MBT 2009

- Preprint -

Computing Optimal Tests for Non-deterministic Systems Using DNNF Graphs

Anika Schumann[1]

Institut für Informatik
Technische Universität München
85748 Garching, Germany

Martin Sachenbacher[2]

Institut für Informatik
Technische Universität München
85748 Garching, Germany

Jinbo Huang[3]

National ICT Australia and
The Australian National University
Canberra, ACT 0200 Australia

Abstract

The goal of testing is to distinguish between a number of hypotheses about a system—for example, different diagnoses of faults—by applying input patterns and verifying or falsifying the hypotheses from the observed outputs. Optimal distinguishing tests (ODTs) are those input patterns that are most likely to distinguish between hypotheses about non-deterministic systems. Finding ODTs is practically important, but it amounts in general to determining a ratio of model counts and is therefore computationally very expensive.
In this paper, we present a novel approach to this problem, which uses structural properties of the system to limit the complexity of computing ODTs. We first construct a compact graphical representation of the testing problem via compilation into *decomposable negation normal form*. Based on this compiled representation, we show how one can evaluate distinguishing tests in linear time, which allows us to efficiently determine an ODT. Experimental results from a real-world application show that our method can compute ODTs for instances that were intractable for previous approaches.

Keywords: testing, DNNF graphs, model counting.

[1] Email: anika.schumann@in.tum.de

[2] Email: sachenba@in.tum.de

[3] Email: jinbo.huang@nicta.com.au

1 Introduction

Testing asks whether a system can be stimulated with input patterns such that different hypotheses about the system can be verified or falsified from the observed output patterns. Applications include model-based fault analysis of technical systems, autonomous control (acquiring sensor inputs to discriminate among competing state estimates), and bioinformatics (designing experiments that help distinguish between different possible explanations of biological phenomena).

In many real-world applications of testing, the underlying models are non-deterministic; applying an input can lead to several possible outputs. Different notions of testing with such non-deterministic models have been introduced. In the area of diagnosis, [11] introduced *definitely* and *possibly discriminating tests* (DDTs and PDTs) for systems modeled as constraints. For a DDT, the sets of possible outputs are disjoint and thus it will necessarily distinguish among hypotheses, whereas for a PDT, the sets partially overlap and thus it may or may not distinguish among hypotheses. In automata theory, [1] studied the analogous problem of generating *strong* and *weak distinguishing sequences* for non-deterministic finite state machines; for sequences of length at most $k \in \mathbb{N}$, this can be reduced to the problem of finding DDTs and PDTs [6]. [7] introduced *optimal distinguishing tests* (ODTs), which generalize DDTs and PDTs by maximizing the ratio of distinguishing over non-distinguishing possible outcomes. Finding ODTs is important as it reduces the number of tests to be executed and the overall costs of the testing process. [7] proposed and analyzed a simple greedy-type algorithm to approximate ODTs, which in some real-world applications produces test inputs whose distinguishing ratios are close to those of ODTs.

In this paper, we present a novel search algorithm to compute ODTs (and thus DDTs and PDTs), which exploits structural properties of the model to limit the complexity of optimal test generation. Its main feature is a carefully constructed graph—through manipulation of logical theories and compilation into *decomposable negation normal form* (DNNF) [5]—that allows efficient computation of good upper bounds on ratios of model counts. These upper bounds are used to prune the search in a way motivated by a recent planning algorithm [8]. We show that our method can compute ODTs for instances that were intractable for previous approaches.

2 Background

Following the framework in [7,10,11], we assume that the system can be modeled as a *constraint satisfaction problem* (CSP), which is a triple $M = (\mathcal{V}, \mathcal{D}, \mathcal{C})$, where $\mathcal{D} = D(v_1) \times \ldots \times D(v_n)$ are the finite domains of finitely many variables $v_j \in \mathcal{V}$, $j = 1, \ldots, n$, and $\mathcal{C} = \{C_1, \ldots, C_m\}$ is a finite set of constraints with $C_i \subseteq \mathcal{D}$, $i = 1, \ldots, m$. We denote by X the set of all solutions, that is, assignments $\boldsymbol{x} \in \mathcal{D}$ to the variables such that all constraints are satisfied. That is, $X = \{\boldsymbol{x} \mid \boldsymbol{x} \in \mathcal{D},\ \mathcal{C}(\boldsymbol{x})\}$, where $\mathcal{C}(\boldsymbol{x})$ denotes $\boldsymbol{x} \in C_i$ for all $i = 1, \ldots, m$.

