Yale University
Department of Computer Science

Learning a Circuit by Injecting Values

Dana Angluin[1] James Aspnes[2] Jiang Chen [3]
Yinghua Wu[4]
Department of Computer Science
Yale University

YALEU/DCS/TR-1341
December, 2005

[1]Email: angluin@cs.yale.edu
[2]Email: aspnes@cs.yale.edu. Supported in part by NSF grants CNS-0305258 and CNS-0435201.
[3]Email: criver@cs.yale.edu
[4]Email: y.wu@yale.edu. Supported by NSF grant CNS-0305258.

Abstract

We propose a new model for exact learning of acyclic circuits using experiments in which chosen values may be assigned to an arbitrary subset of wires internal to the circuit, but only the value of the circuit's single output wire may be observed. We give polynomial time algorithms to learn (1) arbitrary circuits with logarithmic depth and constant fan-in and (2) Boolean circuits of constant depth and unbounded fan-in over AND, OR, and NOT gates. Thus, both **AC0** and **NC1** circuits are learnable in polynomial time in this model. Negative results show that some restrictions on depth, fan-in and gate types are necessary: exponentially many experiments are required to learn AND/OR circuits of unbounded depth and fan-in; it is NP-hard to learn AND/OR circuits of unbounded depth and fan-in 2; and it is NP-hard to learn circuits of bounded depth and unbounded fan-in over AND, OR, and threshold gates, even when the target circuit is known to contain at most one threshold gate and that threshold gate has threshold 2. We also consider the effect of adding an oracle for behavioral equivalence. In this case there are polynomial-time algorithms to learn arbitrary circuits of constant fan-in and unbounded depth and to learn Boolean circuits with arbitrary fan-in and unbounded depth over AND, OR, and NOT gates. A corollary is that these two classes are PAC-learnable if experiments are available. Finally, we consider an extension of the model called the synchronous model. We show that an even more general class of circuits are learnable in this model. In particular, we are able to learn circuits with cycles.

1 Introduction

We introduce a new model of active learning for acyclic circuits in which we may inject chosen values on an arbitrary subset of wires but can observe only the value of the circuit's output wire. Our results illuminate the relative importance of manipulation and observation in discovering the structure of networks modeled as circuits.

Gene regulatory networks are an important area in which Boolean network models have been used. In one variant of the basic model, each node in a finite network represents a gene, which has a current state of active or inactive. The states of all nodes in the network are updated synchronously; for each node there is a Boolean function giving its new state in terms of the current states of some subset of the other nodes. A key point is that the node states are *fully observable*: it is assumed that gene expression data gives the state of every node in the network at every time step. The discovery problem is to learn the updating functions of all the nodes (note that one needs to learn both the set of inputs and the functionality of each node). Of course this is difficult if the updating function may be an arbitrary Boolean function; further assumptions generally restrict the fan-in or types of the possible updating functions. One of the main difficulties in this problem is to discover the topology of the network (which nodes are inputs to which node).

Akutsu et al. [1] describe an approach to the discovery problem that models the experimental capability of multiple gene disruption and overexpression. At each time step several selected genes may be disrupted (put in the inactive state), several other selected genes may be overexpressed (put in the active state), while unaffected genes are updated as usual. In this model the states of the nodes are *fully controllable* as well as fully observable. For networks of N nodes and fan-in bounded by k, Akutsu et al. give an $O(N^{2k})$ algorithm for the discovery task. Ideker, Thorsson, and Karp [7] also consider this model and give more practical discovery methods for acyclic networks, using information theoretic criteria to select genes to disrupt or overexpress. These results show that if the class of updating functions is sufficiently restricted, the problem of learning the structure of a network in this model is tractable.

By contrast, there is ample evidence that learning Boolean circuits or formulas from their input-output behaviors may be computationally intractable. Positive learnability results include those for fairly limited classes, including propositional Horn formulas [2] general read once Boolean formulas [3], and decision trees [5], and those for specific distributions, including **AC0** circuits [12], DNF formulas [8] and **AC0** circuits with a limited number of majority gates [9]. (Note that algorithms in both papers [12] and [9] for learning **AC0** circuits and their variants run only in quasi-polynomial time.) Valiant gives cryptographic evidence for the difficulty of PAC learning general Boolean circuits [16]. Kearns and Valiant [10] show that specific cryptographic assumptions imply that **NC1** circuits and **TC0** circuits are not PAC learnable in polynomial time. These negative results have been strengthened to the setting of PAC learning with membership queries [4], even with respect to the uniform distribution [11].

For these results on learning circuits and formulas, observation and control are both restricted: values on internal wires cannot be observed or manipulated. A natural question is: *What are the relative contributions of full observation and full control to the tractability of learning Boolean networks?*

Our new model addresses this question: we postulate full control and restricted observation. Our results show that the ability to inject values into the circuit gives the learner considerable power, but not as much as would be the case with full observation. In particular, with value injection queries, **NC1** circuits and **AC0** circuits are exactly learnable in polynomial time, but our negative results show that the depth limitations are necessary. However, if behavioral equivalence queries are also available, the depth limitations can be removed, which also implies the polynomial time PAC-learnability of these classes with value injection queries.

In the other direction, Rivest and Sloan [14] propose an interesting model of hierarchical learning of Boolean formulas and acyclic circuits, in which a teacher teaches the circuit one gate at a time to the learner in an order consistent with the graph of the circuit; examples at each stage are drawn from a fixed initial distribution. This increases the observability of the values on internal wires, but not does not provide for their control. Rivest and Sloan give a polynomial time algorithm that successfully learns arbitrary acyclic Boolean circuits in their model.

We also consider an extension of our model called *the synchronous model*, in which we assume that the circuit runs in discrete time and gates are synchronized. The circuits are allowed to have cycles in this extension. In this model, we show that an even larger class of circuits are learnable with experiments only.

2 The Model

2.1 Circuits

We define a variant of the usual circuit model that has no distinguished inputs and permits a finite set Σ of at least two different values on wires. A *circuit* C consists of N *wires*, $W = \{w_1, w_2, \ldots, w_N\}$, and for each wire w_i a *gate* g_i that determines the value on this wire. The *size* of the circuit is N. The wire w_N is assumed to provide the output of the circuit as a whole. A *gate* consists of a function mapping Σ^k to Σ, and a vector of k integers from $[1, N]$ specifying the *input wires* of the gate. The value k is the *fan-in* of the gate. Gates of fan-in zero compute constant functions. The maximum fan-in taken over all the gates in the circuit is the *fan-in* of the circuit. We define the *circuit graph* to have a node for each wire/gate pair and a directed edge from node i to node j if w_i is one of the input wires to gate j. Until Section 7, we assume that the graph of the circuit is acyclic. We define the *depth* of the circuit to be the number of edges in the longest path in the circuit graph.

2.2 Behavior

We focus on the behavior of a circuit in response to experiments in which we fix the values of certain wires and observe the final output of the circuit. Define an *experiment* to be a vector s in $(\Sigma \cup \{*\})^N$, where s_i specifies the value of w_i (if it is in Σ) or leaves the value of w_i unaltered (if it is $*$). If $s_i \in \Sigma$, we say w_i is *fixed* in s; otherwise, it is *free* in s. The value of w_i given s, written $w_i(s)$, is defined as

$$w_i(s) = \begin{cases} g_i(w_{i_1}(s), w_{i_2}(s), \ldots, w_{i_{k_i}}(s)) & \text{if } s_i = *, \\ s_i & \text{if } s_i \neq *. \end{cases} \quad (1)$$

where gate i has function g_i and inputs $(i_1, i_2, \ldots, i_{k_i})$. We remark that gates of fan-in zero, which compute constant functions, give the base cases for the above recursive definition. The output of the circuit given an experiment s is the output of wire w_N, that is, $w_N(s)$; this is also denoted $C(s)$.

The *behavior* of a circuit is the function mapping experiments s to $C(s)$. Two circuits C and C' are *behaviorally equivalent*, if they have the same behavior, that is, if $\forall s \in (\Sigma \cup \{*\})^N, C(s) = C'(s)$. To compare our work with previous work on learning circuits, we treat the gates of fan-in zero as the *input gates* and denote the number of input gates by n. An experiment is *input-only* if it fixes every input gate and leaves every other gate free. The *input-output behavior* of a circuit C is the restriction of its behavior function to input-only experiments. Clearly behavioral equivalence implies equality of input-output behaviors but not conversely. Behavioral equivalence is more constrained by the internal structure of the circuits.

An important special case is *Boolean circuits*, for which $\Sigma = \{0, 1\}$. The input-output behavior of a Boolean circuit is just the usual concept of a circuit computing a Boolean function of n inputs. **NC1** circuits are Boolean circuits with constant fan-in and depth $O(\log n)$. **AC0** circuits are Boolean circuits of constant depth and polynomial size whose gates are unbounded fan-in AND, OR, and NOT. The *threshold function* Θ_t is the Boolean function that is 1 if and only if at least t of its inputs are 1. **TC0** circuits are Boolean circuits of constant depth and polynomial size whose gates are unbounded fan-in threshold, AND, OR and NOT.

