p-21

NASA

Ames Research Center

In Defense of Compilation:
A response to Davis' "Form and Content in Model-based Reasoning"

Richard M. Keller

Report FIA-90-06-25-1
May, 1990

RIA
Artificial Intelligence
Research Branch

In Defense of Compilation

A response to Davis' "Form and Content in Model-based Reasoning" [1]

Richard M. Keller [2]

Artificial Intelligence Research Branch
NASA Ames Research Center
Mail Stop 244-17
Moffett Field, CA 94035-1000

(415) 604-3388
keller@pluto.arc.nasa.gov

June 1990

Abstract

In a recent paper entitled "Form and Content in Model-Based Reasoning" [Davis 89], Randy Davis argues that model-based reasoning research aimed at compiling task-specific rules from underlying device models is "mislabeled", "misguided", and diversionary. In this paper I examine some of Davis' claims and challenge his basic conclusions about the value of compilation research to the model-based reasoning community. In particular, I refute Davis' claim that model-based reasoning is exempt from the efficiency benefits provided by knowledge compilation techniques. In addition, I clarify several misconceptions about the role of representational form in compilation. I conclude that compilation techniques have the potential to make a substantial contribution to solving tractability problems in model-based reasoning.

[1]To appear in the *Proceedings of the AAAI-90 Workshop on Model-based Reasoning*, Boston, MA, July 1990.
[2]Employed under NASA contract #NAS2-11555 to Sterling Federal Systems.

Table of Contents

1. Introduction

Randy Davis, in a paper entitled "Form and Content in Model-Based Reasoning" [Davis 89], makes some provocative statements about the intrinsic value of recent research aimed at compiling task-specific rules from underlying structure/behavior device models. Specifically, he characterizes such work as "mislabeled", "misguided", and diversionary. This paper examines some of Davis' assumptions and challenges his basic conclusions about the value of compilation research to the model-based reasoning (MBR) community.

A primary motivation for applying knowledge compilation techniques to MBR systems is to improve their run-time efficiency on tasks such as diagnosis, design, and simulation. For devices of only moderate complexity, MBR procedures can be computationally intractable, and this presents a significant barrier to their adoption in practical applications [Falkenhainer & Forbus 88, E.Davis 87]. The compilation approach attempts to improve efficiency by customizing and streamlining the general-but-inefficient procedures and models of structure and behavior typically used in MBR (for examples, see [Bobrow85]). Specifically, the approach advocates automatically compiling these models into special-purpose, task-specific models that can be used with efficient, customized reasoning procedures at run-time. Compilation circumvents run-time computation using a variety of techniques including precomputation, partial evaluation, abstraction, and approximation. If the compiled, special-purpose models prove inadequate at run-time, underlying models can still be accessed and reasoned with using traditional MBR techniques. Several recent research efforts have focused on applying compilation techniques to MBR, including those described in [Chandrasekaran & Mittal 83, Sembugamoorthy & Chandrasekaran 86, Araya & Mittal 87, Brown & Sloan 87, Pearce 88, Singh 88, Console & Torasso 88, Steels & Van de Velde 85, Pazanni 86, Keller *et al.* 89, 90].

In following subsections, I review Davis' claims about the application of knowledge compilation to MBR, and present my basic response to these claims.

1.1 The claims

Davis' main contention is that although compilation research focuses on a laudable goal -- improving the computational tractability of MBR -- the basic approach to achieving that goal is fundamentally flawed. According to Davis, compilation work seeks to improve tractability by changing the *form* of a model -- rather than its *content*. The emphasis on form is misplaced, Davis contends, because "... speed is primarily a property of model content (level of detail), not form (conditional statements or causal models)." As an example, Davis focuses on recent research involving rule compilation (e.g., [Keller *et al.* 90, Pearce 88]). This type of compilation focuses on transforming general-purpose device models into task-specific associational rules. Davis sees no advantage to representing structure, behavior, and causality using associational rules rather than the component-centered models advocated by MBR. As Davis points out, rules are not intrinsically more efficient than other representational forms. Moreover, rules tend not to be as compact or transparent as typical MBR models.

A secondary set of Davis' claims focus on the general inapplicability of knowledge compilation techniques to MBR. Davis claims that the attempt to compile diagnosis rules is "largely futile" because it involves precomputing all potential diagnostic outcomes, and this is infeasible except in the most trivial cases. If precomputing is infeasible, he argues, then it will be necessary to use predictive models in conjunction with rules at run-time. And this seems to defeat the purpose of compiling special-purpose rules in the first place. More broadly, Davis contends that any attempt to reduce deliberation and search using precomputation or related techniques is "simply not applicable" to MBR, because MBR employs only "sharply focused" (i.e., not excessively deliberative) procedures.

Based on his analysis, Davis concludes that to combat intractability, researchers in MBR should focus on changes in a model's content, not form. In particular, his apparent suggestion is that MBR researchers should reject both compilation and rules, and focus on the development of automated methods for reducing the information content of causal, structural, and behavioral models -- i.e., "techniques for producing [from a detailed model] a series of ever more abstract and approximate ... models".

1.2 The response

A rebuttal to Davis involves responding to several fundamental fallacies about knowledge compilation that underlie his conclusions:

1. ***The "form is irrelevant" fallacy:***

   While I heartily agree that methods which reduce a model's information content are important weapons in the war on intractability, they are *not* the only methods at our disposal. Contrary to Davis' assertion, significant efficiency improvements in MBR *can* be derived from changes in the *form* of a model;

2. ***The "compilation as change-in-form" fallacy:***

   The compilation paradigm encompasses not only changes in the form of models, but changes in their *content*, as well;

3. ***The "rule emphasis" fallacy:***

   Production of rules, per se, is *not* intrinsic to compilation, nor is it the aim of that process. The fact that rules have been the target representation in some compilation applications is only incidental. Compilation seeks to produce an *operational* (i.e., efficiently usable) representation in whatever target form is suited to the reasoning engine;

4. ***The "MBR special exemption" fallacy:***

   Despite Davis' claims to the contrary, MBR is *not* exempt from the type of performance improvement afforded by knowledge compilation techniques. In general, there are many opportunities to substitute knowledge for search in MBR procedures. In particular, for many types of devices, it is possible to avoid predictive use of the device model at run-time by consulting alternative knowledge sources to determine device behavior.

