)) (r1 (l1 l2)) (d4 (l2 gd)))))
  : blk.box39 = ((s.region75 (in l1) ((d1 (in n)) (r0 (n l1)))))
- When the input is (+ 0) and (+ -), the configuration remains the same.
- When the input is: (voltage (in gd) (0 -)), the configuration is empty.
- When the input is: (voltage (in gd) (- -)), the configuration becomes:
  ((s.region87 (gd in) ((d2 (gd l1)) (r1 (l1 l2)) (d3 (l2 in)))))
- When the input is: (voltage (in gd) (- 0) and (- +)), the configuration remains the same.
- When the input is: (voltage (in gd) (0 +)), the configuration is empty.
- When the input is: (voltage (in gd) (+ +)), the configuration becomes:
  ((s.region77 (in gd) ((blk.box39 (in l1)) (r1 (l1 l2)) (d4 (l2 gd)))))
  : blk.box39 = ((s.region75 (in l1) ((d1 (in n)) (r0 (n l1)))))
- When the input is: (voltage (in gd) (+ 0) (+ -)), the configuration remains the same.

• • •



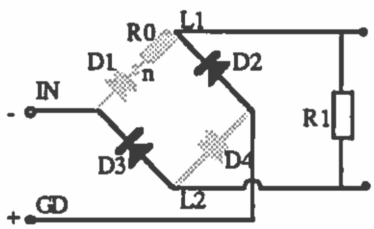


FIGURE 52. Configurations of Full-Wave Bridge Rectifier

The two distinct active configurations show two different current flow paths in the simulation. Although the circuit is nonlinear, each of the configurations behaves linearly, which makes the analysis possible [Rugh 1980].

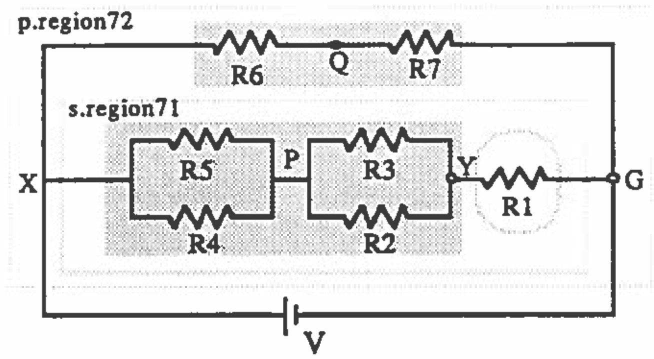
Causal analysis of nonlinear circuits, such as the rectifier we have discussed, depends crucially on our ability to identify appropriate configurations which serve to linearize the circuit. Previous approaches to circuit analysis [de Kleer 1984, Williams 1984] did not consider automating the configuration process. Although they also model a nonlinear device piecewise linearly [de Kleer & Brown 1984], the choice of which operating region the device is in is made by hand, when dealing with a single perturbation.

Reasoning at Both Macroscopic and Microscopic Levels

In addition to topological configuration and structural configuration, ARC can reason about circuits at both the macroscopic and microscopic levels of electronics. The examples presented in this chapter so far have all been carried out in the macroscopic device ontology. We now show two tasks that require shifting between the device ontology and the CC ontology.

The first example involves ontological shift from the device ontology to the CC ontology. Again, we use a circuit from Weidner (1985), as shown in Figure 53. The task is to derive what happens to the drift speed of charge carriers through *R1* when the voltage between *X* and *G* increases.

In Figure 53, notice that the input and output specifications are stated in different ontologies. This indicates an inevitable ontological shift somewhere during the derivation. According to our ontological choice rule #3, the causal propagation proceeds in the input device ontology until it comes to the region of the output structural unit. It then shifts to the output CC ontology to complete the reasoning.



Task:
- Target System: [weidner-circuit2]
- Input Specification: (voltage (x g) (+ +))
- Output Specification: (cc-velocity (y g))

Active Configuration:
((p.region72 (x g) ((s.region71 (x g) ((blk.box10 (x y)) (r1 (y g))))
(blk.box9 (x g)))))

** Begin reasoning in device-ontology **

- In the parallel region, p.region72, between nodes (x g), since voltage@g remains constant, we deduce that voltage@x's increase causes voltage@x_g to increase.
- Also, in p.region72, since resistance@p.region72 remains constant, it is clear that voltage@x_g's increase causes current@x_g to increase.
- In the serial region, s.region71, between nodes (x g), since resistance@s.region71 remains constant, we deduce that voltage@x_g's increase causes current@s.region71 to increase.
- current@s.region71's increase means current@x_y's increase because they two originate from the same source & share the same path in the configuration.
- According to KCL with regard to node y for a serial connection, we know that current@x_y's increase leads to the increase of current@y_g.
- For single device r1 between nodes (y g), since resistance@r1 remains constant, current@y_g's increase causes voltage@y_g to increase.

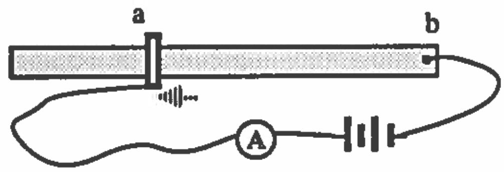
** Cross a bridge to charge-carrier-ontology **

- In the region between nodes (y g), we deduce that voltage@y_g's increase means field@y_g's increase.
- In the electric field between nodes +y and -g, field@y_g's increase causes force-on-cc@y_g to increase.
- force-on-cc@y_g's increase causes cc-velocity@y_g to increase.

FIGURE 53. Ontological Shift from Device Ontology to CC Ontology

This example shows not only the ontological shift in the derivation, but also the equivalent circuit that is created with structural aggregation to suppress irrelevant detail. As shown in Figure 53, because the output structural unit is *R1* between nodes *Y* and *G*, the individual devices *(R2, R3, R4,* and *R5)* between nodes *X* and *Y* are grouped together as a black box. Similarly, *R6* and *R7* are also grouped as a black box. In doing so, the uninteresting detail inside these two black-boxes is ignored by ARC during causal propagation.

Ontological shift during qualitative causal reasoning about circuits can be carried out from the macro level to the micro level, or vice versa. The next example explains why increasing the length of a resistor causes the current through it to decrease (Figure 54):




```
(define-circuit! resistat
  ;; Components =
        ((RESISTOR: R1))
  ;; Connections =
        ((a (terminal1 R1)) (b (terminal2 R1))))
```

Task:
- Target System: [resistat]
- Input Specification: (length (a b) (+ +))
- Output Specification: (current (a b))

** Begin reasoning in charge-carrier-ontology **

- In the electric field between nodes +a and -b, length@a_b's increase causes field@a_b to decrease.
- field@a_b's decrease causes force-on-cc@a_b to decrease.
- force-on-cc@a_b's decrease causes cc-velocity@a_b to decrease.
- cc-velocity@a_b's decrease causes cc-flow@a_b to decrease.

** Cross a bridge to device-ontology **

- In the region between nodes (a b), we deduce that cc-flow@a_b's decrease is equivalent to current@a_b's decrease.

FIGURE 54. Ontological Shift from CC Ontology to Device Ontology

Previous systems on circuit analysis only use a single device ontology. Those systems that considered two ontologies in non-circuit domains are limited to either the "parasitic" approach to ontological shift or only one-way mapping from the micro level to the macro level. ARC overcomes this limitation by representing the bridging relations as constraints that allow ontological shift in both directions. The present implementation of ARC does not fully represent the spatiotemporal aspect of the CC ontology. This imposes a limitation on reasoning about charge carriers as explicit pieces of stuff, and represents a fertile research area for the future.

Table 4 summarizes the examples shown in this chapter with respect to the critical components of ARC. The table indicates the source of each problem and the components of ARC that contribute to carrying out the task. We list four critical components in the table: topological configuration, structural aggregation, ontological shift, and QUALEX. Although each of the four components may play a role in carrying out a task, we only check the components critical to solving the given problem.

Expert Practitioner's Critiques

This section discusses the experience of three physics professors, who were asked to review ARC's performance. To preserve anonymity, I will refer to them as Prof. A, Prof. B and Prof. C. Their fields are in electrical engineering. The meeting with each was conducted separately. We discussed various aspects of ARC, including the examples presented in this chapter. They also posed tasks and gave helpful suggestions during the meeting. The following is an informal description of our discussions and is not a transcribed protocol.

To begin, they all agreed that in order to analyze a circuit quantitatively, the circuit must be understood in qualitative terms. Qualitative causal reasoning to them is tacit. Prof. B said that qualitative and quantitative reasoning in fact go hand in hand. He pointed out that quantitative knowledge can also be crucial in helping us understand physical systems in qualitative terms. He described the scenario that in some situations when the value of a quantity changes slightly, the circuit may exhibit very different behaviors. He almost used

TABLE 4. Summary of Examples with Respect to Critical Components of ARC

Problem	Source			Critical components of ARC			
	old AI	new AI	Textbook	configure	aggregate	onto-shift	qualex
0-order reasoning	√			√			
1st-order reasoning	√			√	√		√
creation of slices	√			√	√		
elimilating primitive feedback analysis		√	√		√		
nonlinear clipping circuits		√	√	√			
bridge rectifier		√	√	√			
reasoning from macro level to micro level		√	√		√	√	√
reasoning from micro level to macro level		√	√			√	√

the word "landmark values" when saying that. When I introduced to him the terminology used in AI's qualitative physics' approach to quantizing continuous circuit parameters, he found it interesting and important.

For the simulation of rectifier circuits with sinusoidal input voltage, the professors all seemed ready to accept the configuration-wise method. They pointed out that this is a common approach used by electronic engineers to analyzing such circuits. Prof. A said that the textbook he once used has an identical diagram showing the two configurations of the bridge rectifier. He noted that topological configuration is vital not only for qualitative reasoning, but also for quantitative reasoning. He, however, pointed out that a configuration is only approximately linear. Linearality to him means straight lines. A circuit is linear if it only contains linear elements. A linear element is one in which a plot of voltage across the element against current through the element yields a straight line. He said that the relationship between voltage and current across a diode when forward-biased is not exactly a straight line, although it is generally approximated as a straight line.

They all agreed with ARC's structural aggregation that generated black boxes to suppress irrelevant detail to explain circuit behaviors. Prof. C remarked that circuit analysis can become extremely complex unless the circuits are simplified to reduce the number of meshes involved. Before we discussed ARC's output using White and Fredricksen's circuit (Figure 55 below), I asked each of them to explain why the light bulb becomes dimmer if the resistance of R5 increases. Prof. B's informal explanation went like this:

"Since all these (circling *R1*, *R2*, *R3*, and *R4*) are not changing, if the resistance here (pointing to *R5*) increases, then the current through the light bulb will decrease." He used the phrase "all these..." as his aggregation of individual components *(R1*, *R2*, *R3*, and *R4)* as a single entity in his explanation.

