Query-based Diagnostics

John M. Agosta and Thomas R. Gardos
Intel Corporation
2200 Mission College Boulevard
Santa Clara, CA 95054, USA

Marek J. Druzdzel
Faculty of Computer Science, Białystok Technical
University, Wiejska 45A, 15-351 Białystok, Poland
and Decision Systems Laboratory
School of Information Sciences
and Intelligent Systems Program
University of Pittsburgh
Pittsburgh, PA 15260, USA

Abstract

We describe an approach to modeling diagnostic problems that is based on a passive observation of a diagnostician's work-flow and recording their findings and final diagnosis, from which the model can be modified directly, or improved by learning from cases so acquired. While the probabilistic model of a system under diagnosis is necessarily simplified, based on three-layer Bayesian networks with canonical interactions among the network variables, we are able to reduce greatly the most important bottleneck—the knowledge engineering effort that goes into model building. Our initial experience with an implementation of this idea suggests that the sacrifices in diagnostic quality are not large, while gains are tremendous.

1 Introduction

It is common knowledge that in today's stage of development of normative methods, i.e., methods based on probability theory and decision theory, it is not the speed of the algorithms that is a bottleneck. Even though probabilistic inference has been shown to be NP-hard (Cooper, 1990) and there exist models that will be too challenging even for a supercomputer, practical systems are capable of solving models consisting of hundreds or even thousands of variables within seconds. The true bottleneck in this approach is the knowledge engineering effort that goes into constructing models. It is our experience, in the domain of machine diagnosis at Intel and elsewhere, that diagnosing a complex manufacturing device requires an initial investment of possibly person-years of knowledge engineering effort. Given that building probabilistic models is a task that requires considerable knowledge and experience, the initial costs are too prohibitive in most practical environments. In order to disseminate the otherwise attractive probabilistic approach, one needs to find ways of overcoming the knowledge engineering bottleneck.

The conventional procedure for implementing knowledge-based systems separates model-building from model usage. Different actors perform tasks of building and running the model at different times (Schreiber et al., 2000). Conventionally a *diagnostic session* consists of a series of observations, each, in turn, generating a query to a previously formulated model that returns recommendations on the next step in the diagnosis. (For ease of exposition, we will

abbreviate the phase "during the steps in a diagnostic session" as simply "during diagnosis.") As the diagnosis progresses, possible causes that initiated the session are clarified, ending in a recommendation of a repair or, in clinical cases, of a treatment and prognosis.

In this paper, we consider an approach that relaxes this constraint by interleaving model building and refinement, and execution. A similar situation occurs with models derived from data: Learning a model has been traditionally a separate activity, often quite computationally intensive, that precedes use of a model for inference. However, recent interest in *online-learning* augments this approach, so that learning and inference are inter-twined (Blum, 1998). *Query-based* diagnosis is analogous in that the model may change during use. However our approach is quite different, and we do not draw upon techniques developed for on-line learning. We explore here two approaches to mingling model refinement with the diagnostic session, one derived from study of work-flow, the other of learning from cases.

By *query-based diagnosis* we mean that the diagnostic model is created and modified in the course of the diagnosis. Is is fluid and not entirely determined before the diagnosis begins. The model applied during the diagnostic session is conditioned on the initial query (sometimes called the *primary-complaint*) and the queries made during the session. It is the structure of the model that changes, not only the state of information consisting of the observations used for inference. The implication of the term is also that the case data on which the model is based may be retrieved after the session begins, and learning from the case data may take place simultaneously with the diagnosis. This is made possible, in part, by the computational power of the current hardware.

In this paper, which is the first time this idea has been applied to simplify and improve elicitation, we first review a few pre-cursors in the literature, explain the foundations of our approach, discuss the system's architecture as it evolved from study of work-flow, give some initial insights on how the computational problems can be addressed, and finally present the first prototype, which can be accessed by the reader on-line.

