F-15

Model Compilation

An approach to automated model derivation

RICHARD KELLER AND CATHERINE BAUDIN

AI RESEARCH BRANCH, MAIL STOP 244–17
NASA AMES RESEARCH CENTER
MOFFETT FIELD, CA 94035

YUMI IWASAKI, PANDURANG NAYAK,
AND KAZUO TANAKA

KNOWLEDGE SYSTEMS LABORATORY
COMPUTER SCIENCE DEPARTMENT
STANFORD UNIVERSITY
STANFORD, CA 94305

NASA Ames Research Center

Artificial Intelligence Research Branch

Technical Report FIA–90–04–06–1

April, 1990

Model Compilation:
An approach to automated model derivation

Richard M. Keller **Catherine Baudin**
NASA Ames Research Center[1]
Artificial Intelligence Research Branch
Mail Stop 244-17
Moffett Field, CA 94035
(keller/baudin@pluto.arc.nasa.gov)

Yumi Iwasaki **Pandurang Nayak** **Kazuo Tanaka[2]**
Knowledge Systems Laboratory
Computer Science Department
Stanford University
Stanford CA 94305
(iwasaki/nayak/tanaka@sumex-aim.stanford.edu)

AAAI-90 Review Category: Automated Reasoning (Rule-based reasoning)
Keywords: Knowledge compilation, automated rule generation, hybrid systems

ABSTRACT

In this paper, we introduce an approach to automated model derivation for knowledge based systems. The approach -- called model compilation -- involves procedurally generating the set of domain models used by a knowledge-based system. With an implemented example, we illustrate how this approach can be used to derive models of different precision and abstraction, and models tailored to different tasks, from a given set of 'base' domain models. In particular, we describe two implemented model compilers, each of which takes as input a base model that describes the structure and behavior of a simple electromechanical device -- the Reaction Wheel Assembly of NASA's Hubble Space Telescope. The compilers transform this relatively general base model into simple task-specific models for troubleshooting and redesign, respectively, by applying a sequence of model transformations. Each transformation in this sequence produces an increasingly more specialized device model. The compilation approach lessens the burden of updating and maintaining consistency among models by enabling their automatic regeneration.

---

[1]Keller employed under contract NAS2-11555 to Sterling Federal Systems; Baudin employed under contract NAS2-12952 to Recom Software, Inc. This work was done while the authors were at Stanford Knowledge Systems Laboratory.

[2]On leave from NTT Human Interface Laboratory.

1. INTRODUCTION AND MOTIVATION

In recent years, there has been a growing awareness that to increase the flexibility, robustness, and efficiency of knowledge-based systems, it is necessary to move away from systems that utilize a single monolithic domain model, and toward multi-model systems that incorporate a heterogeneous collection of domain models. This includes models developed at different levels of abstraction and precision, and models organized according to different task perspectives. The idea behind providing multiple models is to enable systems to dynamically select and use the models that are most efficient and effective for the current problem solving task. If those models prove inadequate, systems can attempt to switch to more appropriate models rather than failing without recourse.

The simplest examples of multi-model systems are hybrid rule-based/model-based systems. These systems incorporate two models: a "shallow" associational domain model, often expressed as a set of rules, and a "deep" first principles domain model, incorporating causal, structural, and behavioral knowledge. Examples of such systems can be found in [Simmons & Davis 87, Steels & Van de Velde 86, Fink 85, Koton 88]. Systems incorporating more than two models or perspectives include those described in [Patil 81, Falkenhainer & Forbus 88, Murthy & Addandi 87, Davis 84, Sussman 78, Abbott 88]. There are many important issues to address in building multi-model systems, including those associated with model acquisition, representation, usage, and maintenance .

In this paper, we describe an approach to the problems of updating and maintaining consistency among models. In most multi-model systems, the different models are constructed separately and linked together manually, if at all. Often the interrelationships among the models are left unspecified. This leads to considerable difficulty in updating a system's models when there are changes in the domain, or when the system is adapted to a slightly different domain. Depending on how frequently such changes are required, considerable time and effort may be involved in manually regenerating the models. Aside from time spent simply editing the models, system maintainers must ensure that all changes are consistent across models, as well.

Our approach to these problems is to enable automatic regeneration of models using a technique we call *model compilation.* This technique is applicable providing there exists one or more assumedly independent *base models* from which all other models of interest in the domain can be computationally derived. The basic idea is to procedurally generate all models from the base models using one or more *model compilers*. A model compiler applies a sequence of transformations (typically, information-losing approximations and/or abstractions) to the set of base models to produce a sequence of new models. Compilation reduces the problem of maintaining multiple models to the problem of maintaining each independent base model. When a base model is updated, its derived models can be automatically recompiled, and consistency across models is automatically reestablished.

Note that in principle, all possible inferences can be drawn within the base models, and the derived models are extraneous. In practice, however, the computational complexity associated with using the full set of base models may be overwhelming, and the use of specialized derived models may prove necessary or expedient. In effect, the model compilation framework provides a mechanism for incorporating simplifying assumptions into the base models, yielding more approximate and/or abstract models. When the simplifying assumptions are valid, these models are likely more efficient to use than the base models, yet just as adequate for problem solving.