In addition, the system under investigation defines a set of *controllable* (input) variables $\mathcal{I}$ and a set of *observable* (output) variables $\mathcal{O}$. Formally, a hypothesis M for a system is then a CSP whose variables are partitioned into $\mathcal{V} = \mathcal{I} \cup \mathcal{O} \cup \mathcal{L}$, such that $\mathcal{I}$ and $\mathcal{O}$ are the input and output variables of the system, and for all assignments to $\mathcal{I}$, the CSP is satisfiable. The remaining variables $\mathcal{L} = \mathcal{V} \setminus (\mathcal{I} \cup \mathcal{O})$ are called internal state variables. We denote by $\mathcal{D}(\mathcal{I})$ and $\mathcal{D}(\mathcal{O})$ the cross product of the domains of the input and output variables, respectively: $\mathcal{D}(\mathcal{I}) = \bigtimes_{v \in \mathcal{I}} D(v)$ and $\mathcal{D}(\mathcal{O}) = \bigtimes_{v \in \mathcal{O}} D(v)$.

The goal of testing is then to find assignments of the input variables $\mathcal{I}$ that will cause different assignments of output variables $\mathcal{O}$ for different hypotheses. In the general case of non-deterministic systems, an input assignment can yield more than one possible output assignments. This case is frequent in practice; one reason is that in order to reduce the size of a model, it is common to aggregate the domains of system variables into small sets of values such as 'low', 'med', and 'high'; a side-effect of this abstraction is that the resulting models can no longer be assumed to be deterministic functions, even if the underlying system behavior was deterministic. Another reason is the test situation itself: even in a rigid environment such as an automotive test-bed, there are inevitably variables or parameters that cannot be completely controlled while testing the device.

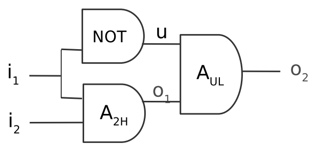
In order to capture sets of outputs, for a given hypothesis M and an assignment $\boldsymbol{t} \in \mathcal{D}(\mathcal{I})$ to the input variables, we define the *output function* $\mathcal{X} : \mathcal{D}(\mathcal{I}) \to 2^{\mathcal{D}(\mathcal{O})}$ with $\boldsymbol{t} \mapsto \{\boldsymbol{y} \mid \boldsymbol{y} \in \mathcal{D}(\mathcal{O}), \exists \boldsymbol{x} \in X : \boldsymbol{x}[\mathcal{I}] = \boldsymbol{t} \wedge \boldsymbol{x}[\mathcal{O}] = \boldsymbol{y}\}$, where $2^{\mathcal{D}(\mathcal{O})}$ denotes the power set of $\mathcal{D}(\mathcal{O})$, and $\boldsymbol{x}[\mathcal{I}]$, $\boldsymbol{x}[\mathcal{O}]$ denote the restriction of the vector $\boldsymbol{x}$ to the input variables $\mathcal{I}$ and the output variables $\mathcal{O}$, respectively. Note that since M will always yield an output, $\mathcal{X}(\boldsymbol{t})$ is non-empty.

2.1 Distinguishing tests

Non-deterministic models have given rise to the introduction of so-called *possibly* and *definitely distinguishing tests*, for short PDT and DDT, respectively [11]. The first type of test (PDT) might distinguish between hypotheses, as the sets of possible outputs partially overlap, whereas the second type (DDT) will necessarily do so, as the sets of possible outputs are disjunct:

Definition 2.1 (Distinguishing Tests) Consider $k \in \mathbb{N}$ hypotheses $M_1, \ldots, M_k$ with input variables $\mathcal{I}$ and output variables $\mathcal{O}$. Let $\mathcal{X}_i$ be the output function of hypothesis M_i with $i \in \{1, \ldots, k\}$. An assignment $\boldsymbol{t} \in \mathcal{D}(\mathcal{I})$ to $\mathcal{I}$ is a *possibly distinguishing test* (PDT), if there exists an $i \in \{1, \ldots, k\}$ such that $\mathcal{X}_i(\boldsymbol{t}) \setminus \bigcup_{j \neq i} \mathcal{X}_j(\boldsymbol{t}) \neq \emptyset$. An assignment $\boldsymbol{t} \in \mathcal{D}(\mathcal{I})$ is a *definitely distinguishing test* (DDT), if for all $i \in \{1, \ldots, k\}$ it holds that $\mathcal{X}_i(\boldsymbol{t}) \setminus \bigcup_{j \neq i} \mathcal{X}_j(\boldsymbol{t}) = \mathcal{X}_i(\boldsymbol{t})$.

For testing with non-deterministic automata models instead of logical theories or CSPs, there exists the analogous notion of so-called *weak* and *strong distinguishing sequences* [1,3]. Finding such sequences with a length bounded by some $k \in \mathbb{N}$ can be reduced to the problem of finding PDTs and DDTs, by unrolling automata into a constraint network using k copies of the automata's transition and observation relation [6]. Therefore, in this paper we restrict ourselves to models given as networks of finite-domain constraints.



A_{2H}:

i_1	i_2	o_1
L	L	L
L	H	L
L	H	H
H	L	H
H	H	H

A'_{2H}:

i_1	i_2	o_1
L	L	L
L	H	L
H	L	H
H	H	H

NOT:

i_1	u
L	H
H	L

A_{UL}:

u	o_1	o_2
L	L	L
L	H	L
H	L	L
H	L	H
H	H	H

A'_{UL}:

u	o_1	o_2
L	L	L
L	H	L
H	L	L
H	H	L

Fig. 1. Circuit with two possibly faulty adders: A'_{2H} and A'_{UL}.