When I asked him to formally explain it with regard to the power source at the pole nodes, he started with almost exactly the same route until he came to the point where the

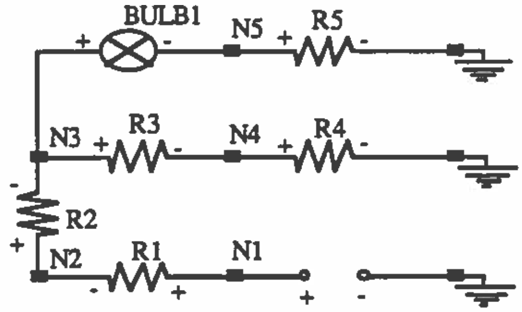


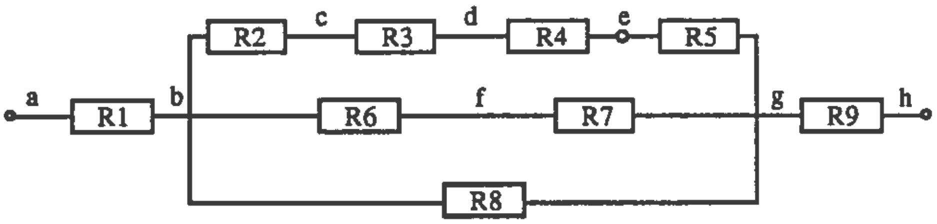
FIGURE 55. A Light-Bulb Circuit

voltage between *N3* and *N5* increases. He said that this voltage applies to both the parallel branches. Unfortunately, he did not finish the explanation. He realized that it was a tricky question. Why should the light get dimmer, whereas voltage across the parallel region increases? When we examined ARC's answer that used Kirchhoff's Current Law to disambiguate the situation and concluded that the current through the light bulb decreases, he indicated he was impressed.

Prof. A felt the ability to recognize parallel and serial constructs in a circuit very vital. He showed great interest in ARC's ability to recognize the parallel and serial structure of a circuit. We tested the circuit as shown in Figure 56 for our discussion. When Prof. A was told that ARC can recognize the parallel and serial constructs for different pole nodes, he suggested using node *e* and *h* as the positive and negative poles. He was enthusiastic enough to try out the solution on paper himself.

To demonstrate ARC's performance to answer the question posed by Prof. A, we gave ARC the following task:

- Target System: [p-circuit4]
- Input Specification: (voltage (e h) (+ -))
- Output Specification: (current (b g))




```
(define-circuit! p-circuit4
  ;; devices =
        ((resistor: r1 r2 r3 r4 r5 r6 r7 r8 r9))
  ;; connections =
        ((a  (terminal1 r1))
         (b  (terminal2 r1) (terminal1 r2) (terminal1 r6) (terminal1 r8))
         (c  (terminal2 r2) (terminal1 r3))
         (d  (terminal2 r3) (terminal1 r4))
         (e  (terminal2 r4) (terminal1 r5))
         (f  (terminal2 r6) (terminal1 r7))
         (g  (terminal2 r5) (terminal2 r7) (terminal2 r8) (terminal1 r9))
         (h  (terminal2 r9))))
```

FIGURE 56. A Circuit for Structural Aggregation

For the given input and output, ARC generated the following configuration (Figure 57):

```
((s.region9 (e h)
        ((p.region8 (e g)
                ((s.region7 (e g)
                        ((blk.box11 (e b))
                         (blk.box10 (b g))))
                 (r5 (e g))))
         (r9 (g h)))))

: blk.box11 = ((s.region12 (e b) ((r4 (e d)) (r3 (d c)) (r2 (c b)))))
: blk.box10 = ((p.region6 (b g)
                ((s.region5 (b g) ((r6 (b f)) (r7 (f g))))
                 (r8 (b g)))))
```

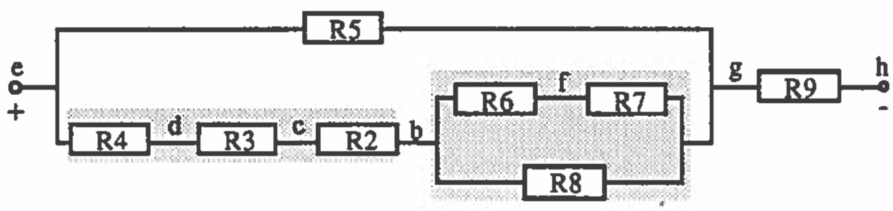


FIGURE 57. Aggregation Based on Chosen Pole Nodes in Circuit Topology

It took Prof. B a while to come up with the same solution, which he compared with ARC's answer. His solution was graphic. He wished that ARC could have produced some graphics. He did not like to read the list representation for circuit structures.

When asked how to aggregate structural constructs, like the Wheatstone (1802) Bridge, that cannot be simplified by parallel-serial reduction, Prof. B said that it is generally a very difficult problem. He did not think that the Wheatstone Bridge should be further simplified. But Prof. C insightfully pointed out that the whole circuit can be viewed as a single resistor, where current flows in at one end and comes out at the other end. From his electronic engineering background, he said there are methods to formally find an equivalent but simpler circuit. He pulled out an electronic engineering textbook [Ryder 1949] and pointed out the relevant discussions. A circuit is equivalent to another if the first circuit can be substituted for the second without change in the currents and voltages appearing at the terminals. Briefly, any delta circuit can have a T equivalent circuit [Ryder 1949], as shown in Figure 58-(1).

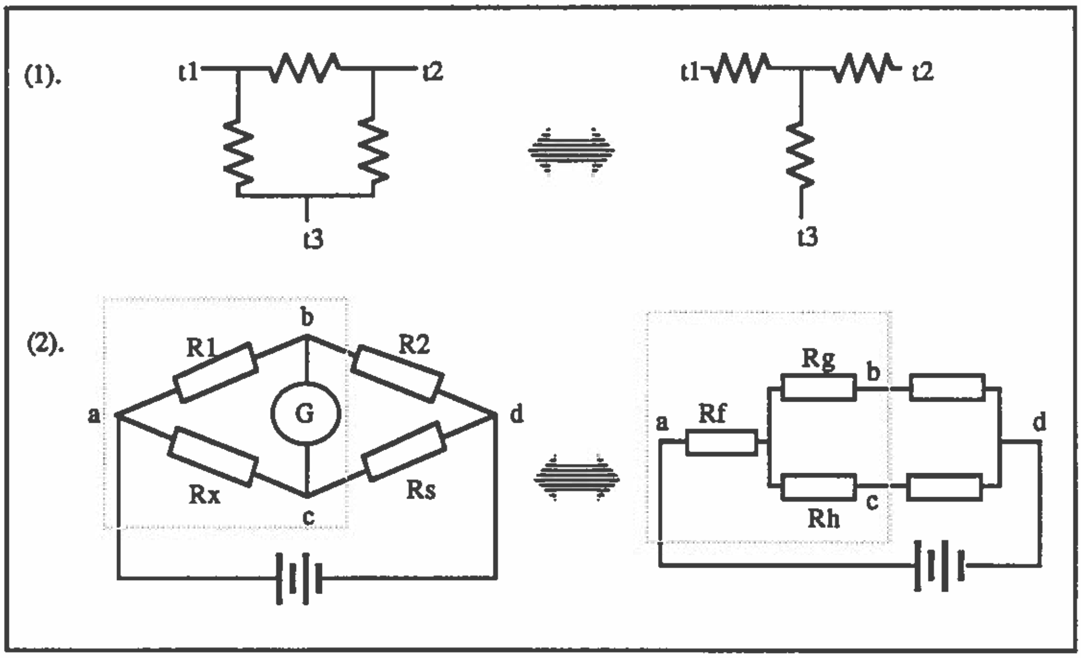


FIGURE 58. Equivalent Circuit of Wheatstone (1802) Bridge

For the Wheatstone bridge, if there is no current flowing through the galvanometer, then the circuit can be easily simplified by parallel and serial reduction. If there is current through the galvanometer, he suggested that we can think of it as a resistor. Then the Wheatstone bridge can have an equivalent circuit as shown in Figure 58-(2). This new circuit can now be simplified by parallel and serial reduction. But he admitted that the equivalent circuit is no longer the Wheatstone bridge.

While agreeing to the fact that physics divides the discussion of electricity in both macroscopic and microscopic levels, Prof. A pointed out that the distinction is not that clear-cut to him. He said that the bridging relations in ARC could also be regarded as axioms. Prof. C, however, liked the division between the macroscopic and microscopic levels. He emphasized that it is important to understand the levels. Prof. A and Prof. B seemed to like ARC's explanation using both the device ontology and the CC ontology.

When asked to explain at the microscopic level why the current through a resistor decreases when its length increases, the first reaction of all three was that it is because the resistance to the charge-carrier movement increases. They felt it natural to tie the physical shape (length and cross-section) directly to the resistance of a resistor. When I changed the question to "Why does the resistance increases when the length of the resistor increases?", they did not seem to have a ready answer. Prof. C offered an explanation using the water flow analogy to explain charge carriers bumping into molecules inside the resistor. He said that this is why a resistor gets heated. He used the Willamette River[14] as an example. Assume the height of water fall does not change. When the length of the river path increases, the water hits more rocks and thus loses energy. This is why the flow decreases. Interestingly, he was still using the scenario that the water experiences more *resistance* caused by the increased number of rocks. His water flow analogy did not seem to have explained well why the current decreases at the microscopic level.

---

[14] The Willamette River flows through Eugene, Oregon — the home of the University of Oregon. Some sections of the river are white water due to the rocks in the river.

Professors A and B preferred ARC's explanation that relates the current with the drift velocity of charge carriers. Their arguments were that given the cross-sectional area remaining unchanged, the change of flow can only be related to the drift speed of the charge carriers passing through a cross-sectional area of the resistor. Prof. A wrote down the equation $V = EL$ (Voltage = Field * Length) to back up his point.

Prof. C argued that when we talk about electric fields, we usually involve some spatial description of electronic devices. He said that it is not advisable to use electric fields to explain the behavior of current flow. He supported his point by citing the recent super-conductor research. When the length of a super conductor increases, the field decreases but the current through the super-conductor does not. In ARC, a wire connecting two devices is generally represented as a node. The spatial aspect of the system topology of a circuit is not modeled. Prof. C pointed out that while most textbooks adopt the same approach, we need to consider the spatial relations between devices which my interfere with each other though electric and magnetic fields.

The overall impression I got from interviewing physicists is that ARC has captured some tacit knowledge physicists use in analyzing electronic circuits. It is encouraging to know that they showed interest in seeing it automated. In addition, they seemed to appreciate the research effort of qualitative causal reasoning to analyze physical systems from structure to behavior. It showed that though the formalism of physics is mostly based on quantitative methods, physicists use a lot qualitative, common-sense knowledge to explain physical phenomena.

CHAPTER VII

RELATED WORK AND DISCUSSION

This chapter reviews previous work in qualitative physics related to the current research. We first review the fundamentals of qualitative physics. We then discuss the work in the areas of ontological choice, multiple models, and structural aggregation, as applied to qualitative causal analysis and automated modeling.