In this paper, we report on the results of an experiment aimed at applying the model compilation approach to problems of modeling and reasoning about engineered devices. Specifically, we describe two implemented model compilers, each of which takes as input a single base model that describes the structure and behavior of a simple electromechanical component -- the Reaction Wheel Assembly of NASA's Hubble Space Telescope. Section 1 describes this base model. The

compilers transform the relatively task-neutral base model into simple task-specific models for troubleshooting and redesign, respectively. The troubleshooting model is represented as a set of associational fault localization rules, and the redesign model is expressed as a set of abstract redesign plans. Both compilers automatically generate a sequence of intermediary device models as a by-product of this compilation process. The compilation processes for troubleshooting and redesign are described in Sections 3 and 4, respectively. Finally, we analyze and discuss the results of our experiment in Section 5.

2. THE REACTION WHEEL ASSEMBLY

The Reaction Wheel Assembly (RWA) is part of the pointing and control subsystem aboard NASA's Hubble Space Telescope. The function of the RWA device is to point the Space Telescope at its visual target. The RWA houses a spinning rotor that is accelerated to induce a torque on the telescope, thus causing the telescope to turn. A cross-sectional view of the RWA is presented in Figure 1.

Our object-oriented base model of the RWA device consists of two basic parts: a structural representation, and a behavioral representation. Device structure is captured by information about component/subcomponent relationships, including physical connectivity and spatial relationships among components. We utilize a simple two-dimensional, bounding box spatial representation to capture shape and layout of the components. Device behavior is represented by a set of behavioral equations that specify constraints among numeric quantities (parameters) associated with components. Behavioral equations are represented in both quantitative and qualitative differential format, as described below in Section 4.1. Further details on the representation used in modeling the RWA can be found in [Keller et al. 89].

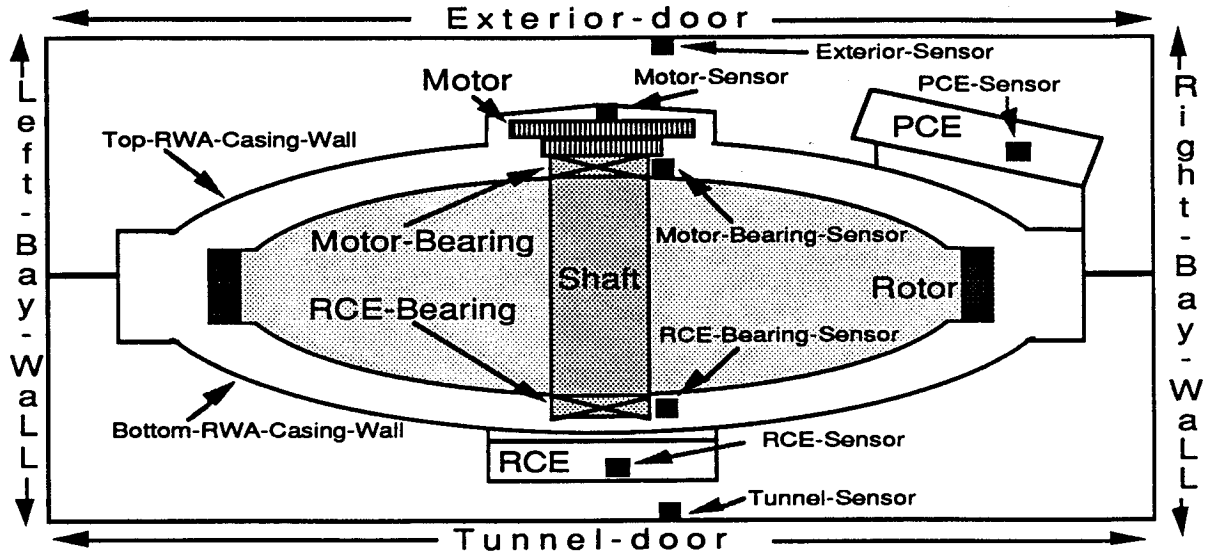

Figure 1: RWA cross-sectional view (adapted from [Perkins & Austin 87]). Each of four reaction wheels is mounted within one of the transport bays that ring the barrel of the telescope. Inside the RWA casing is a hollow aluminum wheel with a steel rim, mounted on a rotating shaft which is connected to a motor. The Rotor Control Electronics (RCE) and the Power Control Electronics (PCE) supply power and control signals to the motor. Each end of the wheel shaft is supported by a bearing. Near each heat-generating component within the RWA (e.g., the bearings) there is a small temperature sensor used to monitor the device's functioning.

3. COMPILING DIAGNOSTIC RULES

This section describes how the general-purpose RWA base model can be compiled into a special-purpose associational model that takes the form of a set of "fault localization" rules for diagnosis. Following is an example of a fault localization rule that the diagnostic model compiler can produce:

R2: *If* *Temperature of RCE-Bearing-Sensor is High*
and *Temperature of RCE-Sensor is OK*
and *Temperature of Tunnel-Sensor is OK*
then *set Malfunction of RCE-Bearing to True.*

To understand this rule, refer back to Figure 1. The rule says that if the sensor for the RCE-Bearing is abnormally high, and nearby sensor readings are normal, then there must be a malfunction within the RCE-Bearing. On first analysis this rule appears incomplete because it only checks the sensor readings for the RCE and the tunnel, and omits other nearby components that function as heat sources and might potentially influence the reading on the RCE-Bearing-Sensor. Actually, the experts deliberately ignore these thermal influences because they consider them negligible.

