Inferential Complexity Control for Model-Based Abduction

Gregory Provan
Rockwell Scientific Company
1049 Camino Dos Rios, Thousand Oaks, CA 91360, USA
gprovan@rwsc.com

Abstract

We describe a technique for speeding up inference for model-based abduction tasks that trades off inference time and/or space for the fraction of queries correctly answered. We compile a knowledge base (for which inference may be intractable) into a set of rules that cover the *most likely* queries using simple criteria that do not entail extensive knowledge engineering effort, such as subset-minimal or most probable query-responses. We demonstrate this approach on the abduction task of model-based diagnosis, and show that this approach can predictably produce order-of-magnitude reductions in time and memory requirements for abductive tasks in which the queries have skewed distributions; for example, in diagnosis the faults are skewed towards being highly unlikely.

Introduction

Abduction can be defined as the task of generating a plausible explanation for a set of observations (Paul 1993). Abductive inference has been applied to several domains, such as plan recognition (Appelt & Pollack 1009), planning (Poole & Kanazawa 1994), natural language understanding (Stickel 1990), user modeling (Poole 1988), and vision (Kumar & Desai 1996), and has shown the most utility in the field of diagnosis (Console, Portinale, & Dupre 1996).

This article focuses on the task of generating *all* plausible explanations for a set of observations, and on approximating this task. This task is important for areas such as diagnosis, where we are interested in knowing all possible causes of some anomalous behavior, in order to be able to repair all faults. Whereas computing a plausible explanation has received significant attention in the literature, computing all plausible explanations has received less attention, e.g., (Eiter & Makino 2003).

We address the task of *approximate compilation*: we trade off compiled model size for reduced coverage of abductive queries, in order to overcome the intractability of abduction. Abductive tasks are intractable (Eiter & Gottlob 1995; Bylander *et al.* 1991), and remain so even when we precompile the system model to speed up inference (Liberatore & Schaerf 2000). Compilation (or preprocessing) for abductive tasks such as diagnosis is an attractive approach, since it offers the possibility of reducing the run-time abductive inference by making use of prior knowledge during the compilation phase. This is possible since part of the data of the problem (namely, the possible assumptions, the possible observations, and the theory relating assumptions and observations) are often known *a priori*. This knowledge has been exploited in (Console, Portinale, & Dupre 1996; Darwiche 1999; Darwiche & Marquis 2002). Compilation approaches that compile the complete solution space, e.g., (Darwiche 1999), make a time/space tradeoff, resulting in compiled models that can be excessively large for complex systems.

The main topic addressed in this article is the nature of the space induced when we trade off inference complexity for query-answering completeness of compiled models. For a propositional model with probabilistic preference criteria, we show how to reformulate a large problem into an approximation-model, specified by a set of rules. Our approach allow a user to smoothly trade off the "quality" of the output with the time/space necessary for inference.[1] This provides a technique for controlling inferential complexity with known tradeoffs.

In this article we adopt a compilation technique called rule generation (Darwiche 1999), and show how expected-query rule generation can significantly reduce inference time/memory by trading off query-answering completeness. We make the following contributions:

1. We define a sound but incomplete compilation procedure for abductive tasks. This approach loses the ability to answer all queries, but provides an upper bound[2] to inference tasks.
2. We propose a metric for "completeness" of inference, where completeness is a measure of the total space of all possible queries that can be answered. Our metric adopts a probabilistic semantics, and assigns a weight (probability) to each assumable.
3. We analyse this approach to approximate compilation along two dimensions:

[1]We can think of solution quality as the likelihood of answering a query correctly.

[2]See (Cadoli & Donini 1997, Section 4)

(a) We set all assumption weights to be equal, and we examine the inference complexity of the resulting rule set. We examine the impact on complexity of the numbers of assumptions and observables, as well as the impact of problem decomposition.

(b) We allow the weights to vary and examine the impact of weights on inference complexity.

For the class of propositional models we have addressed, our results are encouraging. We employ two parameters to identify the tradeoffs of compiled approximate inference: (1) the coverage value χ and (2) the memory reduction factor λ of the compiled model, which provides a measure of the relative complexity of approximate compiled inference versus using the full model. We show that this approach can predictably produce order-of-magnitude reductions in time and memory requirements for abductive tasks in which the queries have skewed distributions; for example, in diagnosis the faults are skewed towards being highly unlikely.

For a diagnostics application, this approach relies on prior knowledge of the observations available (typically sensor and actuator data) and the components for which faults are to be computed (the assumables). When such prior knowledge is complete, this approach can reduce the memory required by an embedded diagnostics model by orders of magnitude; moreover, as the prior knowledge diminishes, this approach has memory requirements that gracefully degrade to the memory requirement of the original abductive (diagnostics) model.

The remainder of the article is organized as follows. We first outline the motivation for the approach we have developed, and then introduce our rule generation approach. Next we provide a theoretical analysis of our approach, and describe the results of experiments used to apply the approach to several real-world models. We then compare the results presented here with prior work in the literature, and conclude by summarizing the main results.

Motivation and Notation

Motivation

We describe model-based abduction as a method for finding explanations for a set of observations $\mathcal{O}$ given a model Φ defined over a set V of variables (Paul 1993). Potential abductive explanations are represented in terms of distinguished variables from $\mathcal{A} \subseteq V$, , called *assumables* or *targets*, and are typically subjected to preference criteria, such as subset minimality, minimal cardinality, minimal weight, etc.

The assumables represent the variables of interest, which are then used to formulate query-responses. For example, in a diagnostics scenario the assumables are defined to be the failure modes of the system components, and responses to all diagnostics queries consist of $\mathcal{A}$-sentences, i.e., combinations of component faults like (motor-faulty $\wedge$ wiring faulty) $\vee$ (power-supply-faulty). Assumables are a key focus of abductive inference, and correspond to the variables over which special-case inference takes place, in contrast to generic satisfiability, where there are no distinguished variables. For example, abductive inference often uses the general notion of parsimony to select, among all responses to a query, the responses that are simplest (e.g., have fewest assumables in a sentence). Many real-world tasks can be defined to have a special subset of variables, in contrast to satisfiability or Bayesian networks representations, which do not select a subset of distinguished variables. We explicitly make use of these variables to obtain computational savings.

The particular instance of abduction on which we focus is diagnosis, which is a domain with clear practical relevance. Note, however, that although we focus on a particular instance of abductive inference, the principles entailed in all algorithms that we develop should be applicable to other abductive domains.

