91490

P-14

Applying Knowledge Compilation Techniques to Model-based Reasoning

Richard M. Keller

Technical Report FIA-91-01
January 1991

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

Applying Knowledge Compilation Techniques to Model-based Reasoning*

Richard M. Keller**

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

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

January 1991

Abstract

Researchers in the area of knowledge compilation are developing general purpose techniques for improving the efficiency of knowledge-based systems. In this article, I attempt to define knowledge compilation, to characterize several classes of knowledge compilation techniques, and to illustrate how some of these techniques can be applied to improve the performance of model-based reasoning systems. I also review and respond to recent criticism of such efforts.

*Invited paper to appear in *IEEE Expert*, 1991.

**Employed under NASA contract #NAS2-13210 to Sterling Federal Systems.

1. Introduction

Over the past five years, a small community of Artificial Intelligence researchers has regrouped and emerged under the banner of "knowledge compilation". This formative community is highly interdisciplinary, and includes researchers from software engineering, automatic programming, expert systems, machine learning, and problem solving. What this diverse collection of researchers shares in common is a set of general programming techniques for improving the efficiency of software systems. Understandably, the focus of these AI researchers has been on improving the performance of "knowledge-based" (rather than conventional) software systems, but the techniques being developed are potentially applicable across the wide spectrum of software systems.

To date, most work reported in this area has described some particular system that employs knowledge compilation techniques. Relatively little effort has gone into cataloging and characterizing these techniques across systems. In this article, I attempt both to characterize several classes of knowledge compilation techniques and to illustrate how some of these techniques can be applied. In particular, I illustrate the application of knowledge compilation techniques to the specialized class of knowledge-based systems known as *model-based reasoning* (MBR) systems[1,2]. MBR systems use a comprehensive model of some system or device to perform "first-principles" reasoning about its structure, behavior, and causality. (In contrast, traditional expert systems use purely pattern-directed associational reasoning, without regard to underlying causal or behavioral mechanisms.) MBR systems have been used principally in the context of various diagnosis, design, and simulation tasks.

Before discussing knowledge compilation in the context of MBR systems, Section 2 introduces knowledge compilation techniques in general, and Section 3 provides some motivation for applying these techniques to MBR. Section 4 illustrates these ideas concretely with some examples from my own work in this area. Finally, in Section 5 I review some recent provocative arguments challenging the wisdom of applying knowledge compilation to MBR.

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 human skill acquisition. The term later took on broader meaning with the convening of the 1986 Workshop on Knowledge Compilation. Although there is no single, universally-accepted definition of the term, there are a number of alternatives to be found in the literature:

Definition #1 (based on Neves & Anderson[3]): Knowledge compilation is the process of shifting from a declarative to a procedural form of knowledge representation.

Definition #2 (based on preface to the 1986 Workshop[4]): Knowledge compilation is the process of automatically transforming explicit, but inefficient, knowledge representations into implicit, but more efficient forms.

Definition #3 (based on Tong[5]): Knowledge compilation is the process of producing knowledge-based systems from higher level specifications.

On the surface, these alternative definitions may appear to have little in common. But what these definitions share at a deeper level is summarized well by Brown[6], who characterizes knowledge compilation as the process of automatically restructuring existing software systems to produce new systems with:

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

To these characteristics, I add:

- Decreased transparency or explicitness.

It is instructive to compare the notion of *knowledge* compilation with more traditional *programming language* compilation by drawing an explicit analogy in terms of the above characteristics. In particular, the "compilation" analogy can be drawn by identifying high-level source code with explicit but inefficient knowledge structures, and target machine code with implicit but more optimized structures. This is depicted in the following table:

**Analogy between
Programming language compilation
and Knowledge compilation:**

	high-level source code	*programming language* *—compilation—>*	**machine code**
	initial knowledge structure	*knowledge* *—compilation—>*	**"optimized" knowledge structure**
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-neutral (or task-neutral) representation		• machine-specific (or task-specific) representation

So in more concrete terms, knowledge compilation can be viewed as a method of improving the run-time efficiency of a reasoning system by optimizing its knowledge structures. To accomplish knowledge compilation, a sequence of transformations is applied to a "source" knowledge structure to produce an efficiently-usable "target" structure. (More generally, note that compilation can affect system efficiency by modifying either the knowledge structures *or* procedures of a reasoning system. In fact, for those working on compilation within the logic programming paradigm[7,8] the distinction between knowledge structures and procedures is meaningless.)

