Equivalence Queries and DNF Formulas

Dana Angluin*, Yale University
YALEU/DCS/RR-659
November 1988

*Supported by the National Science Foundation, IRI-8718975

Equivalence Queries and DNF Formulas

Dana Angluin *
Yale University

November 1988

Abstract

We show there is no polynomial time algorithm that exactly identifies DNF formulas over n variables using only equivalence queries. This result holds even if the target formula is known to be monotone and to contain n terms each with at most $f(n)$ literals for any unbounded function $f(n)$. Dual results hold for CNF formulas. This solves an open problem in [3] and complements similar results for dfa's, nfa's, and cfg's in [2].

1 The Idea

The idea of the proof is to exhibit a target class of DNF formulas and an adversary strategy for answering equivalence queries concerning DNF formulas that "gives away" very little information with each counterexample, so that polynomially many queries do not suffice to narrow down the possible functions in the target class to just one. The key fact that we use about DNF formulas is that every DNF formula is either satisfied by a truth value assignment that assigns 1 to "very few" variables, or is falsified by an assignment that assigns 0 to "very few" variables. Using this fact, the adversary strategy answers an equivalence query about a DNF formula with one of these two types assignments as the counterexample. Then we show that less than a polynomial fraction of the target class of DNF formulas is satisfied by a particular assignment with "very few" 1's, and similarly, less than a polynomial fraction are falsified by a particular assignment with "very few" 0's. Hence, each query eliminates less than a polynomial fraction of the target class, and more than a polynomial number of queries are required.

2 Definitions

2.1 Formulas

To keep the combinatorial analysis simple, we choose a slightly redundant method of naming DNF formulas. Let n be a positive integer, and let V_n denote the set of variables $\{x_1, \ldots, x_n\}$. The *negation* of a variable x_i is denoted $\neg x_i$. A *literal* is a variable or the negation of a variable.

*Supported by NSF grant IRI-8718975

A *term* is a finite sequence of literals, possibly with repetition. The *size* of the term is the number of literals in the sequence. A term is *monotone* if it contains no negations of variables. If L is a literal and τ is a term, we write $L \in \tau$ if and only if L is in the sequence of literals composing τ. A term is *contradictory* if and only if for some i the term contains both x_i and $\neg x_i$.

A *DNF formula* is a finite sequence of terms, possibly with repetition. The *size* of a DNF formula is the sum of the sizes of the sequence of terms. A DNF formula is monotone if it contains only monotone terms. If τ is a term and ϕ is a DNF formula, we write $\tau \in \phi$ if and only if τ is in the sequence of terms that composes ϕ. A DNF formula is *reduced* if and only if it contains no contradictory tems.

We write terms as the concatenation of the literals they contain, and DNF formulas as the sequence of terms they contain, separated by "+". Thus, an example of a DNF formula is:

$$x_1 \neg x_2 x_1 x_4 + x_2 \neg x_5 x_5 \neg x_1 x_7 + x_1 \neg x_2 x_1 x_4.$$

The size of this formula is 13; note that it is not reduced, since it contains a contradictory term.

An *assignment* a is a mapping of the variables V_n to the set $\{0,1\}$. This is extended to literals, terms, and DNF formulas in the usual way, that is, $a(\neg x_i) = 1 - a(x_i)$, if τ is a term, $a(\tau) = \bigwedge_{L \in \tau} a(L)$, and if ϕ is a DNF formula, $a(\phi) = \bigvee_{\tau \in \phi} a(\tau)$.

Two DNF formulas ϕ and ψ over V_n are *logically equivalent*, denoted $\phi \equiv \psi$, if and only if for all assignments a to V_n,

$$a(\phi) = a(\psi).$$

The DNF formula given as an example above is logically equivalent to the formula $x_1 \neg x_2 x_4$ of size 3. Note that any DNF formula ϕ is logically equivalent to the reduced formula obtained by dropping all the contradictory terms from ϕ.