The intuition for expected-query diagnostics is as follows: whereas the full diagnostic model over set A of failure-mode variables can diagnose all $2^{|A|}$-1 possible multiple-fault combinations, in reality one needs only to diagnose multiple-fault combinations with some fixed number of simultaneous faults, since a large number of simultaneous faults is extremely unlikely.[3] If we assign a probability distribution over the discrete values of each failure-mode variable, and define the size of a multiple-fault diagnosis D as the number of assumables in the diagnosis, then it is simple to show that the smaller diagnoses are more likely, i.e., $Pr\{D_i\} < Pr\{D_j\}$ whenever $|D_i| > |D_j|$ and every failure has a non-zero chance of occurring.[4]

Underlying Representation

We represent our KB using $\Phi = (\mathcal{A}^w, \mathcal{O}, \Delta)$, where $\mathcal{A}^w$ is the set of weighted assumables or *target variables* (denoting the primitive entities that we are interested in), $\mathcal{O}$ is the set of observables (variables that can be observed), and Δ is the set of domain axioms. We define these more precisely below.

Definition 1 (System Variables) *We assume that every variable $\{V_1, ..., V_l\}$ has an associated discrete finite domain, denoted by $\{x_1, ..., x_l\}$, respectively, which describes the possible states of that variable.*

For a system defined using a set V of variables, we identify two disjoint subsets: $\mathcal{A} = \{\gamma_1, ..., \gamma_n\}$, the set of n target variables, *and ϑ, the set of non-target variables, such that $V = \mathcal{A} \cup \vartheta$, with $\mathcal{A} \cap \vartheta = \emptyset$.*

We assign a set of weights to assumables so that we can rank the abductive solutions.

Definition 2 (Target Variables) *Each weighted* target variable, *$\mathcal{A}_i^w$, consists of a target variable $\mathcal{A}_i$ together with a real-valued weight $p_i \in [0,1]$ associated with $\mathcal{A}_i$, i.e., $\mathcal{A}_i^w = (\mathcal{A}_i, p_i)$.*

Definition 3 (Observables) *The set of* observables *is a subset of ϑ, i.e., $\mathcal{O} \subseteq \vartheta$.*

Definition 4 (Observation) *An observation is an instantiation of a conjunction of the observable literals $\mathcal{O}$.*

[3]TopN (Henrion 1991) uses a similar approach to develop an algorithm that searches over a complete model for the most likely diagnoses; we compile a model that describes only those most likely diagnosis combinations.

[4]Even if we don't have probabilities for assumables, then we can use parsimony to argue that the most likely explanations are those that are simplest.

Definition 5 (Domain Axioms) *The set Δ of domain axioms is a set of propositional clauses defined over a set V of variables.*

In this article, for simplicity of exposition we restrict each target variable to have binary-valued domain; there is a straightforward generalization to n-ary domains, for $n \geq 3$.[5]

Another key aspect of abductive inference is the preference ordering $\preceq$ over the target variables. The ordering captures the notion of likelihood of an explanation: given two explanations $\mathcal{R}$ and $\mathcal{R}'$, $\mathcal{R} \prec \mathcal{R}'$ if $\mathcal{R}$ is more likely to be the true cause of the observations than $\mathcal{R}'$.

The literature contains a broad range of preference relations; see, for example, those described in (Paul 1993). In this article we assume that we have either (1) a discrete probability distribution $\vec{P}$ over the target variables, i.e., for target γ_i, $p_i = Pr\{\gamma_i = true\}$, or (2) order-of-magnitude probabilities (Goldszmidt & Pearl 1991).

The solution to an abductive problem $\Phi = (\mathcal{A}, \mathcal{O}, \Delta)$ is given by

$$\Gamma(\mathcal{A}, \mathcal{O}, \Delta) = \{\mathcal{A}' \subseteq \mathcal{A} | \mathcal{A} \cup \Delta \text{ is consistent, and } \mathcal{A}' \cup \Delta \models \mathcal{O}\}.$$

Given an ordering $\preceq$ on the weights assigned to assumables, the set of minimal solutions is defined as:

$$\Gamma_{\preceq}(\mathcal{A}^w, \mathcal{O}, \Delta) = min(\Gamma(\mathcal{A}^w, \mathcal{O}, \Delta), \preceq).$$

Model-Based Abductive Rule Generation

Rule Generation Method

This section briefly summarizes the technique for generating rules from a declarative database encoded as a causal network (Darwiche 1999). Note that this approach applies to different database representations and different compilation techniques, and is not limited to the particular approach we describe.

We define a rule as follows (adapted from the robust rule of (Darwiche 1999)):

Definition 6 *Given a KB $\Phi = (\mathcal{A}, \mathcal{O}, \Delta)$, let α be an instantiation of a subset $\mathcal{O}'$ of $\mathcal{O}$, and β be an instantiation of a subset $\mathcal{A}'$ of $\mathcal{A}$. A rule $\mathcal{R}_i$ for $(O', \mathcal{A}')$ is a sentence of the form $\alpha \Rightarrow \bigvee_i \beta_i$, where the β_i are the minimum-cardinality instantiations of $\mathcal{A}$ that are consistent with $\Delta \wedge \alpha$, such that $\alpha \wedge \alpha' \Rightarrow \bigvee_i \beta_i$ is a rule for $(O, \mathcal{A}')$ for every instantiation $\alpha' \subseteq \mathcal{O} - \mathcal{O}'$.*

(Darwiche 1999) shows that we can always compile a system Φ into a set of such rules, such that the rules are guaranteed to provide complete target coverage for the system.

Given that we have compiled a set $\mathcal{R}$ of rules, rule inference is linear in the size of the rule-base. The main issue to consider then is the size of the rule-base, which is what we focus on in the remainder of the article.

[5]Under this assumption we denote $\gamma_i = true$ using γ_i, and $\gamma_i = false$ as $\bar{\gamma}_i$.

Expected-Query Rule-Generation

The task that we are trying to solve is to compute query coverage, a set of minimal (weighted or most likely) query responses that covers some fraction of all possible query responses from a given theory, where we determine minimality using the assumption weight associated with the query response. By computing the relationship over query coverage versus size and complexity of coverage, we show that we can smoothly trade off query coverage for inference complexity, and thereby "control" inferential complexity for abductive inference.

We now define the metrics we propose to use for our approximation task. We use a notion of "most likely" query, where the likelihood of a query is the probability of the occurrence of that query. We represent a query as a conjunction of target variables, $Q = \wedge_i \bar{\gamma}_i$, and we can derive the probability of query Q as $\prod_i p_i$.[6] More generally, we could extend this notion to the utility of query, or the expected utility of a query.

This semantics of query likelihood adheres to Occam's razor, in that it favors the simplest queries as the most likely. In addition, it holds in several domains. For example, in diagnostic inference, if we assign prior probabilities to fault-modes, then this corresponds to our proposed semantics.

We adopt a sound but incomplete compilation approach, and our compiled KB Φ' provides an upper bound on any query response. As defined by Cadoli and Donini (1997), an approximation Φ' of a knowledge based KB is sound when for every query Q, if $\Phi' \models Q$ then $KB \models Q$, in which case Φ' is called an upper bound for KB.

Evaluation Metrics The objective of our approximate compilation is to provide coverage for a fixed percentage of possible queries. We adopt two evaluation metrics to measure how well our compilation approach is doing.

Hence if the space of all possible queries is $2^{\mathcal{A}}$, and the full KB Φ can respond to all queries, then we want to compile a reduced KB Φ' that can respond to some fraction of those queries that Φ can respond to.