To generate rules like R2, the diagnostic model compiler makes use of a simple, but general fault localization rule. Suppose $\{SRC_1, SRC_2, ..., SRC_n\}$ is a set of source components that produce some substance S (e.g., thermal energy), and suppose $\{SEN_1, SEN_2, ..., SEN_n\}$ is a set of corresponding sensors that measure the amount of S at each source component. The following general diagnostic rule captures the fault localization idea:

R1:[3] *If* *Reading of SEN_i is Abnormal*
and *(forall $k \neq i$ / influences(SRC_k, SEN_i))*
Reading of SEN_k is Normal
then *set Malfunction of SRC_i to True.*

R1 states that if the reading of some sensor is abnormal, and the readings associated with potentially influential sources are normal, then there must be a malfunction in the local source measured by the abnormal sensor. Notice how the *influences* predicate captures the notion that only certain sources are capable of influencing the reading of a given sensor. To automatically compile rule R2 from rule R1, the compiler produces a specialized version of R1 by using available domain knowledge to substitute RWA-specific terms for the general terms in R1. This yields the following intermediary rule, with the substituted terms underlined:

R1.5: *If* *Temperature of RCE-Bearing-Sensor is High*
and *(forall $k \neq i$ / influences(SRC_k, RCE-Bearing-Sensor))*
Temperature of SEN_k is OK
then *set Malfunction of RCE-Bearing to True.*

The final step from R1.5 to R2 can be achieved if we know the identity of all heat sources in the RWA, and know whether each heat source is capable of *influencing* RCE-Bearing-Sensor or not. So before executing the steps that take R1 to R1.5 and finally to R2, the diagnostic model compiler first generates the necessary model of thermal influence from the base RWA device model. This generation sequence is detailed in the of two compilation steps described below.

---

[3]Note that this simple rule applies only when there is a 1-1 correspondence between sources and sensors, and when sensors are assumed to be reliable.

3.1. Compiling the Thermal Resistance Model

The first compilation step is to produce a simple, quantitative thermodynamic model of the RWA from the base model by computing *thermal resistance*. The thermodynamic model associates a numeric thermal resistance value with each heat flow path linking a heat source and a heat sensor within the RWA. The higher the resistance value, the harder it is for heat to flow along the path. A portion of one possible thermal resistance model for the RWA is illustrated in Figure 2a.

The compiler computes the resistance between heat source and sensor by tracing a path between them and calculating resistance as the weighted average of the thermal constants associated with the materials of each component along the path. The weighting is proportional to the thickness of the components in the heat flow path. Depending on modeling assumptions, the compiler can either assume that heat radiates along a straight-line path between source and sensor, or conducts along a physically-connected path. Although the compiler's calculation is not thermodynamically precise, it provides a heuristic estimate of the thermal resistance.

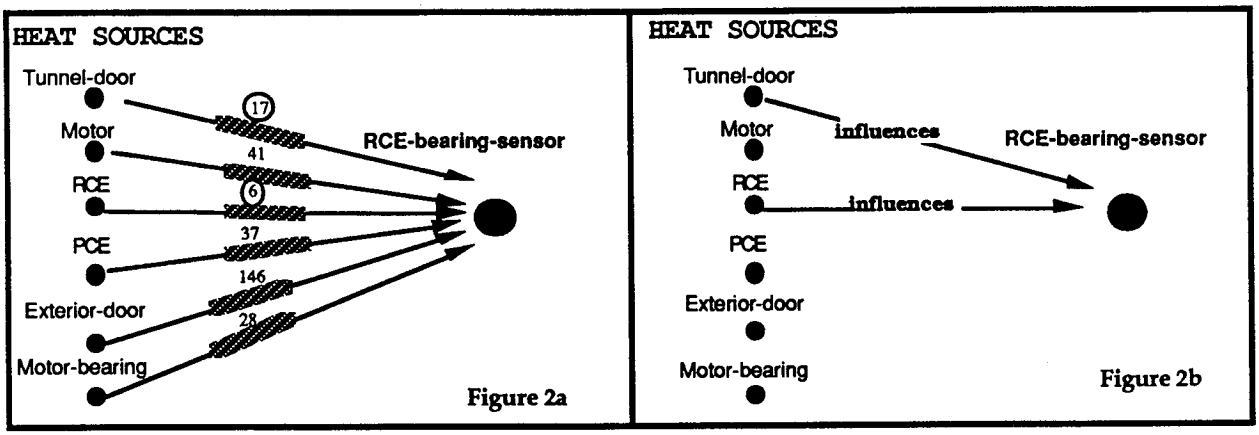

Figure 2: Portions of RWA thermal models pertaining to RCE-bearing-sensor. Thermal resistance model in Figure 2a is thresholded at 20 to produce thermal influence model in Figure 2b. Only the circled links are retained in the influence model.

3.2. Compiling the Thermal Influence Model

Once the compiler has produced a quantitative thermal resistance model of the RWA, the next step is to convert it to a more qualitative thermodynamic model that captures the expert's notion of thermal "influence". Intuitively, the amount of thermal influence imposed by a given heat source on a given heat sensor is inversely proportional to the amount of thermal resistance along the path between the source and sensor. One way of deriving a simple qualitative thermal influence model from the numeric thermal resistance model is to simply threshold the thermal resistance value. Any heat flow path with a resistance below a preset expert-defined threshold will be considered a path of thermal influence. For example, by thresholding the portion of the resistance model shown in Figure 2a at a resistance value of 20, the compiler would produce the partial influence model shown in Figure 2b.