2.2 Optimal Distinguishing Tests

Due to limited observability or a high degree of non-determinism, it is not uncommon that a DDT for the hypotheses does not exist, and one can instead only find PDTs. This motivates a *quantitative measure* for tests that refines and generalizes the previous notions of PDTs and DDTs. The intuition is that if we assume the possible outcomes (feasible assignments to the output variables) to be (roughly) equally likely, a PDT will be more likely to distinguish among two given hypotheses compared to another PDT, if the ratio of possible outcomes that are unique to a hypothesis versus all possible outcomes is higher. The notion of optimal distinguishing tests introduced in [7] formalizes this goal of finding tests that discriminate among two hypotheses as good as possible:

Definition 2.2 (Distinguishing Ratio) Given a test input $\boldsymbol{t} \in \mathcal{D}(\mathcal{I})$ for two hypotheses M_1, M_2 with input variables $\mathcal{I}$ and output variables $\mathcal{O}$, we define $\Gamma(\boldsymbol{t})$ to be the ratio of feasible outputs that distinguish among the hypotheses versus all feasible outputs:

$$\Gamma(\boldsymbol{t}) := \frac{|\mathcal{X}_1(\boldsymbol{t}) \cup \mathcal{X}_2(\boldsymbol{t})| - |\mathcal{X}_1(\boldsymbol{t}) \cap \mathcal{X}_2(\boldsymbol{t})|}{|\mathcal{X}_1(\boldsymbol{t}) \cup \mathcal{X}_2(\boldsymbol{t})|}$$

Γ is a measure for test quality that refines the notion of PDTs and DDTs: if Γ is 0, then the test does not distinguish at all, as both hypotheses lead to the same observations (output patterns). If the value is 1, then the test is a DDT, since the hypotheses always lead to different observations. If the value is between 0 and 1, then the test is a PDT (there is some non-overlap in the possible observations). Note that Γ is well-defined since for any chosen $\boldsymbol{t} \in \mathcal{D}(\mathcal{I})$, the sets $\mathcal{X}_1(\boldsymbol{t})$ and $\mathcal{X}_2(\boldsymbol{t})$ are non-empty.

An *optimal distinguishing test* (ODT) is one that has the maximal distinguishing ratio. Figure 1 shows an example consisting of three components: one NOT component and the two adders: A_{2H} and A_{UL}. The former is high dominant upon receiving input $i_2 = H$ and the latter is low dominant upon receiving input $u = L$. Here we consider the hypotheses that either both adders function normally, i.e. $M_1 = \{\text{NOT}, A_{2H}, A_{UL}\}$ or that both adders are faulty, i.e. $M_2 = \{\text{NOT}, A'_{2H}, A'_{UL}\}$.

This example has the two PDTs: $[-i_1, i_2]$ and $[-i_1, -i_2]$ which have the following distinguishing ratios:

$\Gamma([-i_1, i_2]) = \frac{|\{o_1,o_2\},\{o_1,-o_2\},\{-o_1,o_2\}|}{|\{o_1,o_2\},\{o_1,-o_2\},\{-o_1,o_2\},\{-o_1,-o_2\}|} = \frac{3}{4}$ and

$\Gamma([-i_1, -i_2]) = \frac{|\{-o_1,o_2\}|}{|\{-o_1,o_2\},\{-o_1,-o_2\}|} = \frac{1}{2}$.

Therefore, the input $[-i_1, i_2]$ is an ODT for this example.

2.3 Deterministic DNNF

We briefly review graph-based representations of propositional theories. A propositional theory f is in negation normal form (NNF) [5] if it only uses conjunction (and, $\wedge$), disjunction (or, $\vee$), and negation (not, $\neg$), and negation only appears next to variables. An NNF is *decomposable* (DNNF) if conjuncts of every conjunction share no variables. A DNNF is *deterministic* (d-DNNF) if disjuncts of every disjunction are pairwise logically inconsistent. A d-DNNF is *smooth* if disjuncts of every OR node mention the same set of variables. In the rest of the paper we also assume that every variable of the logical theory appears in a smooth d-DNNF graph (this can always be ensured in polynomial time [5]). Figure 2 illustrates a smooth d-DNNF graph representing the set of PDTs for the example shown in Figure 1.

d-DNNF graphs can be generated for propositional theories in conjunctive normal form (CNF) using the publicly available C2D compiler [4]. The complexity of this operation is polynomial in the number of variables and exponential only in the treewidth of the system in the worst case. Furthermore, given a DNNF graph G one can compute its projection $\Pi_\Sigma(G)$ on variable set Σ in linear time. Without impact on the computation time we therefore assume that M_1 and M_2 are defined over input and output variables only.