We can formalize the memory savings of expected-query rule generation as follows. Let $|\Delta|$ be a measure for the size of the original KB, and $|\Delta_q|$ be a measure for the size of the compiled KB that contains rules with only up to q targets per rule.

Definition 7 (Memory Reduction Factor) *The memory reduction of expected-query rule generation, with respect to using the full KB, is given by*

$$\lambda = \frac{|\Delta_q|}{|\Delta|}. \quad (1)$$

Example: Space Requirements of Expected-Query Rule-Generation Generating rules from a KB (e.g., causal network model) can offer significant space advantages over using the standard inference approach, provided that the observables and assumables (target variables) are known *a priori*. If we assume a system with n target variables, then we

[6]If we do not have a probability measure, we can use an order-or-magnitude probability to rank query weights, as in (Darwiche 1998).

must generate $O(2^n)$ rules to specify all target cases. To define the target cases when there are q targets, we need O($\binom{n}{q}$) rules. It is simple to prove that there are far fewer rules that include only up to q targets per rule than rules that include up to all n targets, for $q \ll n$, i.e.,

$$\sum_{i=1}^{q} \binom{n}{i} \ll 2^n \qquad \text{for } q \ll n.$$

For example, if $n = 10$ and we enumerate single- and double-target rules, we will generate 10 rules, as opposed to the space of 1023 total rules; i.e., we generate only about 1.3% of all rules. As n grows, the percentage of all rules that we must generate diminishes.

Expected-Query Analysis

This section analyses expected-query compilation along two dimensions: (1) holding weights equal and examining the impact of the model structural parameters, such as number of observables and target variables, as well as model decompositions, and (2) the impact of weights on target coverage.

Target Spaces

The following discussion characterizes the space of solutions in which we are interested, which is those abductive solutions in which at least one assumable is *true*. In diagnosis, such a space corresponds to those solutions where at least one component if faulty, which is exactly the class of solution in which we are interested. We will then use this characterization to define the most-likely solutions of interest.

To begin our analysis we first define our notion of target space. A target space corresponds to assumable instantiations where at least one assumable is *true*. We make use of the following definition:

Definition 8 (Assumable Instantiation) *An instantiation of all n assumables $\mathcal{A}_i \in \mathcal{A}$.*

In terms of abductive solutions, an assumable instantiation (or state vector) is an extension of $\Gamma_{\preceq}(\mathcal{A}^w, \mathcal{O}, \Delta)$ in which all assumables not set truth values in $\Gamma_{\preceq}(\mathcal{A}^w, \mathcal{O}, \Delta)$ are instantiated to their minimum-weight values.

We define Ω to be the set of all assumable instantiations. We define an assumable instantiation in which all assumables are *false*,[7] as

$$\mathcal{S}_{\emptyset} \equiv \bigwedge_{i=1}^{n} \bar{\gamma}_i.$$

Definition 9 (Target Instantiation) *The set of target instantiations $\mathcal{S}_T$ of a system is the set of all assumable instantiations in which at least one assumable is* true*:*

$$\begin{aligned} \mathcal{S}_T &= \Omega - \mathcal{S}_{\emptyset} \\ &= \Omega - \bigwedge_{i=1}^{n} \bar{\gamma}_i. \end{aligned}$$

[7]In terms of a diagnosis application, this corresponds to the state where all components are functioning normally.

We define an assumable instantiation that assigns *true* to m simultaneous assumables as follows. Assume that the assumables with *true* values are indexed by $i \in \mathcal{I}$. We can thus define:

$$\mathcal{S}_m = \bigwedge_{i \in \mathcal{I}} \gamma_i \bigwedge_{j=1}^{n} \{\bar{\gamma}_j | j \notin \mathcal{I}\}.$$

Model Structural Parameters

This section examines the impact of the number of observables, assumables, and problem structure on approximate compilation.

The space required by a set of compiled rules is exponential in both $\mathcal{A}$ and $\mathcal{O}$ in the worst case, hence these are the two most critical parameters in evaluating the resulting compiled model. It is possible to decompose a model into a representation that is space-optimal with respect to complete query response: this is done by decomposing the set of assumables $\mathcal{A}$ into independent subsets—see (Darwiche 1999) for details. However, this optimal compilation is still exponential in $\mathcal{O}$ and the maximal subset of $\mathcal{A}$. One key advantage of decomposing a model is that we can compute rules for each individual component, and obtain the rules for the full model simply by merging the component rule-sets. This is formally stated as follows.

Theorem 1 *If we decompose a model into m independent sub-models $\Phi_i, i = 1, ..., m$, each with corresponding rule-base $\mathcal{R}_i, i = 1, ..., m$, then the model rule-set is given by*

$$\mathcal{R} = \bigcup_{i=1}^{m} \mathcal{R}_i.$$

Proof: Assume that for every $\Phi_i, \Phi_j, i \neq j$ and $i, j \in \{1, ..., m\}$, the sub-models share no variables, i.e., $\Phi_i \cap \Phi_j = \emptyset$. If we generate rules for each sub-model, we must have $\mathcal{R}_i \cap \mathcal{R}_j = \emptyset$, when $i \neq j$ and $i, j \in \{1, ..., m\}$ since $\mathcal{R}_i, \mathcal{R}_j$ share no variables by virtue of being in different model partitions. If $\mathcal{R}_i$ is complete for Φ_i, for $i = 1, ..., m$, then since $\Phi = \bigcup_{i=1}^{m} \Phi_i$, we must have $\mathcal{R} = \bigcup_{i=1}^{m} \mathcal{R}_i$. $\diamond$

We experimentally analyse the effect of structural parameters (observables and assumables) and model decomposition later in the article.

Weight-Based Rule Approximations

This section describes how we can use the weights associated with target variables in order to approximate the model compilation. In this approach, we compile a set of rules that is incomplete with respect to the number of solutions that it covers. The set of rules will omit target scenarios containing the least likely solutions. We will use a probabilistic analysis to compute the percentage of the total set of solutions that would be omitted. For this analysis, we assume that all assumables are mutually independent.

We define the *assumable instantiation likelihood* $\Psi_{\mathcal{S}}$ as:

Definition 10 (Assumable Instantiation Likelihood) *the probability measure associated with assumable instantia-*

tion $\mathcal{S}$ is given by

$$\Psi_{\mathcal{S}} \;=\; Pr\{\mathcal{S}\} = \prod_{i=1}^{n} Pr\{\mathcal{A}_i = \xi_i\},$$

where assumable $\mathcal{A}_i$ is assigned the value ξ_i in $\mathcal{S}$.