Fundamentals of Qualitative Physics

To reason about a physical system, one may start by using physics. While physics remains one of the crowning successes of the scientific method [Forbus 1988], it has serious limitations for automated reasoning because the tacit and common-sense knowledge, upon which its formalism relies, is implicit and not available for reasoning. Consider what we need to know to get by in the real world. We carry out most of our daily activities interacting with the physical world without the myriad numerical calculations. Even when numerical equations exist for some problems, people who know nothing about them can often reason fluently in the situation [de Kleer and Brown 1984]. When we do use quantitative methods to problem-solve in the real world, we first have to decide which equations to use based on our understanding of the meaning of each of the variables involved and the relations among them. This knowledge, mostly qualitative and causal, is critical for predicting and explaining the behavior of real-world physical systems.

The studies by Larkin (1983), de Kleer and Brown (1984), Forbus and Gentner (1986), Kuipers (1986), and White and Fredricksen (1986) have shown that qualitative causal reasoning is not only a central and coherent aspect of human mental life but also of expertise. Qualitative physics, as an active area in Artificial Intelligence research, arises from the desire to share this expertise about the physical world with computers. Seeking to represent, simulate, and explain physical systems through symbolic, non-numeric techniques and supplementing traditional physics, qualitative physics aims to lay bare the insights and tacit knowledge of scientists and engineers and make them sufficiently explicit and formal, so that they can be directly reasoned with and about [Forbus & Gentner 1986, Kuipers 1986, Forbus 1988].

Lumped-Circuit Approximation and Causal Analysis

In physics, circuit behavior is formally described by ordinary differential equations in equilibrium models involving carefully selected circuit-parameters. For example, the behavior of a circuit can be described in terms of Maxwell equations. As de Kleer (1984) and Williams (1984) point out, these quantitative equations rely on the lumped-circuit approximation that all input signals propagate in zero time and all circuit quantities change simultaneously when an input perturbation is applied to one of the circuit parameters. The quantitative equations do not account for causalities underlying changes of circuit behaviors. This is because they do not take into account how an input perturbation causes a disequilibrium in the components near the input and how the disequilibrium propagates through the circuit at a finite speed until the circuit reaches a new overall equilibrium.

de Kleer (1984) and Williams (1984) observe that when an electrical engineer reasons about a circuit, he or she re-introduces a kind of causality that the lumped-circuit model throws away by using local causal models and imposing a time flow on the changes in circuit quantities. This is crucial especially when the analysis concerns feedback[13] or nonlinear behavior of the circuit. As Williams (1984) points out, the causal models are not based on Maxwell equations, rather each circuit component is viewed as an active agent

[13] Feedback cannot be directly observed in parameters' values and cannot be analyzed with quantitative methods [de Kleer 1984, Williams 1984].

which attempts to re-establish a local equilibrium within its own vicinity in the circuit topology. The changes in and around the circuit components are partially ordered in a time sequence in which each change is caused by changes earlier in the sequence and earlier in time. Such sequential descriptions are ubiquitous in engineers' verbal and textbook explanations. de Kleer (1984) insightfully notes that during the equilibrating process and causal propagation, quantitative equations do not apply.

Since causal analysis of electronic circuits concerns what happens during the period of disequilibrium, the reasoning is accomplished by perturbing a declared input parameter, and using the laws associated with devices and interconnections to propagate effects throughout the system [de Kleer 1984, Williams 1984].

Reasoning from Structure to Behavior

The virtue of causal analysis is to reason from structure to behavior to explain qualitatively how a physical system works [Davis 1983, Bobrow 1985]. The structure of a physical system is generally described in terms of a set of parameters and the constraints that hold among them: essentially a 'qualitative differential equation' (QDE) [Kuipers 1986]. The behavior description consists of a discrete set of time-points, at which the values of the system parameters are described in terms of the qualitative value (Qval) and the direction of change (Qdir), as formalized below [Kuipers 1986, Weld 1989]:

$Qval(P,t)$	=	P_j	if $P(t) = \text{landmark } P_j$
		$P_{j,j+1}$	if $P_j < P(t) < P_{j+1}$
$Qdir(P,t)$	=	increasing	if $\partial P(t) > 0$
		steady	if $\partial P(t) = 0$
		decreasing	if $\partial P(t) < 0$

Kuipers (1986) summarizes six basic types of relations for qualitative analysis. They are time-differentiation, addition, subtraction, multiplication, and monotonic increasing functions, and monotonic decreasing functions [Kuipers 1986]. He points out that a variable reaches a transition when its qualitative value changes to or from one of the landmark values considered of interest to the analysis. A system is said to reach a transition when any parameter transitions. The system state changes whenever the state of any variable changes. Thus, the behavior of a system is a sequence of the system states alternating between distinguished time points and intervals.

A qualitative representation is an abstraction over its quantitative counterpart. As such, it is capable of describing a wide range of behaviors [de Kleer & Brown 1984, Kuipers 1986]. On the other hand, it tends to be ambiguous at times when exact values of system variables are desired. Note that this ambiguity of qualitative representation is a feature, not a bug. That is what "abstraction" means. The ability of qualitative physics to represent this ambiguity explicitly is sometimes beneficial because it can handle not only imprecise and incomplete information, but also provide a signal to indicate when more detailed knowledge is required. The abstract nature of qualitative representation often requires considering alternate possible behaviors, in the form of an 'envisionment' [de Kleer & Brown 1984], in qualitative simulation. Envisioning generates all possible behaviors, postponing control issues.

In qualitative causal analysis, time is individuated by the occurrence of interesting events, rather than some regular, fixed increment as used in numerical simulation [Forbus 1988]. A shared criterion of causal reasoning is that the behavior of a physical system is compositional. That is, the description of a system's overall behavior must be derivable from the interactions of the behaviors of the components through specified structural connections in the system [de Kleer & Brown 1984]. This differs from the traditional expert system approach using large collections of rules based on empirical associations. Davis (1983) and Forbus (1988) call it "reasoning from first-principles" to promote a general theory of modeling. Reasoning from first principles refers to a system's schematics for deriving the behavior of the system as opposed to the empirical associations specific to past experiences.

White and Fredricksen (1986) distinguish tasks that reason on the basis of the mere presence or absence of current, resistance, voltage, which are called "zero-order" reasoning, from those that reason on the basis of changes in current, resistance, and voltage, which are called "first-order" reasoning because they involve the first-order derivatives or change. These "zero-order" and "first-order" models correspond, respectively, to what de Kleer and Brown (1984) call "quiescent" and "incremental" models. For linear resistive circuits, the "zero-order" (quiescent) and "first-order" (incremental) models are identical [de Kleer & Brown 1984]. The "order" only indicates the mode of reasoning for a particular task.

The Modeling Problem

The structure and behavior of a physical system can be understood and described from different perspectives [Davis 1984, Falkenhainer & Forbus 1988]. As Smith (1985) points out, to model a physical system is to conceive of it in a certain delimited way. You identify the objects you are interested in, the properties you care about, the relations between them that matter, etc., and discard the rest. Davis and Hamscher (1988) observe that a model, inevitably and in principle, can never be complete. One can only hope for the good news that the things the model fails to capture about a target system may have no pragmatic consequences to the task at hand. The less good news is that, for various analytic tasks, our program may reason from a model that does not capture the appropriate perspective of the system. This issue is real and crucial to the robustness of our programs, and exerts an unavoidable impact on model-based reasoning in general.

The intriguing research problem is thus brought into focus by acknowledging the fact that all model-based reasoning is only as good as the model [Davis & Hamscher 1988]. Hobbs (1985), Davis and Hamscher (1988), Addanki, et al. (1989), Weld (1989), and Falkenhainer and Forbus (1988, 1991) propose that to achieve robust performance when reasoning about complex physical systems, our programs must do what human experts do:

use multiple models that embody distinct perspectives and dynamically select the one appropriate to the task at hand.

In the following sections, we review previous work with respect to the three dimensions of perspective-taking we consider in this research.

Ontological Choice

Rescher (1978) notices that whatever representational formalisms we use to describe the world, a description embodies a set of individual entities, which we use to designate the true nature of things in the world and our conceptions of them [Rescher 1978, Hayes 1985]. However, many problems in philosophy turn on the debate of how we determine the existence (or being) of the individual entities in the world and there is no universal agreement [Munitz 1974, Rescher 1978].

The world is initially unlabeled. Mervis, et al. (1981) points out that the real world is structured. Real-world attributes do not occur independently of each other. For instance, creatures with feathers are more likely to have wings than creatures with furs [Mervis 1981]. In other words, the choices of the entities in an ontological perspective are not random but reflect a particular underlying structure for the domain.

Hayes (1985) argues that division of the world into separate entities is made by us and our language, not by nature. He criticizes a naive ontological realism that things just are by pointing out that the actual world does not have a priori individual entities at all. The universe can be conceptualized as a continuum, perhaps a huge quantum-mechanical wave function [Hayes 1985]. The carving up of the universe's space-time fabric into individual entities is done by us for setting up a convenient conceptual framework of the world.

Like philosophy, qualitative physics is concerned with the operations of reasoning and with the entities that make up the true nature of things. But qualitative physics approaches the issue of ontology from a much more practical light. Qualitative physics views ontological choice as providing an organizing structure for modeling a physical system in terms of a specific language [Hayes 1985, Davis & Hamscher 1988, Forbus 1989]. Such a language involves a particular set of terms, predicates and axioms as the primitive elements for modeling and subsequent analysis of the physical world. As Forbus (1988) notes, developing an appropriate ontology in qualitative physics is usually the most difficult part because it shows how well we can describe and formalize a physical domain.

Hayes's Ontologies of Liquids

Hayes (1985) introduced two classic representations for liquids. He points out that sometimes an engineer must think of "the liquid in a container" as an object while also reasoning about a hypothetical collection of molecules traveling together through the system as an object. Likewise, a river may be viewed either as a static container of water defined by its banks, i.e., the same river it was a century ago, or as a dynamic collection of little pieces of stuff, each of which retains its identity as it flows to the sea [Hayes 1985].

Hayes points out that considering the liquid in a container as a single object leads to the contained-liquid ontology. This object has a continuous quantity "amount" and it may appear or disappear, as when a cup of tea is emptied and refilled. On the other hand, considering the liquid as a collection of molecules leads to the piece-of-stuff ontology. The collection of molecules has a fixed mass and a continuous position in space, which is influenced by various forces acting upon the object. Unlike a contained-liquid object which may appear or disappear, a piece of stuff is never created or destroyed for conservation of matter. Hayes argues that neither ontology alone suffices to explain common-sense reasoning about liquids.

Forbus (1988) notes that much qualitative physics research has focused on applying and generalizing Hayes's contained-liquid ontology as the contained-stuff ontology. Recent attempts are also made to specialize the piece-of-stuff ontology as the Molecular-Collection (MC) ontology [Collins & Forbus 1987] to reason about thermodynamic cycles of liquids. A variety of behaviors of complex physical systems can be explained when these ontologies are intertwined.