Based on a smooth d-DNNF graph G the number of models consistent with a partial assignment X to the d-DNNF variables can be computed in linear time. This follows from the fact that each model in G is represented by a subgraph G_s that satisfies the properties: (i) every OR node in G_s has exactly one child, (ii) every AND node in G_s has the same children as it has in G, and (iii) G_s has the same root as G.

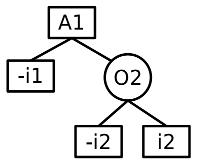


Fig. 2. Smooth d-DNNF graph representing $\neg i_1 \wedge (i_2 \vee \neg i_2)$. "A" and "O" indicate an And and an Or node, respectively. Numbers in non-leaf nodes are their identifiers.

For instance, for the graph in Figure 2 the model $[-i_1, -i_2]$ is represented by the subgraph with nodes $A1, O2, -i_1, -i_2$. Henceforth we will denote subgraphs with these properties as *m-subgraphs*. Those that satisfy only the first two properies we will denote as *s-subgraphs*. Further we say that a subgraph G_s is *labeled* by a literal l if G_s has a leaf node l.

Model counting with respect to X then consists of computing the number of m-subgraphs that are labeled by the literals in X. This is done by a bottom-up traversal of the graph as shown in Algorithm 1. The $\Lambda(N)$ value of each node N represents the number of s-subgraphs rooted in N whose labels are consistent with

Algorithm 1 Model counting with respect to instantiation X

$$\Lambda(N) = \begin{cases} 1 & \text{if } N \text{ is a leaf node consistent with X} \\ 0 & \text{if } N \text{ is a leaf node inconsistent with X} \\ \sum_i \Lambda(N_i) & \text{if } N = \bigvee_i N_i \\ \prod_i \Lambda(N_i) & \text{if } N = \bigwedge_i N_i \end{cases}$$

X. Hence, the $\Lambda(N)$ value of the root of the graph denotes the total number of consistent models.

3 ODT Computation Using DNNF

In this section we show how we can represent the ODT problem as a single DNNF graph and thus obtain a representation that directly exploits the system structure to achieve compactness. Furthermore we describe how we can modify the model counting algorithm described in Section 2.3 to determine the precise distinguishing ratio for any complete instantiation of a test vector (CITV) and an upper bound for any partial instantiation of a test vector (PITV), both in linear time. The latter allows an efficient search for an ODT without computing the Γ value for all test vectors. For some cases, depending on the structure of the DNNF graph, an ODT can be determined directly without a search.

3.1 Encoding the ODT problem as single DNNF

We now describe how we can represent the ODT problem as a single d-DNNF graph $\mathcal{G}'$. Since the size of that graph impacts the efficiency of the ODT computation we will also show how we can reduce the size of $\mathcal{G}'$ by taking into account that we are only interested in the model counts for the test vectors, not in the actual observations. We will show that $\mathcal{G}'$ can be projected on its input variables only, if we compute the number of models before they get lost through the projection.

A graph on whose basis one can compute the distinguishing ratio Γ needs to represent both hypotheses M_1 and M_2. These two hypotheses could be encoded by a graph of the form $M_1 \vee M_2$. However, in addition we need to distinguish the models that belong to the numerator of Γ, i.e., that are consistent with $f_{PDT\bar{O}} = (M_1 \vee M_2) \wedge \neg(M_1 \wedge M_2)$, and those that do not. We do this by introducing an additional variable d and by labelling every m-subgraph representing a model consistent with the numerator with the literal $-d$ and by adding the literal d to the remaining m-subgraphs. Thus $\mathcal{G}'$ is defined as follows:

$$\mathcal{G}' = (f_{PDT\bar{O}} \wedge \neg d) \vee (M_1 \wedge M_2 \wedge d)$$

As stated in Subsection 2.3 this d-DNNF graph can be generated based on the CNF representations of M_1 and M_2 using the available C2D compiler. Figure 3 depicts this graph $\mathcal{G}'$ for our example based on which we can compute the distinguishing ratio for any CITV. For instance, to compute $\Gamma([-i_1, -i_2])$ we count the number of m-subgraphs labeled by $-i_1$, $-i_2$ and $-d$ to obtain the numerator of Γ. In this example there is only one such m-subgraph that consists of the additional nodes:

$A1, O2, A4, O6, A8, -o_1$, and $o2$. To obtain the denominator of Γ we need to count the number of models consistent with $M_1 \vee M_2$ (see Definition 2.2), that is, the number of models consistent with $f_{PDT\bar{O}} \vee (M_1 \wedge M_2)$. Hence we add the number of m-subgraphs labeled by $-i_1$, $-i_2$ and $-d$ and the number of those labeled by $-i_1$, $-i_2$ and d. For our example there is only one subgraph labeled by the latter, namely the one consisting of the additional nodes: $A1, O2, A4, O6, A9, -o_1$, and $-o2$. Hence we obtain $\Gamma([-i_1, -i_2]) = \frac{1}{2}$, as given in Subsection 2.2. Note that, since the definition of graph $\mathcal{G}$' ensures that each of its m-subgraphs is labeled by either d or $-d$, we can also determine the denominator of Γ by counting the number of m-subgraphs labeled by $-i_1$ and $-i_2$.