2 Background

The possibility of learning a model for inference, conditional on specific circumstances arose as advances in algorithms and hardware (e.g. "Moore's law") made real-time learning practical. (Wellman et al., 1992) was an early proponent of building a query-specific probabilistic decision model from a knowledge base. This line of research has matured and proliferated numerous automated Bayes network construction methods, for instance, MEBN that constructs networks from "fragments" based on attributes of the problem. These methods are insightful on translating a database representation into a Bayes network, but do not address the question of elicitation during diagnosis, but raise a different set of elicitation questions.(Laskey, 2008)

An example of a full-fledged system that demonstrates query-driven model-building is the PRIORITIES system, and the COORDINATE system that grew out of it. They implement inference models to predict time intervals of individual's activities by retrieving relevant cases from a database of activities. As the authors describe it:

> Rather than attempting to build a massive static predictive model for all possible queries, we instead focus the analysis by constructing a set of cases from the event database that is consistent with the query at hand.(Horvitz et al., 2002)

Conceivably the range of models that the system is capable of generating makes preconstructing the models practically impossible.

Similarly, Visweswaran and Cooper have proposed supervised classifier learning, under the term "instance-based" classifiers, that generalizes "lazy" learning of classifiers. (Visweswaran and Cooper, 2004) Their approach averages over related instance models to avoid pitfalls of relying on too restrictive a query.

All these approaches condition the model on the query addressed to it; in contrast, we attempt in addition to learn from user's actions while the user is applying the model.

3 Origin of our Approach

As knowledge-engineering practitioners are well aware, elicitation of causal relationships and model structure places large cognitive demands on domain experts. Part of the challenge is the accessibility of knowledge; during an elicitation session, the domain expert is faced with a set of hypothetical circumstances, that often consist of questions about rare occurrences (since descriptions of common occurrences can often be derived from data). One would like to elicit this knowledge when it is current with the activities of the expert. Why not gather this knowledge at the time it is used? During diagnosis the expert is immersed in the problem and the causal relationships have a cognitive immediacy not available generally. Incidentally this may be true at other times, like at the time that the target system is designed and validated.

In our experience, we find that combining model building with receiving recommendations from the model is natural for our users. This became apparent during an application project-review with our client, who had designed an alternate approach for supporting diagnosis. Our application, in conventional fashion, ran inference cycles, based on the user's current observations, to compute posterior probabilities of the faults present in the model. Our client's application closely followed the steps outlined in the work flow for the task. It had options for users to create new observations and to suggest possible causes relevant to the problem during the diagnostic session. It lacked any capability for inference, however, and merely collected cases that could be reviewed in subsequent situations. The challenge was put to us to combine inference with the ability to modify the diagnostic model as evinced by the user's suggestions of unmodeled causal relationships. The manufacturing equipment domain to which this is applied is dynamic with method and process modifications occurring monthly, and without software support that can keep pace with the changes. And it is fair to say that our client has a sophisticated user in mind, whose understanding of the diagnostic problem at hand is on a par with the model they are running. We embraced their approach as one solution to the well-recognized "knowledge elicitation bottleneck."

4 Diagnostic Work-flow

Diagnostic work-flow is the choice of steps available to the user to pursue the diagnosis. The software design codifies the process both to guide the user, and to capture cases for solving similar problems should they arise again. In addition to the existing work-flow we add a mechanism for elicitation of causes in the midst of the problem-solving work-flow that they are familiar with.

The process consists of an interaction that forms a dialog where at each step the application suggests the next actions by ranking possible causes by their posteriors and tests by their diagnostic value, based on observations already entered into the model. This fault set is displayed as a ranked list of possible diagnoses along with an indication of the posterior probabilities of the faults, as computed by the underlying model. In addition a list of suggested observations is displayed from those observations that reside in the model. In suggesting observations, the Bayesian network engine ranks them by expected gain in entropy among the faults.

The user has four choices at each step:

1. Select an existing observation variable to instantiate;
2. Create new test observation;
3. Create a new hypothesized fault; or
4. End the diagnostic cycle by selecting a cause and making the associated repair.

Creating new observations or faults modify the causal structure of the model, requiring the user to indicate what observations a fault manifests, and correspondingly, what faults an observation implies. Causal relationships are the

primary semantics of the model and the user deserves some guidance on when they are necessary. For this we bring up a "pop-up" editor, asking the user to associate a new fault with observations and vice versa. A screen shot of this editor is shown in Figure 1.

A primary intent of building the application around work-flow is to hide the Bayes network from the user. In doing so we make several simplifying assumptions on model structure: The model is a bi-partite graph with occasional context nodes to condition faults; all nodes have binary state spaces; all observation nodes assume Independence of Causal Influences (ICI); and all nodes when initially created are assigned default probabilities so that the user does not need to enter numeric values. These assumptions may appear restrictive, however they should be considered in the context of the feeble guidance that current support tools offer.