Definition 11 (System Target Likelihood) *The* system target likelihood Ψ *is the probability that at least one assumable in an assumable instantiation is* true.

$$\Psi \;=\; Pr\{\mathcal{S}_T\} \;=\; 1 - Pr\{\mathcal{S}_\emptyset\} \;=\; 1 - \prod_{i=1}^{n} \bar{\gamma}.$$

The probability measure Ψ_q assigned to a set of rules each containing q simultaneous targets, assuming that the assumables with *true* values are indexed by $i \in \mathcal{I}$, is

$$\Psi_q \;=\; \prod_{i \in \mathcal{I}} Pr\{\gamma_i\} \prod_{j=1}^{n} \{Pr\{\bar{\gamma}_j\} | j \notin \mathcal{I}\}$$

In our analysis we generate rule sets with rules each containing a fixed number of assumables with *true* values. The probability measure Ψ_Q assigned to a set of rules containing *up to* q simultaneous targets is defined below.

We define the index set corresponding to the case where q assumables have a *true* value as $\mathcal{I}_q = \{1, 2, ..., q\}$.

$$\Psi_Q \;=\; = \sum_{k=1}^{q} \prod_{i \in \mathcal{I}_k} Pr\{\gamma_i\} \prod_{j=1}^{n} \{Pr\{\bar{\gamma}_j\} | j \notin \mathcal{I}_k\}$$

We use these notions to define a key parameter for our experimental analysis, the target coverage.

Definition 12 (Target Coverage Ratio) *We define the* target coverage χ *of a set* Λ *of rules, with target space* $\mathcal{S}_\Lambda$*, as the fraction of the system target likelihood provided by* Λ*:*

$$\chi = \frac{\Psi_\Lambda}{\Psi}. \tag{2}$$

Example We now outline an example in which we assume that the prior probability of all target variables is equal. Our rule generation heuristic translates to selecting the rules with the fewest simultaneous targets.

The probability measure assigned to the total target space Ψ is

$$\Psi \;=\; 1 - (1-p)^n.$$

The probability measure Ψ_q assigned to a set of rules containing q simultaneous targets is

$$\Psi_q \;=\; \binom{n}{q} p^q (1-p)^{n-q}.$$

If we generate rules containing up to q simultaneous targets, the cumulative probability measure is

$$\Psi_Q \;=\; \sum_{i=1}^{q} \binom{n}{i} p^i (1-p)^{n-i}.$$

The coverage of the cumulative q-target set is

$$\chi(Q) \;=\; \frac{\sum_{i=1}^{q} \binom{n}{i} p^i (1-p)^{n-i}}{1-(1-p)^n}.$$

From this term we can calculate bounds on the target coverage value χ in terms of n and p. For example, for the set of single-target rules, we can have

$$\chi \;=\; \frac{np(1-p)^{n-1}}{1-(1-p)^n}.$$

For a system with 3 target variables with priors p_1, p_2, p_3 respectively, the system target likelihood is given by

$$\Psi = 1 - (1-p_1)(1-p_2)(1-p_3).$$

The target space likelihood for the single- and double-target scenario is given by

$$\Psi_{\mathcal{S}_{12}} = \Psi_{\mathcal{S}_1} + (p_1p_2(1-p_3) + p_1(1-p_2)p_3 + (1-p_1)p_2p_3).$$

The target coverage of the set of single-target rules is given by:

$$\begin{aligned}\chi \;=\; & \frac{1}{1-(1-p_1)(1-p_2)(1-p_3)}[p_1(1-p_2)(1-p_3) + \\ & p_2(1-p_1)(1-p_3) + p_3(1-p_1)(1-p_2)].\end{aligned}$$

We will use this framework in the following section to show experimentally the influence of these probability values on rule generation and on target coverage.

Experimental Analysis

We have performed a set of empirical studies of target coverage. This section describes these studies, focusing first on the structural parameters, and then on the weights.

We start off by describing the approach that we use for our experiments. Our objective is to compare the diagnostic performance of the full model Φ (or corresponding set of full rules) with the approximate rules $\mathcal{R}$. We call the diagnosis produced by the full model D_Φ, and the diagnosis produced by the rules $D_\mathcal{R}$. An approximate rule covers a case if the rule contains all the diagnoses that would be generated by the full model. The *relative coverage* can be understood as a ratio between covered cases and total cases, where each case is weighted by its likelihood.

We randomly generate a set of test cases for each model, where each test case is an observation, and compare the diagnostic performance of Φ and $\mathcal{R}$ on each test case. If the test case is physically unrealizable and Φ produces an "invalid observation" output, then the test case is thrown out. Otherwise, for test case i Φ will compute a diagnosis, and we score the corresponding rule-base $\mathcal{R}_i$ using the target coverage ratio (equation 2) as follows:

$$s(\mathcal{R}_i) = \begin{cases} 0 & \text{if } D_\mathcal{R} = \emptyset \\ 1 & \text{if } D_\mathcal{R} = D_\Phi \\ \chi & \text{if } D_\mathcal{R} \subset D_\Phi \end{cases}$$

The overall weighted score is based on the product of (a) the relative performance of rule-set i in covering the faults, $s(R_i)$, and (b) the likelihood of the faults Ψ_i that occur within test case i: $s(R_i) \cdot \Psi_i$.

The coverage ratio for the set of all t test cases, each of which has an associated probability of occurrence of $\Psi_i, i = 1, ..., t$, is then given by:

$$\chi = \frac{\sum_{i=1}^{t} s(\mathcal{R}_i) \cdot \Psi_i}{\sum_{i=1}^{t} \Psi_i}.$$

In our results we assume a relative equivalency between memory and inference-time, since evaluating the rules takes time linear in the number of rules. The relative memory of a set $\mathcal{R}$ of rules is the ratio of the memory of the approximate rules (or model) and the complete rules (or model):

$$\text{Relative memory} = \frac{|\Phi_{\mathcal{R}}|}{|\Phi|}.$$

Structural Parameters

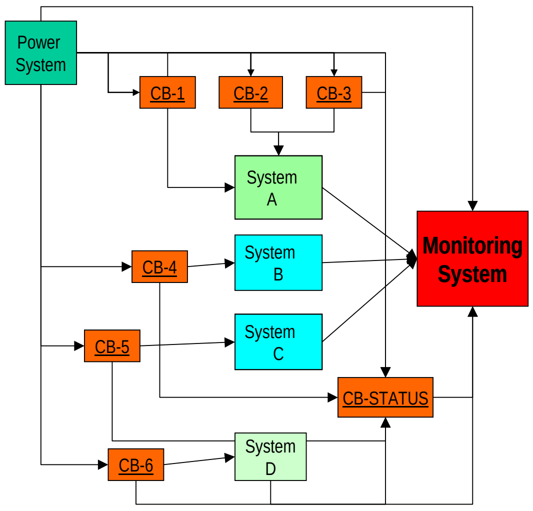

Figure 1: Example of a system that we analyze. This is an abstract view of aircraft subsystems, noted SystemA through SystemD, provided power through circuit breakers CB-1 through CB-6.