The following are just a few 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"[9] content of the source structure. Some of the transforms maintain the "knowledge-level" content, and achieve their speed-up by reformulating and optimizing what the system already "knows". Other transforms alter the "knowledge-level" content of a reasoning system by adding to, or subtracting from the system's overall knowledge in order to improve its performance.

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

- **Simplification** – Eliminates redundancies and extraneous syntactic detail. For example:
  - ° In a program, moving the assignment of a constant outside of a loop[10];
  - ° Eliminating a replicated disjunctive subexpression or collapsing nested conjunctive expressions[11].
- **Macro-operator formation** – Creates a single problem solving operator that has the effect of applying several existing operators in sequence[12].

- **Pre-computation** -- Stores and reuses the results of a computation to avoid subsequent re-execution; a type of caching or chunking[13].
- **Partial evaluation** – Produces a specialized version of a program by incorporating knowledge about restrictions on the program inputs[14].

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

- **Threshold application** – Converts a real-valued variable into a binary feature by applying a numerical cut-off [15].
- **Qualitative transformation** – Converts a quantitative equation into a qualitative equation based on assumptions about the signs of variables[16].
- **Information elimination** -- Ignores information that is costly to use, yet contributes little to accuracy[17,18].
- **Functional approximation** -- Treats a function as invariant with respect to one or more of its arguments[19].

3. Motivation for applying knowledge compilation to Model-Based Reasoning

As previously mentioned, knowledge compilation techniques are general, and can be applied to any type of software system. Recently, there has been interest in applying these techniques to systems developed in several AI subareas, including MBR, planning, and problem solving. The particular focus of this article is on knowledge compilation as applied to MBR systems. As such, the source knowledge structure to be compiled consists of a model of the structural, behavioral, and/or causal properties of some device or system. Compilation of the source model into a variety of target knowledge structures has been explored, including diagnostic rules [13,20,21,22,23], design plans[13,24], design rules[25], "specialist" hierarchies for diagnosis[26] and design[27], and decision trees[28].

The 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. For example, candidate generation for multiple-point-of-failure diagnosis has been shown to be NP-complete. Model-based design and simulation tasks run into similar complexity barriers. The compilation approach attempts to improve efficiency by customizing and streamlining the general, but inefficient-to-use, models of structure and behavior typically employed in MBR. Specifically, the approach advocates automatically compiling these models into special-purpose, customized, task-specific models that can be used efficiently to solve a restricted subclass of problems. Thus, in applying knowledge compilation, we may sacrifice some generality for efficiency. Fortunately, if the special-purpose compiled models fail to cover a specific problem at hand, underlying models can still be accessed and reasoned about using traditional MBR techniques.

Now that I have provided some background on knowledge compilation, I turn to an example that illustrates more concretely the application of knowledge compilation techniques to model-based reasoning.

4. An Illustration

In this section, I describe a prototype knowledge compilation system capable of compiling both diagnosis and design rules from a general-purpose structure/behavior model of an engineered device -- the Reaction Wheel Assembly of NASA's Hubble Space Telescope. First, I briefly describe the Reaction Wheel Assembly and our general-purpose model of the device. Then, I describe one of our two knowledge compilers -- a diagnostic rule compiler, which transforms the structure/behavior model into a specialized set of fault localization rules for troubleshooting. Due to space limitations, I will not describe the second compiler, which produces abstract redesign plans from the same structure/behavior model. However, the interested reader is referred elsewhere for a more complete discussion[13,29].