Formally $\Gamma(\boldsymbol{t})$ based on $\mathcal{G}$' is defined as follows:

$$\Gamma(\boldsymbol{t}) := \frac{|\mathcal{G}'(\boldsymbol{t} \wedge \neg d)|}{|\mathcal{G}'(\boldsymbol{t} \wedge \neg d)| + |\mathcal{G}'(\boldsymbol{t} \wedge d)|} = \frac{|\mathcal{G}'(\boldsymbol{t} \wedge \neg d)|}{|\mathcal{G}'(\boldsymbol{t})|}$$

where $|\mathcal{G}'(X)|$ denotes the model count under the instantiation X. Hence we can compute the distinguishing ratio for any CITV using Algorithm 1.

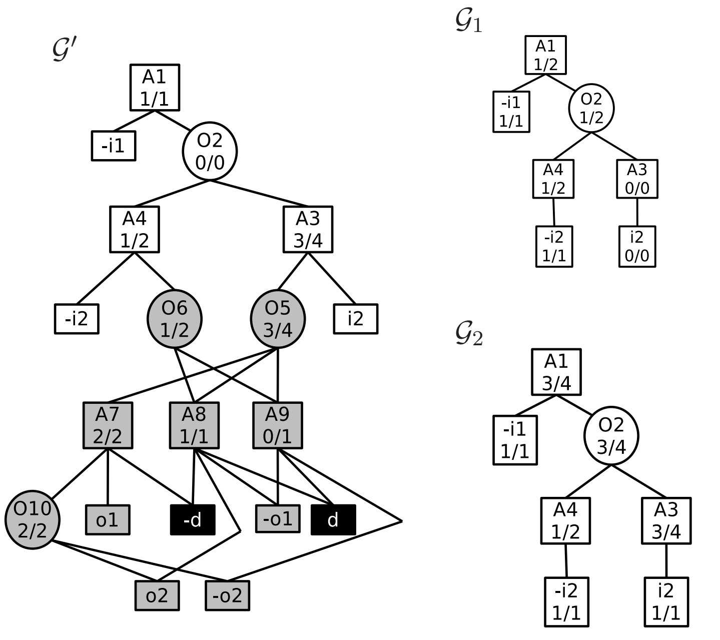


Fig. 3. Graph $\mathcal{G}$' for the example shown in Figure 1. The bottom label of a node N refers to its local counts: α_N/β_N. In graph $\mathcal{G}_1$ the bottom label denotes the test assessment values computed by Algorithm 2 when running it with $\boldsymbol{t} = [-i_1, -i_2]$ and in $\mathcal{G}_2$ these values present the results of running Algorithm 3 with respect to instantiation *true*.

We now describe how we abstract the non-input variables from $\mathcal{G}'$ without losing information relevant to the ODT problem. This is done by first computing the nodes $\mathcal{N}_I$ of the resulting graph $\mathcal{G} = \Pi_{\mathcal{I}}(\mathcal{G}')$ and then by counting the s-subgraphs that are in $\mathcal{G}'$ but not in $\mathcal{G}$. The node set $\mathcal{N}_I$ is composed of those nodes that have an input variable as descendant. For the graph $\mathcal{G}$' in Figure 3 these nodes are the "white" ones. Formally, a node $N \in \mathcal{N}$ is in $\mathcal{N}_I$ iff:

- N is a leaf labeled by an input variable or
- N has a child in $\mathcal{N}_I$.

In order to count the s-subgraphs that are in $\mathcal{G}'$ but not in $\mathcal{G}$ we partition the set of children for each OR and AND node in $\mathcal{G}'$ into those that belong to $\mathcal{N}_I$, i.e., $N_J = \{N_i \mid N_i \in \mathcal{N}_I\}$ and those that do not, i.e., $N_H = \{N_i \mid N_i \notin \mathcal{N}_I\}$. For these nodes the counting procedure of Algorithm 1 can be refined as follows:

$$\Lambda(N) = \begin{cases} \sum_h \Lambda(N_h) + \sum_j \Lambda(N_j) & \text{if } N = \bigvee_i N_i \\ \prod_h \Lambda(N_h) \cdot \prod_j \Lambda(N_j) & \text{if } N = \bigwedge_i N_i \end{cases}$$

where for $y = \{h, j\}$ and $N_Y = \emptyset$ we define $\prod_y \Lambda(N_y) = 1$.

The point is that the local count value $\gamma_N = \sum_h \Lambda(N_h)$ (resp. $\gamma_N = \prod_h \Lambda(N_h)$) does not change when computing the model count for a different $\boldsymbol{t}$. This results from the fact that the s-subgraph rooted in N_h is not labeled by any input variables and hence is consistent with any test vector. Thus it is sufficient to compute these values for each node only once. More precisely we label each non-leaf node N by α_N (resp. β_N) representing the local count value of the numerator (resp. denominator) of $\Gamma(N)$. α_N is obtained by running the refined version of Algorithm 1 with respect to instantiation $\neg d$ and for β_N we count the models with respect to instantiation *true*, i.e., $\Lambda(N) = 1$ for all leaves. Graph $\mathcal{G}$' in Figure 3 shows this labelling for our example.