If n, t, and m are positive integers, let $M(n,t,m)$ denote the set of all monotone DNF formulas over the variables V_n that have t terms, each consisting of a product of m positive literals. Thus, one element of $M(4,3,2)$ is

$$x_1 x_1 + x_2 x_3 + x_2 x_4.$$

Note that the size of any formula from $M(n,t,m)$ is tm. The number of monotone terms over V_n of size m is n^m, and the number of formulas in $M(n,t,m)$ is n^{tm}. In general $M(n,t,m)$ contains many logically equivalent formulas.

2.2 Learning algorithms and equivalence queries

A learning algorithm is given inputs n and s and an equivalence oracle for an unknown DNF formula ϕ_* of size s over V_n. The oracle will answer *equivalence queries*, that is, given an input consisting of any DNF formula ϕ over V_n, the oracle answers *yes* if $\phi \equiv \phi_*$ and *no* otherwise. If the answer is *no*, the oracle also supplies a *counterexample*, that is, an assignment a to V_n such that $a(\phi) \neq a(\phi_*)$. For definiteness, assume that the assignment is represented as a string of n bits, representing the values of a on $x_1, \ldots, x_n$. If $a(\phi) = 0$ then a is *positive* counterexample; otherwise a is a *negative* counterexample. The oracle's choice of a counterexample is assumed to be arbitrary.

Let $T(n,s)$ be a function from pairs of integers to integers. We say that algorithm A exactly identifies DNF formulas using only equivalence queries in time $T(n,s)$ if and only if for every positive integer n and for every DNF formula ϕ_* over V_n, when A is run with inputs n and $size(\phi_*)$ and an equivalence oracle for ϕ_*, A halts in at most $T(n, size(\phi_*))$ steps and outputs a DNF formula ϕ that is logically equivalent to ϕ_*.

Algorithm A is a *polynomial-time algorithm* that exactly identifies DNF formulas using only equivalence queries if and only if there exists a two-variable polynomial $p(n,s)$ such that A exactly identifies DNF formulas using only equivalence queries in time $p(n,s)$.

3 The Main Result

Theorem 1 *There is no polynomial-time algorithm that exactly identifies DNF formulas using only equivalence queries.*

4 The Proof

The general idea is to show that given a polynomial $p(n,s)$, there are constants T and M such that for all sufficiently large n, no learning algorithm running in time $p(n,s)$ can exactly identify all the elements of $M(n,T,M)$ using only equivalence queries. We describe an adversary strategy that answers equivalence queries in such a way that a very small fraction of hypotheses in $M(n,T,M)$ are eliminated by each counterexample, so that too many queries are required to pin down one hypothesis in $M(n,T,M)$.

4.1 A key fact

The property of DNF formulas that we use in the proof is that every DNF formula is satisfied by an assignment with "few" 1's or is falsified by an assignment with "many" 1's. If a is an arbitrary assignment on V_n, let $p(a)$ denote the number of variables assigned 1 by a, that is,

$$p(a) = |\{i : a(x_i) = 1\}|.$$

Then the fact we use is the following.

Lemma 2 *Let $n \geq 4$ be an integer, and let ϕ be any reduced DNF formula with $q \geq 1$ terms over V_n. If ϕ contains some term with fewer than $\sqrt{n}$ positive occurrences of literals, then there is an assignment a_1 such that $p(a_1) < \sqrt{n}$ and $a_1(\phi) = 1$. If every term of ϕ contains at least $\sqrt{n}$ positive occurrences of literals, then there is an assignment a_0 such that $p(a_0) \geq n - 1 - (\sqrt{n} - 1)\log_2 q$ and $a_0(\phi) = 0$.*

This is a corollary of the following simple lemma.

Lemma 3 *Let n be any positive integer. Let α be a real number such that $0 < \alpha < 1$. Suppose ϕ is a DNF formula over V_n with $q \geq 1$ terms such that each term contains at least αn distinct positive literals. Then there is a set $V \subseteq V_n$ of at most $1 + \lfloor \log_b q \rfloor$ variables such that every term of ϕ contains a positive occurrence of some element of V, where $b = 1/(1-\alpha)$.*

Proof. We construct V as follows. Let x_i be a variable that maximizes the number of terms of ϕ that contain a positive occurrence of x_i. Add x_i to V and remove from ϕ any term that contains a positive occurrence of x_i. Iterate this process until no terms are left in ϕ.