We have performed experiments on a large collection of real-world diagnostics models, including models of hydraulic, electrical and mechanical systems. Table 1 shows the basic parameters of the models we used for experimentation. The model classes are as follows: (a) the WoW-v1 through WoW-v6 models are for an autopilot subsystem in which velocity, altitude and other parameters determine autopilot settings; (b) the AC-v1 through AC-v4 models are for integrating multiple aircraft subsystems; (c) the MCP models are for a control system. Each model class has multiple models, denoted by version numbers (e.g., v2), which denote the increased size and complexity over the basic model (v1). In addition, we have experimented on randomly-generated models with varying numbers of variables, assumptions and observables.

Table 1 shows data for a variety of models. One of the key factors to note is the memory required for the compiled model, displayed in the last column. In particular, the models cover memory values ranging from small (12.1KB) to large (52.4MB). As noted earlier, the memory of the compiled model is our metric for evaluation complexity, since evaluating a model is linear in the size of the compiled data. As a consequence, it is important to note that models with compiled data in excess of 20-30MB are computationally expensive to evaluate.[8] Figure 1 shows the system corresponding to the AC-v2 model. This figure displays an abstract view of a collection of aircraft sub-systems, denoted system-A, etc., which receive power from a Power subsystem through circuit breakers (CB-1, CB-2, etc.), and then send their status information to a Monitoring System.

Name	V	$\mathcal{A}$	$\mathcal{O}$	Memory (KB)
WoW-v1	17	5	4	12.1
WoW-v2	19	6	5	22.4
WoW-v3	21	7	6	46.3
WoW-v4	25	9	6	49.0
WoW-v5	23	8	7	87.7
WoW-v6	27	10	7	89.4
AC-v1	12	5	8	1439
AC-v2	41	9	13	37,626
AC-v3	64	17	12	52,376
AC-v4	70	19	13	52,447
MCP	40	20	5	20.1
MCP-extended-v1	66	32	5	22.5
MCP-extended-v2	66	32	8	359.9

Table 1: Model Statistics for Real-world Models. We report data for the total number V of variables, number $\mathcal{A}$ of assumables, number $\mathcal{O}$ of observables, and memory for the full compiled model.

Model Size Parameters We have performed experiments to study the dependence of compiled-model performance on the parameters $\mathcal{A}$ and $\mathcal{O}$. To accomplish this, we have studied a broad range of models. Figure 2(a) shows the growth of the compiled memory required for instances of several model classes. For each model class, we augment the size of the basic model by adding additional model structure. For example, for the aircraft model shown in Figure 1, we add extra subsystems, with associated circuit-breakers. Figure 2(a) shows that the growth, in terms of model variables, is basically linear, except for the class of circuit models named Circuit2.

Figure 2(b) shows the growth of the compiled memory required for instances of several model classes, measured versus the number of observable variables. Again, we see that the growth is roughly linear with respect to the number of observables. The Circuit2 class of models shows much steeper growth than the other models, which is a function of the model structure. These experiments have shown that adding extra structure to the model classes under study created a linear relationship between the number of variables/observables and relative model memory.

Our next experiments studied the coverage of model classes. Figure 3 shows the coverage versus relative memory for four different aircraft sub-system models, AC-v1 through AC-v4. Note that every such coverage/memory curve has a similar shape, with the coverage asymptotically approaching 1 as memory increases. This particular graph shows how the

[8]Even though the models AC-v2, AC-v3 and AC-v4 have relatively few variables, each variable has many (up to 20) values, and diagnosing these models took hours on a 1GHz Pentium3 machine.

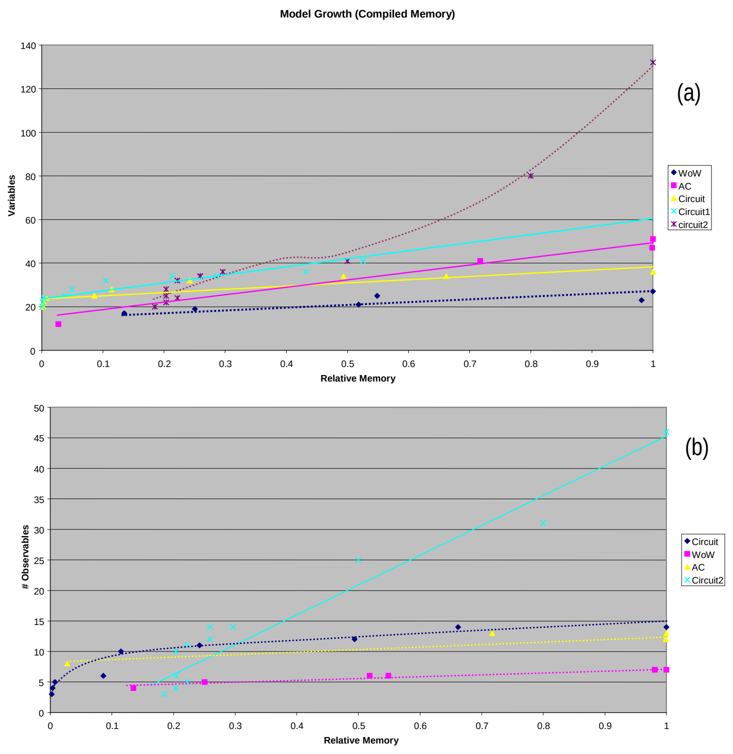

Figure 2: Graph showing tradeoff of coverage versus relative memory for several model classes. Graph (a) shows the memory growth versus the number of variables, and Graph (b) shows the memory growth versus the number of observables.

curves are displaced downwards (meaning reduced cover-

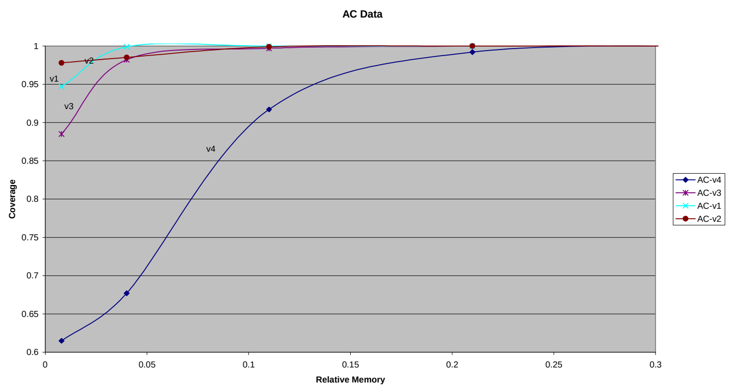

Figure 3: Graph showing tradeoff of coverage versus relative memory for four different aircraft sub-system models, AC-v1 through AC-v4.

Model Decomposition We now report on our experiments with decomposing models. We have adopted a causal network representation for our models, since it is possible to decompose such models by choosing observables appropriately (Darwiche & Provan 1997b). If we choose observables that form a cutset for a model, then the model is effectively decomposed into disjoint sub-models.