The following sections examine and debunk the above fallacies in some detail, and also discuss some terminological issues raised by Davis. I argue that compiled reasoning procedures are not incompatible with model-based approaches, and are not intended to supplant them; each approach has its own contributions. The challenge is to combine these two approaches to advantage. Before I begin my examination of Davis' points, I describe what is meant by the term "knowledge compilation" and review some general motivation for pursuing research in this area.

2. What is "knowledge compilation"?

The term "knowledge compilation" was first coined by Neves and Anderson in 1981 to refer to specific cognitive phenomena in their work on modeling human problem solving [Neves & Anderson 81]. The term later took on broader meaning with the convening of the 1986 Workshop on Knowledge Compilation [Dietterich 86]. Although there is no single, universally-accepted definition for the term, there are a number of alternatives to be found in the literature:

Definition #1 (based on [Neves & Anderson 81]): Knowledge compilation is the process of shifting from a declarative to a procedural form of knowledge representation.

Definition #2 (based on preface to [Dietterich 86]): Knowledge compilation is the process of automatically transforming explicit, but inefficient, knowledge representations into implicit, but more efficient forms.[3]

Definition #3 (based on [Tong 89]): Knowledge compilation is the process of producing knowledge-based systems from higher level specifications.

What these definitions have in common is summarized well in [Brown 89], where knowledge compilation is characterized as a process that results in:

- Increased efficiency or usability;
- Altered representation level;
- Reduced or "short-circuited" reasoning;

To these characteristics, I add:

- Decreased transparency or explicitness.

For the purposes of this paper, we can view knowledge compilation as ***the process of transforming some of the knowledge structures used by a given reasoning system in order to improve the system's run-time efficiency.***[4] To accomplish knowledge compilation, a sequence of transformations is applied to a "source" knowledge structure to produce an efficiently-usable "target" structure. As applied to MBR, the source structure to be transformed consists of a model of the structural, behavioral, and/or causal properties of a device or system. Compilation into a variety of target knowledge structures has been explored, including diagnostic rules [Pearce 88, Keller *et al.* 90], design plans [Araya & Mittal 87, Keller *et al.* 90], "specialist" hierarchies for diagnosis [Sembugamoorthy & Chandrasekaran 86] and for design [Brown & Sloan 87], and decision trees [Singh 88].

Following are some examples of transformations used in knowledge compilation. I divide these transforms into two categories based on the effect that their application has on the "knowledge-level" content of the source structure [Newell 82].

[3]A somewhat generalized variant of Definition #2 is based on [Keller 88], which substitutes the more general goal of operationality for the stated goal of efficiency. An *operational* representation is one that can be used by a given problem solver to achieve a set of specified performance objectives. The time efficiency objective mentioned in Definition #2 is only one possibility out of many. Others include improved space efficiency, improved accuracy, and improved explanatory power, for example.

[4]Note that in general, compilation can affect system efficiency by modifying knowledge structures *or* procedures (see Section 4.1 below). For those working on compilation within the logic programming paradigm (e.g., [Genesereth & Hsu 89, Preditis & Mostow 87]), there is no distinction between knowledge structures and procedures. I emphasize the knowledge structure viewpoint in this paper because research in MBR tends to distinguish between the device model and the reasoning engine.

Content-preserving transforms: maintain "knowledge-level" content

- Simplification -- Eliminates redundancies and extraneous syntactic detail. For example:
  - ° Moving the assignment of a constant outside a loop in a program [Gries 71];
  - ° Eliminating a replicated disjunctive subexpression or collapsing nested conjunctive expressions [Minton 88].
- Macro-operator formation -- Creates a single problem solving operator that has the effect of applying several existing operators in sequence [Fikes *et al.* 72].
- Pre-computation -- Stores and reuses the results of a computation to avoid subsequent re-execution; a type of caching or chunking [Laird *et al.* 87].
- Partial evaluation -- Produces a specialized version of a program by incorporating knowledge about restrictions on the program inputs [Kahn 84].

Content-modifying transforms: change "knowledge-level" content

- Threshold application -- Converts a real-valued variable into a binary feature by applying a numerical cut-off [Keller *et al.* 90].
- Qualitative transformation -- Converts a quantitative equation into a qualitative equation based on assumptions about the signs of variables [Iwasaki 90].
- Information elimination -- Ignores information that is costly to use, yet contributes little to accuracy [Keller 87, Subramanian & Genesereth 88].
- Functional approximation -- Treats a function as invariant with respect to one or more of its arguments [Ellman 88].

3. Motivation for applying knowledge compilation to MBR

There are various motivations for studying knowledge compilation in the context of MBR. These include:

- **Generality/efficiency trade-off:** In MBR, as with many types of reasoning, there is often a trade-off between generality (in the sense of problem solving coverage or scope) and efficiency. If we wish to perform diagnosis using general representational structures and general procedures capable of handling every possible diagnostic eventuality, we will pay a price in terms of efficiency. On the other hand, if we are willing to settle for less generality, we may be able to develop specialized diagnostic procedures and structures that can handle a restricted subclass of problems more efficiently.[5] This poses a dilemma because ideally, we do not wish to sacrifice the generality afforded by MBR procedures and structures in order to solve the efficiency problem.

  Knowledge compilation provides a solution that bridges the gap between the two endpoints of the generality/efficiency spectrum. Knowledge compilation techniques can automatically derive efficient, specialized representational structures from more general-purpose structures by capitalizing on problem class restrictions. Because the derivation path is retained, it is possible to maintain a dual representation and reasoning scheme -- reasoning with the efficient

[5]First generation expert systems can be viewed as special-purpose systems that handle only a subset of the problems in a given domain. The knowledge engineer ensures that the rules perform reasonably on the most typical, or frequently-recurring cases. These systems have proven useful despite their lack of generality.

specialized structures when possible, and reverting to reasoning with the general structures when necessary.