By assuming Independence of Causal Influences for all observation nodes, the addition or removal of a cause translates directly into the addition or removal of one arc and modular change in the CPT of the child node. This is critically necessary for modification of the model to incorporate causal knowledge elicited during diagnosis, and makes model modification straightforward. The notion of Independence of Causal Influences (ICI) has been described many times in the literature, usually under a somewhat confusing term *causal independence*. Please see (Díez and Druzdzel, 2008) for a review of canonical models, including ICI models.

4.1 Causal Work-flow Example

Aside from the modeling and computational challenges to updating the model within a diagnostic session, the work-flow of such a session presents no surprises to the user. Consider this "use-case" example of diagnostic support for a simplified automobile diagnosis.

The user begins the session by entering the primary complaint, of *smoke visible in the exhaust.* The application responds by referring to the appropriate, if incomplete, model, and suggests a list of possible causes, such as *worn piston rings*, etc. To isolate the likely cause, the user proposes to do a *compression test*, a test that is not in the existing model. When the user enters the description of the test, she also designates other possible causes which are relevant to it, both from a list of those already in the model, and those she might create, in addition to *worn piston rings*, such as an *exhaust valve leak*. More significant is that she is steered to not designating causes to which the test is not relevant, making the test a good differentiator among them. Similarly the user could add a new possible cause of the primary complaint. She would then be prompted at that step in the session to designate the relevant observations. The session continues through diagnostic cycles by the user taking one of the four actions listed above.

The result of this interaction style is a model that makes tradeoffs that lean toward a more dynamic, up-to-date model at the cost of current model accuracy and completeness. The appropriate degree to which to shift the interaction style toward dynamicism depends strongly on the domain.

5 A Working Prototype

We have implemented the ideas presented in this paper in a prototype diagnostic system, called MARILYN, available to interested readers at the Decision Systems Laboratory's web site at the following location: `http://barcelona.exp.sis.pitt.edu/`. A sister implementation to MARILYN is in preparation at Intel, to be evaluated on machinery maintenance tasks. We will describe the academic prototype below, understanding that the two implementations share many features.

From the point of view of user work-flow, MARILYN appears as a smart data-entry module that collects observables, such as symptoms and test results, context information, such as the age of a device, prior problems with it, etc., and finally, the final diagnosis. MARILYN involves its user in a dialog that focuses on entering a diagnostic case and does not reveal the underlying probabilistic model. The target users are diagnosticians, possibly working in a machine