Device and Process Ontologies

Mirroring the engineering paradigm of system dynamics [Shearer 1971], a theory based on the device ontology is oriented toward modeling a physical system in terms of its component devices and their interconnections. The objects for modeling are system components such as resistors and diodes, or valves and tanks in the system, each described only in terms of its internal laws to adhere to the "no-function-in-structure" principle [de Kleer & Brown 1984]. The internal laws are generally represented as QDEs ("confluences"), often decomposed into distinct states or operating regions. To model a system, one builds a network of devices, which is then analyzed by using the QDEs from the devices and the device interconnections through constraint propagation. An important assumption made by the device ontology is that flow of information in the model reflects flow of causality in the real world.

In contrast, a theory based on the process ontology [Forbus 1985] views changes in the physical world as caused by explicit causal mechanisms called processes [Forbus 1988]. To model a system, one identifies a set of processes, such as "flowing" and "boiling", that account for the salient features of the behavior of the system. A process involves a specific set of entities and a set of preconditions and constraints on these entities. Such a process is not a property of any physical object, but a new type of entity to describe the behavior of a physical system. These explicit causal mechanisms act as the "agencies of causation". When a change occurs, the process observes specific physical laws and constraints, and its influences may activate other processes causing further changes in the system.

The device ontology and the process ontology provide two distinct guidelines for modeling physical systems in qualitative physics [Forbus 1988]. The device ontology is particularly suitable for modeling those systems such as electronic circuits where the need to model system-components and their interconnections is prominent. But it lacks the facilities to abstract out the local features into system-level descriptions. In contrast, the process ontology shows strength when modeling phenomena such *bending*, *boiling*, and *flowing*, which have a sense of on-going process [Forbus 1988]. A process also abstracts out the salient features of a system for organizing our knowledge of the world. Forbus (1989) notes that the process ontology, however, seems incapable of describing certain domains, e.g., electronics, where the distinction between endogenous and exogenous parameters varies according to the kind of analysis being performed. For example, a common occurrence in circuit analysis is that a change of the potential difference can cause the current to change, and vice versa [de Kleer 1984, Forbus 1989]. It would be hard to write process descriptions for these cases [Forbus 1988].

Shifting Ontological Perspectives

Molecular-Collection and Contained-Stuff Ontology

Most qualitative models built to date use either the device ontology [de Kleer 1984, de Kleer & Brown 1984, Williams 1984, Douglas & Liu 1989] or the process ontology [Forbus 1985, Forbus 1986, Weld 1986], describing only the macroscopic features of physical systems. As the first ground-breaking work in implementing the piece-of-stuff ontology, Collins and Forbus (1987) extend Hayes's piece-of-stuff ontology by showing how it can be used to produce a simplified model of a compression refrigerator [Collins & Forbus 1987]. They introduce the molecular-collection (MC) ontology as a specialization of Hayes's piece-of-stuff ontology for reasoning about thermodynamic cycles. They define a little piece of stuff, MC, which is large enough to have macroscopic properties such as temperature and pressure yet small enough never to split up when traversing a fluid system.

A difficulty introduced by two ontologies in a system is how and when to shift between the two. In Collins and Forbus's work, the ontological shifts from the contained-stuff ontology to the MC ontology follow if-then rules, such as the one shown below (Collins & Forbus 1987, p. 600):

if Boiling (sub, container) then
Transition (MC, STATE, GAS)

if Condensation (sub, container) then
Transition (MC, STATE, LIQUID)

These rules, associated with particular processes, such as *boiling* and *condensation*, describe how the MC's state changes as a consequence of activating these processes. The rules act as bridges between the contained-stuff ontology and the pieces-of-stuff ontology. Complex conclusions such as thermodynamic cycles can be drawn using this approach.

Unfortunately, the MC ontology cannot by itself describe the microscopic properties of its own as its name of "molecular collection" would suggest. Collins and Forbus notice that the MC ontology is parasitic on the contained-stuff ontology [Collins & Forbus 1987, Forbus 1988] because the MC descriptions can only be instantiated by the processes at the contained-stuff level. It relies completely on macroscopic-level reasoning to determine state changes and other information of interest.

By its essence, an ontology ought to be able to provide an organizational structure for modeling a target system. Different ontological choices of a domain should normally have distinctive set of primitives to describe a target system and rules of inference to use in reasoning. It is thus undesirable for one ontology to be parasitic on another. The absolute dependence of one ontology on another makes the dependent one inaccessible without the other. Sometimes, it indicates our incomplete understanding of the problem world. In addition, ontological shift involving parasitic ontologies allows only one-way mapping.

Multi-level Modeling of Populations

Franz and Weld (1990) recently attempt to reason about populations by summarizing the properties of the individuals at the microscopic level up to the macroscopic system as a whole. The summation aggregates the microscopic properties using statistical operators such as Σ, *mean*, *min*, and *max* [Franz & Weld 1990]. These statistical operators play the role of linking the micro level (individuals) to the macro level. For example, the summation of kinetic energy over all molecules of a gas in a container defines heat at the macro level. The mean kinetic energy of the molecules proportionally defines the temperature [Franz and Weld 1990].

Different from Collins and Forbus's work that converts from the macroscopic level (the contained-stuff ontology) to the microscopic level (the MC ontology), this work converts in the opposite direction, from the micro level to the macro level. Changes in the individual properties due to interactions at the individual level are summarized and aggregated to the higher levels by statistical operators. Similar to Collins and Forbus's work, it only allows a one-way transition due to the nature of statistical operators. Franz and Weld point out that the transition from the micro level to the macro level reflects our intuitions about causality. While their work represents just a start in this direction, it promises important insights in the future.

Task-Driven Ontological Shift?

Is ontological shift task-driven in these systems, as in ARC? To the extent that each of these systems was set up to run only a specific task, the choice of which ontology to select and when to shift from one ontology to another was made by the program designer, not the reasoning process in the program. These systems did not have explicit tasks given as input and, therefore, could only follow a pre-defined reasoning path. For example, Collins and Forbus's "if-then rules" for ontological shift are event-driven, rather than task-driven.

They will always fire if the corresponding processes become active, regardless of what task is at hand.

Reasoning with Multiple Models

Graph of Models

Choosing an appropriate ontology can simplify the modeling task. But sometimes a model of an ontological choice can still be overwhelmingly complex due to a large set of possible parameters for describing a target system. Addanki, et al. (1989) and Weld (1989) have used an approach of declaratively representing large amount of domain knowledge in a graph of models (GoM) [Addanki, et al. 1989, Weld 1990]. In this approach, each node denotes a local model of the target system with a set of underlying assumptions. Each edge connecting two nodes represents a mapping from one set of assumptions to another. The goal of this approach is to simplify problem-solving by permitting analysis in a simple model that satisfies the underlying assumptions for the task at hand.

However, there is no explicit task definition given for model selection. As a result, Addanki, et al. (1989) and Weld (1990) both used an algorithm that involves three phases. First, an initial model is arbitrarily selected from the GoM. Then, analysis is performed using this model. Finally, the results are checked for discrepancies with the experimentally observed values.

The third phase, called "validation" [Weld 1990], is to ensure that the model being chosen is valid. A model is valid if the predicted results by the model match the experimentally observed values. In case the current model is invalid, a procedure for model switching is initiated. It is a discrepancy-driven search to determine whether any of the neighboring models in the GoM can account for the discrepancies. If it can, then program switches to the newly found model. If not, the program tries the next neighbor, and the neighbor's neighbor, until it finds a valid model.

If a valid model in the GoM is found, then the program has found an appropriate model for the task at hand. While the GoM paradigm introduces a theoretically sound framework to represent multiple models, it has serious limitations. A GoM is pre-enumerated and the size of a GoM is inherently exponential in the number of possible assumptions one could have about a problem domain. Since model formulation is not part of the reasoning process for a given task, selecting a valid model in the GoM is solely based on trial-and-error search, as driven by the discrepancies found.

Consider-Assumptions

Falkenhainer and Forbus (1988, 1991) have suggested an alternative approach to a graph of models. They note that as the size of a complex system description gets increasingly large, complete instantiation of the system model becomes undesirable. They observe that since an analytic task usually concerns only part of a whole system [Falkenhainer & Forbus 1988], one method for ignoring irrelevant details is to "turn off" certain aspects in the system descriptions.

Instead of a static GoM, this technique uses a fine-grained modular approach to modeling. The fragments of a general domain model are attached with explicit modeling assumptions, each describing various aspects of the domain. Based on a query, it uses "consider-assumptions" to trim the domain model to generate one scenario model, which suffices to answer the query while minimizing extraneous detail. For example, in reasoning about a steam plant, one may use "CONSIDER(thermal-properties)" to focus on the thermal properties of the target system by ignoring the rest of the properties such as volumetric properties of the contained stuff in the system [Falkenhainer & Forbus 1988]. As in ARC's task definition, the terms in the query provide significant constraint in identifying a set of modeling assumptions and associated model fragments.

By alternately going through all consistent sets of consider-assumptions, one may produce a set of local models of the same target system. In contrast to a pre-enumerated GoM, the local models can be dynamically generated. Considerable space savings can be realized this way. Using "consider-assumptions" thus allows the system to focus on a particular local model for the analytic task at hand. "Consider-assumptions" suggest a promising task-driven approach to perspective-taking.

However, for each task, their system only generates one scenario model. By contrast, ARC can generate several models for each task and reason in the space of possible dimensions, shifting perspective as commensurate with the needs of the task. In addition, a difficulty brought out by using consider-assumptions is that it is hard to be sure that some of the assumptions in a scenario model are mutually compatible. For example, one "consider-assumption" may ignore 'friction' while another may depend on it. In light of ARC's experience, it appears that this conflict can be avoided if we allow only one position taken on any one dimension of perspective-taking.

Abstraction with Structural Aggregation

Since the behavior of a physical system is dependent on the structure of the system, if we can simplify the structural descriptions in a model, we can expect to suppress irrelevant details of the complex interactions in the original structure, thus to simplify the behavioral descriptions of the system.

A physical system can be partitioned structurally into aggregated pieces which are more or less disjoint and which together cover the entire system. For causal analysis of a complex physical system containing a large number of structural components, one strategy is to deal with aggregated structural units rather than the original individual components. If analysis can be carried out with less structural units than the original components, it is clearly a computational advantage to re-cast the system description in terms of the aggregated structural units. This reduces the conceptual complexity of the structure of the system. For example, a complex system with a large number of components may not be immediately understood in its entirety (by Gestalt), but it may be easily understood in clusters of physically-connected components.

Slices of Electronic Circuits

In reasoning about electronic circuits, Sussman and Steele (1980) have introduced slices, a notation for formally describing the view of equivalent circuits used in electrical engineering [Sussman & Steele 1980]. For example, two resistors in series can be replaced by a single resistor whose resistance equals the sum of the two in series. This way of viewing a group of components as a high level unit leads to a simplified view of a circuit. Sussman and Steele (1980) focus on quantitative reasoning using numerical constraints associated with slices to avoid algebra.