- **Representational mismatch:** Sometimes the forms of explicit models traditionally used by MBR systems are simply incompatible with the form required by the designated problem solving architecture. For example, suppose that we wish to use the Soar learning and problem solving architecture [Laird *et al.* 87] to do diagnosis. A device model developed for a particular MBR system is a potentially valuable knowledge resource, but it cannot be used by Soar without some translation into a "problem space" representation. Potentially, this type of translation can be accomplished with the aid of knowledge compilation techniques.

- **Learning:** Knowledge compilation can be considered a form of "speed-up" learning, and is closely related to work on explanation-based learning (EBL) [Mitchell *et al.* 86]. The first-principles model corresponds to a domain theory in EBL, and the compiled target structures correspond to the operational descriptions learned in EBL. The derivation of a target structure can be viewed as a type of "explanation" or justification of the structure based on the underlying model. Model-based reasoning is an application area in which to apply knowledge compilation techniques and extend our understanding of the strengths and limitations of EBL.

Now that I have provided some background on knowledge compilation, I turn to the main thrust of this paper. Section 4 addresses the first three of Davis' fallacies about knowledge compilation, as outlined in the Section 1. The fourth fallacy is addressed in Section 5.

4. The Form Fallacies

The first three misconceptions about knowledge compilation all relate to issues of representational form. They include the "form is irrelevant" fallacy, the "compilation as change-in-form" fallacy, and the "rule emphasis" fallacy.

4.1 The "form is irrelevant" fallacy

One of Davis' central tenets is that "... speed is primarily a property of model content ... not form ...". Davis never defines what he means by "content" or "form", but we can attempt to interpret his statement in terms of the knowledge level / symbol level dichotomy introduced in [Newell 82]. Davis apparently believes that changes in the knowledge level description of a system typically have a more profound impact on efficiency than changes at the symbol level. Thus, Davis downplays the importance of implementation (i.e., symbol level) concerns such as data structures and algorithms, and instead focuses on reducing the information content in a model as the best way to improve efficiency.

Although information content is one important factor to consider, it is essential to maintain a balanced perspective on efficiency improvement. The efficiency of a model is a function of *several* factors, and the model's information content is only one of those factors. To make legitimate claims about efficiency, it is necessary to go beyond the knowledge level and address symbol level concerns. A model by itself is neither intrinsically efficient nor inefficient; questions of efficiency must be addressed in the context of a specific *representational form* and a specific *interpreter*. To affect a model's performance one can either 1) modify the model's information content, 2) modify the model's interpreter, or 3) modify the model's representational form[6].

[6]These types of modifications are not necessarily independent. For example, modifications made to the model's representational form will necessitate changes in the model interpreter.

4.2 The "compilation as change-in-form" fallacy

A corollary to the "form is irrelevant" fallacy is the notion that compilation merely effects changes in form. As described in Section 2, certain knowledge transformations preserve content and others modify content. Thus knowledge compilation can be used to produce the kinds of approximate and/or abstract models that Davis desires. POLLYANNA [Ellman 88] and MetaLEX [Keller 87] are examples of knowledge compilers that produce approximations of a given model or theory.

4.3 The "rule emphasis" fallacy

Part of Davis' point seems to be that he does not see any advantage to rules over other representational forms -- in terms of efficiency or other measures, such as transparency and conciseness. Certainly Davis is correct that rules are not intrinsically more efficient than other representational forms. If unrestricted, rules may perform arbitrarily complex computation. However, in the work cited on rule compilation, the objective is to produce rules with a special property that makes them efficient to use. The crucial property of the desirable rules is that they test *easily recognizable patterns in the data.* In other words, the attributes tested in the conditions of the rules must either be directly available in the observed data or must be calculable from the observed data with minimal cost. In a troubleshooting setting, a rule involving a device state that can be detected using only low-cost probes is an example. The point is that rule compilation focuses on producing these types of desirable, low-cost rules, not arbitrary rules.

Furthermore, there is nothing special about compiling into rules, per se, versus other representational formats. In fact, rules are just one of the target forms used in compilation research; others have been described in Section 2. The essential common element of these compilation approaches is that they take as input a relatively inefficient source representation and produce as output a more efficient target representation.

5. The "MBR special exemption" fallacy

A significant portion of Davis' paper is devoted to arguing that knowledge compilation has little to offer to MBR. He claims that MBR is exempt from the benefits provided by knowledge compilation, despite the fact that knowledge compilation techniques are broad-ranging and widely applicable. In particular, Davis claims that MBR procedures involve "no avoidable search" or computation, so there is little room for improving efficiency with knowledge compilation techniques.

My basic response to this assertion is that considerable search *can* be avoided in the execution of MBR procedures providing we are willing to narrow the scope of the problems being addressed. MBR procedures, such as dependency tracing and guided probe selection [Davis & Hamscher 88], are computationally intractable for large devices precisely because they fail to make restrictive assumptions about the class of diagnostic problems to be solved. These procedures exhaustively consider the space of consistent hypotheses, and ignore problem-specific information that may serve to limit the search. In contrast, knowledge compilation techniques trade coverage for efficiency by exploiting known problem class regularities. In particular, the techniques generate efficient, specially-tailored representations that are applicable to only a limited class of problems. If a problem falls outside of this class, the system can revert to the more comprehensive "first principles" approach. Note however that to actually reap efficiency benefits from diagnostic compilation, the cost of compilation must be successfully amortized over many diagnostic episodes. If the assumptions underlying a compiled representation are violated too often, compilation will lose; otherwise it may win big, depending on just how costly it is to do the diagnosis at run-time.

The following subsections examine in detail Davis' basic argument that knowledge compilation does not apply to MBR. I will attempt to show that his argument is based on restrictive assumptions about both the types of devices being modeled and the types of tasks being addressed by MBR. In particular, many of his observations are specific to diagnostic reasoning about digital circuits, and do not extend to other types of reasoning tasks (e.g., design or training) or to other types of devices (e.g., analog electrical or mechanical). These and other points can best be illustrated in terms of the main example given in the Davis paper.