Since every term of ϕ contains at least αn positive occurrences of variables, at least one variable must occur positively in a fraction α of the terms remaining, so after r elements have been added to V, there must be at most $(1-\alpha)^r q$ terms left in ϕ. Hence, for $b = 1/(1-\alpha)$, when

$$r = 1 + \lfloor \log_b q \rfloor,$$

we have

$$(1-\alpha)^r q < (1-\alpha)^{\log_b q} q = 1.$$

Thus fewer than 1 term must be left in ϕ, that is, ϕ must be empty. Q.E.D.

Proof of Lemma 2. Let $n \geq 4$ and let ϕ be any reduced DNF formula with $q \geq 1$ terms over V_n. If some term τ of ϕ has fewer than $\sqrt{n}$ positive occurrences of literals, let $a_1(x_i) = 1$ if and only if x_i occurs positively in τ. Since ϕ is reduced, τ is non-contradictory, and $a_1(\tau) = 1$. Then $p(a_1) < \sqrt{n}$ and $a_1(\phi) = 1$.

If every term of ϕ has at least $\sqrt{n}$ positive occurrences of literals, then we apply the preceding lemma with $\alpha = 1/\sqrt{n}$ to conclude that there is a set of variables V such that every term of ϕ has a positive occurrence of some variable from V and $|V| \leq 1 + \lfloor \log_b q \rfloor$ where $b = 1/(1-\alpha)$. Take $a_0(x_i) = 0$ if and only if $x_i \in V$. Then $a_0(\phi) = 0$, and $p(a_0) = n - |V|$. For $n \geq 4$, $\log_b q \leq (\sqrt{n}-1)\log_2 q$, so $p(a_0) \geq n - 1 - (\sqrt{n}-1)\log_2 q$. Q.E.D.

4.2 How to answer queries

We now can describe the adversary's strategy for answering queries. Assume $n \geq 4$. Suppose input to the query is a DNF formula ϕ over V_n with q terms. We may assume without loss of generality that ϕ is reduced. There are three cases; in each case the reply is "no", and the counterexample a is as described.

1. If $q = 0$, that is, ϕ contains no terms, then $a(\phi) = 0$ for all assignments a to V_n. In this case, the counterexample is the assignment $a(x_i) = 1$ for all $x_i \in V_n$. Since $a(\phi) = 0$, a is a positive counterexample.

2. If there exists some term τ that contains fewer than $\sqrt{n}$ positive literals, let a be the assignment a_1 guaranteed by Lemma 2 such that $a_1(\phi) = 1$ and $p(a_1) < \sqrt{n}$. Since $a(\phi) = 1$, a is a negative counterexample.

3. If ϕ has at least one term and every term of ϕ contains at least $\sqrt{n}$ positive occurrences of literals, let a be the assignment a_0 guaranteed by Lemma 2 such that $a_0(\phi) = 0$ and $p(a_0) \geq n - 1 - (\sqrt{n}-1)\log_2 q$. Since $a(\phi) = 0$, a is a positive counterexample.

Now we need to analyze how many elements of the hypothesis space $M(n,t,m)$ are eliminated by each of answers above. A formula ϕ over V_n is *eliminated* by a counterexample a if and only if a is positive and $a(\phi) = 0$ or a is negative and $a(\phi) = 1$.

In case (1), the counterexample a is positive and $a(\phi) = 1$ for every $\phi \in M(n,t,m)$, so no elements of $M(n,t,m)$ are eliminated by a. In case (2), the counterexample a is negative, so $\phi \in M(n,t,m)$ is eliminated by a if and only if $a(\phi) = 1$. In case (3), the counterexample a is positive, so $\phi \in M(n,t,m)$ is eliminated by a if and only if $a(\phi) = 0$. In the next subsection we determine how many elements of $M(n,t,m)$ are assigned 0 by a given assignment.

