

Qualitative Reasoning about Physical Systems with Multiple Perspective

Zheng-Yang Liu

My dissertation (Liu 1991) describes an approach to automatically formulating or selecting models of a target physical system for a given qualitative reasoning task. It was motivated by two observations regarding modeling in general and work in qualitative physics in particular. First, all model-based reasoning is only as good as the model used (Davis and Hamscher 1988). Second, no single model is adequate or appropriate for a wide range of tasks (Weld 1989). A model of a real-world system is but an abstraction of some aspects of the system. To formulate a model of a physical system for a given task, we inevitably take certain perspectives of the system to capture proper scenarios by deciding what to describe and what to ignore (Hobbs 1985). We simply could not describe everything about a physical system in a single model; we would be drowned in detail, and computation would become intractable. The narrowness of most expert systems can be attributed to this situation because they work within fixed, single models. These systems are brittle because they usually cannot reformulate or shift to new models when the models supplied to them become inappropriate.

To address this problem and study perspective taking in commonsense reasoning, my research focused on three dimensions of representational choice: (1) *ontological commitment*, adopting distinct language forms or vocabularies to describe a target system; (2) *topological configuration*, focusing on a dominant subset of the structure of the system and ignoring the rest; and (3) *structural aggregation*, abstracting structural elements into

black boxes and leaving out irrelevant structural and behavioral details.

In this approach, a perspective embodied in a model is defined as a position taken in each of the possible dimensions. *Perspective taking* as a process is defined as formulating or selecting a scenario model of a target system that reflects a dominant configuration of the system topology, an appropriate structural granularity, and a distinct language form. Perspective taking is an integral part of reasoning about real-world systems. Different models are generally required for different tasks. Even for a single task, multiple models with distinct perspectives are often needed because multiple perspectives are often intertwined in effective, efficient reasoning about complex real-world systems. My research explored the issues of intertask and intratask model shift. The central thesis of my dissertation is that by using a task-driven, perspective-taking approach, we can help overcome the brittleness problem and extend the range of automated reasoning about complex physical systems.

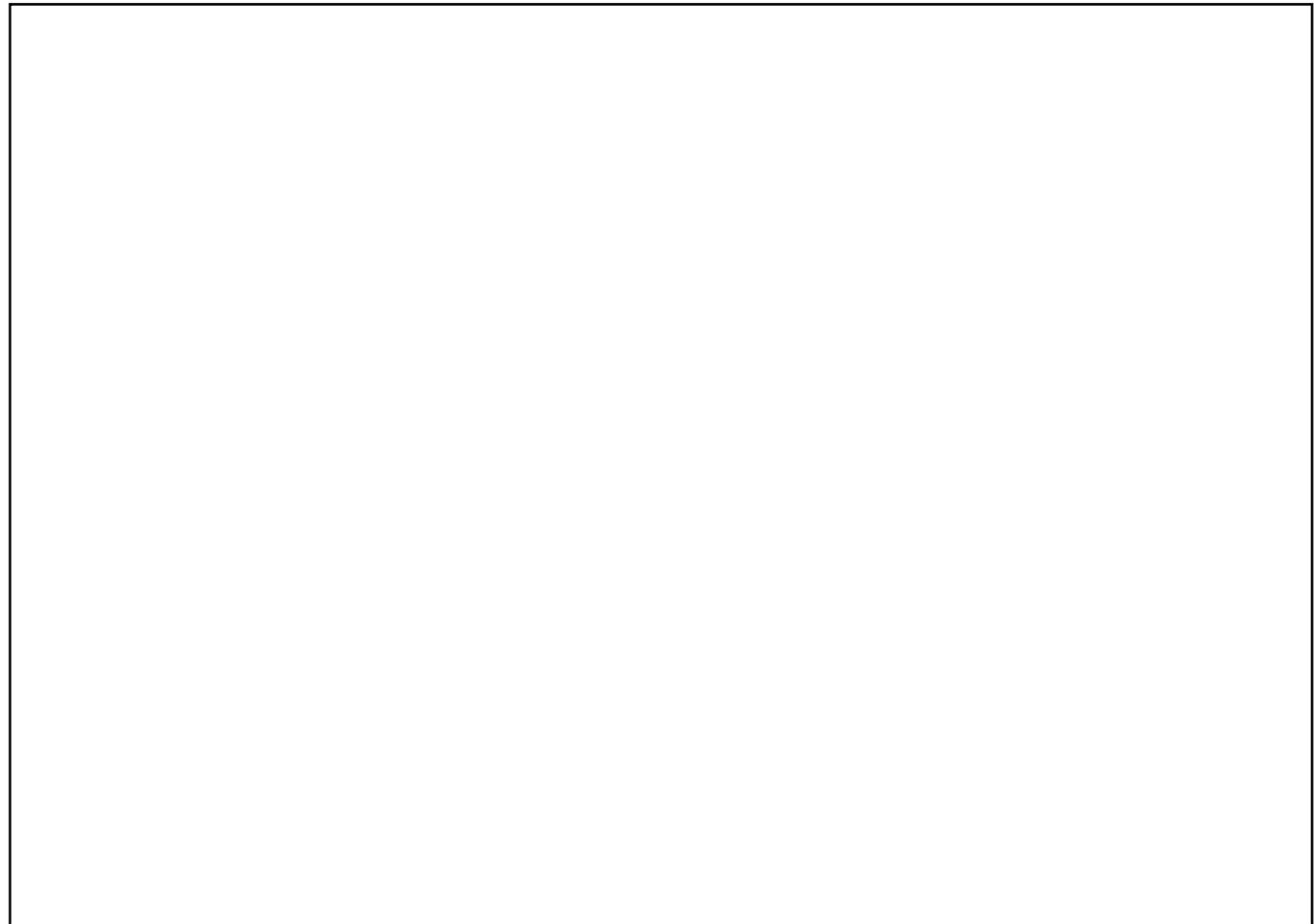
Three research issues are prominent in perspective taking given a task: (1) which perspectives to take, (2) how to represent them, and (3) when to shift from one perspective to another. To address these and other issues, I built a system called ARC (automated reasoning about circuits) that integrates perspective taking with qualitative simulation to generate causal explanations for tutorial purposes, using electronic circuits as an example domain. ARC focuses on standard perturbation analysis to explain circuit behaviors and supports a simple language to define such tasks. ARC extends previous work