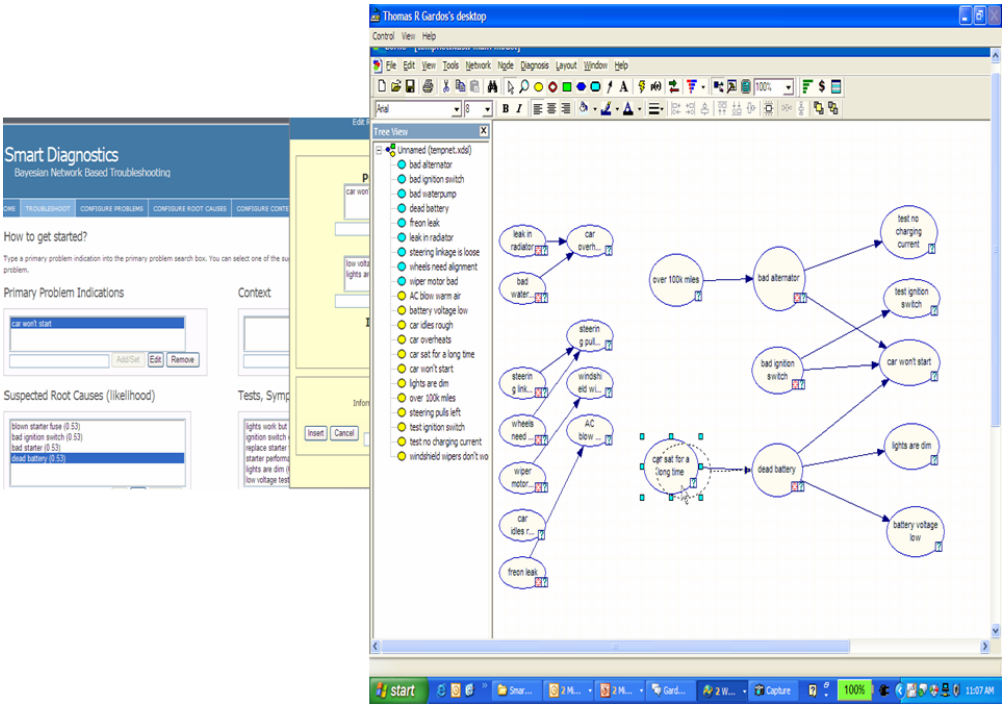

Figure 1: When the user enters a variable name that is not already in the current model, they are presented with a pop-up editor (here shown in yellow and largely hidden from view), for entering causal information. We see here a screen shot of the *GeNIe* editor that shows the model fragments generated by previous diagnoses super-imposed over the panes of the web application. The model network fragments are *not* shown to the user in the application.

repair shop. MARILYN takes a very gentle attitude towards the user—it merely suggests a set of possible diagnoses from among those diagnoses that have been entered in the past that are contained in the current model and implied by the observations entered.

5.1 The System Architecture

The intent of making MARILYN a web-based program is to make the system accessible from any computer through the Internet. MARILYN prototype consists of three elements: (1) user interface running on the client, (2) database of cases, and (3) a Bayesian reasoning engine based on SMILE, both running at the server. Information that is entered by the user is stored in a database (Figure 2 shows the database schema, which illustrates the components of the information collected from the user during a diagnostic session). The database is used by the user interface module in its auto-complete function. The database is also accessed directly by the

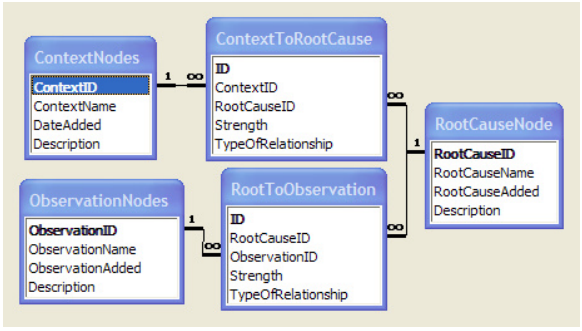

Figure 2: Entity-relationship diagram for the database storing diagnostic information in MARILYN

Bayesian reasoning engine, which constructs on the fly a three-layer Bayesian network that is used in deriving suggested diagnoses and suggested observations. As revealed in Figure 1, it is possible to export the underlying network to GeNIe for the purpose of the examination by a knowledge engineer.

5.2 The Quality of the System's Advice

In our experience, the quality of Marilyn's initial advice (during the first 10-20 diagnostic cases) is low, as the system is not aware of most of the possible diagnoses. As time progresses and the number of cases entered increases, the quality of the model increases and so does the quality of the system's advice. It is our expectation that in such environments as help desks, where tens of hundreds of cases are entered daily, Marilyn's approach will outperform novice diagnosticians fairly quickly.

5.3 Managing Variables

A downside of the system's flexibility toward the users being able to enter any text as the name of variable is the possible proliferation of similar variable names. While its user is typing a name, Marilyn offers suggestions from the database, based on simple techniques for string matching (along the lines of the "auto-complete" function of web browsers). We believe that for a system like Marilyn to truly scale, this flexibility is necessary; however this invites confusion due to model variables that are named slightly differently but stand for the same concept.

In the future we hope to augment the string matching auto-complete feature of Marilyn's user interface with more sophisticated statistical language processing techniques. The intent would be to accommodate common typographic errors using string edit distance measures as well as more challenging synonymous labels through semantic similarity techniques. The ultimate goal is to make this system scale to large models with as little expert user intervention as possible.

Also to support scaling, we believe that it is necessary to equip the system with what we call *expert mode* in which an expert knowledge engineer downloads the underlying Bayesian network, analyzes it, corrects it as necessary, and uploads it back to the system. The frequency of this knowledge base maintenance task depends on the speed of growth of the model. Similarly, the expert might review cases generated by diagnostic sessions that will be used in learning to refine the model. Over time the frequency of reviewing the expert model should converge to a low level.