4.3 How many hypotheses are assigned 0?

Lemma 4 *If a is any assignment on V_n, the number of elements of $M(n,t,m)$ such that $a(\phi) = 0$ is $(n^m - p(a)^m)^t$.*

Proof. If τ is a monotone term over V_n of size m, then $a(\tau) = 1$ if and only if each variable in τ is assigned 1 by a. Hence the number of monotone terms τ over V_n of size m such that $a(\tau) = 1$ is just the number of ways of choosing (with repetition) a sequence of m variables out of the $p(a)$ assigned 1 by a, that is, $p(a)^m$. Thus the number of monotone terms τ over V_n of size m such that $a(\tau) = 0$ is $n^m - p(a)^m$.

For each element $\phi \in M(n,t,m)$, $a(\phi) = 0$ if and only if $a(\tau) = 0$ for each term τ in ϕ. Thus, the number of elements $\phi \in M(n,t,m)$ such that $a(\phi) = 0$ is the number of ways of choosing (with repetition) a sequence of t terms from the set of monotone terms τ over V_n of size m such that $a(\tau) = 0$. That is, the number of $\phi \in M(n,t,m)$ such that $a(\phi) = 0$ is $(n^m - p(a)^m)^t$. Q.E.D.

If a is any assignment on V_n, let

$$f_0[n,t,m](a) = |\{\phi \in M(n,t,m) : a(\phi) = 0\}|/|M(n,t,m)|.$$

This is the fraction of elements of $M(n,t,m)$ that are assigned 0 by a. Also, let

$$f_1[n,t,m](a) = 1 - f_0[n,t,m],$$

which is the fraction of elements of $M(n,t,m)$ that are assigned 1 by a. Then, dividing the expression in the preceding lemma by $|M(n,t,m)| = n^{tm}$, we get the following.

Corollary 5

$$f_0[n,t,m](a) = (1 - (p(a)/n)^m)^t,$$

and

$$f_1[n,t,m](a) = 1 - (1 - (p(a)/n)^m)^t.$$

4.4 Two lemmas

Lemma 6 *Let n be any positive integer and x any real number such that $0 \leq x \leq 1/n$. Then $(1-x)^n \geq 1 - 2nx$.*

This is not difficult to prove using the binomial theorem.

Lemma 7 *For all positive integers n, t, m such that $n > t$ and $n > m$, if S is any subset of more than $1/2$ of the elements of $M(n,t,m)$, then S contains at least two logically distinct formulas.*

Proof. Let n, t, and m be positive integers such that $n > t$ and $n > m$. For each formula $\phi \in M(n,t,m)$, let $E(\phi)$ be the set of all formulas $\psi \in M(n,t,m)$ such that $\psi \equiv \phi$. It suffices to show that for all $\phi \in M(n,t,m)$, $|E(\phi)| \leq (1/2)|M(n,t,m)|$.

If π is any permutation of the set of integers from 1 to n and $\phi \in M(n,t,m)$, let $\pi(\phi)$ be the formula obtained from ϕ by substituting $x_{\pi(i)}$ for x_i for all $i = 1,\ldots,n$. Thus π induces a permutation on $M(n,t,m)$. Note that if $\phi \equiv \psi$, then $\pi(\phi) \equiv \pi(\psi)$. Moreover, $\pi(E(\phi)) = E(\pi(\phi))$.

We now show that for any formula $\phi \in M(n,t,m)$, there exists a permutation π such that $\phi \not\equiv \pi(\phi)$. This will complete the proof, because for this π, $E(\phi)$ and $E(\pi(\phi))$ are disjoint subsets of $M(n,t,m)$ of equal cardinality, and therefore $|E(\phi)| \leq (1/2)|M(n,t,m)|$.

Order the set of all assignments by $a \leq a'$ if and only if for each $i = 1,\ldots,n$, $a(x_i) \leq a'(x_i)$. Let A be the set of minimal elements of the set of assignments satisfying ϕ. Then $1 \leq |A| \leq t$. Let $a \in A$ minimize $p(a)$. Then $1 \leq p(a) \leq m$. Also, at most t assignments a' satisfying ϕ have $p(a') = p(a)$.