2.3 Queries

We assume that the learner can get information about the circuit by specifying an experiment s and observing $C(s)$, the output of the circuit. Such an action is termed a *value injection query*, abbreviated VIQ. We also define a *behavioral equivalence query*, abbreviated BEQ: the learner proposes a circuit C', and, if it is not behaviorally equivalent to the target circuit C, receives in response a *counterexample*, that is, an arbitrarily chosen experiment s such that $C'(s) \neq C(s)$. To be consistent with previous usage, we use the term *membership query*, for a VIQ restricted to an input-only experiment s, although $C(s)$ may be non-binary, and the term *equivalence query*, for a query that tests whether the proposed circuit C' has the same input/output behavior as the target circuit C and returns an arbitrary input-only experiment s witnessing $C'(s) \neq C(s)$ if not. Thus, membership queries and equivalence queries necessarily refer to the input/output behavior of the circuit. An algorithm that learns the (full) behavior of any circuit from a given class using VIQ's and BEQ's also learns the input/output behavior of any circuit from the class using VIQ's and equivalence queries. Then a standard polynomial time transformation yields a PAC learning algorithm using VIQ's for input/output behavior of circuits in the class, which implies the following.

Proposition 1. *If a class of circuits is learnable in polynomial time with VIQ's and BEQ's, then the input/output behaviors of circuits in the class are PAC-learnable in polynomial time using VIQ's.*

2.4 The Problem

The learning problems we address are: by making VIQ's (respectively, VIQ's and BEQ's) to a target circuit C, find a behaviorally equivalent circuit C'. Both C and C' use gates from a specified class $\mathcal{F}$.

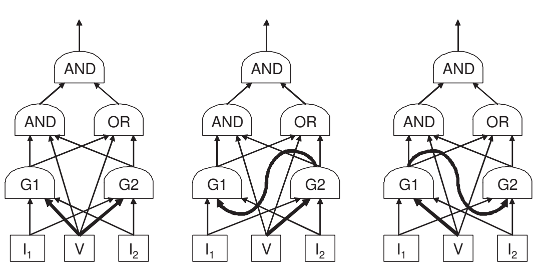

Figure 1: Three equivalent circuits

We remark that it is not possible to discover the exact structure of the target circuit, even when all wires are relevant. The following example shows that the same behavior may be exhibited by structurally distinct circuits. The three circuits C_1, C_2, and C_3 shown in Figure 1 are behaviorally

Table 1: Summary of our results for acyclic circuits

Depth	**Fan-in**	**Gates**	**Query types**	**Learnability**	**Reference**
Unbounded	Unbounded	AND/OR	VIQ	$2^{\Omega(N)}$ queries	Theorem 2
Unbounded	2	AND/OR	VIQ	NP-hard	Theorem 3
Constant	Unbounded	AND/OR/Θ_2	VIQ/BEQ	NP-hard	Theorem 6
Logarithmic	Constant	Arbitrary	VIQ	Poly-time	Theorem 19
Constant	Unbounded	AND/OR/NOT	VIQ	Poly-time	Theorem 21
Unbounded	Constant	Arbitrary	VIQ/BEQ	Poly-time	Theorem 22
Unbounded	Unbounded	AND/OR/NOT	VIQ/BEQ	Poly-time	Theorem 25

equivalent, where G_1 and G_2 are arbitrary gate functions. Only when G_2 and V both have value 1 (respectively, 0) can the value of G_1 propagate through the depth 1 AND gate (respectively, OR gate). Therefore we cannot decide which one of G_2 and V is an input of G_1, and similarly in the other case.

3 Result summary

We investigate the computational tractability of these problems for classes of circuits defined by restrictions on depth, fan-in, and $\mathcal{F}$. Our results are summarized in Table 1.

Section 4 contains negative results for exact learning with VIQ's and BEQ's. In Section 5 we give an algorithm, CircuitBuilder, that takes a class of gates $\mathcal{F}$ and a set U of experiments and constructs a circuit C' by making VIQ's on experiments in U and their one-symbol perturbations, and then finding a gate for each wire consistent with the results. If U contains for every wire and every gate that is wrong for that wire a witness experiment that excludes the incorrect gate, then the resulting circuit is behaviorally equivalent to the target circuit. We then show how to construct appropriate sets of experiments U for the class of log-depth constant fan-in circuits and for the class of **AC0** circuits. In Section 6, we extend these methods to use BEQ's as well as VIQ's, and show that the limitations on circuit depth can be removed for both classes.

Finally, in Section 7, we study the synchronous model. We show that the class of circuits with gates that are closed under two natural operations *projection* and *blurring*, which are defined in Section 7, are learnable in this model. This includes any circuits with constant fan-in gates and AND/OR gates with unbounded fan-in. There is no limitation on circuit depth and we do not restrict circuits to be acyclic.

3.1 Learnability of the gates

What is the relationship between the learnability of circuits in our model and learnability of the class $\mathcal{F}$ of permitted gates? A depth 1 circuit consists of n input gates and one gate g depending on some subset of the inputs; if any nontrivial circuits are to be learnable, then depth 1 circuits must be learnable in the same sense. Therefore, we assume that the class of permitted gates $\mathcal{F}$ is learnable.

For depth 1 circuits, a VIQ reduces to a membership query. Classes $\mathcal{F}$ of gates for which depth 1 circuits are learnable in polynomial with membership queries include (1) the class of gates with fan-in at most some constant k over an arbitrary finite value set Σ and (2) the class of all symmetric Boolean gates (which includes unbounded fan-in AND, OR, NAND, NOR, threshold and parity gates.)

Another aspect of the learnability of depth 1 circuits is the *consistency problem*, defined as follows. The input is a set E of *prohibited pairs* (s, σ) where s is an input-only experiment (recall that an input-only experiment fixes all input gates), $\sigma \in \Sigma$ is a value, and the desired output is a depth 1 circuit C' over $\mathcal{F}$ such that $C'(s) \neq \sigma$ for every pair $(s, \sigma) \in E$. For Boolean circuits, because $C'(s) \neq 0$ implies $C'(s) = 1$ and similarly $C'(s) \neq 1$ implies $C'(s) = 0$, the consistency problem can be stated as finding C' that agrees with given values of experiments in E. We remark that the consistency problem is a computational problem and therefore each s should fix all input gates for the problem to be well defined. One of the major differences between the consistency problem and the problem of learning with membership queries is that in the consistency problem no information is provided beyond the set E and hence one may not be able to query Hamming neighbors of an experiment. For the class of arbitrary gates with fan-in at most k, the consistency problem can be solved in time $O(n^k)$. There is also a polynomial time algorithm to solve the consistency problem over the class of unbounded fan-in AND, OR, NAND, NOR, NOT and parities. However, Lemma 5 shows that the consistency problem is NP-hard over the class of unbounded fan-in AND, OR, and thresholds. CircuitBuilder makes use of algorithms for the consistency problem. Lemma 4 shows that polynomial time learnability with VIQ's and BEQ's implies a polynomial time algorithm for consistency in certain cases.

3.2 Relation to circuit testing

A central challenge for learning algorithms in our model is to propagate the effects of a changed value on some internal wire to the observable output of the circuit. Our methods are similar in some respects to the idea of *path sensitization* in circuit testing, used to detect whether the output of some gate is "stuck" at a constant value instead of computing the correct value of its inputs [6]. In path sensitization, a path in the correct circuit from a gate g to the output is constructed, and an input x is sought such that the output of g would be the complement of the "stuck" value, and the values of the other inputs to gates along the chosen path are set so as to propagate the output of g (or its complement) to the output of the circuit. Path sensitization is not a complete method: there are examples of single stuck-at faults in acyclic circuits that cannot be detected by path sensitization, though they are detectable by other tests. In our model, the ability to inject values on internal wires gives the approach greater power. However, this power does not trivialize the problem; the negative results in Section 4 illustrate the subtle "shadowing" or "filtering" effects that limit the power of VIQ's.

Fujiwara [6] considers the computational problem of deciding, for a given circuit, gate, and stuck-at value, whether there is any test to detect the fault. He shows that the problem is NP-complete, even when restricted to AND/OR circuits of depth three. By contrast, since this class is contained in **AC0**, we show that it is polynomial time learnable using VIQ's.

4 What Cannot be Learned Efficiently?

Figure 2 presents a *gadget*, which is an AND/OR circuit of fan-in 2 that computes $Y = \Theta_2(X, V, W)$. We can view V and W as controlling a switch: only when their values are different will the value of X be passed on to Y. Gadgets can be concatenated to get a gadget chain (see Figure 3), in which the value of X is passed on to Y only when every pair V_i and W_i have different values. The chain can be used to "hide" part of the circuit unless the values of V_i and W_i are complements of each other. In Figure 3, exactly one of each pair V_i and W_i is an input to the big AND gate. The learner has to guess which combination of them are the inputs to the AND gate, which yields the following negative result.