(Addanki, Cremonini, and Penberthy 1989; Weld 1989; Falkenhainer and Forbus 1991) by performing not only intertask model shift but also intratask model shift. ARC is able to answer a user's queries about a circuit's behavior by formulating or selecting appropriate scenario models and reasoning with these models. A query specification is considered a task definition.

Three items make up a task definition in ARC: (1) the name of a target system, (2) the specification of input perturbations, and (3) the specification of output desired. The name of a target system provides access to a description of the system topology of the target circuit—a network of circuit components and their connections by nodes. This information is the only circuit-specific information provided to ARC. To specify input perturbations, ARC extends the common single-perturbation method (de Kleer 1984) by allowing a sequence of input signals as perturbations based on a discrete set of time points. A signal consists of two values: a qualitative value and its derivative. Each signal is processed in the context created by these signals early in the sequence and early in time. To specify output, ARC allows the user to indicate a specific system variable whose behavior the user wants explained. A variable is defined with respect to two nodes in the circuit topology, such as $V_{p,n}$. These nodes, such as (p, n) , determine an output structural unit, encapsulating either a single device or a set of devices.

The input perturbations influence the actual behaviors of the circuit in the simulation. The output specification indicates a structural unit as the focus for explanation. The vocabulary used in specifying the input and output variables reveals the language forms used. For example, the user can specify voltage, current, field intensity, or charge flow with respect to any two nodes in the circuit topology. For model formulation, a global knowledge base of the subject domain—the physics of electricity—is also provided to ARC.

Given a task, ARC extracts crucial pieces of information from the task



specification and takes positions along each of the possible dimensions about configurations, structural granularities, and ontologies to formulate appropriate scenario models for carrying out the task. Model shift during reasoning reflects changing positions along the dimensions of representational choice.

A configuration of a circuit reflects an active current flow path in the circuit. For any circuit with a set of possible configurations, only one configuration is dominant, or active, at any point in time. One crucial property of a configuration is that the individual devices within the configuration, linear or nonlinear, behave linearly over the duration of the given qualitative state. Thus, identifying each possible configuration of a nonlinear circuit provides a configuration-based linearization method for automating the analysis of nonlinear systems.

To identify a dominant configura-

tion, ARC searches the topology of the circuit based on the input perturbations of a given task and the internal qualitative state of the circuit. A clustering technique was developed (Liu and Farley 1991) to reveal the "big picture" of a circuit. The process makes explicit the implicit structural hierarchy in the circuit by clustering parallel and serial graphs in the system topology. Reasoning can then traverse the hierarchy, as needed. At any level, lower-level details are ignored. To further facilitate searching, the clustering process gives each cluster a "road sign," such as "capacitive," "resistive," "one-way-path," or any combination of these. For example, a capacitive cluster acts as an impasse for a direct-current signal whose derivative is zero, and the search inside this cluster can simply be avoided.

The qualitative state associated with a specific configuration serves as the underlying modeling assumption

for the configuration model. Any violation of the modeling assumptions during simulation indicates that a new configuration must be identified.

For structural aggregation, the output specifications in different tasks can suggest a different structural granularity for efficient reasoning by grouping subsets of structural units in the active configuration as black boxes. Because each task indicates an output structural unit as the focus of explanation, this unit is typically treated as a black box. The rest of the circuit is then mapped to appropriate constructs with maximum possible granularity to suppress extraneous detail.