Similarly, Davis (1984) and Genesereth (1985) also use hierarchical structural description of a physical systems for troubleshooting electronic devices. Given a hierarchical structural description, reasoning can shift between levels of structural abstraction to achieve efficiency [Genesereth 1985] because at any particular level, the lower level details are suppressed.

However, the structural abstraction in these systems are defined by the program designers. In other words, the structural granularity of a target physical system is pre-defined and remains fixed regardless of what task is at hand. The programs are not equipped with any structural aggregation procedures to reformulate the structural description to suit the task at hand, as in ARC.

Cluster-Based Reasoning

Capturing the notion of structural aggregation for complex hydraulic systems, Farley [1988] has introduced a cluster-based reasoning technique in his system ORS. ORS was designed to handle pressurized systems that embody a large number of component devices such as valves, tanks, and pipes. A cluster in ORS consists of a maximal, connected subset of system components that does not contain a closed valve. Farley observes that valves, when closed, functionally isolate clusters of components in the hydraulic system. When such valves are opened, they merge neighboring clusters. A cluster in a hydraulic system can only be in one of two qualitative states: (i) stable, where pressures are equal throughout and no flow occurs, or (ii) unstable, where flow from high pressure source(s) to low pressure sink(s) occurs. Thus, a set of clusters partitions a complex system into functionally independent subsystems.

The dynamic grouping of components into clusters in a system introduces multiple structural granularities for reasoning about the system. The representation is shown to be useful in designing troubleshooting and standard operating procedures [Farley 1989]. To create a liquid flow past a certain location in a certain direction requires that one put that location in an unstable cluster on a flow path with a high pressure source and a low pressure sink on appropriate sides of the location in question. A cluster-based representation reduces the complexity of search for relevant flow paths because search can proceed in the cluster graph to determine a path between clusters having the desired pressure relationship. The complete flow path is determined by joining sub-paths that connect valves on cluster boundaries or that connect a valve with a source or sink of pressure. This shows reasoning at both levels of cluster and individual devices.

Summary

In this chapter, we have reviewed previous work related to this dissertation. We started out by discussing the fundamentals of qualitative physics. We then addressed the general modeling problem and discussed previous approaches to reasoning with multiple perspectives. While it is clear that significant progress to formalize common-sense reasoning and to advance qualitative causal reasoning about complex physical systems with multiple perspectives have been made, there remain several inadequacies which the current research seeks to address.

- Most systems do not have adequate task definitions given as input so as to extract crucial information for selecting appropriate perspectives to formulate or select models of the problem domain. As a result, most work assumes that appropriate models are given to the program as input. No attempts are made to integrate perspectives of multiple dimensions for automated reasoning. This limits the range of qualitative causal reasoning about complex physical systems.
- A parasitic ontology, though useful, is weak. Since it has terms, predicates, but no axioms, a parasitic ontology can not be viewed as a complete representational language for modeling. Ontological shift involving a parasitic ontology can only be a one-way mapping. Ontological shift using statistical operators suggests a promising framework. Unfortunately, a statistical operator only allows one-way conversion from the micro level to the macro level of behavioral description.
- The work of multiple local models that aims at reducing the parameter space of a complex physical system appears ad hoc. One problem is that no explicit organization structure is used to formulate and select models by analysis programs. As such, pre-storing a graph of models presents the challenge of how to search for a valid model from the graph. For the approach using consider-assumptions to trim a global model, the issue of how to detect incompatible assumptions underlying a model to avoid inconsistency becomes critical.
- Structural abstraction in the previous work, as it stands, is mostly static. Curiously, most approaches do not represent the physical organization of a physical system. Instead, the structural description of a target system is represented as a set of parameters and the constraints that exist among them. This kind of structural description assumes implicit, fixed configurations, rarely making any provision for dynamic topological configuration crucial for analyzing nonlinear systems.

The current research of ARC seeks to address some of the research questions discussed above. In the next chapter, we discuss the contributions as well as the limitations of this research, and outline some open problems for future work.

CHAPTER VIII

CONCLUSION

This research has developed a task-driven approach to perspective-taking for qualitative causal reasoning that integrates techniques of topological configuration, structural aggregation and ontological shift. This chapter discusses the contributions as well as the limitations of this work. Through evaluating this research, we revisit the major issues addressed in this dissertation, draw boundaries of the current framework and look beyond for future extensions.

Contributions

This section discusses four contributions of this research to the Artificial Intelligence field of qualitative physics. The primary contribution is an integrated framework of task-driven perspective-taking for qualitative reasoning. As sub-problems, we make three related contributions of techniques for configuration-wise simulation, task-driven structural aggregation, and ontological shift. Together, they provide a novel paradigm for qualitative causal analysis of complex physical systems.

Multi-Dimensional Perspective-Taking

The central theme of this dissertation is that model generation and selection is an integral part of effective reasoning about complex physical systems. The initial structural description of a target system given to ARC is the system topology representing the connectivity of component devices of a circuit. This is the only information given that is specific to a target system. No assumptions are made about the possible flow paths in the topology, whether it is a linear or a nonlinear system, or how black boxes are to be created. From the input and output specifications in a given task, ARC extracts crucial information to select appropriate perspectives of the target system and formulate associate models.

A model generated by ARC embodies a specific perspective of the target system. Such a perspective can involve single or multiple dimensions each reflecting a partial perspective. For example, after identifying active configurations, ARC reveals the different dynamic structures associated with a static system topology. Different configurations of the same target system involve different subsets of the system components, constituting one dimension of perspective-taking.

Structural aggregation over a given active configuration considers different ways to black-box system components. Different structural aggregations reflect different structural granularities. This constitutes a choice on another dimension of possible aggregations. An active configuration with an appropriate structural aggregation thus embodies a 2-D perspective.

Finally, a QDE model of a target system shows not only the active configuration of a target system with structural aggregation, but also the constraint relations among system variables in a chosen ontology. Different ontological choices for describing the behavior of an aggregated configuration constitute the third dimension we have investigated. Therefore, a QDE model generated by ARC represents a 3-D perspective.

When shifts in perspective become appropriate during qualitative simulation due to the need to consider new configurations, or different choices of structural granularity, or different levels of behavioral description, ARC dynamically generates and selects models embodying appropriate perspectives to carry out the task at hand.

A number of researchers in qualitative physics have described systems that reason with multiple models to great advantage [Weld 1988, Falkenhainer & Forbus 1988, Addanki et al. 1989, Weld 1990]. Their models, however, usually consider only one perspective dimension. In addition, most of the modeling work in qualitative physics assumes that a hierarchy of models is given to the program as input rather than dynamically formulating appropriate models for different tasks. This approach is effective only to the extent that all the possible models required by future tasks can be pre-defined, but incurs an enormous amount of storage and trial-and-error search for a valid model in a graph of models [Addanki et al. 1989, Weld 1990]. Pre-enumerating all possible models without clear knowledge of what the future tasks may be would be difficult, if not impossible. This is especially true in situations like constructive simulation environments for qualitative simulation [Douglas & Liu 1989] where the user is in control of constructing target systems and posing questions.

Recently, Falkenhainer and Forbus (1991) have proposed an interesting compositional modeling method that uses a fine-grained modular approach to modeling. The fragments of a general domain model are attached with explicit modeling assumptions. Based on a query, it uses "consider-assumptions" inferred from the query to trim the domain model to generate one scenario model, which suffices to answer the query while minimizing extraneous detail.

By contrast, ARC can generate several models for each task and reason in the space of possible dimensions, shifting perspective as commensurate with the needs of the task. This dissertation contributes by providing an operational framework that automatically formulates models embodying appropriate perspectives for effective qualitative causal reasoning for different tasks. It is precisely this ability of task-driven model formulation that offers ample opportunity for improving the flexibility and efficiency of automated reasoning systems.

Configuration-Wise Simulation

The techniques used in ARC to identify active configurations in the system topology of a target system provides new techniques for reasoning about complex physical systems. Since the qualitative behavior of each configuration is linear, this approach extends the present qualitative reasoning paradigm with the ability to automate reasoning about nonlinear systems through configuration-wise linearization. Through identifying active configurations for a given task, ARC recasts a nonlinear system into a series of linear configurations, performing zero-order analysis [White and Fredricksen 1986] of current or charge-carrier flow in the system topology. QDE models are then generated for each of the configurations from either the device ontology or the CC ontology for the first-order causal analysis.

To date, work in qualitative physics has not dealt effectively with problems of how to automate reasoning about nonlinear systems. Curiously, most systems do not have an effective structural description language to model the physical organization of a system. Instead, the structure of a physical system is described in terms of a set of parameters and the constraints that hold among them, essentially a "qualitative differential equation" [de Kleer & Brown 1984, Kuipers 1986]. This kind of structural description, having originated from quantitative equations, deal exclusively with constraints among system variables, rarely making any provision for topological configuration of nonlinear systems. Other work [Davis 1984, Williams 1984] that uses descriptions of the system topology - the connectivity of components, with possible hierarchical views - appears to assume a single static topological configuration throughout reasoning.

The traditional approaches to studying nonlinear systems include (1) theoretical methods, (2) experimentation and numerical simulation, and (3) piece-wise linear approximation. Theoretical methods are powerful, but they are rarely used by engineers due to their theoretical formalisms. Numerical simulations and experimentations are widely used, due to recent advances in computational power, but generate results which only experts can recognize and interpret. Piecewise linearization methods use a set of predefined linear approximations for analysis [Sacks 1987]. The results are plotted as graphs to show the qualitative characteristics of behaviors. However, the responsibility for constructing adequate piecewise linear approximations rests solely with the programmer.

In the spirit of piecewise approximation, current qualitative physics considers the behavior of a nonlinear device over several operating regions or qualitative states, each described by a different set of confluences or causal rules [de Kleer & Brown 1984]. But as de Kleer (1977) notes, since the correct state cannot be determined when a nonlinear device is examined by the simulation, the choice of which causal rules to use in any state can only be made under the assumption that the device is operating within that particular state. This assumption is used to envision all possible behaviors of the device. Unfortunately, that assumption simply postpones the problem of determining which operating state the nonlinear device is in. To date, most qualitative physics work avoids the issue of multiple configurations altogether by assuming that all behaviors are linear within a given set of confluences or causal rules. As a result, the structural aspects underlying nonlinear behaviors of complex physical systems are ignored.

Dynamic Structural Aggregation

This research has implemented commonly-used techniques of structural aggregation in circuit analysis. These serve two purposes for qualitative causal reasoning. First, it lets the reasoning program see the big picture of a target system by revealing the implicit structural hierarchy of the system. Second, it lets the reasoning program focus on a particular region of interest by creating black boxes suppressing unnecessary details to simplify qualitative causal reasoning. Structural aggregation in ARC is task-driven. For different tasks, ARC can view the same target system with different structural granularities.