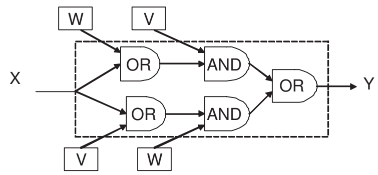

Figure 2: The gadget

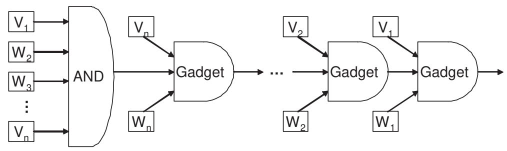

Figure 3: A hidden AND gate and the gadget chain

Theorem 2. *Learning the class of acyclic Boolean AND/OR circuits requires $2^{\Omega(N)}$ VIQ's.*

Proof. Suppose an adversary reveals the gadget chain and the fact that the last gate is an AND gate with exactly one of each pair V_i and W_i as an input, but hides the exact combination. When there exists a pair V_i and W_i that are both set 0 or 1, the output of the circuit is determined by the gadget chain. In particular, it is determined by the pair with smallest index that are both set 0 and 1 (the pairs is ordered according to their distances to the output gate as in Figure 3). The adversary answers 0 if the pair are set 0 and 1 if the pair are set 1.

Only when for every pair, V_i and W_i are set differently, will the value of the big AND gate affect the output of the circuit. There are 2^n such settings of V's and W's. The adversary answers 0 until only one setting remains. The theorem then follows because $N = O(n)$. □

Theorem 2 gives an exponential information-theoretic lower bound, using a deep circuit and a gate of large fan-in. The following construction uses the gadget chain to give a computational hardness result for deep circuits with fan-in 2.

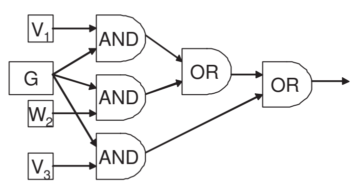

Figure 4: $G \wedge (x_1 \vee \overline{x_2} \vee x_3)$

Theorem 3. *Learning the class of fan-in 2 AND/OR circuits using VIQ's is NP-hard.*

Proof. We use the implicit negation enforced by the gadget chain to construct a circuit representing a CNF formula, for which a satisfying instance must be found in order to expose a hidden part of the circuit.[1] We associate a Boolean variable x_i with each pair V_i and W_i, and let $x_i = 1$ if $V_i = 1, W_i = 0$ and 0 if $V_i = 0, W_i = 1$ (we only deal with the case that V_i and W_i are set differently thanks to the gadget chain). Then the circuit in Figure 4 computes $G \wedge (x_1 \vee \overline{x_2} \vee x_3)$. We can chain such clauses by replacing G with the output of the succeeding clause and finally connect the output of the first clause to the gadget chain shown in Figure 3 (note that the big AND gate is

[1]Using a NOT gate would not achieve the same effect because we can override its output in an experiment.

not part of the gadget chain). Let g be an AND gate of fan-in 2 with different inputs than V's and W's. We connect the output of g to the last clause of the clause chain (by replacing G of the last clause by the output of g). In order to learn the circuit, we have to be able to observe the output of g because we are not able to distinguish between this circuit and the circuit with g being replaced by an OR gate if we can not observe g's output. In this construction, the functionality of g matters if there exists an satisfying assignment to all clauses.

As in the proof of Theorem 2, when there exists a pair V_i and W_i that are both set to 0 and 1, the output of the circuit is determined by the gadget chain regardless of other parts of the circuit. Suppose every pair of V_i and W_i are set differently. In order to observe g's output, we must compute an assignment to V's and W's that satisfies all clauses in the clause chain, since otherwise the circuit output will simply be 0. In other words, we have to solve the 3-SAT problem. Because the gadget chain consists of AND/OR gates of fan-in 2, the theorem follows. $\square$

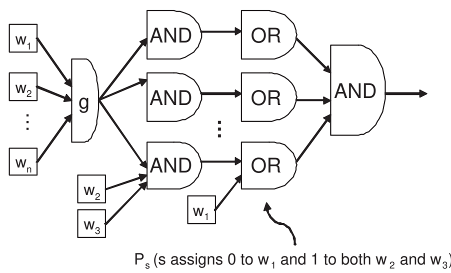

Figure 5: The "filtering" circuit and an illustration of a filtering path P_s with three input wires.

The gadget chain is a deep circuit, but in the following construction we use AND and OR gates to achieve a constant depth filter that forces a learning algorithm to solve the consistency problem for $\mathcal{F}$ (defined in Section 3.1). Given (1) any depth 1 circuit with input wires $w_1, w_2, \ldots, w_n$ and gate g from a class $\mathcal{F}$ of Boolean gates and (2) a set E of input-only experiments, we add the following structure to construct another circuit C as follows (see Figure 5). For each $s \in E$, which assign each w_i to 0 or 1, we add a distinct directed path P_s of length 3 consisting of g, a new AND gate, a new OR gate, and the output gate. Let each wire w_i that is set 0 in s be an input of the OR gate of P_s. Let each wire w_i that is set 1 in s be an input of the AND gate of P_s. The construction has the property that if the assignment to $w_1, w_2, \ldots, w_n$ is not s, either the output of the AND gate or the output of the OR gate in P_s is determined, and hence the output of g can not be passed through the path P_s. Therefore, P_s "filters" out all assignments in E except s. Finally, we take the output gate of the whole circuit to be an AND gate.[2]

Lemma 4. *There is a polynomial time algorithm using VIQ's and BEQ's to learn circuits from the above class if and only if there is a polynomial time algorithm for the consistency problem over* $\mathcal{F}$.

Proof. Suppose the consistency problem is solvable in polynomial time. The structure of a target circuit C from the class, (except for the identity of gate g) can be learned using VIQ's in a straightforward fashion, which determines the corresponding set E. For each $s \in E$ (which corresponds to a path), we construct an experiment s' that agrees with s on $w_1, w_2, \ldots, w_n$, and sets the output gate and the OR gate and the AND gate in P_s free, sets all the other OR gates 1. It is clear that $C(s') = g(s)$. Therefore, we can learn the value $g(s)$ for each $s \in E$ and learn g using the algorithm

[2]Note that we cannot use the same method to replace the gadget chain because it would require $|E|$ to be exponential.

that solves the consistency problem. Let g' be the function learned. g and g' are equivalent on *experiments* of E. Because of the filtering structure of C, the only experiments in which the value of g matters are those that assign $w_1, w_2, \ldots, w_n$ in the exactly same way as some $s \in E$ does. Therefore, we conclude that the learned circuit is equivalent to C.

Conversely, suppose that there is a polynomial time learning algorithm. We can construct a circuit that corresponds to the consistency problem and simulate VIQ's by evaluating the circuit. We simulate BEQ's only when the proposed gate for g does not solve the consistency problem since otherwise the learning algorithm solves the consistency problem and we are done. Denote the proposed gate by g'. Let C' be the proposed circuit. We show that whenever g' does not solve the consistency problem, we can find a counterexample to C'. Therefore, the learning algorithm must propose a gate that solves the consistency problem.

There are two cases. When the inputs of g' are contained in the set $\{w_1, w_2, \ldots, w_n\}$, there must exist an $s \in E$ such that $g'(s) \neq g(s)$ as g' does not solve the consistency problem. In fact, we can find such an s in polynomial time by simulating a membership query algorithm on g' (recall that we assume that gates are learnable with membership queries). Among all the queries the algorithm makes, a query must be made to distinguish g and g' and the query can serve as s. Suppose w.l.o.g. that $g(s) = 1$ and $g'(s) = 0$. We construct s' as before. We give s' as a counterexample if $C(s') \neq C'(s')$. Otherwise, we construct two experiments s'_0 and s'_1 that set in s' the corresponding wire of g to 0 and 1 respectively. By our construction, we know that $C(s'_0) = 0$ and $C(s'_1) = 1$. However, we have that $C(s') = C(s'_1)$ (since $g(s) = 1$) and that $C'(s') = C'(s'_0)$ (since $g'(s) = 0$). Therefore $C'(s'_0) = C'(s') = C(s') = C(s'_1) = 1 \neq C(s'_0)$. Thus s'_0 is a counterexample to C'.

If g' depends on an AND/OR gate h that lies in a path P_s for some $s \in E$, construct s', s'_0 and s'_1 as before. Let w be the corresponding wire of g and w' be the corresponding wire of h. Suppose that C and C' agree on all three experiments. Compare s'_0 and s'_1. In the proposed circuit C', w' must take the same value on s'_0 and s'_1, because the fact that g' depends on w' and that C' is acyclic implies that in C', w' is not reachable from w and therefore changing the value on w will not affect the value on w' (see also Proposition 9). However, by construction, in C, w' takes value 0 on s'_0 and value 1 on s'_1, which means that, on either of s'_0 and s'_1, w' takes different values in C and C'. Suppose w.l.o.g. it is s'_0. Then for s'_0, w takes value 0 in C and value 1 in C'. Set w' to 1 in s'_0 and let the resulting experiment be s'_2. We have that $C'(s'_2) = C'(s'_0) = C(s'_0) = 0$ but $C(s'_2) = 1$. Thus, s'_2 is a counterexample to C'. ☐

Lemma 5. *The consistency problem is NP-hard for the class of unbounded fan-in AND, OR, and Θ_2 gates.*

Proof. We reduce a 3-SAT instance ϕ over the variables x_i for $i \in [1, n]$ to the consistency problem for this class. The input wires are $\{I_1, I_2, I_3, V_1, W_1, V_2, W_2, \ldots, V_n, W_n\}$. Let G denote the output wire of the unknown gate. We define a correspondence between literals of ϕ and wires: literal x_i corresponds to wire V_i literal $\overline{x_i}$ corresponds to wire W_i. We design the set of experiments and their outputs as follows, so as to constrain the unknown gate to be a Θ_2 gate with its inputs corresponding to a satisfying assignment of ϕ.