The aggregation process generates simpler, yet equivalent circuits. Each aggregate is treated as an individual device during reasoning. The hiding of irrelevant structural details inside such an aggregate ultimately leads to suppressing the behavioral details of the structural unit. Different tasks

can suggest different ways of aggregation over the same circuit, indicating distinct focuses on various structural and behavioral details of the circuit.

For ontological choice, ARC currently uses the standard device ontology at the macroscopic level (de Kleer 1984) and the charge-carrier ontology at the microscopic level (Liu and Farley 1990) to reason about electronic circuits. The input and output of a task are specified in these vocabulary terms. ARC creates qualitative differential equation (QDE) models in chosen language forms and uses a set of ontological choice rules to control ontological shift.

For example, one rule specifies that "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 task." Another rule specifies that "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." Ontological shift is carried out by switching QDE models by way of *bridging relations*, which associate comparable elements from the two related ontologies.

ARC was evaluated using examples from related AI literature and standard textbooks (Liu 1991). Sussman and Steele (1980) and Genesereth (1985) used multiple levels of structural descriptions of a physical device for efficient reasoning. Their approaches relied on predefined or fixed structural hierarchies, or *slices*. ARC is able to generate different structural aggregates for different tasks rather than resort to a fixed structural hierarchy.

The graph-of-models approach by Addanki, Cremonini, and Penberthy (1989) and Weld (1989) is effective to the extent that all the possible models needed by future tasks can be preenumerated, but the process requires a large amount of storage and entails a problematic trial-and-error search for a valid model. ARC does not assume a graph of models given as input. Instead, scenario

models are generated based on given tasks.

The approach closest to ARC is compositional modeling, which was described by Falkenhainer and Forbus (1991). They used a fine-grained modular method. The fragments of a general domain model are attached with explicit modeling assumptions, each describing various aspects of the domain. From a query that provides constraints to identifying a set of assumptions and associated model fragments, the domain model can be trimmed to one scenario model for answering the query and minimizing extraneous detail.

By contrast, ARC can generate not just one scenario model for a single task but several. In addition, the modeling assumptions for a scenario model are generated automatically in the simulation process based on the task at hand and then stored with the model in the knowledge base for possible reuse by future tasks. At any point in time, if these assumptions become invalidated because of a change in the task or the qualitative state of the circuit, ARC formulates or selects a new scenario model that satisfies the state. A scenario model is guaranteed to be consistent because only a single position is taken on each possible dimension of representational choice.

Acknowledgments

I am grateful to Art Farley for his inspiration and guidance. This work benefited from discussions with Keith Downing, Pat Hayes, and Dan Weld and was supported by fellowships from Tektronix and the University of Oregon.

Note

1. A copy of this dissertation is available as a technical report (CIS-TR-91-04) from the Department of Computer and Information Science, University of Oregon, Eugene, OR 97403.

References

Addanki, S.; Cremonini, R.; and Penberthy, J. S. 1989. Reasoning about Assumptions in Graphs of Models. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 1432-1438. Menlo Park, Calif.: International

Joint Conferences on Artificial Intelligence.

Davis, R., and Hamscher, W. 1988. Model-Based Reasoning: Trouble-Shooting. In *Exploring Artificial Intelligence: Survey Talks from the National Conferences on Artificial Intelligence*, eds. H. E. Shrobe and AAAI, 297-346. San Mateo, Calif.: Morgan Kaufmann.

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

Falkenhainer, B., and Forbus, K. 1991. Compositional Modeling: Finding the Right Model for the Job. *Artificial Intelligence* 51:95-143.

Genesereth, M. 1985. The Use of Design Description in Automated Diagnosis. In *Qualitative Reasoning about Physical Systems*, ed. D. Bobrow, 411-436. Cambridge, Mass.: MIT Press.

Hobbs, J. R. 1985. Granularity. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 432-435. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Liu, Z.-Y. 1991. Qualitative Reasoning about Physical Systems with Multiple Perspectives. Ph.D. diss., Dept. of Computer and Information Science, Univ. of Oregon.

Liu, Z.-Y., and Farley, A. M. 1991. Structural Aggregation in Commonsense Reasoning. In Proceedings of the Tenth National Conference on Artificial Intelligence, 868-873. Menlo Park, Calif.: American Association for Artificial Intelligence.

Liu, Z.-Y., and Farley, A. M. 1990. Shifting Ontological Perspectives in Reasoning about Physical Systems. In Proceedings of the Ninth National Conference on Artificial Intelligence, 395-400. Menlo Park, Calif.: American Association for Artificial Intelligence.

Sussman, G. J., and Steele, G. L. 1980. CONSTRAINTS—A Language for Expressing Almost-Hierarchical Descriptions. *Artificial Intelligence* 14:1-39.

Weld, D. S. 1989. Automated Model Switching. Paper presented at the Third International Qualitative Reasoning Workshop, Palo Alto, California, 9-11 August.

Zheng-Yang Liu is a research scientist at EDS Research Laboratory in Austin, Texas. His current work involves applying AI techniques to software-engineering problems, including perspective taking in application domain modeling and large-scale software reengineering and software evolution.