Consider the image of a under all possible permutations of the variables; we must obtain all possible assignments a' with $p(a') = p(a)$. Since $1 \leq p(a) < n$, there are at least n different assignments a' with $p(a') = p(a)$. However, there are at most t assignments a' that satisfy ϕ and have $p(a') = p(a)$, so for at least one permutation π, $\pi(\phi)$ must be satisfied by an assignment that does not satisfy ϕ, that is, $\pi(\phi) \not\equiv \phi$. Q.E.D.

4.5 How many hypotheses are eliminated?

Now we are ready to analyze the fraction of elements of $M(n,t,m)$ that are eliminated in cases (2) and (3) of the adversary strategy.

Case (2). In this case, $p(a) < \sqrt{n}$, and the counterexample a is negative, so we need to bound the fraction of hypotheses from $M(n,t,m)$ that are assigned 1 by a, that is, $f_1[n,t,m](a)$. By Corollary 5,

$$f_1[n,t,m](a) = 1 - (1 - (p(a)/n)^m)^t.$$

Since $p(a) < \sqrt{n}$,

$$f_1[n,t,m](a) \leq 1 - (1 - (1/\sqrt{n})^m)^t.$$

Let

$$x = (1/\sqrt{n})^m.$$

Provided that $tx \leq 1$, by Lemma 6,

$$(1-x)^t \geq 1 - 2tx.$$

Hence,

$$f_1[n,t,m](a) \leq 2tx,$$

and

$$f_1[n,t,m](a) \leq 2t(1/\sqrt{n})^m.$$

Case (3). In this case, $p(a) \geq n - (1 + (\sqrt{n} - 1)\log_2 q)$. Assuming that $q \geq 2$,

$$p(a) \geq n - (\sqrt{n})\log_2 q.$$

In case (3) the assignment a is a positive counterexample, so an element of $M(n,t,m)$ is eliminated by a if and only if it is assigned 0 by a. From Corollary 5,

$$f_0[n,t,m](a) = (1 - (p(a)/n)^m)^t,$$

so

$$f_0[n,t,m](a) \leq (1 - ((n - (\sqrt{n})\log_2 q)/n)^m)^t,$$

or

$$f_0[n,t,m](a) \leq (1 - (1 - \log_2 q/\sqrt{n})^m)^t.$$

Let $x = \log_2 q/\sqrt{n}$. Provided that $mx \leq 1$, by Lemma 6 we have

$$(1-x)^m \geq 1 - 2mx,$$

so

$$f_0[n,t,m](a) \leq (1 - (1 - 2mx))^t,$$

and

$$f_0[n,t,m](a) \leq (2m\log_2 q/\sqrt{n})^t.$$

4.6 Proof of Theorem 1

To prove Theorem 1, we assume to the contrary that A is an algorithm that exactly identifies DNF formulas in polynomial time using only equivalence queries. Let $p(n,s)$ be a polynomial bounding the running time of A. Without loss of generality we may assume that $p(n,s)$ is increasing in both arguments and positive for all pairs of positive integers n and s. Let $q(n) = p(n,n^2)$. Choose N_1 sufficiently large that for all $n \geq N_1$, $q(n) \geq 2$.

Choose M and N_2 sufficiently large that for all $n \geq N_2$,

$$2n(1/\sqrt{n})^M \leq 1/3q(n).$$

This is possible because the left side is $O(1/n^{M/2-1})$. Now choose T and N_3 sufficiently large that for all $n \geq N_3$,

$$(2M\log_2 q(n)/\sqrt{n})^T \leq 1/3q(n).$$

This is possible because $2M\log_2 q(n)/\sqrt{n}$ is $O(\log_2 n/\sqrt{n})$.

Fix some n such that $n > \max\{4, N_1, N_2, N_3, M, T\}$ and consider A with inputs n and $s = TM$, and assume that the answers to equivalence queries of A are determined using the strategy described in Section 4.2. Recall that the size of each hypothesis in $M(n,T,M)$ is TM.

We claim that for all $r \leq q(n)$, after A has made r queries and been answered according to the adversary strategy, there are at least

$$(1 - r/3q(n))|M(n,T,M)| \geq (2/3)|M(n,T,M)|$$

hypotheses in $M(n,T,M)$ that are consistent with all the answers given by the adversary strategy to this point.