- For each of the eight experiments assigning 0 and 1 to I_1, I_2, and I_3, with all $V_i = W_i = 0$, the output value is $\Theta_2(I_1, I_2, I_3)$. This guarantees that G cannot be AND or OR, and must therefore be a Θ_2 gate whose inputs include I_1, I_2, and I_3.

- For each i, on the experiment with $I_1 = V_i = W_i = 1$ and all other input wires assigned 0, the output value is 1. This implies at least one of V_i and W_i is an input wire of G.

- For each i, on the experiment with $V_i = W_i = 1$ and all other input wires assigned 0, the output value is 0. This implies not both V_i and W_i are inputs to G.
- For each clause of ϕ, on the experiment that sets I_1 and the three wires corresponding to the three literals in the clause to 1 and all other wires to 0, the output is 1. This ensures that at least one wire corresponding to a literal in the clause is an input to G.

It is easily verified that ϕ is satisfiable if and only if there is a gate G from the specified class of gate functions consistent with these experiment/value pairs. $\square$

Lemma 4 and Lemma 5 establish the following theorem.

Theorem 6. *Learning constant depth AND/OR/Θ_2 circuits with VIQ's and BEQ's is NP-hard.*

5 Learning with experiments

In this section, we give algorithms for arbitrary circuits with logarithmic depth and constant fan-in and Boolean circuits of constant depth and unbounded fan-in over AND, OR and NOT gates. Therefore, we show that both **AC0** and **NC1** circuits are learnable in polynomial time with VIQ's. One of the main issues is to learn the set of inputs for each gate. We remark that the gates in the circuit that the algorithms output may not have the same set of inputs as the target circuit (see Figure 1).

First we develop some tools. A *partial experiment* is a partial function from $[1, N]$ to $\Sigma \cup \{*\}$, where some wires are unspecified. Let s be an experiment and τ be a partial experiment. Define $s|_\tau$ to be the experiment obtained by replacing in s the settings of all wires that are specified in τ by the corresponding settings in τ. Let s and t be two experiments. We say that $t \preceq s$ if the set of free wires in t is a subset of the set of free wires in s. We say $t \prec s$ if $t \preceq s$ and there is at least one free wire in s that is fixed in t. Let s be an experiment with wire w set free. We call $s|_{w=\sigma}$, where $\sigma \in \Sigma$, the (w, σ)*-perturbation* of s. If $C(s) \neq C(s|_{w=\sigma})$, we say s is (w, σ)*-exposing*.

Consider any gate g with inputs $(i_1, i_2, \ldots, i_l)$. We overload g to take an experiment s as an argument. That is, let $g(s) = g(w_{i_1}(s), w_{i_2}(s), \ldots, w_{i_l}(s))$, where $w_i(s)$ is the value of wire w_i on s in the target circuit C. The following useful facts are easily verified.

Proposition 7. *Let s and t be two experiments with the output wire set free. If s and t agree on every wire that is either free in s or an input to a wire that is free in s then $C(s) = C(t)$.*

Proposition 8. $C(s) = C(s|_{w=w(s)})$.
(This is meaningful only when w is set free in s. In this case, $w(s)$ is the value the corresponding gate computes. The proposition simply says that if we fix w to the value it takes on an experiment s, the circuit output stays the same.)

Proposition 9. *Let w and u be two wires and suppose there is no path from w to u in the graph of the circuit. Then $u(s) = u(s|_{w=\sigma})$ for any experiment s and $\sigma \in \Sigma$.*

The main task of our learning algorithms is to find a "correct" gate function for each wire. Formally speaking, a gate g is *wrong* for a wire w, if there exists an experiment s that fixes all of g's inputs and is $(w, g(s))$-exposing. We call such an s a *witness* experiment for g and w. Otherwise, we say that g is *correct* for w.

Lemma 10. *Let C' be a circuit with the same set of wires as C. If C' is acyclic and every gate of C' is correct for the corresponding wire in C, C' is behaviorally equivalent to C.*

Proof. Suppose to the contrary that C' is not behaviorally equivalent to C. Let s be a minimal experiment such that $C'(s) \neq C(s)$. Let w be a free wire in experiment s and g be its corresponding gate in C', chosen so that all g's inputs are fixed in s (such a wire exists because C' is acyclic and the input gates of C' are considered to have fixed inputs because they have no inputs). By Proposition 8, we have $C'(s) = C'(s|_{w=g(s)})$. By the minimality of s, we have $C'(s|_{w=g(s)}) = C(s|_{w=g(s)})$, which then implies that $C(s|_{w=g(s)}) \neq C(s)$. This contradicts the fact that g is correct for w. □

5.1 Constructing a circuit

Let $\mathcal{F}$ be a class of gates containing all the gates in the target circuit C. We describe an important subroutine, *CircuitBuilder* (Algorithm 1), that takes a set of experiments U and constructs an acyclic circuit C' using gates from $\mathcal{F}$. CircuitBuilder builds C' from the bottom up, starting with gates of fan-in zero. At each iteration, CircuitBuilder attempts to add another wire to C' by choosing a gate in $\mathcal{F}$ among those that depend only on wires that are already in C'. This method has the advantage of building an acyclic circuit, which is crucial because the dependence between gates is not always clear, as in Figure 1.

Define U to be a *sufficient set of tests* for C and $\mathcal{F}$ if for every wire w_i in C and every gate $g \in \mathcal{F}$ that is wrong for w_i, U contains at least one witness for g and w_i. In the remainder of this section we prove the following.

Theorem 11. *If U is a sufficient set of tests for C and $\mathcal{F}$ then C' is behaviorally equivalent to C, where C' is the circuit constructed by CircuitBuilder. Moreover, the queries can be made non-adaptively.*

Let U_w denote the set of experiments in U with w set free. In CircuitBuilder, since before we replace s by $s|_{w=g(s)}$, we check whether $C(s) = C(s|_{w=g(s)})$ for s in U_w, we do not need to make queries on the replacing experiments in U. However, we may have to make queries on the perturbations of replacing experiments. This would require the algorithm to make queries adaptively. Instead, in CircuitBuilder, we maintain another set of experiments V which contains all possible perturbations of experiments in U at the beginning of the algorithm. Similarly, let V_w denote the set of experiments in V with w set free. In Lemma 13, we will show that after replacement, V still contains all necessary perturbations of experiments in U. In Lemma 14, we show that even in V, a replacing experiment will have the same circuit output as the original one. Therefore, we only need to make queries on $U \cup V$ at the beginning of the algorithm. The algorithm is non-adaptive.

At each iteration of the algorithm, experiments in $U \cup V$ may be replaced. We make the following claims about the replacements.

Lemma 12. *At any iteration, for any $s \in U \cup V$, no wire in C' is set free in s.*

Proof. This is because we fix the wire to a value whenever we add it to C'. □

The following lemma says that if $s \in U$ and $t \in V$ are an experiment and perturbation pair, and w is the corresponding wire, they will continue to be such a pair until w is added to C'.

Lemma 13. *Consider any iteration, any $w \in Z$ and any $s \in U_w$, and let s_0 be the version of s at the start of the algorithm. For any $\sigma \in \Sigma$, let t_0 be the (w, σ)-perturbation of s_0 at the start of the algorithm and t be the replacement of t_0 at the iteration considered. Then t is a (w, σ)-perturbation of s.*

Algorithm 1 CircuitBuilder

INPUT: U and $\mathcal{F}$.

OUTPUT: C'.

$\forall w$, $\forall s \in U_w$, $\forall \sigma \in \Sigma$, let V contain the (w, σ)-perturbation of s.
Make a VIQ on every experiment $s \in U \cup V$.
$C' \leftarrow \emptyset$. $Z \leftarrow W$.
while Z is not empty **do**
  for $w \in Z$ **do**
    if there exists a function $g \in \mathcal{F}$ that depends only on wires in C', such that $\forall s \in U_w, C(s) = C(s|_{w=g(s)})$ **then**
      Add w and g to C' and remove w from Z.
      $\forall s \in U_w \cup V_w$, replace s by $s|_{w=g(s)}$.
      break

Proof. The statement is clearly true at the start of the algorithm. At each previous iteration, at most one setting of s and t will be changed. We only need to show that each replacement will replace the same value for s and t. The lemma then follows by observing that the replaced value is an output of a function that depends on wires that do not include w (w has not been added to C') and that s and t differ only at their settings of w. □

Together with Lemma 13, the following lemma shows that although we need to compare the circuit outputs of replacement experiments and their perturbations, we do not need to make any further VIQ's.