3.3. Summary of Diagnostic Model Compilation

In summary, there are three overall steps in the diagnostic model compilation process. In the first step, the base RWA structure/behavior device model is compiled into a thermal resistance model

via a simple heat flow computation. Next, the thermal resistance model is reduced to a thermal influence model by applying a thresholding procedure. Finally, an associational diagnostic model (which takes the form of a set of RWA-specific fault localization rules) is produced by instantiating a generalized fault localization rule based on information in the thermal influence model.

4. COMPILING ABSTRACT REDESIGN PLANS

The second model compiler takes as input the same base structure/behavior device model as the diagnostic model compiler, but it produces abstract redesign plans rather than fault localization rules as output. Here is a sample of the type of abstract redesign plan generated by the compiler:

If *the goal is to decrease the temperature of RCE-Bearing*
then *consider the following actions, in the following order:*
increase the bearing-width of RCE-Bearing;
increase the thickness of Bottom-RWA-Casing-Wall;
increase the thermal-constant of Bottom-RWA-Casing-Wall;
increase the body-width of RCE-body;
increase the thermal-constant of RCE-body

Notice that this plan is abstract in the sense that it does not specify exactly *how* to achieve the desired decrease in temperature, but merely narrows the set of potential actions for achieving the goal down to a small ordered set of recommended actions. Furthermore, the actions do not suggest exact quantitative values for increasing and decreasing the specified quantities. This type of plan merely serves as a starting point for a redesign system, and must be elaborated further to produce a complete redesign plan in the form of a set of executable actions.

The process by which an abstract redesign plan is compiled differs markedly from the diagnostic rule compilation process described in the previous section. In the redesign case, the compilation process resembles macro-formation. The plan is extracted from a tree of redesign goals generated during the compilation process. The root of the redesign goal tree constitutes the "if" part of the plan, and the leaves of the goal tree, suitably ordered, form the "then" part. The following sections detail the series of compilation steps used to transform the general-purpose RWA device model into a redesign goal tree from which an abstract redesign plan can be extracted.

4.1 Equation Set Assembly

The first step in the process of compiling redesign plans is to assemble a set of qualitative differential equations, which can be viewed as a type of qualitative model of the RWA's behavior. The qualitative behavior model abstracts away precise numeric constraints and retains only imprecise knowledge of functional relationships that describe changes in quantities. For redesign, this model will be used to draw inferences about how to modify the values of controllable quantities in order to achieve a specified redesign goal.

To form an equation model, the compiler gathers together the set of qualitative differential equations that interrelate the quantities associated with the RWA's components. A subset of these qualitative equations is shown in Table 1. In the table, the notation [Q'] stands for the sign (i.e., +, -, or 0) of the time derivative of the quantity Q. For example, the qualitative equation *[MotorSpeed'] = [MotorCurrent']* says that when the motor speed is increasing (decreasing), the motor current is also increasing (decreasing).

```
[BearingTemp'] = [TunnelContribution'] + [RCEContibution'] + [MotorContribution']
                 + [BearingFriction']
[MotorSpeed'] = [BallRadius'] + [BearingFriction']
[MotorSpeed'] = [MotorCurrent']
[BallRadius'] = [BearingWidth']
    etc.
```

Table 1: Portion of qualitative equation set for RWA

4.2 Causal Dependency Analysis

The compiler's next step is to infer causal relationships among the quantities represented in the qualitative equation model. Whereas equations are inherently acausal, in redesign it is necessary to make use of causal dependencies to determine how changes in specific quantities propagate through the device and affect other quantities. Iwasaki's causal ordering procedure [Iwasaki 88] is used to analyze causal dependencies and produce a directed graph structure that encodes these dependencies. In order to apply the procedure, it is necessary to define which quantities are *exogenous*. Exogenous quantities are causally primitive in the sense that their value is not determinable from any quantities within the scope of the system under study. If an equation relates an exogenous quantity to a non-exogenous quantity, the causal ordering procedure defines the non-exogenous quantity to be causally dependent on the exogenous quantity. The ordering procedure then iterates on this process to find the quantities that are causally dependent on the non-exogenous quantities. Eventually, an entire causal dependency graph is constructed bottom-up from the given exogenous quantities in this manner. A portion of the causal dependency graph derived from the qualitative equations in Table 1 is presented in Figure 3. Exogenous quantities are prespecified in the RWA device model.