5.1 The decision tree example

The example introduced in Section 3.2 of [Davis 89] is based on the simple device illustrated in Figure 1. A traditional model-based diagnosis program, such as the program described in [Davis 84], would decide dynamically at diagnosis time which components to probe and in what sequence. These decisions would be made using some combination of dependency tracing and guided probe selection [Davis & Hamscher 88].

Davis points out, however, that a *decision tree* can be used in lieu of this dynamic decision procedure under certain conditions. Essentially, the decision tree caches the results of executing the (potentially expensive) dependency tracing and guided probe selection procedures. Rather than making the same types of run-time decisions repeatedly over many similar diagnostic episodes, a decision tree is precomputed only once -- at compile time. The decision tree in Figure 2 can be computed for the device in Figure 1 assuming 1) faults lie causally upstream of their effects, 2) there is a single point of failure, and 3) there is no fault masking in the device.

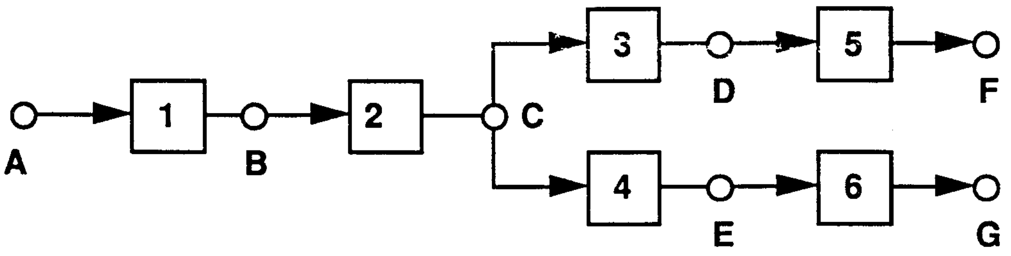

Figure 1: Simple device from [Davis 89]

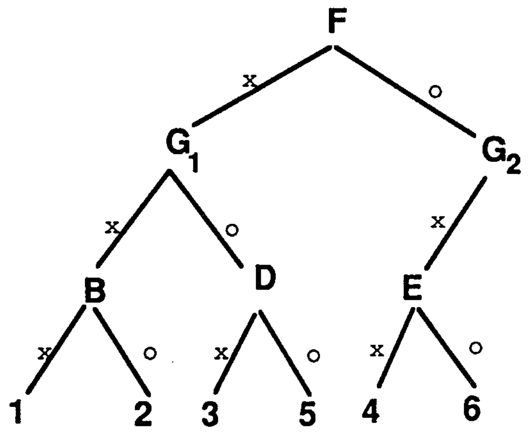

Figure 2: A decision tree for diagnosing the device in Figure 1. Also from [Davis 89]. Letters are measurement points in Figure 1; x indicates the value at that point was bad (i.e., not as predicted); o indicates it was ok (as predicted). Numbers at the leaves indicate which component is faulty.

After presenting the decision tree idea, Davis proceeds to critique this approach. In the following subsections, I paraphrase each point of his critique, and then present a counterargument.

5.2 Trading time for space won't help

<u>Davis' Critique (#1)</u>: *The creation of a decision tree merely trades execution time for execution space. And as the tree gets larger, memory overhead increases, and execution time will ultimately increase.*

The question of whether or not it is worthwhile to trade space for time is largely an empirical one; the answer will vary according to the particular device in question and the characteristics of the task environment. For example, if probe selection is very costly for a given device and if similar diagnostic problems are encountered frequently in the task environment, then the time/space conversion will be favorable. On the other hand, if probe selection is relatively cheap and each diagnostic problem faced is unique, the compilation process will not be worthwhile. The exact *extent* of improvement achievable via compilation for a given problem depends on a variety of factors ranging from particular device characteristics (e.g., component connectedness, complexity, fault patterns) to system implementation details (e.g., on what hardware the system is running). Hence, I find highly speculative Davis' claim that "the model based approach to the device in Figure 1 will not be slow, hence there is not much speed to reduce".

5.3 Run-time model usage is unavoidable

<u>Davis' Critique (#2)</u>: *There is no way to know which way to branch in the decision tree without comparing the actual values at the probe points with the nominal, expected measurements at those points. The only way to compute these nominal measurements is by executing the device model at run-time, which defeats the purpose of computing the decision tree in the first place.*

Davis stresses this point about the inevitability of run-time model usage several times in his paper, setting it up as the 'achilles heel' of the knowledge compilation approach to diagnostic reasoning. It is therefore critical to understand the weakness of the argument on this point.

Davis' statement that "use of the model at diagnosis time cannot be avoided" is simply not true. Certainly simulation is not the only way to generate expectations for device behavior. Perhaps for the types of digital devices Davis has focused upon in his work, expectations cannot be derived readily from alternative sources. But in general, expectations can also be generated from the following sources, for instance:

- *historical data*: empirical trends constitute a source of expectation for device behavior (e.g., "a temperature reading of 106°C is abnormal for component B because the reading has remained between 30°C and 35°C for the past 5 months");
- *engineering knowledge about nominal device function*: knowledge about the specifications of a device can provide expectations (e.g., "a temperature reading of 106°C is abnormal for component B because it is designed to function properly only in the range 20°C-40°C");
- *engineering knowledge about device structure*: knowledge about the structure and composition of a device can also provide expectation (e.g., "106°C is abnormal because B is made of a plastic that melts at 100°C");

- *alternative models*: rather than simulating the full, complex device model at run-time, simpler analytic models may be available to provide first-order expectations of sufficient quality for diagnosis (e.g., "106°C is an abnormal reading for component B because simple mechanical and thermal models indicates that under normal operation, B's temperature will not rise above 50°C").