Lemma 14. *Suppose U is a sufficient set of tests for C and $\mathcal{F}$. At any iteration consider any $s \in U \cup V$ and let s_0 be the version of s at the start of the algorithm. Then we have $C(s) = C(s_0)$.*

If $s \in U$, there is nothing to prove since the algorithm checks the equality before making the replacement. The case that $s \in V$ is a little bit trickier. We prove an even more general lemma, from which the case $s \in V$ follows.

Lemma 15. *Suppose U is a sufficient set of tests for C and $\mathcal{F}$. Let g be the function that the algorithm chooses for gate w. Then g is correct for w. That is, $\forall s$ with g's input wires fixed, $C(s) = C(s|_{w=g(s)})$.*

Proof. W.l.o.g., let w be the first wire added to C' for which the statement in the lemma does not hold. Suppose g is wrong for w. By the assumption that U is sufficient for C and $\mathcal{F}$, there exists an experiment $s_0 \in U$ at the start of the algorithm such that s_0 fixes all g's inputs, and $C(s_0) \neq C(s_0|_{w=g(s_0)})$. Let $s \in U$ be the replacement of s_0 at the iteration w is added to C'. We have that $C(s) = C(s_0) \neq C(s_0|_{w=g(s_0)}) = C(s|_{w=g(s_0)})$, by the assumption that w is the first wire violating the condition. Moreover, $g(s) = g(s_0)$ because s_0 and s both fixes all g's input wires and therefore should agree on them. Therefore, we have

$$C(s) \neq C(s|_{w=g(s)})$$

which contradicts the choice of g. □

Lemma 15 together with Lemma 10 show that if U is a sufficient set, C' is behaviorally equivalent to C. Lemma 13 and 14 show that all the queries can be made at the beginning of the algorithm, which establishes Theorem 11. Lemma 12 validates the operation of picking a function g, which amounts to solving the following consistency problem (defined in Section 3.1). Let E be the projection of U_w to C' (note that all wires in C' are fixed) and let the prohibited pairs (t, σ) be those $t \in E$ and $\sigma \in \Sigma$ such that there is an experiment in U_w that agrees with t and is (w, σ)-exposing.

The next lemma shows that the algorithm terminates in N iterations.

Lemma 16. *At each iteration, the algorithm adds one wire to C'.*

Proof. First we observe that there is at least one wire w in Z whose input wires are all contained in C', because the circuit graph of C is acyclic. The true gate of w in C will survive every *if-test* in the algorithm. □

5.2 Test paths

One of the key ideas at the heart of our algorithms is to use test paths. A *test path* is an experiment whose free wires are a directed path from some wire w to w_N, through which w is exposed. Let a *side wire* of a test path be a fixed wire that is an input to a gate whose corresponding wire is set free. The meaning of test paths is made clear in the following lemma, which says that using test paths is sufficient.

Lemma 17. *Let s be a (w, σ)-exposing experiment, where $\sigma \in \Sigma$. Let $s^* \preceq s$ be a minimal (w, σ)-exposing experiment. Then the free wires in s^* are a directed path in the graph of C, which starts with w and ends with the output wire w_N. (s^* is a test path.)*

Proof. When $w = w_N$, the directed path is just w_N itself. Suppose the claim is true for any free wire whose corresponding gate has w as an input. First we claim that only those wires that w can reach (in the underlying digraph) can be free in s^*. This is because these wires take the same values in s^* and the perturbations $s^*|_{w=\sigma}$ (see Proposition 9). Thus, we can set them to the corresponding values and the resulting experiment is still (w, σ)-exposing, which contradicts the minimality of s^*.

Let u be a free wire in s^* whose only free input wire is w. u must exist, because the underlying digraph is acyclic. Let $\sigma_0 = w(s^*)$ and $\beta_0 = u(s^*)$ and $\beta = u(s^*|_{w=\sigma})$. We claim that $s^*|_{w=\sigma_0}$ is a minimal (u, β)-exposing experiment. Let us view the circuit as a function of values of w and u. That is, let $F(x, y) = C(s^*|_{w=x,u=y})$. By the assumption, we have $F(\sigma_0, \beta_0) \neq F(\sigma, \beta)$. By the minimality of s^*, we have $F(\sigma_0, \beta) = F(\sigma, \beta)$. Thus, we have

$$F(\sigma_0, \beta) \neq F(\sigma_0, \beta_0)$$

which implies that $s^*|_{w=\sigma_0}$ is (u, β)-exposing. Suppose on the contrary that there exists $s' \prec s^*$ is (u, β)-exposing. We set both w and u free in s' and let the resulting experiment be s''. Let $F''(x, y) = C(s''|w = x, u = y)$. Again, by the assumption and the minimality of s^*, we have

$$F''(\sigma_0, \beta_0) \neq F''(\sigma_0, \beta) = F''(\sigma, \beta)$$

Therefore, we conclude s'' is (w, σ)-exposing by observing that $w(s'') = \sigma_0$, $u(s'') = \beta_0$ and $u(s''|_{w=\sigma}) = \beta$, which leads to a contradiction.

Therefore, we conclude that $s^*|_{w=\sigma_0}$ is a minimal (u, β)-exposing experiment. By induction, its free wires consist of a directed path starting with u and ending with w_N. We append w to this path to obtain the directed path in s^*. □

5.3 Learning log depth circuits with constant fan-in

We use CircuitBuilder to give an algorithm that learns an arbitrary log depth bounded fan-in circuit. The algorithm does not perform any additional queries and hence is non-adaptive. The main idea is based on the observation that in an acyclic circuit of depth d and fan-in k, there are at most d free wires and at most kd side wires in a test path. There are at most $|\Sigma|^{O(kd)}$ settings of these wires. If we randomly assign one of $\Sigma \cup \{*\}$ to each wire with equal probabilities, the probability that we generate one of the settings is $1/|\Sigma|^{O(kd)}$. We can generate all settings using $|\Sigma|^{O(kd)} \log \frac{1}{\delta}$ random experiments, which succeeds with probability at least $1 - \delta$. We can also generate them deterministically using a universal set construction. The following definition of universal set is adapted from Seroussi and Bshouty [15]. An experiment set U is called (N, l)-*universal* if for every set of indices $R = \{r_1, r_2, \ldots, r_l\} \subseteq [N]$, the projection of U to R contains all $(|\Sigma| + 1)^l$ l-tuples. It is shown in [13] that a (N, l)-universal set of size $2^{O(l \log |\Sigma|)} \log N$ can be constructed in polynomial time.[3] Let U be a $(N, (d + 1)(k + 1))$-universal set and let $\mathcal{F}$ be the class of all gates of fan-in at most k. We show that U is sufficient for C and $\mathcal{F}$.

Lemma 18. *U is a sufficient set for C and $\mathcal{F}$.*

Proof. Let s be a witness experiment that g is wrong for w; all g's inputs are fixed in s. Let $s^* \preceq s$ be a minimal $(w, g(s))$-exposing experiment. According to Lemma 17, there are at most d free wires and dk side wires in s. Since U is a universal set, there exists an experiment $s_0 \in U$ at the beginning of the algorithm, such that s_0 agrees with s^* in all s^*'s free wires and side wires and also all g's inputs (there are at most $(d + 1)(k + 1)$ wires). Proposition 7 shows that s_0 is a witness experiment that g is wrong for w. □

Whenever $kd = O(\log N)$, the size of U is polynomial in N and so is that of V. It takes time $N^{O(k)}$ to solve the consistency problem for a function of k variables, by checking every possible combination of k inputs. When k is a constant, this is polynomial.

Theorem 19. *Log depth bounded fan-in circuits can be learned non-adaptively in polynomial time using VIQ's.*

5.4 Learning AC0 circuits

Theorem 6 precludes polynomial time algorithms for learning constant depth unbounded fan-in circuits with fairly simple gates. In this section, we show that if we allow only AND and OR gates (it is easy to extend it to NAND, NOR and NOT), constant depth unbounded fan-in Boolean circuits are learnable. Thus we show that **AC0** circuits are learnable with VIQ's.

We are not able to use a universal set, since k can be as large as $\Omega(N)$. Instead, we use Algorithm 2 to gather the necessary test paths adaptively. Algorithm 2 begins with learning the output gate, which can be easily done for AND and OR gates. It then sets one of its input wires free and fixes the other input wires so that the free input wire is still relevant. In particular, it sets the other input wires to 1 if the output gate is an AND gate, or 0 if the output gate is an OR gate. This partial experiment is then used to find (some of) the inputs of the corresponding gate. The algorithm goes on exploring the whole circuit. We remark that Algorithm 2 alone is not sufficient, because some input wires may be fixed as side wires and therefore hidden to the learner.

Let U contain all tests that are made in Algorithm 2 and $\mathcal{F}$ be all AND and OR gates. The following lemma shows the correctness of the learning algorithm.

[3] The paper [13] only deals with binary vectors. But it can be easily extended to the non-binary case by viewing each non-binary literal in $\Sigma \cup \{*\}$ as a binary vector of size $\log(|\Sigma| + 1)$.