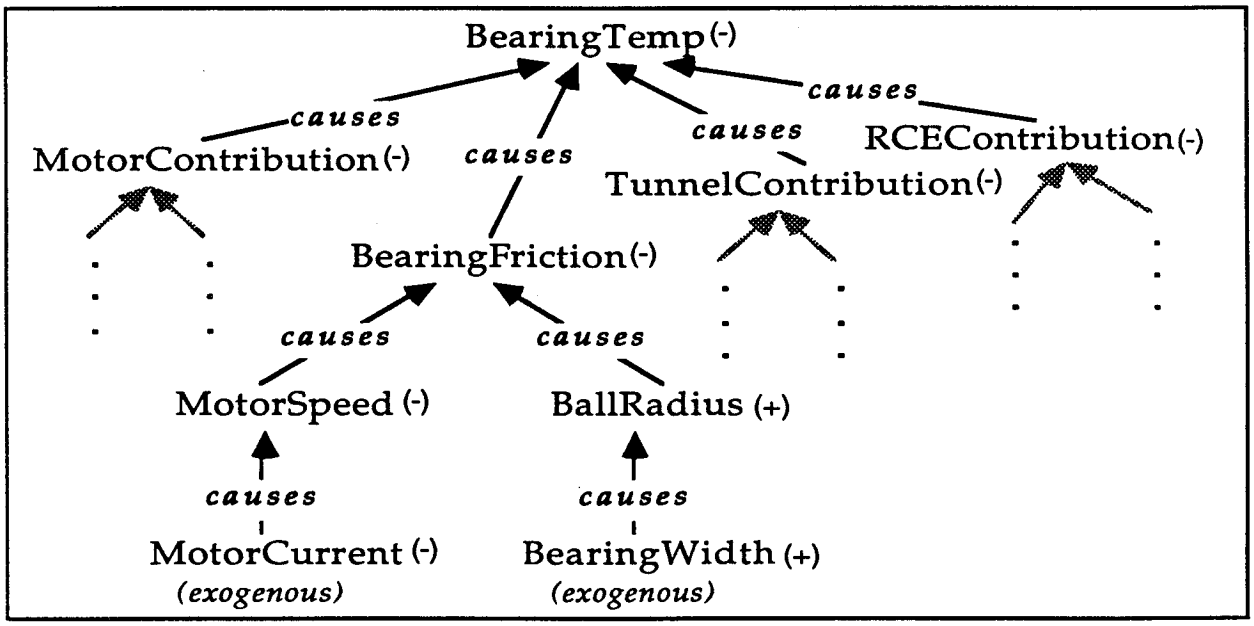

Figure 3: Causal dependency subgraph derived from Table 1 equations

4.3 Redesign Goal Tree Generation

Once the causal dependency graph has been generated, the compiler can use the graph plus the qualitative equation model to produce a tree of redesign alternatives for achieving a user-specified redesign goal. Goals take the form **"{increase or decrease} quantity Q of component C"**. To produce a set of redesign subgoals for achieving a goal of this type, the compiler augments each causal dependency node in the causal subgraph rooted at quantity Q with either the action "increase" or "decrease", based on an analysis of the associated qualitative equations. For example, suppose the goal is to decrease the quantity *BearingFriction*, which is causally dependent on the quantities *MotorSpeed* and *BallRadius* (see Figure 3). Further assume that the qualitative differential equation relating these quantities is [*MotorSpeed'*] = [*BallRadius'*] + [*BearingFriction'*]. In order for *BearingFriction* to decrease, either *MotorSpeed* must decrease (assuming *BallRadius* is constant) or *BallRadius* must increase (assuming *MotorSpeed* is constant). In this case, the compiler generates two corresponding subgoals in the redesign goal tree. (Note: the quantities in Figure 3 are labeled with + or - depending on whether the generated goal is to increase or decrease the quantity.) The leaves of the goal tree correspond to a set of "increase" or "decrease" actions on externally-controllable (exogenous) quantities, and thus constitute an unordered set of executable actions for achieving the top-level goal.

4.4 Goal Pruning and Ordering

The next step in the compilation process is to prune and order the nodes in the goal tree according to a set of redesign heuristics. This has the effect of transforming the set of *possible* redesign actions (represented by the leaves of the goal tree) into a smaller, prioritized sequence of *recommended* redesign actions.

The compiler currently uses two pruning heuristics and one ordering heuristic. The first pruning heuristic eliminates -- based on an analysis of the causal dependency graph -- any subgoals whose execution would violate any pre-specified design constraints on quantities. For example, suppose a side-effect of decreasing the *MotorSpeed* is a decrease in *MotorTorque*. If *MotorTorque* were constrained to be constant, the compiler would prune the subtree rooted at this subgoal. The second pruning heuristic is specific to situations involving redesign of thermodynamic properties. The basic idea is to prune any branch leading to a leaf redesign subgoal that recommends modifying a parameter of a "non-influential" component. A "non-influential" component is one that is thermally insulated from the component undergoing redesign. The procedure used by the redesign compiler for computing a thermal influence model is the same as the one used by the diagnostic model compiler (see Section 3.2.). After applying this pruning heuristic, leaves of the goal tree that remain are ordered by value of increasing thermal resistance based on a thermal resistance model (see Section 3.1).

4.5 Redesign Plan Synthesis

The final compilation step involves synthesizing an abstract redesign plan that caches the recommended sequence of redesign actions for accomplishing the user-specified redesign goal. The synthesized plan is a type of macro-rule formed from the root and leaves of the final redesign goal tree. In particular, the root of the tree forms the rule's antecedent (the plan applicability conditions) and the ordered leaves of the tree form its consequent (the abstract plan, itself). The final rule that the redesign compiler produces from the pruned and ordered goal tree was given at the beginning of Section 4.

4.6 Summary of Redesign Plan Compilation

During the compilation process, the initial RWA device model is first transformed into a *qualitative* behavior model, then into a *causal* behavior model, and finally into a tree of redesign goals from

which the final plan is extracted. The compiler takes a number of different types of knowledge as input, ranging from problem specific information (e.g., the redesign goal, exogenaity assumptions) to task specific information (e.g., thermal redesign heuristics).