We applied this cutset-based approach to several models, and report results for three circuit models in Table 2. The results for the first two models show dramatic reductions of model size, and the last network shows roughly a 50% reduction. For the first two models, it turned out that adding the new observables reduced the multiple-fault rules of the original model to all single-fault rules.[9] As an example,

Model	Memory (KB)	Relative Memory
Circuit1	4,176.2	1.0
Circuit1-Decomposed	45.8	0.010
Circuit2	2,026.6	1.0
Circuit2-Decomposed	43.0	0.021
Circuit3	20.0	1.0
Circuit3-Decomposed	10.3	0.515

Table 2: Model Statistics for decomposition of Real-world Models. The relative memory is the fraction of the memory of the full model.

for Circuit2 the decomposition reduced 3597 multiple-fault rules to 10 single-fault rules. We provide examples of these rules below. We first two two rules derived from the full model with 4 faults (or Failure-Modes) in each:

[9]In terms of underlying causal model parameters, this corresponds to converting a multiply-connected model with large clique-sizes (with inference complexity exponential in the size of the largest clique) to a singly-connected model, in which inference is linear.

——— Full Model: rule 362 ——————

$([\text{Tx1-out=}\textit{false}] \wedge [\text{Tx2-out=}\textit{true}] \wedge [\text{Tx3-out=}\textit{false}] \wedge$

$[\text{Tx4-out=}\textit{false}] \wedge [\text{Tx5-out=}\textit{false}] \wedge [\text{Tx6-out=}\textit{false}] \wedge$

$[\text{B-out=}\textit{false}] \wedge [\text{B1-out=}\textit{false}] \wedge [\text{B2-out=}\textit{false}] \wedge$

$[\text{B3-out=}\textit{false}] \wedge [\text{B4-out=}\textit{false}] \wedge [\text{B5-out=}\textit{false}])$

$\Rightarrow ([\text{B-Mode=BAD}] \wedge [\text{B2-Mode=BAD}]$

$\wedge [\text{B4-Mode=BAD}] \wedge [\text{B5-Mode=BAD}])$

——— Full Model: rule 363 ——————

$([\text{Tx1-out=}\textit{true}] \wedge [\text{Tx2-out=}\textit{false}] \wedge [\text{Tx3-out=}\textit{false}] \wedge$

$[\text{Tx4-out=}\textit{true}] \wedge [\text{Tx5-out=}\textit{true}] \wedge [\text{Tx6-out=}\textit{false}] \wedge$

$[\text{B-out=}\textit{true}] \wedge [\text{B1-out=}\textit{false}] \wedge [\text{B2-out=}\textit{true}] \wedge$

$[\text{B3-out=}\textit{false}] \wedge [\text{B4-out=}\textit{true}] \wedge [\text{B5-out=}\textit{true}])$

$\Rightarrow ([\text{B-Mode=BAD}] \wedge [\text{B3-Mode=BAD}]$

$\wedge [\text{B4-Mode=BAD}] \wedge [\text{B5-Mode=BAD}]))$

In contrast, all rules derived from the decomposed model have a single fault each, as shown below:

———Decomposed Model: rule 9 ——————

$(\text{6AND-out=}\textit{false}] \wedge [\text{B3-out=}\textit{false}]) \Rightarrow [\text{B3-Mode=BAD}]$

———Decomposed Model: rule 10 ——————-

$(\text{ 6AND-out=}\textit{false}] \wedge [\text{B4-out=}\textit{false}]) \Rightarrow [\text{B4-Mode=BAD}]$

Weights

We have performed a variety of experiments to study the influence of assumption probability values on coverage. In these experiments, we have assigned different probability values to the assumptions, and report our results using the mean probability value, averaged over all the model probabilities.

Real-World Models Figure 4 shows the effect of mean assumption probability values on the coverage for two control models, a basic model and an extension of that model. These figures show how the mean probability value reduces the coverage values. For example, Figure 4(b) shows that for small probability values, the coverage asymptotes very quickly to coverage values near 1 at relatively small relative memory values, but as the probability gets larger, greater memory is needed to achieve high coverage values.

Randomized Models Computing a closed-form solution for Ψ_Λ defined over an arbitrary distribution $\vec{P}$ over $\mathcal{A}$ is a complex task, and we have derived results for some special cases, such as the case where are target prior probabilities are the same. In the following, we present results of an empirical analysis of rule generation. We use the parameters $(\chi, n, \vec{P})$ to perform an empirical tradeoff analysis of the rules compiled from a randomly-generated model.

We make the following assumptions:

1. Each target probability is skewed, i.e., either $p_i \gg \bar{p}_i$ or $p_i \ll \bar{p}_i$.
2. All targets are conditionally independent.

In our experiments, we assigned skewed probabilities to the target variables ranging from $p = 0.1$ to $p = 0.001$ for a true instantiation. In reporting our results, we averaged all target probability values and plot graphs using this average, $\bar{p}$. We used diagnostic models of digital electrical circuits with a rough ratio of 5 non-target variables for every target variable. We report results for models with between 3 and 1000 target variables.

Figure 5 shows a sample of some data, plotting expected outcome (or target coverage) versus average target probability. This figure describes plots for 5 different models, containing 10, 25, 50, 100 and 1000 target variables. For each model, we compiled 3 different expected case rule-sets. For example, for the 1000-target model we compiled rule-sets containing up to 2, 3 and 4 target variables per rule. In the figure we use the pair (i, j) to tag each graph, where i denotes the number of target variables per rule, and j the number of targets in the KB.

One of the key observations of this data is that for small KBs (10 or fewer targets), the target coverage remains high even when the average target probability ($\bar{p}$) rises to around 0.05 (and up to 0.1, even though it is not shown). However, as the KB size increases, the coverage is high only for quite small values of $\bar{p}$.

A second observation is that the graphs for a particular KB tend to cluster together. For example all graphs for the KB with $n = 1000$ are clustered, and increasing the number of target variables per rule produces graphs that occur very close to the group of three shown in the figure. Contrast this with the graphs for $n = 100$, which occur a region quite different to those for the cluster for $n = 1000$.

Our results indicate that we can smoothly trade off target coverage for decreased memory/inference time within a particular range of $\bar{p}$ for a given KB. For example, for a KB with $n = 10$ this $\bar{p}$ range is about (0,0.5), whereas for a KB with $n = 1000$ this $\bar{p}$ range is about (0,0.01).

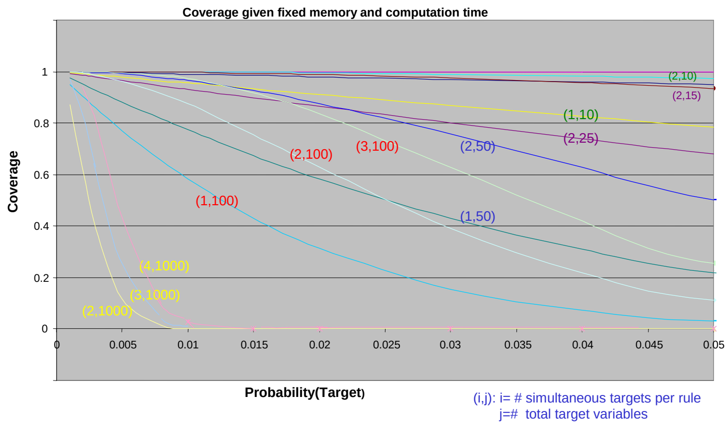

Figure 5: Graph showing tradeoff of solution quality versus time/memory.