Most work in qualitative physics does not have explicit description of the physical organization of a target system, as discussed above. However, some researchers, such as Sussman & Steele (1980), Davis (1984) and Genesereth (1985) have included notions of structural abstraction in their work by providing an explicit structural hierarchy of system constructs. The structural abstraction facilitates problem solving, but because those structural units are pre-defined and remain fixed during reasoning, they suit only a limited range of tasks.

Farley (1988) developed a cluster-based representation for pressurized hydraulic systems. His system dynamically computes clusters based on closed valves. The cluster representation is shown to be useful in designing troubleshooting and standard operating procedures [Farley 1989]. The structural aggregation techniques in the current work, borrowing ideas from the cluster-based reasoning, extends this work by focusing on a task-driven approach to aggregating system components. Although our structural aggregation research represents only an initial investigation, it reflects a promising direction for the current qualitative reasoning paradigms to scale up to deal with more complex physical systems.

Ontological Shift Paradigm

This research has developed a novel paradigm for ontological shift. It uses domain-independent ontological-choice rules to select appropriate ontological perspectives when formulating models for a given task. In this research, we view an ontological choice as a language with a set of terms, predicates and axioms for describing a problem domain. Although syntactic in nature, this approach makes ontological choice rules possible, by noting what terms, predicates and axioms are involved in a particular task. When ontological shift becomes necessary, ARC switches models among different ontologies via bridging relations.

To preserve spatiotemporal continuity for causal propagation during ontological shift, we introduced the structural compatibility principle for defining bridging relations, which associate elements of compatible structural entities from different ontologies. Based on this principle, spatiotemporal continuity of causal propagation is maintained when ontological shift takes place following these bridging relations.

Related work in using multiple ontologies includes Collins and Forbus's (1987) system that reasons about liquids from both a contained-stuff ontology and a molecular-collection (MC) ontology, and Franz and Weld's (1990) recent work on reasoning about populations from both macroscopic and microscopic levels. Collins and Forbus note that the MC ontology is parasitic to the contained-stuff ontology. Their bridging relations consist of rules for one-way conversion of process descriptions into MC descriptions. Specifically, the predicates and axioms of the MC ontology itself are not represented in their system. As a result, the overall reasoning is done only in the contained-stuff ontology. Franz and Weld's work uses statistical operators to aggregate the microscopic behaviors to the macroscopic level. Only one-way conversion is possible because only aggregation via statistical operators is used to link the two perspectives. Neither of the two systems uses any explicit ontological choice rules to control ontological shift. They simply follow a pre-defined reasoning path for lack of a task-driven approach.

In contrast, ARC does not have parasitic views of ontological perspectives. Reasoning can proceed in either ontology, as determined by the domain-independent ontological choice rules. The decision as to which ontology to use and when to shift from one to the other is based on the specific task at hand. When a shift is necessary from one ontology to another, reasoning follows one of the bridging relations. Because these bridging relations are represented as constraints, the ontological shift framework developed in this research allows two-way conversions for ontological shift during causal analysis.

Limitations

There are several limitations in the current work of task-driven perspective taking. Some of them were deliberately chosen so we could focus on the central issues of this research. Some of them are due to the present implementation choice within ARC. We discuss the limitations of this work in this section.

Structure, Behavior, Function

This research has focused entirely on qualitative causal reasoning from structure to behavior. The teleological aspect of physical systems regarding their purposes or functions are not addressed. This choice is made deliberately because we want to study and understand qualitative causal reasoning about a physical system with multiple perspectives from structure to behavior without being confined to only the purposeful behavior of the system, either as designed or perceived.

But, complete qualitative physics should involve the functional aspects of physical systems. For artifacts, the functional knowledge used in design often helps resolve severe complexities encountered in causal reasoning. Structural aggregation also can be done from a functional perspective as well. For example, a circuit can be aggregated as a black box and described as an *adder* or a *multiplexer*. One feasible way for doing this would be to access the design knowledge to guide the structural aggregation. A complete causal explanation of how a physical device works requires the knowledge of purpose. Recent work by Downing (1990) has examined teleological aspects of biological circulatory systems in an attempt to extract comprehensive explanations of complex systems.

Structural Abstraction

The structural aggregation techniques developed in this research allow automated reasoning systems to recast the original structural descriptions into representations favorable to the task at hand. However, this research has only examined structural aggregation based on parallel-serial reductions. While series-parallel circuits are the norm, there are circuits that cannot be aggregated using parallel-serial reduction, such as the full-wave bridge rectifier, as shown in Figure 59. Fortunately, the orientation of the devices (clusters) in the rectifier circuit allows ARC to recognize the active configurations for structural aggregation.

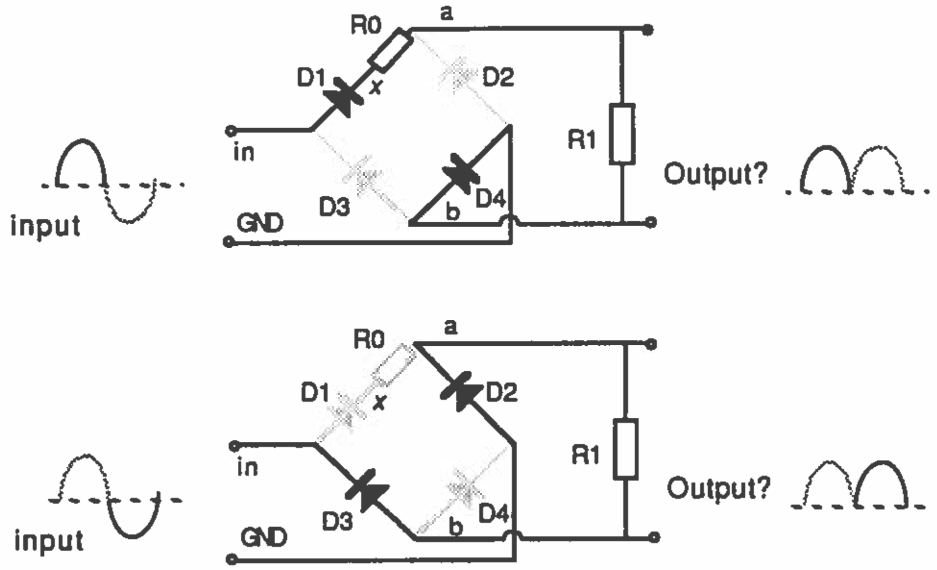


FIGURE 59. Two Configurations of Full-Wave Bridge Rectifier.

But sometimes, the orientations are not sufficient. The Wheatstone bridge circuit is one such example (Figure 60).

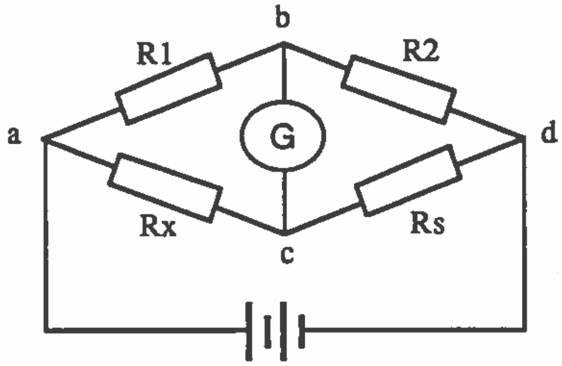


FIGURE 60. Wheatstone Bridge

When there is current between nodes *b* and *c* (in either direction), then the configuration of the circuit cannot be aggregated by parallel-serial reduction. Situations like this usually indicate the need for quantitative knowledge. Interestingly, the Wheatstone bridge was designed to take advantage of parallel and serial circuit principles. If there is no current flow between *b* and *c*, then the active configuration of the circuit is a perfect parallel-serial circuit with voltages at nodes *b* and *c* equal. In this situation, the bridge is said to be balanced [Weidner 1985].

But when the bridge is not balanced, parallel-serial reduction fails. The general qualitative physics solution in this situation is to generate an envisionment [de Kleer & Brown 1984] for all possible behaviors of the circuit. For example, Williams (1984) analyzed this circuit, using qualitative techniques, by considering three cases: (1) $Vb > Vc$, (2) $Vb = Vc$, and (3) $Vb < Vc$. Clearly, each case indicates a different configuration of the circuit.

Task Definition Language

The tasks given to ARC are queries made by the user, as is shown in all the examples in this thesis. Although ARC specifies tasks with both input and output, capable of large signal analysis, the current version of the task definition language is not sufficient to express tasks beyond the parameter-perturbation analysis problems.

Improvement of a task definition language like the one developed in this work will be tightly coupled with the advances of theories and techniques in automated reasoning research. In ARC, a task consists of terms in chosen ontologies with nodes implying the structural units in a target circuit. This suggests an important framework for task analysis with improved task definition languages in the future. Given a more general task definition language, how can its form be used to suggest appropriate perspectives for modeling and reasoning in automated reasoning systems?

Spatiotemporal Reasoning

The representation of charge carriers as pieces of stuff is rather limited in the current system, following the notion of Molecular-Collection (MC) by Collins and Forbus (1987). Complex analytic tasks require additional spatial and temporal reasoning about charge carriers as pieces of stuff. Some aspects of pieces of stuff reasoning for charge carriers are discussed in [Liu 1989, 1991], which show why a diode conducts current only in one direction, as in Figure 61. Tasks such as this require following charge carriers that travel in space and time under electric forces in electric or magnetic fields. This direction for future research offers the possible synergy with recent work on spatial and temporal reasoning [Throop 1989, van Beek 1990].

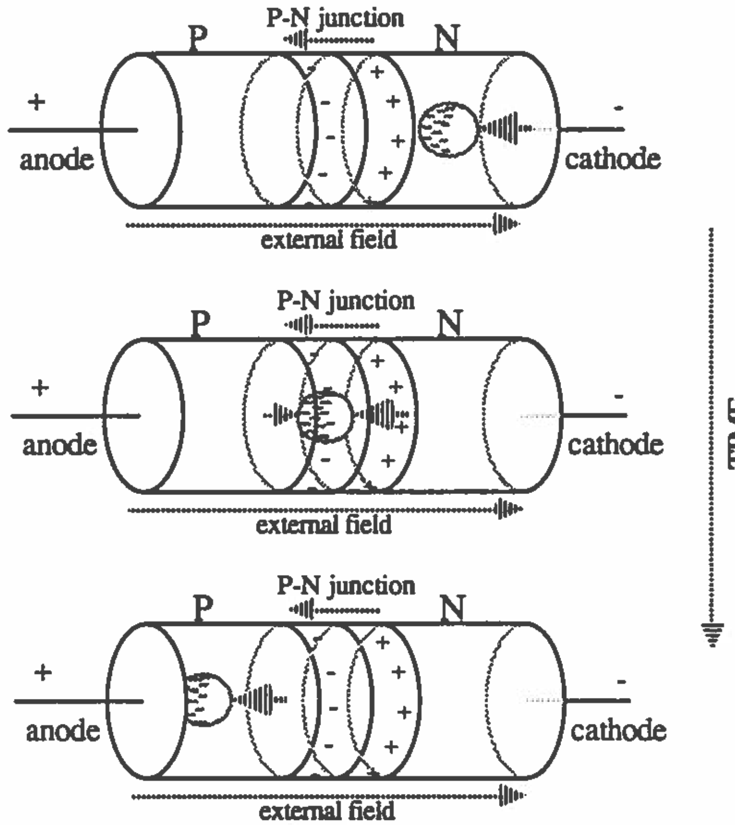


FIGURE 61. Spatiotemporal Aspect of Charge-Carrier Flow inside Diode

Envisionment and Control

The current implementation of ARC does not generate complete envisionment [de Kleer 1977] of all possible behaviors when ambiguity occurs. Envisioning intentionally ignores control issues in qualitative simulation by explicitly generating the entire search space [Forbus 1988]. The disadvantage of envisioning is that it could generate behavior not realizable in the real situation or not of interest to the analytic task at hand.