5.4 The decision tree incorporates too many built-in assumptions

Davis' Critique (#3): *The decision tree builds in assumptions about the diagnostic process that may not be valid for the current diagnostic problem. For example, the tree incorporates assumptions about fault masking, fault transmission, and simultaneous fault occurrence, as well as assumptions about which kinds of diagnostic information are available and the order in which diagnostic information can be obtained.*

These types of assumptions are precisely the source of power in the compilation approach. In general, the greatest efficiency gains in compilation come about as a result of making powerful classwide assumptions. Unfortunately, these assumptions also limit the generality of compiled structures, such as the decision tree. If the assumptions underlying a compiled structure cannot be made with a reasonable degree of confidence, then the structure cannot be used, and a more comprehensive "first principles" reasoning approach is warranted.

Presumably, Davis reacts against the notion of built-in assumptions because traditionally, MBR research has strived toward an ideal of developing completely general reasoning methods and representations for tasks such as diagnosis. But in the limit, generality comes at a high price – one we may not be willing to pay. Knowledge compilation takes a different philosophical perspective and focuses on the automatic generation and use of more specialized, efficient methods and representations that are tailored to specific subclasses of reasoning problems (e.g., large, important, or frequently recurring subclasses). Another way to put this is that traditional MBR work focuses on *arbitrary-case reasoning,* whereas knowledge compilation focuses on *typical-case reasoning.* Yet because the compiled structures have been automatically derived, it is always possible to "fall back" on more detailed structures to handle atypical cases.

To illustrate the critical role of assumptions in compiling efficient representational structures, I return to Davis' example involving the device in Figure 1. Suppose we know that components 5 and 6 are highly reliable and may be exonerated *a priori* with a high degree of confidence. Under these assumptions, the collapsed decision tree in Figure 3 is more appropriate and more efficient for diagnosis than the original tree in Figure 2. In this tree, the value of the probes at E, D, and G_2 have been compiled away based on the no-fault assumptions. To see why, notice that because component 6 is assumed faultless, we know that the result of probing E at the branch point in Figure 2 must be a "bad" value; otherwise component 6 would be identified as faulty. If we know that the probe at E will always produce a "bad" value, then it is unnecessary to probe E at all, and the test can be eliminated from the decision tree. A similar line of reasoning eliminates the probe at D. To eliminate G_2 we utilize some knowledge about diagnosis. In particular, we know that diagnosis is not initiated unless one of the outputs is bad. If this is so, the value of output G must be "bad" because the value of the other output, F, is "good". So the test at G_2 is unnecessary.

Taking the example one step further, suppose we assume that component 2 is also faultless, in addition to components 5 and 6. Now we are able to eliminate probe B as well. This produces the further simplified decision tree in Figure 4. Making these various simplifying "no-fault" assumptions corresponds to making structural simplifications in the device. Figure 5 illustrates a simplified device structure that ignores the effects of components 2, 5, and 6.

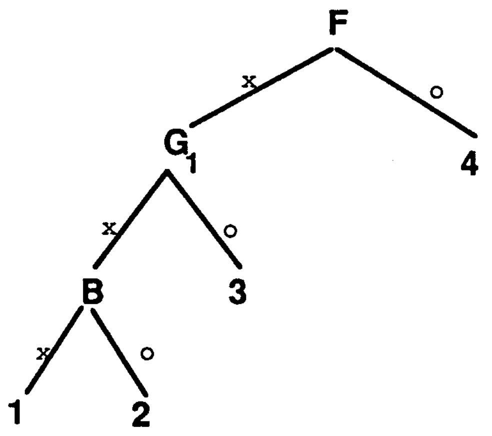

Figure 3: Collapsed decision tree assuming components 5,6 faultless

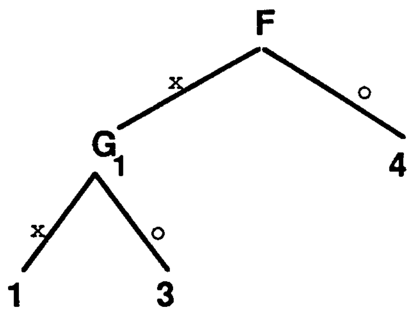

Figure 4: Collapsed decision tree assuming components 2,5,6 faultless

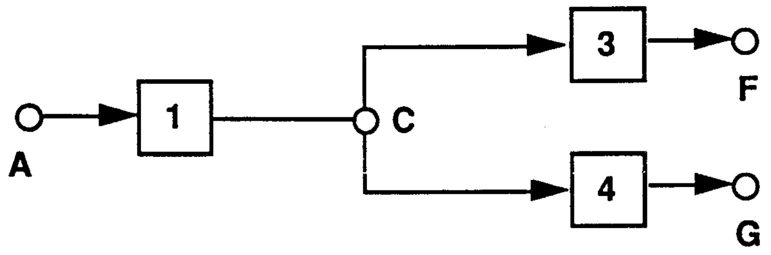

Figure 5: Abbreviated device structure assuming components 2,5,6 faultless

Note that the average number of probes required to diagnose a failure[7] is reduced from 3 probes for the original decision tree, down to 2.25 for the first abbreviated tree (Figure 3), and finally to 1.66 for the second abbreviated tree (Figure 4). However, the number of components being diagnosed also decreases with each simplification step. This is the classic tradeoff between generality and efficiency. The key is to improve efficiency by sacrificing as little generality as possible, *and without affecting diagnostic accuracy*. Under the stated assumptions, the decision trees in Figures 3 and 4 are more efficient and just as accurate for diagnosis as the original tree. However, if those assumptions are violated, the diagnostic results will be suspect.

5.5 And rules won't help either

In response to his observation that the decision tree representation incorporates too many assumptions, Davis proposes a second "strawman" representation. In particular, he suggests representing the diagnostic information in terms of "more modular" rules that relate measurement probes to faulty components. Rules of this type include:

A ok $\wedge$ B bad $\Rightarrow$ component 1 faulty
B ok $\wedge$ C bad $\Rightarrow$ component 2 faulty
C ok $\wedge$ E bad $\Rightarrow$ component 4 faulty
etc.