4.1 The Reaction Wheel Assembly

The Reaction Wheel Assembly (RWA) is part of the pointing and control subsystem aboard NASA's Hubble Space Telescope (HST). 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.

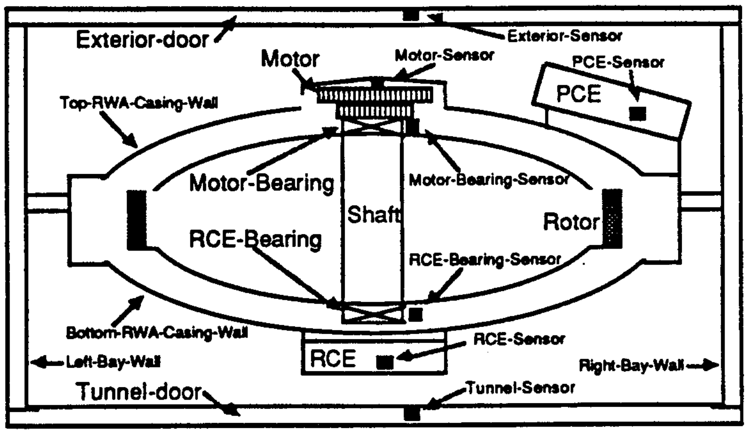

Figure 1: RWA cross-sectional view (adapted from Perkins & Austin[30], used with permission). The outside shell consists of a metal casing, which mounts directly to the telescope bay walls. Inside the casing is a hollow aluminum rotor with a steel rim, mounted on a rotating shaft. The shaft is connected to a motor at the top of the assembly. The Rotor Control Electronics (RCE) and the Power Control Electronics (PCE) supply power and control signals to the motor. Each end of the rotor shaft is supported by a bearing. Near each of the heat-generating components within the RWA (e.g., the bearings) there is a small temperature sensor used to monitor the device's functioning.

Our model of the RWA device was constructed using a frame-based, object-oriented knowledge representation tool. The model consists of two basic parts: a structural representation, and a behavioral representation. The structural part of the device representation consists of information about the component/subcomponent structure of the device, including the physical connectivity of the components and the spatial relationships among the components. For the purposes of our initial prototype, we utilized 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. These equations specify constraints among numeric quantities associated with device components. The behavioral equations are represented in both quantitative and qualitative format to facilitate different degrees of precision in reasoning.

4.2 Compiling Diagnosis Rules

This section describes how the general-purpose RWA model described above can be compiled into a set of special-purpose "fault localization" rules for diagnosis. Following is an example of a fault localization rule that the diagnostic compiler can produce. (This rule was extracted from an actual "shallow" RWA telemetry monitoring expert system built by Lockheed[28].)

```
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 could potentially influence the reading on the RCE-Bearing-Sensor. For example, the motor generates considerable heat, and so do the PCE and the exterior door (which heats up due to the sun). Why aren't these heat sources checked in the compiled rule? The answer is that the experts consider the influence of these heat sources to be negligible.

To produce R2, the rule compiler makes use of a simple, but general diagnostic 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. Here is a general diagnostic rule that captures the fault localization idea:

```
R1:  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.
```

Notice how the predicate *influences* 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 incorporates domain knowledge about specific source components and sensors. By "folding" this specific knowledge into R1 (i.e., by partially evaluating R1) we get the following intermediary rule, with the substituted terms from R1 underlined:

```
R1.5: If    Temperature of RCE-Bearing-Sensor is High
      and   (forall $k \neq i$ | influences($SRC_k$, RCE-Bearing-Sensor))
                    Reading 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. The necessary thermal influence model can be derived from the general-purpose RWA device model using the sequence of two compilation steps described below.