5. DISCUSSION

In this section, we make some observations about the model compilation approach in light of our experience in applying the approach to the RWA device. Figure 4 summarizes the model compilation processes described in Sections 2 and 3 by illustrating the various models along with the transformation steps applied to derive one model from another. When multiple models were used in a given transformation step to derive a new model, this is illustrated by multiple incoming arrows. Aside from the base model, the figure does not illustrate the inputs used by the two compilers to produce the derived models. For example, inputs such as the thermal threshold, the generalized fault localization rule, the exogenaity assumptions, and the redesign goal and heuristics are not shown.

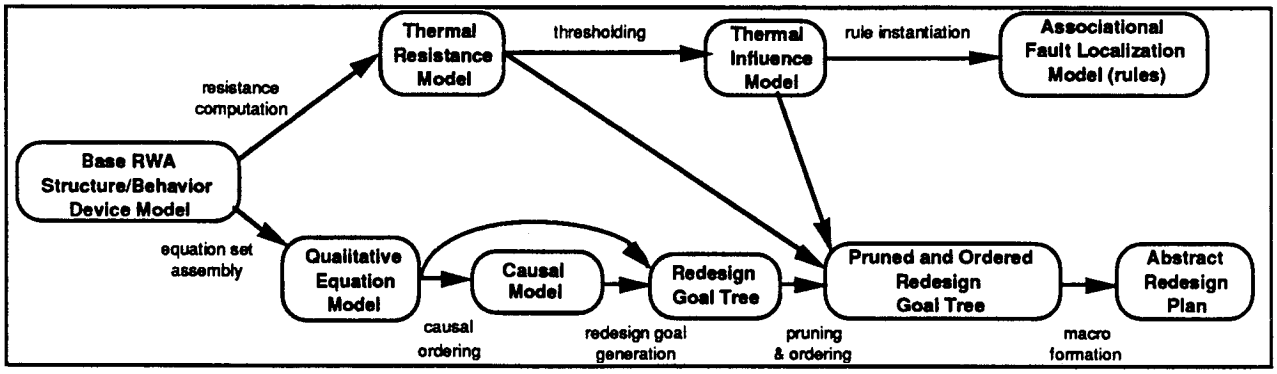

Figure 4: Summary of model compilation for RWA device

5.1 Trade-offs in Model Compilation

A fundamental motivation for compiling and reasoning with multiple models is to combat the complexity associated with detailed base model reasoning. Any reduction in complexity comes with associated costs. At each step in the compilation process, there is a trade-off between efficiency, on one hand, and generality, explicitness, and precision (if approximating model transforms are applied), on the other hand. In particular, at each step, the information content of the derived model is reduced, yielding a potentially more efficient, yet less general-purpose, less explicit, and in some cases less precise model.

Consider the thermal resistance model produced by the RWA diagnostic compiler (Section 3.1). This model caches heuristic thermal resistance values for heat flow paths in the RWA. Assuming the number of heat flow paths is reasonable, resistance can be determined quite efficiently by doing a simple lookup in this model. Note that although the computational cost is greater, the base RWA structure/behavior device model can also be used to compute thermal resistance. Furthermore, the base model is clearly more general because it contains more information, and it can be used to derive inferences that include -- but are not limited to -- inferences involving thermal resistance. Also, knowledge in the base model is also represented more explicitly than in the derived thermal resistance model. For example, the locations of components and their material composition are represented only implicitly in terms of the cached thermal resistance values stored in the resistance model. Finally, the resistance values themselves are heuristic estimates that may be computed with greater precision, if necessary, using the full base model.

In general, the efficiency benefits gained through compilation are difficult to guarantee or quantify without knowledge of the specific problem solving context. In producing compiled models, we are betting that the effort required to compile the models and use them in future problem solving will be less than the effort necessary to use the original base models. To win, the total effort put into compilation must be amortized successfully over the lifetime of the system. The success of this effort is contingent on the appropriateness of our predictions about future problems faced by the system. As an example, consider a device with thousands of thermal sensors, each of which might possibly fail. Compiling a fault localization rule for *each* sensor using the approach outlined in Section 3 would be relatively expensive, and the cost may be unwarranted if most of the rules are never used because many of the sensors rarely fail in practice. On the other hand, if we have knowledge that certain sensors have a higher failure rate, it may make sense to compile rules for only those sensors, and to rely on the base model to diagnose unexpected failures of other sensors.

5.2 Evaluation of Model Compilation

The examples described in Sections 3 and 4 illustrate how model compilation can be applied to derive two types of task-specific models (in the form of fault localization rules and abstract redesign plans) from a common base model. A major advantage to this approach to constructing multi-model systems is that the resulting system exhibits an enhanced degree of robustness to changes in the base models. When the base models change, the derived models can be recompiled to reflect these changes. For example, suppose that after extensive testing, NASA designers decide to reconfigure some of the RWA's components by changing some of their manufacturing materials and rearranging their positions. If the changes are substantial, the thermal properties of the RWA will be affected, and rules like R2 in Section 3 may become invalid. However, a corrected rule set can be automatically recompiled using the model compilers. Simultaneously, all derivative models -- including the thermal resistance model, the thermal influence model, the causal model, the redesign goal trees, and the final redesign plans -- will be updated and made consistent with the changes in the base model.