An issue related to the expert mode and to the initial weak performance of the system is the possibility jump-starting the initial domain model.[1] The quality of this initial model is not critical and the expert can spend just a little bit of time entering the most important domain variables. We expect that the quality of this initial model will be improved fairly quickly as the system is fielded and diagnostic cases entered.

5.4 Managing Model Growth

One technical challenge that we are facing with a system of this type is a possible uncontrolled growth of the underlying model. Model size is normally not a problem, as a reasoning engine such as SMILE is able to handle huge networks and propagate evidence in them within a fraction of a second. What is more troublesome are certain undesirable trends in the network topology. If the number of parents of a single node, for example, grows above 15, this node may pose a considerable challenge to the algorithms. Even though Marilyn represents all conditional probability distributions as *DeMorgan* gates, a variety of ICI model (a detailed exposition of the DeMorgan gate, a natural and intuitive combination of Noisy-OR and Noisy-AND gates can be found as a separate paper in this volume (Maaskant and Druzdzel, 2008)), the size of the conditional probability table is a challenge for the algorithms. Our initial way of dealing with this problem is imposing a maximum on the number of parents of every single node. We control this by means of removing

[1] Such model can be created manually, or by converting an existing Bayesian network to a bipartite graph. GeNIe in fact contains a function to this effect (see the *Diagnosis* menu).

weaker connections in the model. In the next section we speculate about more rigorous methods for modifying models once fielded.

6 Learning from just a Few Cases

It is natural to consider how automated learning can be applied in query-based diagnosis by using a selection of cases to modify an existing model. In this section we state this problem, and make a few observations about how one might go about it; the solution to it is outside the scope of this paper. The logging function of the system will generate supervised cases of session outcomes that can be used to improve the accuracy of the model. Much like a case-based reasoning system, the process envisioned consists of retrieving cases from a database when a new diagnosis starts. The process resembles conventional case-based reasoning, but enforces the strong consistency conditions of a Bayes network model.

Learning from cases is an additional way to exploit user work-flow, however unlike modifications made from elicitation of causal links, there is no direct way to modify the network by inspection of a set of cases. The problem is to determine if the set is consistent, and if so, to learn a model consistent with the set. To do this we offer a precise definition of probabilistic cases and case consistency:

Consider a database of cases indexed by j, corresponding to a selected set of diagnostic sessions that have been resolved. A case is the relevant part of the diagnostic state at the end of a diagnosis that has been validated, perhaps by replace-and-test, or by an expert's opinion. An individual case contains both evidence consisting of the complete set of observations from the session and the posterior of the top (or top few) faults.

Definition 1. Case. Given evidence for case j, $\mathbf{e}^{(j)} = \{e^{(j)}\}_{i=1\ldots I_j}$ and the posterior of faults $f^{(1)} \ldots f^{(k)}$ obtained by inference from the current model M_j, a case j is list of ordered fault marginals for that evidence: $\mathsf{P}\left(f^{(1)} \,|\, \mathbf{e}^{(j)}, M_j\right) \geq \ldots \geq \mathsf{P}\left(f^{(k)} \,|\, \mathbf{e}^{(j)}, M_j\right)$.

Definition 2. Case Consistency. A model M^* is consistent with a case j, to level k, if the list of ordered fault marginals given the evidence $\mathbf{e}^{(j)}$ agrees with the case: $\mathsf{P}\left(f^{(1)} \,|\, \mathbf{e}^{(j)}, M^*\right) \geq \ldots \geq \mathsf{P}\left(f^{(k)} \,|\, \mathbf{e}^{(j)}, M^*\right)$.

Clearly a case j is consistent with the model M_j that generated it.

6.1 Stochastic versus Epistemic Uncertainty

Given a large number of such cases, one may be tempted to apply conventional supervised learning techniques to refine the existing model. Putting aside the problem of a limited number of cases, and that the case consists most likely almost entirely of missing values, we argue that this is inappropriate from the user's point of view. The user considers a case as specifying the exact effect that they expect to see when the case is run on the correct model. The probabilities in the case are not variations due to a random sample, rather they indicate *epistemic uncertainty*—the knowledge, or lack of it in the diagnostic outcome given the observation set.

When a database of observations contains stochastic uncertainty, a statistical learning approach is appropriate, since, roughly, the true model is an average of many imprecise instances. In contrast, a case containing epistemic uncertainty is better considered as a fragment of the correct network model. The case is "supervised" in the sense of revealing a part of the joint distribution of the overall network.