1. **Thermal Resistance Model Compilation:** The first compilation step is to produce a simple, quantitative thermodynamic model of the RWA by associating a numeric *thermal resistance* value with each heat flow path linking a heat source and a heat sensor within the device. The thermal resistance between two components is calculated as a weighted average of the thermal constants associated with the materials of the components in the heat flow path. In the resulting compiled model, determination of thermal resistance can be accomplished with a lookup of a single cached value, as opposed to the lengthy computation that would be required with the original device model.

2. **Thermal Influence Model Compilation:** Once the compiler has produced a quantitative thermal resistance model, the next step is to convert it into 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 can be considered inversely proportional to the amount of thermal resistance along the heat flow path. One way of deriving a binary thermal influence model from the scalar thermal resistance model is to threshold the thermal resistance value. The compiler considers any heat flow path with a resistance below a preset expert-defined threshold to be a path of thermal influence. Determining thermal influences is very efficient using this qualitative model, although influences could be computed -- at increased computational cost -- from either the quantitative resistance model or the original device model.

The thermal influence model can now be used to evaluate the *influences* predicate in R1.5 above, yielding the final compiled rule, R2. The rule compilation process is summarized in Figure 2.

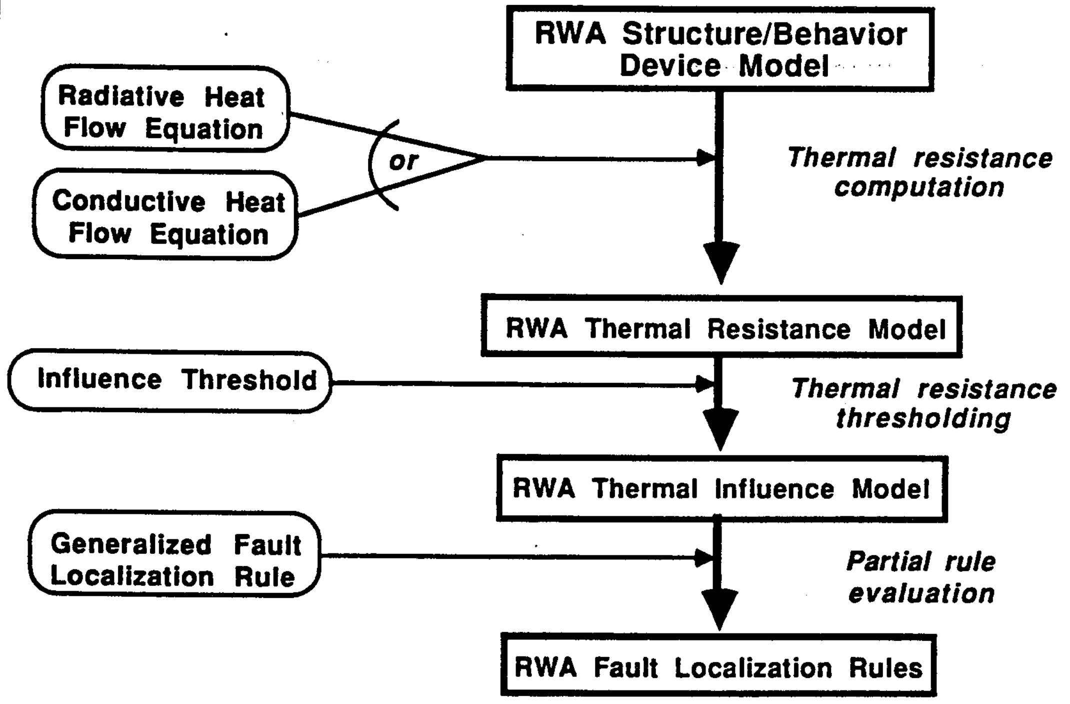
RWA Structure/Behavior Device Model
Radiative Heat Flow Equation
Conductive Heat Flow Equation
or
Thermal resistance computation
RWA Thermal Resistance Model
Influence Threshold
Thermal resistance thresholding
RWA Thermal Influence Model
Generalized Fault Localization Rule
Partial rule evaluation
RWA Fault Localization Rules