Instead of envisioning, ARC handles the ambiguity problem by turning the control over to the user and letting the user make a choice so that only behavior interesting to the user is examined. ARC only allows the user to make one choice in the situation and then focuses reasoning on this particular choice. The current implementation does not try to predict all the possible future behaviors. Part of the reason for doing so is that ARC can follow a given sequence of perturbations that drive the qualitative simulation. Our focus here was on integrating perspectives, not control of multiple envisionments. In some situations, however, it was felt that the user should be given the choice to turn on an envisioner to see all possible future behaviors of the target system at a time. This should be a rather straightforward extension to ARC.

Future Work

Beyond extensions mentioned above, the following areas offer interesting opportunities for new research.

Multiple Ontological Perspectives

In this research, we discuss only two ontologies — the device ontology and the CC ontology. The next step is to see whether the framework developed here can be expanded to more than two ontologies. For example, one may add a digital perspective to the present system, introducing a new language for describing and reasoning about electronic circuits, involving notions of logical gates, logic values, and boolean algebra axioms.

There are many interesting issues to be addressed. For example, given multiple ontologies, do we need pair-wise bridging relations between each pair of ontologies? It is probably domain-dependent. Without pair-wise bridges, ontological shift from one to another could involve crossing more than one bridge with the intermediate ontologies as islands. Given a particular task, do we choose the shortest bridge-path between the two or do we select a path that involves those islands deliberately sought out for explanation purposes? This appears to be task-dependent. What are the rules for controlling ontological shift in the presence of multiple ontologies without pair-wise bridging relations?

Behavioral Abstraction and Ontologies

We have addressed the issues involved in structural aggregation extensively in this dissertation. But we have not specifically discussed behavioral aggregation. To a certain degree, it appears that behavioral aggregation is closely related to introducing new ontologies of the same target system.

One difference between structural aggregation and behavioral aggregation is that the former does not change the vocabularies for describing and reasoning about a physical system while the latter usually does. For example, when aggregating a group of resistors as a single equivalent resistor, Ohm's Law is still applicable to describe its behavior. But if aggregating analog behavior of a circuit to digital behavior, we switch to a different vocabulary, or ontology, for modeling and reasoning.

Given a set of related ontologies, how to incorporate behavioral aggregation with ontological shift in automated reasoning systems is a challenging problem to study in the future.

Structural Perturbation with Comparative Analysis

The next possible extension to the current work would be to allow structural perturbation in addition to the parameter perturbation for a task. For example, if one changes the structure of a target circuit by either adding or deleting some device(s), what is the behavioral change as a result? Such reasoning would be valuable for design, diagnosis and instruction.

Reasoning about behavioral changes requires comparative analysis of two models [Weld 1988]. For structural perturbation, comparative analysis would compare the behaviors of the two models of the circuit before and after the structural perturbation takes place. Since deleting or adding devices to a circuit in simulation can instantly cause the configuration of the circuit to switch, comparative analysis in this case must also consider changes to the topological configuration as well. Studying this problem would advance the current theory of comparative analysis in qualitative physics.

Incorporating Qualitative and Quantitative Reasoning

In part, qualitative physics is motivated to provide a framework for organizing and using quantitative knowledge, such as numerical simulations. Though widely used in science and engineering, current numerical simulations has important limitations [Forbus 1990]. The most noticeable problem is that the underlying physical assumptions are implicit and unaccessible by the simulation program for explaining the behavior being simulated.

ARC is a promising framework to incorporate qualitative and quantitative reasoning about complex physical systems. ARC's ability to generate qualitative models can be extended to creating quantitative models for numerical simulation. These quantitative models will have full accesses to the information of the qualitative states in which they are generated and used. Since before a numerical simulation is run, the behavior of the target system has already been analyzed in qualitative causal terms, the qualitative knowledge can be used to interpret and explain the numerical simulation of complex physical systems.

I have begun exploring generating quantitative models in ARC with encouraging results. For example, piecewise linear quantitative equations can be generated for analyzing nonlinear rectifier circuits, with qualitative causal explanations accompanied by automated graph plotting. Numerical simulations in ARC will also be task-driven. Exciting work lies ahead in this area.

Instructional Systems

If used in an instructional setting, ARC's ability to identify topological configurations, recognize parallel and serial constructs, and reason with both the macroscopic and microscopic perspectives in electric circuits should help teach the basic principles of electricity, especially with its component-based, first-principle nature and task-driven reasoning style.

One of the major difficulties for elementary physics students is to recognize parallel and serial constructs in the circuit diagrams [Palies et al. 1986]. The trouble is caused by the geometrical features of the diagram. Figure 62 below shows two diagrams of the same circuit. The diagram on the left presents many elusive geometric loops and does not reveal the inherent parallel-serial connections as does the one on the right. Obviously, mapping the diagram on the left to the one on the right requires certain expertise.

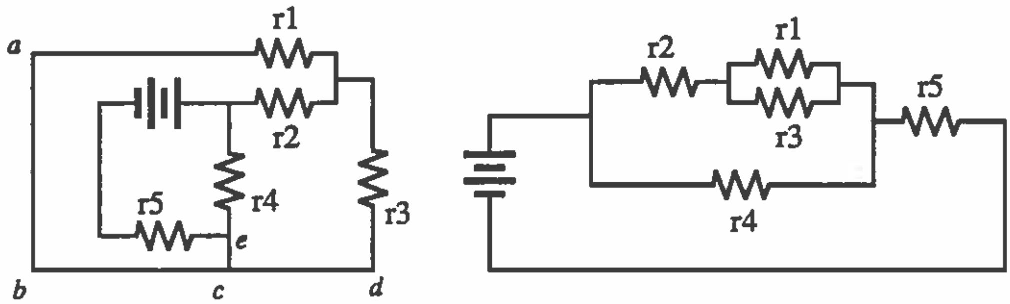


FIGURE 62. Different Drawings of One Circuit

Part of the problem is that novices see extra nodes that are actually the same potential node. For example, locations *a, b, c, d,* and *e* in the left diagram are the same potential node. Currently, ARC's representation of the system topology of a circuit only includes potential nodes. Clearly, locations that are connected to each other only by wires in a circuit diagram should be merged as one potential node during analysis. If enhanced by a graphic interface whereby circuit diagrams can be drawn, it will be interesting to extend ARC to recognizing the implicit structural hierarchy in a circuit diagram and graphically display topological configurations and structural aggregation with explanations in qualitative and numerical simulations.

Conclusion

In this research, I have investigated the general problem of how to automate reasoning about physical systems with multiple perspectives. I have begun specifying a theory of task-driven perspective-taking for effective reasoning about complex physical systems. The results of this research show that by using an integrated framework of topological configuration, structural aggregation, and ontological shift, we can extend the range of qualitative causal reasoning about complex physical systems.

BIBLIOGRAPHY

[Addanki, et al. 1989]
Addanki, S., Cremonini, R. and Penberthy, J. S., Reasoning about Assumptions in Graphs of Models. In Proceedings of IJCAI-89, pp. 1432-1438, Detroit, MI, August 1989.

[Allen 1983]
Allen, J., Maintaining Knowledge about Temporal Intervals. Communications of the ACM, (26):832-843, 1983.

[Arnborg & Proskurowski 1989]
Arnborg, S. and Proskurowski, A., Canonical Representations of Partial 2- and 3-trees. Tech-Report, CIS-TR-89-11, Department of Computer and Information Science, University of Oregon. July, 1989.

[Bell 1980]
Bell, D. A., Electronic Devices and Circuits. Reston Publishing Company, Inc., Reston, VA, 2 edition, 1980.

[Bobrow 1985]
Bobrow, D. G., editor, Qualitative Reasoning about Physical Systems. MIT Press, Cambridge, Massachusetts, 1985.

[Chen & Yang 1987]
Chen, L. S. and Yang, J. H., editors, Analog Electronic Circuits. China Science and Technology University (EE), Hefei, Anhui, China, April 1987.

[Collins & Gentner 1989]
Collins, A. and Gentner, D., Multiple Models of Evaporation Processes. In D. Weld and J. de Kleer, editors, Readings in Qualitative Reasoning about Physical Systems, pp. 508-512. Morgan Kaufmann, San Mateo, CA, 1989.

[Collins & Forbus 1987]
Collins, J. W. and Forbus, K. D., Reasoning about Fluids via Molecular Collections. In Proceedings AAAI-87, pp. 590-594, Seattle, WA, 1987.

[Davis 1983]
Davis, R., Diagnosis via Causal Reasoning: Paths of Interaction and the Locality Principle. In Proceedings of AAAI-83, pp. 88-94, 1983.

[Davis 1984]
Davis, R., Diagnostic Reasoning Based on Structure and Behavior. Artificial Intelligence, 24, 1984.

[Davis & Hamscher 1988]
Davis, R. and Hamscher, W., Model-Based Reasoning: Troubleshooting. In H. E. Shrobe, editor, Exploring Artificial Intelligence, chapter 8, pp. 297-346, Morgan Kaufmann, San Mateo, CA, 1988.

[de Kleer 1977]
de Kleer, J., Multiple Representations of Knowledge in a Mechanics Problem-Solver. In Proceedings of IJCAI-77, pp. 299-304, 1977.

[de Kleer 1984]
de Kleer, J., "How Circuits Work". Artificial Intelligence, 24:205-280, 1984.

[de Kleer & Brown 1984]
de Kleer, J. and Brown, J. S., A Qualitative Physics Based on Confluences. Artificial Intelligence, 24:7-83, 1984.

[de Kleer & Brown 1986]
J. de Kleer and J. S. Brown, Theories of Causal Ordering. Artificial Intelligence, 29(1):33-61, July 1986.

[Douglas & Liu 1989]
Douglas, S. A. and Liu, Z. Y., Generating Causal Explanation from a Cardio-Vascular Simulation. In Proceedings IJCAI-89, pp. 489-494. Detroit, MI., 1989.

[Downing 1990]
Downing, K. L., The Use of Teleology in the Qualitative Evaluation and Explanation of Circulatory Systems, PhD Thesis. University of Oregon, 1990.