3.2 Computation of $\Gamma(\boldsymbol{t})$ based on DNNF graph $\mathcal{G}$

For computing the distinguishing ratio we use the fact that the α_N and β_N values of OR nodes in $\mathcal{G}$ are necessarily 0. This follows from $\mathcal{G}'$ and hence $\mathcal{G}$ being smooth, which means that all s-subgraphs rooted in N are labeled with the same variables. Therefore, either one of these variables is an input variable in which case the local counts are 0 for all children of N or none of these variables is an input variable in which case the OR node itself is not in $\mathcal{G}$. Therefore the distinguishing ratio can be computed as presented in Algorithm 2. It is obtained from the value of the root as stated formally in Theorem 3.1. The bottom label of $\mathcal{G}_1$ of Figure 3 shows the Λ_α and Λ_β values for $\boldsymbol{t} = [-i_1, -i_2]$.

Theorem 3.1 (Test Assessment) *Let G be the root node of the smooth DNNF graph $\mathcal{G}$. Then $\Gamma(\boldsymbol{t}) = \Gamma(G, \boldsymbol{t})$ for any complete instantiation $\boldsymbol{t}$ of input variables.*

3.3 Upper bounds for partial test vectors

While the computation of the distinguishing ratio could have also been done based on two separate DNNF graphs, we now show how this single graph is essential to our new method for computing upper bounds on the distinguishing ratio based on a PITV $\boldsymbol{t}_p$. We start by pointing out that Algorithm 2 also retrieves exactly the subgraph of $\mathcal{G}$ that is consistent with $\boldsymbol{t}$.[4] The $\Gamma(\boldsymbol{t})$ value can therefore also be seen as the Γ value of that subgraph. We will now show how we can compute a subgraph

[4] This graph is composed of the root and of all nodes N for which $\Lambda_\beta(N) > 0$ and for which at least one parent N_p satisfies $\Lambda_\beta(N_p) > 0$.

Algorithm 2 Test assessment with respect to instantiation $\boldsymbol{t}$

For a leaf N consistent with $\boldsymbol{t}$ we set

$$\Lambda_\alpha(N) = \Lambda_\beta(N) = 1$$

and for a leaf N inconsistent with $\boldsymbol{t}$ we set

$$\Lambda_\alpha(N) = \Lambda_\beta(N) = 0.$$

For remaining nodes we compute:

$$(\Lambda_\alpha(N), \Lambda_\beta(N)) = \begin{cases} (\sum_i \Lambda_\alpha(N_i), & \\ \quad \sum_i \Lambda_\beta(N_i)) & \text{if } N = \bigvee_i N_i \\ (\alpha_N \cdot \prod_i \Lambda_\alpha(N_i), & \\ \quad \beta_N \cdot \prod_i \Lambda_\beta(N_i)) & \text{if } N = \bigwedge_i N_i \end{cases}$$

Then we compute the distinguishing ratio for each node:

$$\Gamma(N, \boldsymbol{t}) = \begin{cases} 0 & \text{if } \Lambda_\beta(N) = 0 \\ \frac{\Lambda_\alpha(N)}{\Lambda_\beta(N)} & \text{otherwise} \end{cases}$$

---

for a PITV $\boldsymbol{t}_p$ whose Γ value will necessarily be an upper bound on that of every completion of $\boldsymbol{t}_p$.

This subgraph $\mathcal{G}_S$ is composed of a set of s-subgraphs and the question of which of those should be included in $\mathcal{G}_S$ translates into the question of which children of an *OR* node should be included. The distinguishing ratio for such a node N with children $N_1, \ldots N_j$ is defined as:

$$\Gamma(N, \boldsymbol{t}_p) = \frac{\Lambda_\alpha(N_1) + \Lambda_\alpha(N_2) \cdots + \Lambda_\alpha(N_j)}{\Lambda_\beta(N_1) + \Lambda_\beta(N_2) \cdots + \Lambda_\beta(N_j)}.$$

Therefore, $\Gamma(N, \boldsymbol{t}_p)$ cannot be larger than the Γ value of its "best" child, i.e., the child with the highest distinguishing ratio. This results from the fact that for $x = \frac{a_1+b_1}{a_2+b_2}$ and $\frac{a_1}{a_2} \leq \frac{b_1}{b_2}$ and therefore $a_1 \leq \frac{a_2 \cdot b_1}{b_2}$ we have

$$x \leq \frac{\frac{a_2 \cdot b_1}{b_2} + b_1}{a_2 + b_2} = \frac{a_2 \cdot b_1 + b_1 \cdot b_2}{b_2 \cdot (a_2 + b_2)} = \frac{b_1 \cdot (a_2 + b_2)}{b_2 \cdot (a_2 + b_2)} = \frac{b_1}{b_2}.$$

Since this reasoning also holds if the numerator and denominator of x have more than two summands, we have

$$\Gamma(N, \boldsymbol{t}_p) \leq \max_i \frac{\Lambda_\alpha(N_i)}{\Lambda_\beta(N_i)}.$$

Hence, we can obtain an upper bound for $\Gamma(N, \boldsymbol{t}_p)$ by setting the values of each *OR* node to the values of its "best" child and by computing the values for the other nodes as in Algorithm 2.