Algorithm 2 Gathering test paths for a constant depth AND/OR circuit

Let Γ contain the partial experiment that sets the output wire free, $\{w_N = *\}$.
Let $\mathbf{1}$ ($\mathbf{0}$) be an experiment that sets all wires to 1 (0),
while Γ is not empty **do**
  Pick $\tau \in \Gamma$ and remove it from Γ.
  if $C(\mathbf{1}|_\tau) \neq C(\mathbf{0}|_\tau)$ **then**
    Let $Z = \{w|\ \ w$ unspecified in τ, and $C(\mathbf{1}|_{\tau,w=0}) \neq C(\mathbf{1}|_\tau)$ or $C(\mathbf{0}|_{\tau,w=1}) \neq C(\mathbf{0}|_\tau)$ $\}$.
    for $w \in Z$ **do**
      if $C(\mathbf{1}|_{\tau,w=0}) \neq C(\mathbf{1}|_\tau)$ **then**
        Add $\tau|_{w=*,\forall w' \in Z\setminus\{w\}, w'=1}$ to Γ. $\{$*AND gate*$\}$
      else if $C(\mathbf{0}|_{\tau,w=1}) \neq C(\mathbf{0}|_\tau)$ **then**
        Add $\tau|_{w=*,\forall w' \in Z\setminus\{w\}, w'=0}$ to Γ. $\{$*OR gate*$\}$

Lemma 20. *U is sufficient for C and $\mathcal{F}$.*

Proof. Suppose s is a witness experiment that g is wrong for w and $s^* \preceq s$ is a minimal $(w, g(s))$-exposing experiment. Let u be the successor of w in the directed path from w to w_N (Lemma 17). We define two partial experiments τ_u and τ_w. τ_u sets all free wires in the directed path before u and their side wires as in s^* and sets u free. τ_w is similarly defined. We claim that τ_w is added to Γ in Algorithm 2. We assume inductively τ_u has been added to Γ.

Compare τ_u and τ_w. Those wires unspecified by τ_u but specified by τ_w are side wires that are inputs only to u. They are set to 1 if u is an AND gate and 0 if u is an OR gate so as to keep w relevant. Furthermore, we observe that

1. If u is an AND gate, $C(\mathbf{1}|_{\tau_u}) \neq C(\mathbf{0}|_{\tau_u})$ and $C(\mathbf{1}|_{\tau_u,w=0}) \neq C(\mathbf{1}|_{\tau_u})$;

2. If u is an OR gate, $C(\mathbf{1}|_{\tau_u}) \neq C(\mathbf{0}|_{\tau_u})$ and $C(\mathbf{0}|_{\tau_u,w=1}) \neq C(\mathbf{0}|_{\tau_u})$.

Therefore, τ_w must be added. Thus U must contain the following sets of experiments $\{\mathbf{0}|_{\tau_w}, \mathbf{1}|_{\tau_w}\}$, $\{\mathbf{1}|_{\tau_w,w'=0}|\ w'$ unspecified in $\tau_w\}$, and $\{\mathbf{0}|_{\tau_w,w'=1}|\ w'$ unspecified in $\tau_w\}$. Let g^* be the gate w in the target circuit C. The two projected functions $g|_{\tau_w}$ and $g^*|_{\tau_w}$ (fixing some inputs of the functions) must be different, because otherwise it contradicts the fact that s^* is a witness experiment. By case analysis, we can show that there exists at least one of the above-mentioned experiments s_0, such that $g(s_0) \neq g^*(s_0)$. This s_0 serves our purpose. □

It is clear that each partial experiment collected by Algorithm 2 corresponds to a directed path in the circuit C. Thus the number of partial experiments is bounded by $O(N^d) = poly(N)$ when the depth d is a constant. The size of U and the number of tests are hence polynomially bounded. The theorem then follows from the fact that the consistency problem for AND/OR gates can be solved in polynomial time.

Theorem 21. *AC0 circuits are learnable in polynomial time using VIQ's.*

6 Learning with experiments and counterexamples

BEQ's overcome the obstacles of Theorem 2 and Theorem 3, because the counterexample has to give away the combination when an appropriate hypothesis circuit is presented. However, the result in Theorem 6 still applies. Assuming both VIQ's and BEQ's are available, we give polynomial time algorithms to learn both arbitrary constant fan-in circuits and AND/OR circuits with unbounded depth.

Both algorithms repeatedly make a BEQ on a candidate circuit C' until C' is behaviorally equivalent to the target circuit. Each counterexample s is processed to give a minimal counterexample $s^* \preceq s$ such that $C'(s^*) \neq C(s^*)$. This process, *Minimize*, can easily be done with $O(N)$ VIQ's. The minimal counterexample is then used in rebuilding the candidate circuit C'. As in the proof of Lemma 10, a minimal counterexample is a witness experiment that a candidate gate g is wrong for a wire w. Therefore, each counterexample eliminates at least one candidate gate for at least one wire. For constant fan-in circuits, this immediately leads to a learning algorithm, in which we use a refinement of CircuitBuilder, which instead of checking each gate with respect to U, just picks a gate that is not eliminated for the corresponding wire. The algorithm is polynomial because there are at most $|\mathcal{F}| = N^{O(k)}$ gates to eliminate.

Theorem 22. *Bounded fan-in circuits are learnable in polynomial time using VIQ's and BEQ's.*

However, the same method does not work for AND/OR circuits, as $|\mathcal{F}|$ is exponential. But each minimal counterexample still proves a gate wrong for a wire w. Let us denote each counterexample by a pair indicating the outputs of the true gate and the proposed gate. A $(1, 0)$ counterexample eliminates the constant 0 gate and similarly a $(0, 1)$ counterexample eliminates the constant 1 gate. Counterexamples for AND/OR gates can be divided into two types, namely, *input removing* and *input demanding* counterexamples. For instance, if the proposed gate and the true gate are both AND gates, a $(1, 0)$ counterexample says that the *0-inputs* (inputs that are set 0) of the proposed gate are not inputs of the true gate. Such a counterexample causes the removal of the 0-inputs from the potential input set and is called input removing. Let $R_w^{\wedge}$ contain all inputs removed by input removing counterexamples for wire w.

A $(0, 1)$ counterexample implies that inputs of the proposed gate does not include all inputs of the true gate. Such a counterexample is called input demanding. Let $T_w^{\wedge}$ be a collection of sets of wires. We add the set of inputs of the proposed gate to $T_w^{\wedge}$ when an input demanding counterexample is received. Any AND gate whose inputs are completely contained in any set of $T_w^{\wedge}$ can not be the true gate and is therefore eliminated. We can view $T_w^{\wedge}$ as constraints on candidate gates for wire w. An analogous argument applies when both gates are OR gates. Let $R_w^{\vee}$ be analogous to $R_w^{\wedge}$ and $T_w^{\vee}$ be analogous to $T_w^{\wedge}$.

When the true gate and the proposed gate are of different types, we will process each counterexample as if they were the same type. It can be verified that even when the two gates are of different types, each counterexample will either add some wires to $R_w^{\wedge}$ or $R_w^{\vee}$ or add a set to $T_w^{\wedge}$ or $T_w^{\vee}$ and therefore make some progress. For example, if the true gate is an OR gate and the proposed gate is an AND gate, we treat the counterexample as if the true gate was an AND gate. Thus, we will add 0-inputs of the proposed gate to $R_w^{\wedge}$ upon receiving a $(1, 0)$ counterexample, and add the whole set of inputs of the proposed gate to $T_w^{\wedge}$ upon receiving a $(0, 1)$ counterexample. An important fact is that *the true gate will never be eliminated*. This is because only when the proposed gate is of the same type as the true gate will there be any restriction on the true gate, and the true gate will not be eliminated in these cases as we have already discussed.

We use Algorithm 3 to build the proposed circuit C'. At each iteration, we try to add a wire in Z to C'. We put Z in a queue and let the initial order be $w_1, w_2, \ldots, w_N$. At each iteration, the first wire in the queue will be considered. If it is not added to C', it will be put at the end of the queue. We add the wire to C' only if one of the following is true. One of the two constant functions is not eliminated; $C' \backslash R_w^{\wedge}$ is not contained in any set of $T_w^{\wedge}$; $C' \backslash R_w^{\vee}$ is not contained in any set of $T_w^{\vee}$. We add the wire with a constant function, AND of wires in $C' \backslash R_w^{\wedge}$, or OR of wires in $C' \backslash R_w^{\vee}$, respectively.

Algorithm 3 Building the proposed circuit

INPUT: $\forall w \in W$, $R_w^\wedge, T_w^\wedge, R_w^\vee, T_w^\vee$ and the set of constant functions $\mathcal{C}_w$ for wire w that are not eliminated.