Another advantage to this approach which we have not yet explored relates to model repair. It may be possible to correct a faulty model by inspecting its derivation and evaluating the validity of simplifying assumptions incorporated into the model during the compilation process. For example, if the fault localization rules are performing incorrectly, we may be able to determine that thermal resistance threshold assumed in compiling the rules is incorrect. Or if the redesign plans do not accomplish the desired effect, the exogenaity assumptions used in compiling the causal model -- and in turn, the plans -- may be inappropriate. To address model repair issues, we would need to have a more explicit representation of the derivation path and the assumptions than is currently available in our system. The use of explicit model derivations has been pursued by Swartout in his work on explanation [Swartout 83] and by Smith [Smith et al. 85] and Schlimmer [Schlimmer et al. 89] in their work on rule justification.

A disadvantage of the model compilation approach is that its applicability is limited to domains in which we have sufficient theoretical understanding to interconnect the various problem solving models. When there is no underlying theory for a domain, or when the theory is poorly understood, it will be difficult or impossible to link models using compilers. In these cases, it may be necessary to use disjoint models. However, we believe there are a significant number of theory-rich domains in which this approach is feasible. Another disadvantage of the model compilation approach is that there may be a significant amount of extra knowledge engineering work involved in building the model compilers. Our experience in creating the RWA system testifies to the extra effort involved. Our methodology was to first acquire simple associational models for our troubleshooting and redesign tasks from domain experts. We then proceeded to reverse-engineer these models with the aid of domain experts, creating a sequence of increasingly more detailed models in the process. We then constructed the compilers to produce these models in the "proper"

direction, from more detailed to less detailed. This reverse engineering process can be time-consuming. Still, we believe the benefits of the compilation approach will outweigh its costs in many situations.

Overall, the model compilation approach is promising, but the work reported in this paper takes only the first steps toward fully realizing this approach. We have developed two specific model compilers, but we have not developed a general mechanism for accomplishing model compilation. In particular, the transformations between models have been implemented procedurally, rather than by applying a sequence of explicit transformation operators. The sequence of compilation steps has been predetermined and "hard-wired" into each compiler. Thus we have avoided the difficult problem of automated control of the compilation process, which is addressed in [Mostow 83, Keller 87, Ellman 88, Tong 89].

5.3 Related Work

For two-model systems, there has been some research on automating derivation of so-called "shallow" models from "deeper" models [Chandrasekaran & Mittal 83, Sembugamoorthy & Chandrasekaran 86, Araya & Mittal 87, Brown & Sloan 87, Pearce 88, Singh 88, Console & Torasso 88, Steels & Van de Velde 86, Pazanni 86]. However, there has been little research into automated derivation for systems with more than two models, where the problems associated with consistency maintenance are exacerbated due to multiple interactions. Davis, however, describes a system that is able to generate a sequence of successive component interaction models for use in troubleshooting digital circuits [Davis 84]. This work is quite similar in spirit to the work described in this paper. However, Davis' system generates the models in the reverse-compilation order -- from simpler to more detailed. (This suggests that our notion of a base model must be encoded implicitly in the procedures Davis uses to generate this reverse-compilation sequence.) Also, whereas the example in [Davis 84] focuses on generating a single sequence of models, we present two different sequences originating from the same base model, where there are interactions among models in both sequences. Although we have applied model compilation in the domain of reasoning about engineered devices, [Dietterich 86] describes related work on knowledge compilation in the context of software engineering, machine learning, expert systems, and problem reformulation.

5.4 Summary

We have introduced the notion of model compilation as an approach to automated model derivation for knowledge based systems. With an implemented example, we illustrated how this approach interrelates models represented at different levels of precision and abstraction, and from different task perspectives. This approach lessens the burden of multi-model system maintenance by automatically propagating model changes through a network of interrelated models.

ACKNOWLEDGMENTS

Our thanks to Graham Ross and Singaravel Murugesan, who served as experts on the RWA device. Avis Austin at Lockheed AI Center provided us with the set of RWA fault localization rules that ultimately served as the target for the diagnostic compiler. We would like to thank members of the Stanford LMKB project for their support of this effort. Tom Gruber, Jane Hsu, Smadar Kedar, and Michael Sims provided helpful comments on earlier drafts of this paper. This research has been supported by NASA under Grant # NCC2-537.

REFERENCES

[Abbott 88] K. Abbott, "Robust Operative Diagnosis as Problem Solving in a Hypothesis Space". In *Proceedings of AAAI-88*, St. Paul, MN, pp. 369-374, August 1988.

[Araya & Mittal 87] A. Araya and S. Mittal, "Compiling Design Plans from Descriptions of Artifacts and Problem Solving Heuristics". In *Proceedings IJCAI-87*, Milan, August 1987, pp. 552-558.

[Brown & Sloan 87] D.C. Brown and W.N. Sloan, "Compilation of Design Knowledge for Routine Design Expert Systems: An initial view". In *Proceedings ASME International Computers in Engineering Conference*, New York, NY, Vol. 1, pp. 131-136, 1987.

[Chandrasekaran & Mittal 83] B. Chandrasekaran and S. Mittal, "Deep versus Compiled Knowledge Approaches to Diagnostic Problem-solving". *International Journal of Man Machine Studies*, Vol. 19, pp. 425-436, May 1983.

[Console & Torasso 88] L. Console and P. Torasso, "Compiling Causal Models into Heuristic Knowledge", technical report, Dipartimento di Informatica, Universita di Torino, Italy, March 1988.