Figure 2: Compilation of diagnosis rules

4.3 Discussion

This example illustrates how knowledge compilation techniques can be applied to derive "shallow" rules from an underlying device model. The information content of the original RWA device model is reduced with each successive step taken by the diagnostic compiler. Each successive model generated is, in turn, more specially tuned and more efficient for the requirements of the troubleshooting task. As a result, troubleshooting with the compiled rules is more efficient than with the original model.

Furthermore, by compiling the diagnostic rules from an underlying device model, and generating a sequence of derivative models along the way (see Figure 2), we have provided additional knowledge that can be used to enhance the robustness of the entire reasoning system. In particular, if any information in the underlying model changes, the rules can be recompiled to reflect those changes. Using techniques from explanation and truth maintenance, if any of the shallow rules is found to perform poorly, the system should be able to construct a justification for the suspect rule in terms of one or more of the derivative models, and then isolate potentially incorrect assumptions or approximations used in producing those models. This approach has been examined in detail by Swartout.[31]

Although the run-time costs are clearly decreased by rule compilation, we cannot ignore the cost of compiling the rules when considering efficiency. Knowledge compilation trades off increased compile-time costs for decreased run-time costs. For this trade to be worthwhile in the long run, the compile-time costs must be successfully amortized over the system's performance lifetime. If the compiled rules are used only once before they become invalid due to changes in the underlying device model, then compilation will have turned out to be an expensive gamble. On the other hand, if the device remains stable, and the rules are used frequently, compilation will pay off handsomely.

Note that compilation is not an all-or-nothing decision. If compilation is expensive, we might choose to compile only a subset of the rules -- e.g., the small subset that covers 95% of the troubleshooting problems. First-principles reasoning can be used to handle the remaining 5%, without incurring the high compilation overhead. Of course determining which subset of rules to compile, and which troubleshooting problems are likely to occur may be extremely difficult. Estimating costs and benefits remains one of the most important and challenging research issues in knowledge compilation.

5. Dissenting Voices

Currently, the knowledge compilation subfield is in the midst of a developmental "identity crisis", struggling to identify the common purpose or set of unifying principles that unite its eclectic group of researchers. As a natural outcome of this self-examination process, some critical opposing views have been expressed. In particular, Davis[32] has criticized attempts to apply knowledge compilation to MBR. Although I have addressed these criticisms thoroughly elsewhere[33], I will briefly summarize his critique and my response here.

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)." Davis sees no advantage to compiling structure, behavior, and causal models into associational rules because rules are not intrinsically more efficient than other representational forms. Moreover, rules tend not to be as compact or transparent as typical MBR models. Davis suggests that rather than concentrating on *form*-changing compilation techniques to improve tractability, researchers ought to 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".

A secondary set of Davis' claims focus on the alleged 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 that this is infeasible except in the most trivial cases. Furthermore, 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.

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

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

   Contrary to Davis' contention, the knowledge compilation paradigm encompasses not only changes in the form of models, but changes in their *content*, as well. Traditional compilers are restricted to applying only content-preserving transforms, and thus do not alter the semantic meaning of the source code during the compilation process. In contrast, the knowledge compilation paradigm fully supports such changes by permitting the use of content-modifying transforms, as discussed in Section 2. Thus, for example, a knowledge compiler can produce an abstraction or an approximation of an input device model.

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

   While I heartily agree that methods which reduce a model's information content (such as approximation and abstraction) are important weapons in the war on intractability, they are *not* the only methods at our disposal. In principle, there is every reason to believe that significant efficiency improvements in MBR can be derived from changes in the *form* of a model. In practice, whether content-preserving compilation methods produce significant efficiency improvements is still largely an open empirical question. Insufficient empirical experimentation has been done to answer this question with any authority. Furthermore, a purely content-level analysis of a model reveals nothing about its efficiency characteristics. Any claims about the efficiency properties of a model must be made in the context of a particular representational form and an interpreter for that model.