OUTPUT: C'.

```
C' ← ∅. Z ← (w_1, w_2, ..., w_N).
while Z is not empty do
  Pop the first wire w in Z.
  if C_w ≠ ∅ then
    Add w to C' with one of the functions in C_w.
  else if ∀t ∈ T_w^∧, C'\R_w^∧ ⊈ t then
    Add w to C' with AND of wires in C'\R_w^∧.
  else if ∀t ∈ T_w^∨, C'\R_w^∨ ⊈ t then
    Add w to C' with OR of wires in C'\R_w^∨.
  else
    Put w at the end of Z.
```

Now we bound the number of counterexamples each wire can receive. There are at most 2 counterexamples that eliminate constant functions. There are at most $O(N)$ input removing counterexamples. The most subtle case is input demanding counterexamples. We identify the phase number of the learning algorithm with the number of counterexamples it has received (recall that the algorithm rebuilds C' each time a counterexample is received). In Algorithm 3, let *the round number of an iteration* be the number of times the wire being considered has been added to the queue. Let $I_w(t)$ be the round number of the iteration that w is added to C' at phase t. We will show that $I_w(t)$ will never decrease in the following. The intuition is that we add more constraints on w as we receive more counterexamples.

Let $C'_{(w,i)}(t)$ be the set C' when w is considered at round i in phase t. If we order pairs in $W \times [1, N]$ first by their round number and then by the order of wires in W, we have that $C'_{(w,i)}(t) = \{w' | (w', I_{w'}(t)) \leq (w, i)\}$. Together with R_w's and T_w's, $C'_{(w,i)}(t)$ decides whether w can be added to the circuit at round i in phase t. In the following, we show that $C'_{(w,i)}(t)$ never gets bigger. In other words, the set of wires C' that are available when w is considered to be added at round i is a subset of the corresponding set of wires in the previous phase. Therefore, if w is not added at round i in the previous phase, it is likely that it will not be added at round i at this phase. The key observation in the reasoning is that if a set C' cannot pass the test with $R_w^\wedge$ and $T_w^\wedge$, and $R_w^\vee$ and $T_w^\vee$,

1. no subset of C' can pass the test;
2. C' cannot pass the test if we add more wires to $R_w^\wedge$ and $R_w^\vee$ or more sets to $T_w^\wedge$ and $T_w^\vee$.

Lemma 23. *For any wire w, if $C'_{(w,i)}(t)$ exists and the algorithm does not end at phase t, $C'_{(w,i)}(t+1)$ exists and $C'_{(w,i)}(t+1) \subseteq C'_{(w,i)}(t)$.*

Proof. We do induction on the pairs (w, i). The lemma clearly holds for $(w_1, 1)$ as $C'_{(w_1,1)}(t)$ is always empty. Suppose it holds for all pairs that precede (w, i). Suppose there exists a wire w' in $C'_{(w,i)}(t+1) \backslash C'_{(w,i)}(t)$. We have that $(w', j = I_{w'}(t+1)) \leq (w, i)$. Therefore, w' is added at the j^{th} round at phase $t+1$ but after the j^{th} round at phase t. This implies (using the observations 1 and 2) that $C'_{(w',j)}(t+1) \nsubseteq C'_{(w',j)}(t)$, because $R_w^\wedge$ and $R_w^\vee$, and $T_w^\wedge$ and $T_w^\vee$ only grow when more counterexamples are received. This leads to a contradiction. $\square$

It follows (also using the observations 1 and 2) that

Corollary 24. *$I_w(t)$ is non-decreasing.*

Now let us bound $I_w(t)$. Recall that the true gate will never be eliminated. Therefore, whenever C' contains all inputs of the true gate, w will be added. Thus, if the true gate of w is a constant gate, $I_w(t) = 1$. If the true gate of w depends only on constant gates, $I_w(t) \leq 2$. In general, we have that $I_w(t) \leq \max(I_{w_{i_1}}(t), I_{w_{i_2}}(t), \ldots, I_{w_{i_k}}(t)) + 1$, where $w_{i_j} (j \in [1, k])$ is an input of a true gate of w. Therefore, $I_w(t) \leq d$.

Each input demanding counterexample for wire w at phase $t + 1$ will eliminate the gate that Algorithm 3 picked for w at round $I_w(t)$ in phase t, by adding all inputs of the gate to $T_w^{\wedge}$ or $T_w^{\vee}$. By Lemma 23 and the observations 1 and 2, this means that at phase $t+1$, we cannot pick gates of the same type again for w at or before round $I_w(t)$. Thus $I_w(t)$ will increase when w receives at most 2 input demanding counterexamples, one for AND gates and the other for OR gates. Therefore, we can bound the number of input demanding counterexamples by $O(d)$ per wire.

Theorem 25. *AND/OR/NOT circuits with unbounded fan-in and unbounded depth are learnable in polynomial time using VIQ's and BEQ's.*

7 Learning synchronous circuits with cycles

In this section, we study a new learning model called *the synchronous model*, where time is quantized and we can inject values as well as observe the output of the circuit at each time step. The value of each wire at time $t + 1$ is determined either by the injected value at time $t + 1$ or by its inputs at time t if it is set free. An *experiment sequence* is a series of experiments. We require the first experiment in an experiment sequence to fix all wires for the sake of completeness. Two circuits are *equivalent* in the synchronous model, if and only if they have the same set of wires, and produce the same output sequence given the same experiment sequence. We allow circuits to have cycles because the synchronous model provides more power to the learner by allowing the injection of values at each time step. Roughly speaking, it allows us to isolate and examine individual gates to some extent. In this section, we will show that any circuit over a large class of gates, including constant fan-in gates and AND/OR gates, are learnable in this new model.

Let $g(u_1, u_2, \ldots, u_k)$ be a function from Σ^k to Σ. We define two operations. $\forall \sigma \in \Sigma, i \in [1, k]$, let $g|_{u_i=\sigma} = g(u_1, \ldots, u_i = \sigma, \ldots, u_k)$ be a *projection* of g. Let $P \subseteq 2^{\Sigma}$ be a partition of Σ. Let p from Σ to P be the corresponding partition function, i.e., p maps a symbol to the set that contains it in P. Let $p(g(u_1, u_2, \ldots, u_k))$ be a *blurring* of g. The reason to introduce blurrings is that some values of a gate may be indistinguishable by gates on the way to the output gate. Thus the learner is only able to see a "blurred" version. We remark that a blurring is only interesting when g is non-binary or $|\Sigma| > 2$, since when $|\Sigma| = 2$, the only possible blurring of g is a constant function.

In the synchronous model, because the capability of isolating and examine individual gates, we do not face the consistency problem. As a consequence, a large class of circuits are learnable in this model. Let $\mathcal{G}$ be the collection of classes of gates that are closed under projection and blurring and learnable with membership queries in polynomial time. The main theorem of this section is the following.

Theorem 26. *Circuits (which may have cycles) with gates from a class in $\mathcal{G}$ are learnable in polynomial time in the synchronous model.*

The class of constant fan-in gates and AND/OR gates certainly belongs to $\mathcal{G}$. Therefore, the theorem applies to these gates. In the following, we assume all gates are from a class in $\mathcal{G}$ and hence there is a learning algorithm with membership queries that learns this class of gates.

The main idea is natural. An experiment s^0 that fixes all wires followed by an experiment s^1 that sets w_N free can be used to simulate a membership query on the output gate, as we can observe the value $g_N(s_0)$ after s^1 is applied. In general, given an experiment sequence $\xi = (s^1, s^2, \ldots, s^m)$, we can simulate a membership query s^0 on a gate g by inserting s^0 at the beginning of the experiment sequence, and setting the corresponding wire w free in s^1, i.e., we test the experiment sequence $(s^0, s^1|_{w=*}, s^2, \ldots, s^m)$. However, we can not observe $g(s^0)$ if the gate is not the output gate. But as usual, we can compare the circuit outputs of an experiment sequence and its perturbations. Let $\xi|_\tau = (s^1|_\tau, s^2, \ldots, s^m)$, where τ is a partial experiment. In general, $\xi|_\tau$ may not be an experiment sequence since its first experiment may not fix all wires. We can compare the circuit outputs of $(s^0, \xi|_{w=*})$ and $\xi|_{w=\beta}$ for some $\beta \in \Sigma$ to determine whether $g(s^0) = \beta$. In fact, we are only able to decide that $g(s^0) \neq \beta$ if the circuit outputs are different. More formally, we say an experiment sequence ξ is (w, α, β)-*exposing* if the circuit outputs at the end are different given two experiment sequences, $\xi|_{w=\alpha}$ and $\xi|_{w=\alpha}$. We are only able to distinguish $\alpha = g(s^0)$ and β if the experiment sequence is (w, α, β)-exposing. Therefore, we may only be able to distinguish $g(s^0)$ from some of the symbols in Σ and may not be able to determine $g(s^0)$. If we partition Σ such that α and β is put in the same partition if and only if ξ is not (w, α, β)-exposing, and let g^ξ be the blurring of g under this partition, we can simulate a membership query on g^ξ by comparing the circuit outputs of $(s^0, \xi|_{w=*})$ with $\xi|_{w=\beta}$, $\forall \beta \in \Sigma$.