The learning problem can now be stated as one of modifying an existing model to meet the set of constraints expressed by cases selected as relevant to the current diagnosis. We call this learning problem *case consistency*. Methods to solve this have been studied as applications of Jeffreys' Rule for belief revision (Chan and Darwiche, 2005). Such belief revision changes the joint distribution of the model. Jeffrey's rule is often presented as an alternate means of applying evidence, by comparison with applying likelihoods as evidence (sometimes called "virtual" or "soft" evidence, but the usage in the literature isn't consistent.) In contrast it appears preferable to distinguish the application of ev-

idence from case consistency, if only to make a clear semantic distinction between inference conditioned on observations, and an operation more like "splicing" a fragment of new knowledge into the existing model.

7 Conclusions

We described an approach to solving diagnostic problems that is based on the concept of *query-based diagnostics* and amounts to recovering a normative system based on Bayesian networks from a conventional diagnostic work-flow. Without forcing the user to use a limited vocabulary and variables, and by capturing the user's causal understanding, the system allows for continuous refinement of the model, while offering suggestions to avoid possible repetition of terms. While this carries with it a danger of uncontrolled model growth, we believe that with incorporation of workflow-generated cases for belief revision, and possibly off-line intervention of a knowledge engineer, the accuracy of the system can be maintained. Despite that the probabilistic model of a system under diagnosis is necessarily simplified, based on three-layer Bayesian networks with canonical interactions among the network variables, we are still able to reduce greatly the most important bottleneck—the knowledge engineering effort that goes into model building. We have two implementations of this idea underway, with preliminary results that are promising.

A system based on the principles outlined in this paper has to be thoroughly evaluated in a practical setting. One of our next steps is to employ our prototype in a diagnostic setting and carefully monitoring its use, including user experiences, model creation and growth, and the development of the system's diagnostic accuracy.

Acknowledgments

This work has been supported by Intel Research. Implementation of Marilyn is based on SMILE, a Bayesian inference engine developed at the Decision Systems Laboratory and available at `http://genie.sis.pitt.edu/`. We would like to thank Erik Pols for a considerable effort that went into developing initial ideas into a working MARILYN prototype and Parot Ratnapinda and his classmates at the University of Pittsburgh for refining Erik's initial implementation. We thank Cort Keller, Christine Matlock, Daniel Peters and Bryan Pollard for their work on design and implementation of Intel's application.

References

Avrim Blum. 1998. On-line algorithms in machine learning. In *Online Algorithms*, pages 306–325. Springer.

Hei Chan and Adnan Darwiche. 2005. On the revision of probabilistic beliefs using uncertain evidence. *Artif. Intell.*, 163(1):67–90.

Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using bayesian belief networks. *Artif. Intell.*, 42(2-3):393–405.

F. Javier Díez and Marek J. Druzdzel. 2008. Canonical probabilistic models for knowledge engineering. Unpublished manuscript, available at http://www.ia.uned.es/∼fjdiez/papers/canonical.html.

Eric Horvitz, Paul Koch, Carl M. Kadie, and Andy Jacobs. 2002. Coordinate probabilistic forecasting of presence and availability. In *18th Conference on Uncertainty in Artificial Intelligence*, pages 224–233. Morgan Kaufmann Publishers, July.

Kathryn B. Laskey. 2008. Mebn: A language for first-order bayesian knowledge bases. *Artif. Intell.*, 172(2-3):140–178.

Paul P. Maaskant and Marek J. Druzdzel. 2008. An Independence of Causal Interactions model for opposing influences. In *Proceedings of the Fourth European Workshop on Probabilistic Graphical Models (PGM–08)*, Hirtshals, Denmark.

Guus Schreiber, Hans Schreiber, Anjo Akkermans, Robert de Anjewierden, Nigel Shadbolt Hoog, Walter Van de Velde, and Bob Wielinga. 2000. *Knowledge Engineering and Management: The CommonKADS Methodology*. MIT Press.

S. Visweswaran and G.F. Cooper. 2004. Instance-specific bayesian model averaging for classification. In *Proceedings of the Neural Information Processing Systems Conference (NIPS-2004)*.

M. Wellman, J. Breese, and R. Goldman. 1992. From knowledge bases to decision models. *Knowledge Engineering Review*, 7(1):35–53.