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

   Rules appear to be the scapegoat in Davis' critique of MBR knowledge compilation techniques. Actually, the use of rules in MBR compilation work is only incidental. The production of rules, per se, is *not* intrinsic to compilation, nor is it the aim of that process. Compilation seeks to produce an efficient, usable representation in whatever target form is suited to the available reasoning engine. Rules are just one possible target representation; others are mentioned in Section 3. No claim has been made by compilation researchers that rules *in general* are better, faster, more expressive, or more transparent; certainly such a claim is false. 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 MBR systems with relatively *un*acceptable run-time behavior into systems with more acceptable behavior'.

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

   Despite Davis' claims, MBR systems are *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 – provided that 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.

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. And if one agrees that research on approximation and abstraction is central to MBR, it follows that research on compilation is *also* central because compilation provides a set of techniques for generating new (and possibly more abstract or more approximate) models from existing models.

6. Conclusions

Knowledge compilation is an emerging area of research within Artificial Intelligence. The aim of this research is to devise methods for automatically or semi-automatically improving the performance of knowledge-based systems. In general, knowledge compilation methods improve the performance of systems by tailoring their knowledge structures and/or procedures to a more narrowly-scoped class of tasks. Thus knowledge compilation techniques often (but not always) trade generality for efficiency.

In terms of model-based reasoning, it seems clear that both compiled reasoning approaches and model-based approaches have their 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. An integrative perspective on these two reasoning approaches is illustrated in Figure 3, 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 which approach is superior, and toward a more fruitful dialogue about what the two approaches have to offer each other, and how they may be successfully integrated.

Less compiled
More compiled
Model-based systems
Associational systems

Figure 3: Spectrum of reasoning approaches

7. Acknowledgments

The knowledge compilation system described in Section 4 was built at Stanford Knowledge Systems Lab with the assistance of Catherine Baudin, Yumi Iwasaki, Pandu Nayak, and Kazuo Tanaka, and was supported by NASA under Grant # NCC2-537.

My thanks to Jane Hsu for sharing her perspective both on the Davis paper and on knowledge compilation in general. Several people have helped me gather my thoughts and have provided valuable feedback on ideas expressed in this article. 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. Finally, Guy Boy and Amy Lansky gave me helpful comments on an earlier draft of this paper.

8. References

1. D. Bobrow (ed.), *Qualitative Reasoning about Physical Systems,* MIT Press, Cambridge, MA, 1985.

2. D. S. Weld and J. de Kleer (eds.), *Readings in Qualitative Reasoning about Physical Systems,* Morgan Kaufmann, San Mateo, CA, 1990.

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

4. T. G. Dietterich, *Proceedings of the Workshop on Knowledge Compilation,* Oregon State University technical report, Corvallis, OR, Sept. 1986.

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

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

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

8. A. Preditis and J. Mostow, "Prolearn: Towards a PROLOG interpreter that learns",*Proc. AAAI-87: 6th National Conference on Artificial Intelligence,* Seattle, WA, Aug. 1987, pp. 494-498.

9. A. Newell, "The Knowledge Level", *Artificial Intelligence,* Vol. 18, 1982, pp. 87-127.

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

11. S. Minton, *Learning Search Control Knowledge,* Kluwer Academic Publishers, Hingham, MA, 1990.

12. R. Fikes, P. Hart, and N. Nilsson, "Learning and Executing Generalized Robot Plans", *Artificial Intelligence,* Vol. 3, No. 4, 1972, pp. 251-288.

13. J.E. Laird, A. Newell, and P.S. Rosenbloom, "Soar: An architecture for general intelligence", *Artificial Intelligence,* Vol. 33, No. 1, 1987, pp. 1-64.