[7]The average number of probes required is obtained by averaging the path length of all paths in the decision tree leading from the root to a leaf.

Davis' Critique (#4): *If we were to cover all possible diagnostic contingencies in this fashion, we would experience a tremendous proliferation of rules and would see a significant proportion of time being spent on the associated rule management problems (i.e., on rule storage and rule access).*

First, as discussed above in Section 5.2, the question of efficiency is largely empirical. It is not possible to make simple generalizations that are valid across a wide variety of devices and task environments. Second, compilation advocates two basic approaches to rule management: (1) reduce the number of rules compiled, and (2) prune unproductive rules over time. The rule compilation approach does not advocate precomputing an entire set of rules corresponding to every diagnostic possibility. The objective is to compile a fairly limited set of rules that are relevant in the current problem solving environment. These rules should handle frequently recurring diagnostic problems encountered in the environment. Atypical problems can be handled by more in-depth "first principles" problem solving. Both statistically-based empirical and knowledge-based analytic approaches have been proposed in response to the problem of characterizing the problem solving environment in a form suitable for a knowledge compiler [Chase *et al.* 89, Keller 87]. However, this is a difficult problem and remains an open research issue. An alternative to reducing the number of rules compiled is *post hoc* rule pruning. Work on PRODIGY [Minton 88] illustrates how utility analysis can prune the set of compiled rules to improve a problem solver's performance.

5.6 Rules and decision trees obscure device structure and behavior

Davis' Critique (#5): *Precompiled rules and decision trees obscure the structure and behavior of a device without providing any obvious benefits. These representations are less transparent, less explicit, and less compact than traditional component-centered MBR representations.*

I agree that the component-centered representation is more transparent, explicit, and compact. But my primary argument is that precompiled rules and decision trees of the sort illustrated above *do* in fact yield an important benefit -- an efficiency benefit. Otherwise there is no point to representing them in this format versus a component-centered format (unless there is a representational mismatch, as described in Section 3). In any case, the rule compilation approach advocates *deriving* rules from underlying models, so the tranparency issue is moot. There should be no need to examine the compiled rules any more than one examines the object code output from a programming language compiler.

6. On terminology

Davis voices some criticism of the terminology used in the literature on knowledge compilation and MBR. Here, I review his criticism of the terms "compilation" and "deep versus shallow".

6.1 "Compilation"

Davis finds "compilation" to be an inappropriate metaphor for the work on MBR for a variety of reasons. Most fundamentally, he argues there is no improvement in efficiency analogous to the improvement achieved by a high-level language compiler, such as a FORTRAN compiler. His claim about lack of efficiency gains has been contested extensively in Section 5 of this paper.

Additionally, Davis complains that a programming language compiler *adds detail* to a given high-level specification in order to produce an executable machine-level program, whereas rule compilation *removes detail* by abstracting away aspects of structure and behavior. However, note that the "amount of detail" dimension is just one possible dimension to focus upon in drawing the compilation analogy; three other comparative dimensions identified in Section 2 -- explicitness, usability/efficiency, and representation level -- are just as reasonable. Based on these, the analogy to programming language compilation is quite apt. In particular, the "compilation" analogy can be justified by identifying high-level code with "explicit" knowledge and machine code with "implicit" knowledge, as follows:

Analogy between
Programming language compilation and Knowledge compilation:

programming language compilation:	features of **high-level languages**		features of **machine code**
	&	*—compilation-->*	&
knowledge compilation:	**structure/behavior models**		**associational rules**
decreased explicitness:	• explicit representation		• implicit representation
increased usability/ efficiency:	• not directly executable (or inefficient to execute)		• directly executable and efficient to execute
altered representation level:	• machine-independent source code or task-independent model		• machine-specific instructions or task-specific rules

6.2 "Deep" vs. "Shallow"

The terms "deep" and "shallow" are clearly non-technical[8] and over-used, but it is also clear that the current terminology is lacking when it comes to talking about relationships between models. The terms "causal", "structural", "behavioral", and "empirical" -- which Davis arguably characterizes as well-defined and sufficient -- are model *types*, characterizing the kind of knowledge represented in a given model. The terms "deep" versus "shallow", on the other hand, focus on relative relationships *between* models. The language for MBR is not as rich for describing inter-model relationships, such as relative differences in model veracity, grainsize, and level of abstraction. Recent work in this area [Hobbs 85, Addanki *et al.* 89, Weld 89] takes some important steps toward elaborating this vocabulary of interrelationships.

7. Summary

In this section, I summarize my response to Randy Davis' critique of efforts to apply knowledge compilation to MBR.

7.1 Points of agreement

Despite many points of disagreement, I concur wholeheartedly with Davis' main conclusion that a key research issue for MBR is the development of techniques for generating a series of successively more abstract and approximate models. In fact, our own work on model compilation [Keller *et al.* 90] takes some initial steps in this very direction. As the MBR community attempts to scale up to complex real-world

[8]Note that the terms "deep" and "shallow" originally were introduced in a high-level, prospective paper by Hart [Hart 82]. For an excellent discussion of these terms see [Karp & Wilkins 89].

domains, the fundamental intractability of current qualitative reasoning techniques will become increasingly evident [Falkenhainer & Forbus 88]. Abstraction and approximation are two extremely important weapons in the war on intractability in MBR [Weld 89, Addanki *et al.* 89, Falkenhainer & Forbus 88, Hamscher 88, Patil 83, Davis 84]. If one agrees that research on approximation and abstraction is central to MBR, it follows that research on compilation is *also* central because it focuses squarely on the process of generating new (and possibly more abstract or more approximate) models from existing models.

Lest I leave the reader with the false impression that knowledge compilation is a solved problem, let me briefly review the issues that must be addressed before applying compilation techniques in a given area. First, there is the issue of developing an appropriate set of knowledge compilation transformations to cover the problems in the domain. Second, there is the issue of how to characterize the class of problems for which the compiler should "optimize". This involves representing the problem-solving environment so that the knowledge compiler can determine what class-wide assumptions are reasonable to make. And third, there is the issue of controlling the compilation process. If compilation is viewed as a search through a knowledge reformulation space, it becomes apparent that considerable information is necessary to guide the search. Various control approaches have been investigated, including means-ends analysis [Mostow 83], hill-climbing [Keller 87, Ellman 88], and classification [Tong 89]. Search control is perhaps the single most difficult issue in automating knowledge compilation.