Our experimental analysis has identified, for the set of models analyzed, a pair of parameters for which this ap-

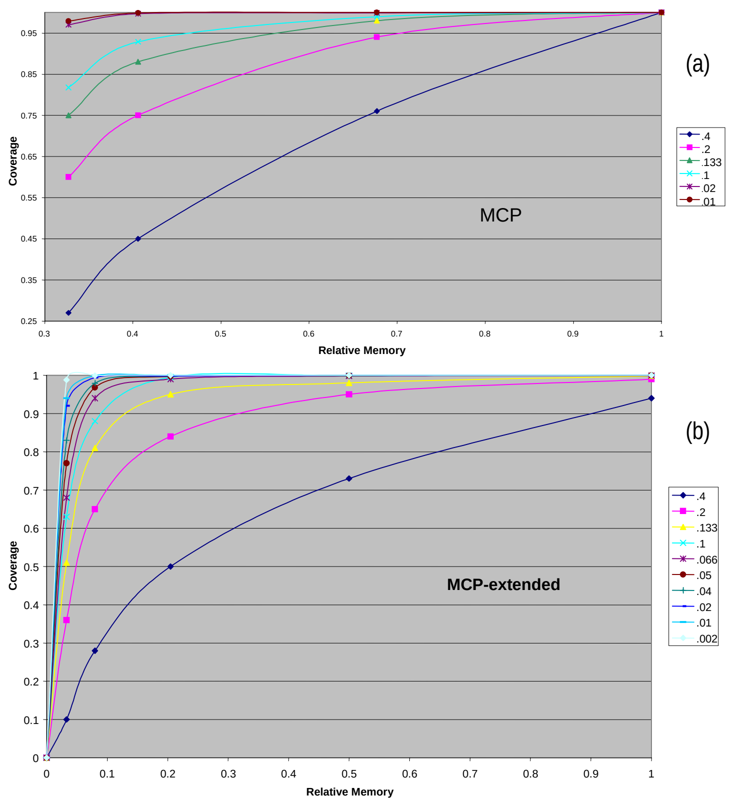

Figure 4: Graph showing tradeoff of coverage versus relative memory for control sub-system model, for various mean probability values.

proach will provide good expected coverage, regardless of KB size. For a target coverage value χ^*, a model with n target variables and mean target probability $\bar{p}$, the following result holds:

$$1.05 \leq \chi^* n\bar{p} \leq 1.625.$$

This result provides some important insights into this approach, as it has been applied to the set of models analyzed.

- For a given KB size, it tells you the type of skewed probability distribution necessary for the approach to work.
- Given an appropriate probability distribution, it can tell you the target coverage value that is achievable.
- Given an appropriate probability distribution and target coverage value, it tells you the size of the compiled KB.

One key issue that needs further work is to examine this result in greater detail, to see if it is more a property of the diagnostic models studied, or if it will generalize across multiple domains.

Another interesting result is the phase transition behavior based on the mean target probability $\bar{p}$: for large KBs, once the mean target probability increases beyond a certain threshold, the number of rules required to maintain a high coverage value increases dramatically. In fact, for some threshold value of n target variables, the savings associated with this approach are negligible if we desire a solution that covers arbitrary values of $\bar{p}$.

Relevance for Diagnosis

Under reasonable assumptions, we have found that the coverage provided just by rules containing few simultaneous targets can be extremely good. For diagnosis the basic intuition is that, assuming components are all well-designed and have low prior target probabilities, all multiple-target outcomes with three or more targets are very unlikely.

If the prior fault probabilities are all small (e.g., $p \leq 0.003$) then the diagnostics coverage is at least 90% using triple-target embedded models for n=1000 and a KB with 25,000 variables; this represents a reduction in the number of possible diagnoses from O(2^{1000}) in the full model to O(2^{27}) in the compiled model, leading to a 1000-fold decrease in memory needed and 100-fold decrease in diagnostics inference time. However, when these prior probabilities grow larger, the coverage diminishes for fixed k-target embedded models; to maintain coverage, the value of k needs to be increased.

Related Work

The task that we are trying to solve is to compute query coverage, using a set of minimal (weighted or most likely) query responses that covers some fraction of all possible query responses from a given theory. We determine minimality using the assumption weight associated with the query response. By computing the curve of query coverage versus size and complexity of coverage database, we show that we can smoothly trade off query coverage for inference complexity, and thereby "control" inferential complexity for abductive inference.

This work straddles a considerable body of work in several different areas, which we group into three primary areas: abductive reasoning, (approximate) knowledge compilation, and test set generation. We review the most relevant literature, and explain the ways in which our work is different.

Abductive Inference: Abduction has received considerable attention; for a review of the field see (Paul 1993). This article focuses on compiling a representation that is tailored to efficiently computing all minimal explanations, particularly for diagnostics applications. As mentioned earlier, the task of computing all plausible explanations has received less attention than that of computing a plausible explanation. This multiple-explanation task has been addressed by (Eiter & Makino 2003), who show the intractability of this task. More generally, the computational complexity of abduction has been studied thoroughly, e.g., in (Bylander *et al.* 1991; Eiter & Gottlob 1995; Liberatore & Schaerf 2000; del Val 2000).

Knowledge Compilation: This area has been surveyed by Cadoli and Donini (1997), who mention that the majority of approaches focus on complete compilation, and that the approximate compilation approaches lack obvious metrics for determining the distance of a compilation from the complete solution. In contrast to these approaches, we provide a clear distance metric for our approximate compilation. Darwiche and Marquis (2002) describe an approach to compiling weighted knowledge bases. Our approach is quite different, in that we assign weights not the clauses but to assumptions, and we approximate the quality of solution coverage.

Many researchers have developed techniques for compiling models for various applications, particularly diagnosis (Darwiche & Marquis 2001; Darwiche 1998). One of the most successful approaches, that of (Darwiche 1998), provides an approach that enables a simple evaluator to be used for embedded applications; the drawback of this approach is that it basically makes a time/space tradeoff, producing an embedded model for which inference is too inefficient for complex real-world systems.

For example, the Query DAG paradigm (Darwiche & Provan 1997a; Darwiche 1998) compiles a Bayesian network into an arithmetic expression (the Query DAG, or Q-DAG) that may be evaluated using a simple numerical evaluation algorithm. The complexity of Q-DAG *evaluation* is linear in the size of the Q-DAG; for example, for a Bayesian network such inference amounts to a standard evaluation of the arithmetic expression it represents. The *value* of the Q-DAG is that it reduces both the software and hardware resources required to utilize Bayesian or causal networks in on-line, real-world applications. Our proposed compilation approach extends the causal network compilation method of (Darwiche 1998): we compile a sub-model that can answer only the *most-likely* queries, given a pre-defined set of observations. This expected-query approach provides a method for making principled coverage/complexity tradeoffs.