Certainly when $r = 0$, this is true, since no answers have been given, and all the hypotheses in $M(n,T,M)$ are consistent with the answers that have been given. Assuming that at least $(1 - r/3q(n))|M(n,T,M)|$ hypotheses from $M(n,T,M)$ are consistent with the answers given to the first $r < q(n)$ queries, then at least one hypothesis remains in $M(n,T,M)$ that is consistent with the answers, so at the next query A must propose a DNF formula ϕ with $q \leq p(n,TM) \leq q(n)$ terms, since $TM < n^2$.

If the counterexample a comes from case (1) of the strategy, then no further hypotheses are eliminated from $M(n,T,M)$ by this counterexample. If the counterexample a comes from case (2), then the fraction of elements of $M(n,T,M)$ that are inconsistent with the counterexample is

$$f_1[n,T,M](a) \leq 2T(1/\sqrt{n})^M \leq 2n(1/\sqrt{n})^M \leq 1/3q(n).$$

If the counterexample a comes from case (3), then the fraction of elements of $M(n,T,M)$ that are inconsistent with the counterexample is

$$f_0[n,T,M](a) \leq (2M \log_2 q(n)/\sqrt{n})^T \leq 1/3q(n).$$

In either of these two cases, a fraction of at most $1/3q(n)$ of the elements of $M(n,T,M)$ are eliminated by the counterexample, so at least

$$(1 - (r+1)/3q(n))|M(n,T,M)|$$

elements in $M(n,T,M)$ are consistent with the first $r+1$ counterexamples.

If A halts after making no more than $q(n)$ equivalence queries then at least $2/3$ of the elements in $M(n,T,M)$ are consistent with all the answers to the equivalence queries. Since $n > T$ and $n > M$, by Lemma 7 there are at least two logically inequivalent hypotheses in $M(n,T,M)$ that are consistent with all the counterexamples to this point, so A must be incorrect for at least one of them. Since A runs in at most $p(n,s) = p(n,TM) \leq q(n)$ steps, it cannot make more than $q(n)$ equivalence queries in identifying any element of $M(n,T,M)$. This contradiction shows that no such A exists. Q.E.D.

5 Comments

Dual results hold, of course, for CNF formulas. This result shows that a polynomial time algorithm for CNF formulas using only equivalence queries can be correct for all k-CNF formulas (CNF formulas with at most k literals per clause) for only a finite number of values of k. This is the behavior achieved by the equivalence query algorithm for k-CNF formulas in [3], adapted from the k-CNF algorithm given by Valiant [6].

Note that this result does not imply anything directly about whether DNF or CNF formulas can be pac-identified in polynomial time, which is an open problem. Kearns and Valiant [5] have shown that for general boolean formulas, a polynomial time prediction algorithm that does slightly better than chance can be used to give random polynomial time algorithms for various apparently hard cryptographic problems.

In [3] we describe a polynomial time algorithm that exactly identifies monotone DNF formulas using equivalence queries and membership queries. The present result shows that membership queries are essential to that result. The question of whether there is a polynomial time algorithm that exactly identifies DNF or CNF formulas using equivalence queries and membership queries is open, even if we consider only Horn form CNF formulas [1,4].

The proofs could be further tuned, possibly to get sharp bounds on the required size of M as a function of the bounding polynomial $p(n, s)$.

6 Acknowledgements

This research was funded by the National Science Foundation, under grant number IRI-8718975.

References

[1] D. Angluin. *Learning propositional Horn sentences with hints.* Technical Report, Yale University, YALEU/DCS/RR-590, 1987.

[2] D. Angluin. *Negative results for equivalence queries.* Technical Report, Yale University, YALEU/DCS/RR-648, 1988.

[3] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1987. Preliminary version appeared as YALEU/DCS/RR-479.

[4] D. Angluin. *Requests for hints that return no hints.* Technical Report, Yale University, YALEU/DCS/RR-647, 1988.

[5] M. Kearns and L. Valiant. *Learning boolean formulae or finite automata is as hard as factoring.* Technical Report, Harvard University Center for Research in Computing Technology, TR-14-88, 1988.

[6] L. G. Valiant. A theory of the learnable. *C. ACM*, 27:1134–1142, 1984.