Naturally, the search for an ODT will be the more efficient the tighter the upper bounds. To that end we include in the subgraph only the "best" child for those *OR* nodes that have a leaf as descendant which is not set by $\boldsymbol{t}_p$; see Algorithm 3. Formally, a node N is set by $\boldsymbol{t}_p$, i.e., $set(N, \boldsymbol{t}_p) = 1$, iff

- N is a leaf node whose variable is set by $\boldsymbol{t}_p$ or
- N is an AND or OR node such that for all children N_i we have $set(N_i, \boldsymbol{t}_p) = 1$.

For instance for node $O2$ of graph $\mathcal{G}_1$ shown in Figure 3 we have $set(O2, [-i_2]) = 1$ and $set(O2, [-i_1]) = 0$. Every child of an OR node set by $\boldsymbol{t}_p$ is necessarily either consistent or inconsistent with a complete instantiation of $\boldsymbol{t}_p$ regardless of how $\boldsymbol{t}_p$ is completed. Therefore we can compute the distinguishing ratio for these OR nodes as in Algorithm 2 and are still guaranteed that $\Gamma(N, \boldsymbol{t}_p)$ is an upper bound for every node N.

Note that resulting from $\mathcal{G}$ being smooth all consistent children of an OR node have necessarily the same set value and thus their Λ_α and Λ_β values are either all the result of a maximization or all the result of a summation.

Algorithm 3 Upper bound with respect to instantiation $\boldsymbol{t}_p$

For an OR node $N = \bigvee_i N_i$ we compute:

$$(\Lambda_\alpha(N), \Lambda_\beta(N)) = \begin{cases} (\sum_i \Lambda_\alpha(N_i), & \\ \ \sum_i \Lambda_\beta(N_i)) & \text{if } set(N, \boldsymbol{t}_p) = 1 \\ (\Lambda_\alpha(N_m), & \text{if } set(N, \boldsymbol{t}_p) = 0 \text{ and} \\ \ \Lambda_\beta(N_m)) & \Gamma(N_m, \boldsymbol{t}_p) = \max_i \Gamma(N_i, \boldsymbol{t}_p) \end{cases}$$

For remaining nodes the computation is same as in Algorithm 2. The distinguishing ratio $\Gamma(N, \boldsymbol{t}_p)$ is also computed as in Algorithm 2.

Note further that we are using the same name for the value function as in Algorithm 2. The reason is that Algorithm 2 can now be regarded as a special case of Algorithm 3 where $\boldsymbol{t}$ is a complete instantiation of a test vector, and hence for all OR nodes the set function returns true. It is precisely in this case that the computed value is guaranteed to be exact as formally stated in the following theorem:

Theorem 3.2 (Upper Bound) *Let $\boldsymbol{t}_p$ be a PITV and G the root node of the smooth DNNF graph $\mathcal{G}$. Then $\Gamma(\boldsymbol{t}) \leq \Gamma(G, \boldsymbol{t}_p)$ for any complete instantiation $\boldsymbol{t}$ of the free variables in $\boldsymbol{t}_p$.*

3.4 ODT computation

With the DNNF graph $\mathcal{G}$ and the linear time algorithms to compute the precise distinguishing ratio for a CITV and an upper bound for a PITV we have obtained the basis for our ODT search method. This consists of a branch-and-bound search over the input variables. Iteratively we set the input variables until either all variables are set and the precise distinguishing ratio is obtained or until the upper bound of the PITV is lower than the Γ value for a previously computed CITV.

Interestingly, if $\mathcal{G}$ has a certain structure we can obtain an ODT without a search by making use of the facts (i) that we can compute an upper bound Γ_{UB} for the distinguishing ratio of the ODT by running Algorithm 3 with respect to instantiation *true*, and (ii) that there is exactly one test vector $\boldsymbol{t}$ that is consistent with the resulting subgraph $\mathcal{G}_S$. The latter results from graph $\mathcal{G}$ being decomposable. Now, since we can compute $\Gamma(\boldsymbol{t})$ using Algorithm 2 we could be able to determine an ODT in linear time as stated in the following theorem:

Theorem 3.3 *Let* $\boldsymbol{t}$ *be a test vector consistent with the subgraph* $\mathcal{G}_S$ *that is obtained from computing the distinguishing ratio with respect to instantiation true.* $\boldsymbol{t}$ *is an ODT that can be computed in linear time if* $\Gamma(\boldsymbol{t}) = \Gamma(\mathcal{G}_S)$*.*

For our example the condition of this theorem holds and hence we could obtain the ODT $\boldsymbol{t} = [-i_1, i_2]$ without a search (see $\mathcal{G}_2$ of Figure 3). Note that the condition of this Theorem always holds if $\mathcal{G}$ is deterministic. Unfortunately only graph $\mathcal{G}$' is guaranteed to be deterministic and the projection operation applied to $\mathcal{G}$' to obtain $\mathcal{G}$, i.e., the abstraction from the observable variables, does not preserve determinism. To ensure determinism we would need to recompile $\mathcal{G}$, but this might alter its structure completely. This would result in the loss of local count values and hence our ability to compute the distinguishing ratio from that graph.