Test Set Generation: The large body of work in test set generation focuses on computing a minimal (smallest cardinality) set of tests, where each test is an instantiation of observables that can isolate faults for a given system (Simpson & Sheppard 1994). In other words, we can define a rule as a test minimal with respect to the number of observables in the test.

The task we address is a strict generalization of test set generation, in that we adopt a more complex model with fewer restrictions, and we compute the equivalent of weighted tests. For the most part, the literature in test set generation focuses on a model that needs to be represented only by the system inputs I, outputs O, failure-modes F, and the relation of outputs to inputs and failure-models, $O = R(I, F)$ (Simpson & Sheppard 1994). In addition, most of the literature focuses on binary-valued variables and deterministic relations, although some approaches relax these restrictions (Deb *et al.* 1994). The primary algorithm for test set compaction is set covering (Flores, Neto, & Marques-Silva 1999), although many other algorithms have been adopted (Hamzaoglu & Patel 1998).

Conclusions

Generating expected-query rules to approximate the complete target coverage provided by a KB (e.g., causal network model) can save significant space and still guarantee high target coverage with respect to using the full model. Further, our analysis has identified a set of parameters $(\chi, n, \bar{p})$, respectively the coverage ratio, number of components and average prior target probability, to perform a tradeoff analysis of the rules generated by a model. This approach provides a method for defining the number and type (based on number of simultaneous targets in a rule) of rule necessary to achieve a prescribed coverage ratio χ.

Analysis of data on randomized models has identified that target coverage is highly sensitive to the average prior target probability $\bar{p}$ (especially for very large models), and is relatively insensitive to the number of components n for small $\bar{p}$. For example, if we fix $\bar{p}$ to some small value such as $p = 0.003$, increasing the number of components from 100 to 1000 causes a reduction in coverage only from 0.99 to 0.81 (for rules containing up to 4 targets); however, for a fixed model size of 1000, coverage falls from 0.81 (for $p = 0.003$) to 0.44 for for $\bar{p} = 0.005$ to 0.028 for for $\bar{p} = 0.01$.

Hence, high-reliability systems need only a relatively small number of rules, each containing up to a few simultaneous targets, to guarantee high diagnostic coverage. In contrast, low-reliability systems need a relatively large number of rules, containing up to 10 simultaneous targets (depending on the system), to guarantee high diagnostic coverage.

Future work is needed to determine, from a theoretical perspective, the significance of the bounds governing the relation among $(\chi, n, \bar{p})$, and to explore the phase transitions observed during compilation. We are also exploring the use of approximate cutset algorithms to identify good model decompositions and key observables that will lead to simplifying decompositions.

References

Appelt, D., and Pollack, M. 1009. Weighted abduction for plan ascription. Technical report, Tech. Rep. AI Center and CSLI, SRI International, Menlo Park, CA.

Bylander, T.; Allemang, D.; Tanner, M. C.; and Josephson, J. R. 1991. The computational complexity of abduction. *Artificial Intelligence* 49(1-3):25–60.

Cadoli, M., and Donini, F. M. 1997. A survey on knowledge compilation. *AI Communications* 10(3-4):137–150.

Console, L.; Portinale, L.; and Dupre, D. T. 1996. Using Compiled Knowledge to Guide and Focus Abductive Diagnosis. *IEEE Trans. on Knowledge and Data Engineering* 8(5):690–706.

Darwiche, A., and Marquis, P. 2001. A perspective on knowledge compilation. In *Proceedings of the Intl. Joint Conference of Artificial Intelligence*. Morgan Kaufmann, San Francisco, CA.

Darwiche, A., and Marquis, P. 2002. Compilation of Propositional Weighted Bases. In *Proceedings of the Ninth International Workshop on Non-Monotonic Reasoning (NMR'02)*, 6–14.

Darwiche, A., and Provan, G. 1997a. Query DAGs: A Practical Paradigm for Implementing Belief Network Infernece. *Journal of Artificial Intelligence Research* 6:147–176.

Darwiche, A., and Provan, G. M. 1997b. The effect of observations on the complexity of model-based diagnosis. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI)*, 94–99.

Darwiche, A. 1998. Model-based diagnosis using structured system descriptions. *Journal of Artificial Intelligence Research* 8:165–222.

Darwiche, A. 1999. On Compiling System Descriptions into Diagnostic Rules. In *Proceedings of the Conference on Principles of Diagnosis*.

Deb, S.; Pattipati, K.; Raghavan, V.; Shakeri, M.; and Shrestha, M. 1994. Multi-Signal Flow Graphs: A novel Approach for System Testability Analysis and Fault Diagnosis. In *Proc. IEEE AUTOTESTCON*, 361–373.

del Val, A. 2000. The complexity of restricted consequence finding and abduction. In *AAAI: 17th National Conference on Artificial Intelligence*. AAAI / MIT Press.

Eiter, T., and Gottlob, G. 1995. Semantics and complexity of abduction from default theories. In Mellish, C., ed., *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, 870–877. San Francisco: Morgan Kaufmann.

Eiter, T., and Makino, K. 2003. Generating all abductive explanations for queries on propositional horn theories. Technical Report Tech. Rep. INFSYS RR-1843-03-09, Institute of Inf.Sys., TU Vienna.

Flores, P.; Neto, H.; and Marques-Silva, J. 1999. On Applying Set Covering Models to Test Set Compaction. In *roceedings of the IEEE Great Lakes Symposium on VLSI (GLS)*.

Goldszmidt, M., and Pearl, J. 1991. System Z^+: A formalism for reasoning with variable strength defaults. In *Proceedings of American Association for Artificial Intelligence Conference*, 399–404.

Hamzaoglu, I., and Patel, J. H. 1998. Test set compaction algorithms for combinational circuits. In *ICCAD*, 283–289.

Henrion, M. 1991. Search-based Methods to Bound Diagnostic Probabilities in Very Large Belief Nets. 142–150.

Kumar, U., and Desai, U. 1996. Image interpretation using bayesian networks. *IEEE Trans. PAMI* 18(1):74–78.

Liberatore, P., and Schaerf, M. 2000. Compilability of abduction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI 2000)*.

Paul, G. 1993. Approaches to abductive reasoning—an overview. *Artificial Intelligence Review* 7:109–152.

Poole, D., and Kanazawa, K. 1994. A Decision-Theoretic Abductive Basis for Planning. In *Proceedings of the AAAI Spring Symposium on Decision-Theoretic Planning*, 232–239.

Poole, D. 1988. A Logical Framework for Default Reasoning. *Artificial Intelligence* 36:27–47.

Simpson, W., and Sheppard, J. 1994. *System Test and Diagnosis*. Kluwer.

Stickel, M. 1990. Rationale and Methods for Abductive Reasoning in Natural Language Reasoning and Interpretation. In Studer, R., ed., *Natural Language and Logic*, 331–352. Springer-Verlag.