14. K. Kahn, "Partial Evaluation as an Example of the Relationship between Programming Methodology and AI", *AI Magazine,* Vol. 5, No. 1, Spring 1984, pp. 53-57.

15. R.M. Keller, C. Baudin, Y. Iwasaki, P. Nayak, and K. Tanaka, "Model Compilation: An approach to automated model derivation", Artificial Intelligence Research Branch technical report # RIA-90-04-06-1, NASA Ames Research Center, April 1990.

16. Y. Iwasaki, "Qualitative Physics", *The Handbook of Artificial Intelligence, Vol. 4,* A. Barr, P.R. Cohen, and E.A. Feigenbaum (eds.), Addison-Wesley, Reading, MA, 1990.

17. R.M. Keller, "Concept Learning in Context", *Proc. 4th International Workshop on Machine Learning,* Irvine, CA, June 1987, pp. 91-102.

18. D. Subramanian and M.R. Genesereth, "The Relevance of Irrelevance", *IJCAI-87: Proc. 10th International Joint Conference on Artificial Intelligence,* Milan, Aug. 1987, pp. 416-422.

19. T. Ellman, "Approximate Theory Formation: An Explanation-Based Approach", *Proc. AAAI-88: 7th National Conference on Artificial Intelligence,* St. Paul, MN, Aug. 1988, pp. 570-574.

20. L. Steels and W. Van de Velde, "Learning in Second Generation Expert Systems", *Knowledge-based Problem Solving,* R. Kowalik (ed.), Prentice Hall, Englewood Cliffs, NJ, 1986, pp. 270-295.

21. M. Pazzani, "Refining the Knowledge Base of a Diagnostic Expert System: An application of Failure-Driven Learning", *Proc. AAAI-86: 5th National Conference on Artificial Intelligence,* Philadelphia, PA, Aug. 1986, pp. 1029-1035.

22. D.A. Pearce, "The Induction of Fault Diagnosis Systems from Qualitative Models", *Proc. AAAI-88: 7th National Conference on Artificial Intelligence,* St. Paul, MN, Aug. 1988, pp. 353-357.

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

24. A. Araya and S. Mittal, "Compiling Design Plans from Descriptions of Artifacts and Problem Solving Heuristics",*IJCAI-87: Proc. 10th International Joint Conference on Artificial Intelligence,* Milan, Aug. 1987, pp. 552-558.

25. G. Cerbone and T.G. Dietterich, "Inductive and Numerical Methods in Knowledge Compilation", Oregon State University technical report, Corvallis, OR, 1990.

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

27. D.C. Brown and W.N. Sloan, "Compilation of Design Knowledge for Routine Design Expert Systems: An initial view",*Proc. ASME International Computers in Engineering Conference,* New York, NY, Vol. 1, 1978, pp. 131-136.

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

29. R.M. Keller, C. Baudin, Y. Iwasaki, P. Nayak, and K. Tanaka, "Compiling Redesign Plans and Diagnosis Rules from a Structure/Behavior Device Model", Knowledge Systems Laboratory technical report # KSL 89-50, Stanford University, June 1989. To appear in *Knowledge Aided Design,* Marc Green (ed.), Academic Press, London, 1991.

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

31. W.R. Swartout, "XPLAIN: A System for Creating and Explaining Expert Consulting Programs", *Artificial Intelligence,* Vol. 21, No. 3, pp. 285-325, 1983.

32. R. Davis, "Form and Content in Model Based Reasoning",*Proc. 1989 Workshop on Model-Based Reasoning,* Detroit, MI, Aug. 1989, Boeing Computer Services technical report, pp. 11-27.

33. R.M. Keller, "In Defense of Compilation: A response to Davis' 'Form and Content in Model-based Reasoning' ", *Proc. 1990 Workshop on Model-Based Reasoning,* Boston, MA, Aug. 1990, Boeing Computer Services technical report, pp. 22-31.