[Falkenhainer & Forbus 1988]
Falkenhainer, B. and Forbus, K. D. 1988, Setting Up Large Scale Qualitative Models. In Proceedings AAAI-88, pp. 301-306, Saint Paul, Minn., 1988.

[Falkenhainer & Forbus 1990]
Falkenhainer, B. and Forbus, K. D, Compositional Modeling of Physical Systems. In Proceedings of 4th QP Workshop, pp. 1-15, July, Lugano, Switzerland, 1990.

[Falkenhainer & Forbus 1991]
Falkenhainer, B. & Forbus, K., Compositional Modeling: Finding the Right Model for the Job. To appear in Artificial Intelligence, special issue on qualitative physics, Fall, 1991.

[Farley 1988]
Farley, A. M., Cluster-Based Representation of Hydraulic Systems. In Proceedings of the 4th Conference on Artificial Intelligence Applications, pp. 358-364, San Diego, CA, March 1988.

[Farley 1989]
Farley, A. M., Diagnostic Mechanism Modeling. In International Journal of Pattern Recognition & Artificial Intelligence, Vol. 3, No. 2 pp.237-259, 1989.

[Forbus 1985]
Forbus, K. D., The Role of Qualitative Dynamics in Naive Physics. In J. R. Hobbs and R. C. Moore, editors, Formal Theories of the Commonsense world, chapter 5, pp. 185-226. Ablex Publishing Corp., Norwood, NJ, 1985.

[Forbus 1988]
Forbus, K. D., Qualitative Physics: Past, Present, and Future. In H. E. Shrobe and AAAI, editors, Exploring Artificial Intelligence, chapter 7, pp. 239-296. Morgan Kaufmann, Inc, San Mateo, California, 1988.

[Forbus 1989]
Forbus, K. D., Pushing the Edge of the (QP) Envelope. In Proceedings of the 3rd International Qualitative Physics Workshop, Palo Alto, CA, August 1989.

[Forbus & Gentner 1986a]
Forbus, K. D. and Gentner, D., Causal Reasoning about Quantities. Cognitive Science, pp. 196-207, 1986.

[Forbus & Gentner 1986b]
Forbus, K. D. and Gentner, D., Learning physical domains: Toward a theoretical framework. Technical Report UIUCDCS-R-86-1247, Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, IL, December 1986.

[Franz & Weld 1990]
Franz, A. and Weld, D., Multi-Level Modeling of Populations. In Proceedings of the 4th International QP Workshop, July, Lugano, Switzerland, 1990.

[Genesereth 1985]
Genesereth, M., The Use of Design Description in Automated Diagnosis, in D. Bobrow, editor, Qualitative Reasoning about Physical Systems, pp. 411-436. MIT Press, Cambridge, MA, 1985.

[Gentner & Gentner 1983]
Gentner, D. and Gentner, D. R., Flowing Waters or Teeming Crowds: Mental Models of Electricity. In D. Gentner and A. L. Stevens, editors, Mental Models, chapter 6, pp. 99-130. Lawrence Erlbaum Associates, Publishers, Hillsdale, New Jersey, 1983.

[Greeno 1983]
Greeno, J. G., Conceptual Entities. In D. Gentner and A. L. Stevens, editors, Mental Models, chapter 10, pp. 227-252. Lawrence Erlbaum Assoc. Inc., Hillsdale, NJ, 1983.

[Hayes 1985a]
Hayes, P., Naive Physics I: Ontology for Liquids. In J. R. Hobbs and R. C. Moore, editors, Formal Theories of the Commonsense world, chapter 3, pp. 71-107, Ablex, Norwood, NJ., 1985

[Hayes 1985b]
Hayes, P., The Second Naive Physics Manifesto. In J. R. Hobbs and R. C. Moore, editors, Formal Theories of the Commonsense World, chapter 1, pp. 1-36. Ablex, Norwood, NJ, 1985.

[Hayes 1989]
Hayes, P. (Conference discussion and personal communications). The 3rd International QP Workshop and IJCAI-89, 1989.

[Hobbs 1985]
Hobbs, J. R., Granularity. In Proceedings of IJCAI-85, pp. 432-435, 1985.

[Hofstadter 1980]
Hofstadter, D. R., Gödel, Escher, Bach: An Eternal Golden Braid, Vintage Books, New York, NY, 1980.

[Iwasaki & Bhandari 1988]
Iwasaki, Y. and Bhandari, I., Formal Basis for Commonsense Abstraction of Dynamic Systems. In Proceedings of AAAI-88, pp.307-312, Saint Paul, Minnesota, August 1988.

[Iwasaki & Simon 1986]
Iwasaki, Y. and H. Simon, A., Theories of Causal Ordering: Reply to de Kleer and Brown. Artificial Intelligence, 29(1):63-72, July 1986.

[Joobbani & Talukdar 1985]
Joobbani, R. and Talukdar, S. N., An Expert System for Understanding Expressions from Electric Circuit Analysis. In Proceedings of AAAI-85, Los Angeles, CA, August 1985.

[Keogh & Suntag 1981]
Keogh, J. E. and Suntag, B., Handbook of Simplified Radio, Phone, and Tape Recorder Repairs: An Illustrated Troubleshooting Guide, Parker Publishing Company, West Nyack, NY, 1981.

[Kuipers 1986]
Kuipers, B., Qualitative Simulation. Artificial Intelligence, (29):289-338, 1986.

[Kuipers 1987]
Kuipers, B., Abstraction by Time-Scale in Qualitative Simulation. In Proceedings of AAAI-87, July 1987.

[Larkin 1983]
Larkin, J. H., The Role of Problem Representation in Physics. In D. Gentner and A. L. Stevens, editors, Mental Models, chapter 5, pp. 75-98. Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.

[Liu 1989]
Liu, Z. Y., A Charge-Carrier Ontology for Reasoning about Electronics. In Proceedings of the 3rd International Qualitative Physics Workshop, Palo Alto, CA, August, 1989.

[Liu & Farley 1990]
Liu, Z. Y. and Farley, A. M., Shifting Ontological Perspectives in Reasoning about Physical Systems. In Proceedings of AAAI-90. pp. 395-400. Boston, August, 1990.

[Liu 1991]
Liu, Z. Y., Integrating Two Ontologies for Electronics. In B. Faltings and P. Struss, editors, Advances in Qualitative Physics, MIT Press, 1991.

[Mervis & Rosch 1981]
Mervis, C. B. and Rosch, E., Categorization of Natural Objects. Annual Review of Psychology, (32):89-115, 1981.

[Munitz 1974]
Munitz, M. K., Existence and Logic. New York University Press, NY, 1974.

[Murphy & Medin 1985]
Murphy, G. L. and Medin, D. L., The Role of Theories in Conceptual Coherence. Psychological Review, 92(3):289-315, 1985.

[Palies et al. 1986]
Palies, O., Caillot, M., Cauzinille-Marmeche, E., Lauriere, J. and Mathieu, J., Student Modelling by a Knowledge-Based System. Computational Intelligence, (2):99-107, January 1986.

[Rescher 1978]
Rescher, N. editor, Studies in Ontology. Monograph. Basil Blackwell, Oxford, 1978.

[Rugh 1981]
Rugh, W. J., Nonlinear System Theory. The Johns Hopkins University Press, Baltimore, MD, 1981.

[Rusgrove et al. 1977]
Rusgrove, J., DeMaw, D. and Grammer, G., Understanding Amateur Radio. American Radio Relay League, Inc., Newington, CT 06111, 1977.

[Sacks 1987]
Sacks, E., Piecewise Linear Reasoning. In Proceedings of AAAI-87, pp. 655-659, Seattle, WA, July 1987.

[Shearer 1971]
Shearer, J., Murphy, A., and Richardson, H., Introduction to System Dynamics. Addision-Wesley, 1971.

[Simmons 1986]
Simmons, R. G., The Use of Qualitative and Quantitative Simulations. In Proceedings of American Association of Artificial Intelligence, pp. 364-368. AAAI, 1986.

[Smith & Davis 1989]
Smith, B. and Davis, R., Expert Systems: How Far Can They Go? AI Magazine, 10(2):65-77, Summer (A panel session at IJCAI-85 in Los Angeles.), 1989.

[Stallman & Sussman 1977]
Stallman, R. and Sussman, G. J., Forward Reasoning and Dependency-Directed Backtracking in a System for Computer-Aided Circuit Analysis. Artificial Intelligence, 9:135-196, 1977.

[Steinberg 1983]
Steinberg, M. S., Reinventing Electricity. In Proceedings of the International Seminar Misconceptions in Science and Mathematics], pp. 388-401, Cornell University, Ithaca, New York, 1983.

[Sussman & Steele 1980]
Sussman, G. J. and Steele, G. L., Constraints - a Language for Expressing Almost-Hierarchical Descriptions. Artificial Intelligence, 1(36):1-39, 1980.

[Throop 1989]
Throop, D. R., Spatial Unification: Qualitative Spatial Reasoning about Steady State Mechanisms - An Overview of Current Work. In Proceedings of the 3rd Int. Qualitative Physics Workshop, Palo Alto, CA, August, 1989.

[van Beek 1990]
van Beek, P., Reasoning about Qualitative Temporal Information. In Proceedings of AAAI-90. pp. 728-734, Boston, August, 1990.

[Weidner 1985]
Weidner, R. T., in collaboration with M. E. Browne, Physics. Allyn and Bacon, Inc., Boston, MA, 1985.

[Weld 1986]
Weld, D. S., The Use of Aggregation in Causal Simulation. Artificial Intelligence, (30) 1986.

[Weld 1988]
Weld, D. S., Comparative Analysis. Artificial Intelligence, (36):333-373, 1988.

[Weld 1989]
Weld, D. S., Automated Model Switching. In Proceedings of the 3rd International Qualitative Physics Workshop, Palo Alto, CA, August 1989.

[Weld 1990]
Weld, D. S., Approximation Reformulations. In Proceedings of AAAI-90, pp. 407-412, Boston, August 1990.

[White & Frederiksen 1986]
White, B. Y. and Frederiksen, J. R., Intelligent Tutoring Systems Based Upon Qualitative Model Evolutions. In Proceedings of IJCAI-86, pp. 313-319, 1986.

[White & Horwitz 1987]
White, B. Y. and Horwitz, P., Thinkertools: Enabling Children to Understand Physical Laws. Report 6470, BBN Laboratories, Inc, Cambridge, MA, March 1987.

[Williams 1984]
Williams, B. C., Qualitative Analysis of MOS Circuits. Artificial Intelligence, 24:281-346, 1984.

[Williams et al. 1983]
Williams, M. D., Hollan, J. D. and Stevens, A. L., Human Reasoning about a Simple Physical System. In D. Gentner and A. L. Stevens, editors, Mental Models, chapter 7, pp. 131-154. Lawrence Erlbaum Associates, Publishers, Hillsdale, NJ, 1983.

[Winograd & Flores 1987]
Winograd, T. and Flores, F., Understanding Computers and Cognition - A New Foundation for Design. Addison-Wesley, reading, MA, 1987.