7.2 Points of disagreement

In what follows, I recap my main points of disagreement in terms of the four fallacies discussed in Sections 4 and 5.

7.2.1 The "form is irrelevant" fallacy

Form is not irrelevant to efficiency; form is critical to efficiency. Claims about the efficiency properties of a model cannot be made without defining both the model's form and its interpreter. A purely content-level analysis of a model reveals nothing about its performance characteristics. As potentially important as abstraction and approximation are in the battle against intractability, it is important not to dismiss -- as Davis does -- other available methods for reducing computational complexity. These methods include the so-called "content-preserving" methods, such as simplification, macro formation, precomputation, and others discussed in Section 2. Davis' blanket statements discounting the complexity-reducing power of these methods are unsubstantiated. Whether these methods produce significant efficiency improvements is still largely an empirical question. Very little experimentation has been done to either validate or refute Davis' claims.[9]

7.2.2 The "compilation as change-in-form" fallacy

Traditional compilers are restricted to applying only content-preserving transforms, and thus cannot produce approximations, in particular. However, in general, the knowledge compilation paradigm fully supports changes in content via the use of approximating transforms.

[9]Minton presents some evidence of the benefits resulting from a change in representational form based on his work with the PRODIGY learning system [Minton 88]. To improve overall system performance, a variety of content-preserving transforms are applied to a learned concept description during PRODIGY's compression phase. Empirical evidence illustrates the effectiveness of these transforms -- an excess of a 50% speedup over no transforms [ibid., pp. 130-131]. On the other hand, Minton's work also graphically illustrates the problem Davis alludes to with exhaustive generation of rules. With indiscriminate learning of new rules, PRODIGY's problem solver actually slows up as a result of learning. PRODIGY conducts a form of utility analysis to determine which rules are worth forming and saving over time.

7.2.3 The "rule emphasis" fallacy

Rules appear to be the scapegoat in Davis' critique of knowledge compilation techniques. Actually, the use of rules in compilation work is only incidental. Furthermore, no claim has been made that rules *in general* are better, faster, more expressive, more transparent, or anything of this nature. The objective of researchers in this area is not to 'turn models into rules', as Davis claims, but rather to develop automated methods to 'turn systems (such as MBR systems) with relatively *un*acceptable run-time behavior into systems with more acceptable behavior'.

7.2.4 The "MBR special exemption" fallacy

MBR is not exempt from the types of benefits afforded by knowledge compilation techniques. In particular, precomputing model predictions and embedding them into diagnostic rules is an effective technique for reducing run-time costs -- providing the compile-time costs are not prohibitive and can be successfully amortized over the lifetime of the system. If the compile-time costs are prohibitive, then selective (rather than exhaustive) compilation may be appropriate. A decision about which predictions to precompile depends on characteristics of the problem solving environment, such as the frequency of repeated problems and the cost of producing a prediction. In any case, Davis' strong claim that the precomputation approach is fatally flawed appears unsubstantiated. In particular, it is *not* necessary to use the original model to evaluate the precompiled rules at runtime. A variety of alternative knowledge sources can be used to produce the necessary normative expectations.

7.3 Points along a continuum

Clearly, compiled reasoning approaches are intended to work in tandem with model-based approaches, and not to supplant them. This paper argues that each style of reasoning has its place in the next generation of knowledge based systems: efficient, compiled approaches for "routine" reasoning, with a fall-back to more comprehensive model-based approaches for "extraordinary" reasoning. The challenge is to integrate these two types of approaches seamlessly. In fact, a less dichotomous and perhaps more realistic perspective on these two reasoning approaches is illustrated in Figure 6, where associational reasoning and model-based reasoning are identified as opposite endpoints along an spectrum of approaches ranging from more compiled to less compiled. Taking this perspective effectively shifts the discussion away from talk about 'models versus rules' and toward a more fruitful dialogue about what the two approaches have to offer each other, and how they may be successfully integrated.

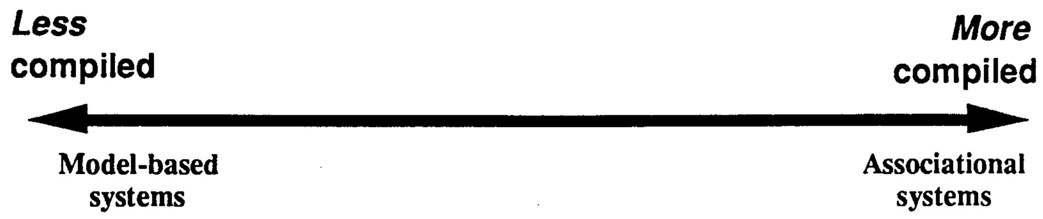

Figure 6: Spectrum of reasoning approaches

8. Conclusion

I hope this paper clears up some misconceptions about the goals of knowledge compilation work as applied to MBR, and further, that it promotes careful consideration of the benefits afforded by the compilation approach.

9. Acknowledgments

My thanks to Jane Hsu for sharing her perspective both on the Davis paper and on knowledge compilation in general. Jane's incisive observations made a significant mark on this paper. Several people have helped me gather my thoughts and have provided valuable feedback on earlier drafts of this paper. My thanks to Kathy Abbott, Nick Flann, Tom Gruber, Yumi Iwasaki, Smadar Kedar, Deepak Kulkarni, Steve Minton, Pandu Nayak, Ethan Scarl, Narinder Singh, and Dan Weld. Of course, the opinions expressed herein are my own, and are not necessarily sanctioned by my esteemed colleagues.

10. References

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

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

[Bobrow 85] D.G. Bobrow (ed.), *Qualitative Reasoning about Physical Systems,* MIT Press, Cambridge, MA, 1985.