4 Experimental Evaluation

We evaluated our DNNF-based testing method on a model of an automotive engine test-bed [9], which consists of three major components: engine, pipe, and throttle. The goal is to find leaks in the pipe by assigning three to four controllable variables, and observing three to four measurable variables. The problem has been turned into sets of discrete instances of varying sizes by applying different abstractions to the original mixed discrete-continuous model, and using a direct encoding from CSP to SAT [12].

inst.	#nodes	time
1a	58	0.04
2a	103	0.06
3a	161	0.09
4a	205	0.10
5a	329	0.20
6a	245	0.21
7a	362	0.40
8a	4766	5.41
9a	2654	36.6
10a	65063	414

inst.	#nodes	time
1b	66	0.06
2b	124	0.07
3b	191	0.10
4b	4865	14.7
5b	48238	396
6b	102817	1566
7b	–	–
8b	–	–
9b	–	–
10b	–	–

Table 1
Results of ODT (left) and DDT (right) computation.

Table 1 shows the experimental results for computing ODTs and DDTs. For each instance, we report the size (number of nodes) of the DNNF graph computed, and the computation time in seconds on a Linux Dual-Core PC with 1GB of RAM.

For instances that have DDTs, we compared our method with the most recent specialized DDT approach [10] based on quantified Boolean formulas and the QBF solver sKizzo [2]. That approach was able to solve instances 1b–3b and ran out of memory for the rest. In contrast, our ODT approach could also solve instances 4b–6b.

Our method is the first that computes exact ODTs; hence we cannot compare it directly with previous approaches. However, we used the greedy algorithm of [7] to compute approximate solutions for the same problem instances. This algorithm was only able to solve instances 1a–7a; that is, for instances 8a–10a we were able to compute an optimal solution where the previous approach could not even compute a suboptimal one.

5 Conclusion and Future Work

Optimal distinguishing tests generalize and refine the notion of possibly and definitely distinguishing and strong and weak tests for non-deterministic systems. Since computing ODTs can be computationally very expensive, previous work has focused on approximate solutions. We presented a new method to compute exact ODTs based on innovative ways of compiling the ODT problem into DNNF and computing upper bounds to prune a systematic search. Experimental results show that the method is able to compute both ODTs and DDTs for instances that were too large for previous methods. Future work may include extensions of this method to finding tests with minimal costs, or to situations where possible output patterns cannot be assumed to be equally likely.

References

[1] R. Alur, C. Courcoubetis, and M. Yannakakis. Distinguishing tests for nondeterministic and probabilistic machines. In *Proc. ACM Symposium on Theory of Computing*, pages 363–372, 1995.

[2] M. Benedetti. skizzo: A suite to evaluate and certify QBFs. In *Proc. CADE-05*, 2005.

[3] S. Boroday, A. Petrenko, and R. Groz. Can a model checker generate tests for non-deterministic systems? *Elec. Notes Theor. Comp. Sci.*, 190:3–19, 2007.

[4] A. Darwiche. The c2d compiler user manual. Technical report, Comp. Sci. UCLA, 2005.

[5] A. Darwiche and P. Marquis. A knowledge compilation map. *Journal of Artificial Intelligence Research*, 17:229–264, 2002.

[6] M. Esser and P. Struss. Fault-model-based test generation for embedded software. In *Proc. IJCAI-07*, pages 342–347, 2007.

[7] S. Heinz and M. Sachenbacher. Using model counting to find optimal distinguishing tests. ZIB Report 08-32, Zuse Institute Berlin, 2008.

[8] J. Huang. Combining knowledge compilation and search for conformant probabilistic planning. In *Proc. ICAPS-06*, pages 253–262, 2006.

[9] J. Luo, K. Pattipati, L. Qiao, and S. Chigusa. An integrated diagnostic development process for automotive engine control systems. *IEEE Trans. on Systems, Man, and Cybernetics*, 37(6):1163–1173, 2007.

[10] M. Sachenbacher and S. Schwoon. Model-based testing using quantified CSPs. In *ECAI-08 Workshop on Model-based Systems*, 2008.

[11] P. Struss. Testing physical systems. In *Proc. AAAI-94*, pages 251–256, 1994.

[12] T. Walsh. SAT vs. CSP. In *Proc. of CP-00*, pages 441–456, 2000.