[Davis 84] R. Davis, "Diagnostic Reasoning Based on Structure and Behavior", *Artificial Intelligence*, vol. 24, 1984.

[Dietterich 86] T. G. Dietterich, *Proceedings of the Workshop on Knowledge Compilation*, Otter Crest, OR, September 1986, Oregon State University technical report.

[Ellman 88] T. Ellman, "Approximate Theory Formation: An Explanation-Based Approach". In *Proceedings AAAI-88*, St. Paul, Minnesota, pp. 570-574, August 1988.

[Falkenhainer & Forbus 88] B. Falkenhainer and K.D. Forbus, "Setting up Large-Scale Qualitative Models". In *Proceedings of AAAI-88*, St. Paul, MN, pp. 301-306, August 1988.

[Fink 85] P. Fink, "Control and Integration of Diverse Knowledge in a Diagnostic Expert System". In *Proceedings of IJCAI-85*, Los Angeles, California, pp. 426-431, August 1985.

[Iwasaki 88] Y. Iwasaki, "Causal Ordering in a Mixed Structure". In *Proceedings AAAI-88*, St. Paul, Minnesota, pp. 313-318, August 1988.

[Keller 87] R.M. Keller, *The Role of Explicit Contextual Knowledge in Learning Concepts to Improve Performance*, Ph.D. thesis (technical report # ML-TR-7), Department of Computer Science, Rutgers University, 1987.

[Keller et al. 89] R.M. Keller, C. Baudin, Y. Iwasaki, P. Nayak, and K. Tanaka, "Compiling Diagnosis Rules and Redesign Plans from a Structure/Behavior Device Model: *The details*". Technical report No. KSL-89-50, Knowledge Systems Laboratory, Stanford University, June 1989.

[Koton 88] P. Koton, "Reasoning about Evidence in Causal Explanations". In *Proceedings of AAAI-88*, St. Paul, MN, pp. 256-261, August 1988.

[Mostow 83] J. Mostow, "A Problem-solver for Making Advice Operational". In *Proceedings of AAAI-83*, Washington, DC, pp. 279-283, August 1983.

[Murthy & Addanki 87] S.S. Murthy and S. Addanki, "PROMPT: An Innovative Design Tool". In *Proceedings of AAAI-87,* Seattle, WA, pp. 637-642, August 1987.

[Patil 81] R. Patil, "Causal Representation of Patient Illness for Electrolyte and Acid-Base Diagnosis", MIT LCS Technical Report #267, 1981.

[Pazzani 86] M. Pazzani, "Refining the Knowledge Base of a Diagnostic Expert System: An application of Failure-Driven Learning". In *Proceedings of AAAI-86,* Philadelphia, PA, pp. 1029-1035, August 1986.

[Pearce 88] D.A. Pearce, "The Induction of Fault Diagnosis Systems from Qualitative Models". In *Proceedings of AAAI-88*, St. Paul, MN, pp. 353-357, August 1988.

[Perkins & Austin 87] W.A. Perkins and A. Austin, "Experiments with Temporal Reasoning Applied to Analysis of Telemetry Data". In *Space Station Automation III*, vol. 851, pp. 39-46, Society of Photo-Optical Instrumentation Engineers, 1987.

[Schlimmer et al. 89] J.C. Schlimmer, T.M. Mitchell, and J. McDermott, "Justification-Based Refinement of Expert Knowledge". In *Proceedings of the IJCAI-89 Workshop on Knowledge Discovery in Databases*, Detroit, Michigan, August 1989.

[Sembugamoorthy & Chandrasekaran 86] V. Sembugamoorthy, and B. Chandrasekaran, "Functional Representation of Devices and Compilation of Diagnostic Problem-Solving Systems". In Kolodner, J.L. and Riesbeck, C.K. (editors), *Experience, Memory, and Reasoning,* Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Simmons & Davis 87] R. Simmons and R. Davis, "Generate, Test and Debug: Combining Associational Rules and Causal Models". In *Proceedings IJCAI-87*, Milan, August 1987, pp.1071-1078.

[Singh 88] N. Singh, "Generating Diagnostic Procedures for Discrete Devices". Logic Group Technical report No. Logic-88-7, Computer Science Department, Stanford University, August 1988.

[Smith et al. 85] R.G. Smith, H.A. Winston, T.M. Mitchell, and B.G. Buchanan, "Representation and Use of Explicit Justifications for Knowledge Base Refinement". In *Proceedings IJCAI-85*, Los Angeles, CA, pp. 673-680, August 1985.

[Steels & Van de Velde 86] L. Steels and W. Van de Velde, "Learning in Second Generation Expert Systems", in *Knowledge-based Problem Solving*, J.S. Kowalik (ed.), Prentice Hall, 1986.

[Sussman 78] G.J. Sussman, "Slices: At the Boundary Between Analysis and Synthesis". In *Artificial Intelligence and Pattern Recognition in Computer-Aided Design*, J.-C. Latombe (Ed.), North-Holland, Amsterdam, 1978, pp. 261-299.

[Swartout 83] W.R. Swartout, "XPLAIN: A System for Creating and Explaining Expert Consulting Programs", *Artificial Intelligence*, vol. 21, no. 3, pp. 285-325, 1983.

[Tong 89] C. Tong, "Toward Knowledge Compilation as A Classification Process". In *Proceedings of the 1989 IJCAI Workshop on Automating Software Design*, Detroit, Michigan, August 1989, pp. 280-289.