[Brown 89] D.C. Brown, "Compilation: The hidden dimension of design systems", in *Proceedings of the Third IFIP Working Group 5.2 Workshop on Intelligent CAD*, Osaka, Japan, September 1989.

[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.

[Chase *et al.* 89] M. Chase, M. Zweben, R. Piazza, J. Burger, P. Maglio, and H. Hirsh, "Approximating Learned Search Control Knowledge". In *Proceedings of the Sixth International Workshop on Machine Learning*, pp. 218-220, Morgan Kaufmann, 1989.

[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.

[E.Davis 87] E. Davis, "Constraint Propagation with Interval Labels", *Artificial Intelligence*, vol. 32, no. 3, pp. 281-332, 1987.

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

[Davis 89] R. Davis, "Form and Content in Model Based Reasoning". In *Proceedings of the 1989 Workshop on Model-Based Reasoning,* Detroit, MI, pp. 11-27, August 1989. E. Scarl (ed.), Boeing Computer Services (publisher).

[Davis & Hamscher 88] R. Davis and W. Hamscher, "Model-based Reasoning: Troubleshooting". In *Exploring Artificial Intelligence,* H.E. Shrobe (ed.), Morgan Kaufmann, 1988.

[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 of AAAI-88,* St. Paul, MN, 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.

[Fikes *et al.* 72] R. Fikes, P. Hart, and N. Nilsson, "Learning and Executing Generalized Robot Plans", *Artificial Intelligence,* 3(4), 1972.

[Gries 71] D. Gries, *Compiler Construction for Digital Computers,* John Wiley & Sons, New York, 1971.

[Hamscher 88] W. Hamscher, *Model-based Troubleshooting of Digital Circuits,* Ph.D. thesis, MIT technical report # 1074, 1988.

[Hart 82] P. Hart, "Directions for AI in the eighties", *Sigart Newsletter,* No. 79, September 1982.

[Hobbs 85] J.R. Hobbs, "Granularity". In *Proceedings of IJCAI-85,* pp. 432-435, Los Angeles, CA, August 1985.

[Genesereth and Hsu 89] M.R. Genesereth and J. Hsu, "Partial Programs", technical report #Logic-89-20, Stanford University Department of Computer Science, 1989.

[Iwasaki 90] Y. Iwasaki, "Qualitative Physics", in *The Handbook of Artificial Intelligence, Vol. 4,* P.R. Cohen and E.A. Feigenbaum (eds.), in press.

[Kahn 84] K. Kahn, "Partial Evaluation as an Example of the Relationship between Programming Methodology and AI", *AI Magazine,* 5(1), Spring 1984.

[Karp & Wilkins 89], "An Analysis of the Distinction Between Deep and Shallow Expert Systems", Stanford Knowledge Systems Lab, technical report #KSL-89-10, January 1989.

[Keller 87] R.M. Keller, "Concept Learning in Context". In *Proceedings of the 4th International Workshop on Machine Learning,* Irvine, CA, pp. 91-102, June 1987.

[Keller 88] R.M. Keller, "Defining Operationality for Explanation-Based Learning", *Artificial Intelligence,* vol. 35, no. 2, pp. 227-241, 1988.

[Keller *et al.* 89] R.M. Keller, C. Baudin, Y. Iwasaki, P. Nayak, and K. Tanaka, "Deriving Shallow Rules from Underlying Device Models". In *Proceedings of the 1989 Workshop on Model-Based Reasoning,* Detroit, MI, pp. 124-128, August 1989.

[Keller *et al.* 90] R.M. Keller, C. Baudin, Y. Iwasaki, P. Nayak, and K. Tanaka, "Model Compilation: An approach to automated model derivation". Submitted to *AAAI-90.*

[Laird *et al.* 87] J.E. Laird, A. Newell, and P.S. Rosenbloom, "Soar: An architecture for general intelligence", *Artificial Intelligence,* vol. 33, no. 1, pp. 1-64, 1987.

[Minton 88] S. Minton, *Learning Effective Search Control Knowledge: An Explanation-Based Approach,* Ph.D. dissertation, Carnegie-Mellon University technical report #CMU-CS-88-133, March 1988.

[Mitchell *et al.* 86] T.M. Mitchell, R.M. Keller, and S.T. Kedar-Cabelli, "Explanation-Based Generalization: A Unifying View", *Machine Learning,* vol. 1, no. 1, pp. 47-80, 1986.

[Mostow 83] J. Mostow, "A Problem Solver for Making Advice Operational". In *Proceedings of AAAI-83,* Washington, DC, 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.

[Neves & Anderson 81] D. Neves and J.R. Anderson, "Knowledge Compilation: Mechanisms for the automatization of cognitive skills," in *Cognitive Skills and Their Acquisition,* J.R. Anderson (Ed.), Lawrence Erlbaum Assoc., Hillsdale, NJ, 1981.

[Newell 82] A. Newell, "The Knowledge Level", *Artificial Intelligence,* vol. 18, pp. 87-127, 1982.

[Patil *et al.* 81] R. Patil, P. Szolovits, and W. Schwartz, "Causal Understanding of Patient Illness in Medical Diagnosis". In *Proceedings of IJCAI-81,* pp. 893-899, August 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.

[Preditis & Mostow], "Prolearn: Towards a PROLOG interpreter that learns". In *Proceeding of AAAI-87,* Seattle, WA, August 1987.

[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.

[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.

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

[Subramanian & Genesereth 87] D. Subramanian and M.R. Genesereth, "The Relevance of Irrelevance". In *Proceedings of IJCAI-87,* Milan, pp. 416-422, August 1987.

[Tong 89] C. Tong, "Toward Knowledge Compilation as a Classification Process", in *Proceedings of the 1989 IJCAI Workshop on Automated Software Design,* Detroit, MI, August 1989, pp. 280-289.

[Weld 89] D.S. Weld, "Automated Model Switching: Discrepency Driven Selection of Approximation Reformulations", technical report #89-08-01, Department of Computer Science and Engineering, University of Washington, October 1989.