Algorithm 4 starts with an arbitrary length 1 experiment sequence (s^1), which is surely an exposing experiment sequence for the output gate w_N and any pair of distinct symbols. The learning algorithm can be used to learn the output gate. Then we extend the exposing experiment sequence to the input wires of the output gate. Let w be an input wire of the output gate. If there exists s^0 such that $g_N(s^0|_{w=\alpha}) \neq g_N(s^0|_{w=\beta})$, we can extend the experiment sequence to $(s^0, s^1|_{w_N=*})$, which is a (w, α, β)-exposing experiment sequence. We then learn the gate function of w and extend $(s^0, s^1|_{w_N=*})$, and go on collecting all exposing experiment sequences by extending known exposing experiment sequences.

In general, given an experiment sequence $\xi = (s^1, s^2, \ldots, s^m)$ and wire u, we resort to the learning algorithm to compute a s^0 such that $(s^0, \xi|_{u=*})$ is (w, α, β)-exposing, if there exists one. Consider the two projections $g^\xi|_{w=\alpha}$ and $g^\xi|_{w=\beta}$. s^0 does not exist if the two projections are equivalent. Otherwise, we simulate the learning algorithm to learn $g^\xi|_{w=\alpha}$. One query must be made to distinguish the two projections as they both belong to the class of gates by assumption. That is, the learning algorithm must make a query on a s^0 such that $g^\xi|_{w=\alpha}(s^0) \neq g^\xi|_{w=\beta}(s^0)$. This s^0 serves our purpose.

We claim that Algorithm 4 collects all exposing experiment sequences.

Lemma 27. *For any wire w, and $\alpha, \beta \in \Sigma$, if there exists a (w, α, β)-exposing experiment sequence, there exists one in V.*

Proof. Suppose $\xi = (s^1, s^2, \ldots, s^m)$ is a (w, α, β)-exposing experiment sequence, but there is no (w, α, β)-exposing experiment sequence in V. Assume further that it is the minimum such exposing experiment sequence and that there is no corresponding exposing experiment sequence in V with the same $W \times \Sigma \times \Sigma$ tuple.

Consider the two experiment sequences $\xi|_{w=\alpha}$ and $\xi|_{w=\beta}$. Let the j^{th} step of a sequence be the time after we applied j experiments and before we apply the $(j+1)^{th}$ one. At the first step, only wire w takes different values in the two sequences, since it is the only wire that is assigned differently in the first experiments.

Algorithm 4 Learning a synchronous circuit

OUTPUT: $C' = \{(w_i, g'_i)\}$.

Let ξ_0 be an arbitrary length 1 experiment. Let V be a database of experiment sequences, with the keys being tuples in $W \times \Sigma \times \Sigma$. Add ξ_0 to V with the key (w_N, α, β) for any $\alpha, \beta \in \Sigma$ and $\alpha \neq \beta$.

while there exists a unmarked experiment sequence ξ with key (u, α', β') in V **do**

  Mark ξ.

  If g is the true gate of wire u, simulate the membership query algorithm to learn g^{ξ}.

  for $w \in W$, $\alpha, \beta \in \Sigma$ **do**

    if the key (w, α, β) is not in V **then**

      Using the simulated learning algorithm to compute s^0 such that $(s^0, \xi|_{u=*})$ is (w, α, β)-exposing or decide that s^0 does not exist.

      if s^0 exists **then**

        Add (s^0, ξ) to V with the key (w, α, β).

$\forall i \in [1, N]$, and $\forall s^0$ that fixes all wires, let $g'_i(s^0) = \sigma$, where $\sigma \in \bigcap_{\xi \in V} g_i^{\xi}(s^0)$.

Consider the second step (after we apply s^2). Suppose wire w_i takes values α_i in $\xi|_{w=\alpha}$ and β_i in $\xi|_{w=\beta}$ and $\alpha_i \neq \beta_i$. We claim that there is no (w_i, α_i, β_i)-exposing sequence in V. We know that w_i must be set free in s^2 and $g_i(s^1|_{w=\alpha}) = \alpha_i \neq \beta_i = g_i(s^1|_{w=\beta})$. If there exists a (w_i, α_i, β_i)-exposing sequence, Algorithm 4 would have extended it to a (w, α, β)-exposing sequence.

By the minimality assumption, there does not exist any (w_i, α_i, β_i)-exposing sequence of length $m-1$. Now let s_α^j (respectively s_β^j) be an experiment that fixes wires to their corresponding values at the j^{th} step in $\xi|_{w=\alpha}$ (respectively $\xi|_{w=\beta}$). Since there does not exist any (w_i, α_i, β_i)-exposing sequence of length $m-1$, we conclude that $(s_\alpha^2, s_3, \dots, s_m)$ has the same circuit outputs as $(s_\alpha^2|_{w_1=\beta_1}, s_3, \dots, s_m)$, as well as $(s_\alpha^2|_{w_1=\beta_1, w_2=\beta_2}, s_3, \dots, s_m)$, $\dots$, $(s_\beta^2, s_3, \dots, s_m)$. This is a contradiction, because at the last step, $(s_\alpha^2, s_3, \dots, s_m)$ has the same circuit output as $\xi|_{w=\alpha}$, and $(s_\beta^2, s_3, \dots, s_m)$ has the same circuit output as $\xi|_{w=\beta}$, whereas $\xi|_{w=\alpha}$ and $\xi|_{w=\beta}$ have different circuit outputs. $\square$

After collecting all exposing experiment sequences and learning blurrings of gates, Algorithm 4 outputs a synthesis function of all blurrings for each gate. Since the number of keys in V is bounded by $\Sigma^2 N$, the running time of Algorithm 4 is polynomial. We now show that the constructed circuit C' is equivalent to C in the synchronous model.

Proof. (of Theorem 26) We first relate a circuit in synchronous model with an acyclic circuit in our previous model. Let m be a number. Given a circuit C in the synchronous model, we create a circuit $\tilde{C}$ with m copies of each wire in C. Each copy is indexed by a distinct number in $[1, m]$ and each number corresponds to a layer in $\tilde{C}$. The gate function of each copy i ($i > 1$) is the same as the original wire, except that its inputs are $(i-1)^{th}$ copies of inputs of the original gate. The gates in first layer are constant gates. For any experiment sequence ξ of length m, if we view the ξ as an experiment for $\tilde{C}$ and identify the m layers with the m experiments in ξ, and assuming that we can observe all m copies of w_N, $\tilde{C}$ simulates C on ξ. We assign m^{th} copy of w_N be the output wire of $\tilde{C}$.

Suppose on the contrary C and C' are not equivalent and ξ is an experiment sequence that witnesses their difference. Suppose ξ is such an experiment sequence of the minimum length and $|\xi| = m$. Let $\tilde{C}$ and $\tilde{C}'$ be the corresponding length m acyclic circuits of C and C'. Because ξ has the minimum length, the circuit outputs of C and C' only differ at the last step when applying ξ.

Therefore, we have $\tilde{C}(\xi) \neq \tilde{C}'(\xi)$. Let $\xi^* \preceq \xi$ be the minimal witness experiment for $\tilde{C}$ and $\tilde{C}'$. By the arguments of Lemma 10, ξ^* is a witness experiment for a wire in $\tilde{C}$ which corresponds to a copy of a wire w in C. Let α and β be the two values that the true gate and the corresponding gate in C' output. We observe that this could happen only when there is no (w, α, β)-exposing experiment sequence. However, we can use Lemma 17 to further minimize ξ^* to obtain (w, α, β)-exposing experiment sequence, which leads to a contradiction. □

References

[1] T. Akutsu, S. Kuhara, O. Maruyama, and S. Miyano. Identification of gene regulatory networks by strategic gene disruptions and gene overexpressions. In *SODA '98: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 695–702, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.

[2] D. Angluin, M. Frazier, and L. Pitt. Learning conjunctions of Horn clauses. *Machine Learning*, 9:147–164, 1992.

[3] D. Angluin, L. Hellerstein, and M. Karpinski. Learning read-once formulas with queries. *J. ACM*, 40:185–210, 1993.

[4] D. Angluin and M. Kharitonov. When won't membership queries help? *J. Comput. Syst. Sci.*, 50(2):336–355, 1995.

[5] N. H. Bshouty. Exact learning boolean functions via the monotone theory. *Inf. Comput.*, 123(1):146–153, 1995.

[6] H. Fujiwara. *Logic Testing and Design for Testability.* MIT Press, 1986.

[7] T. Ideker, V. Thorsson, and R. Karp. Discovery of regulatory interactions through perturbation: Inference and experimental design. In *Pacific Symposium on Biocomputing 5*, pages 302–313, 2000.

[8] J. C. Jackson. An efficient membership-query algorithm for learning DNF with respect to the uniform distribution. *J. Comput. Syst. Sci.*, 55(3):414–440, 1997.

[9] J. C. Jackson, A. R. Klivans, and R. A. Servedio. Learnability beyond AC0. In *STOC '02: Proceedings of the thirty-fourth annual ACM symposium on Theory of computing*, pages 776–784, New York, NY, USA, 2002. ACM Press.

[10] M. Kearns and L. Valiant. Cryptographic limitations on learning boolean formulae and finite automata. *J. ACM*, 41(1):67–95, 1994.

[11] M. Kharitonov. Cryptographic hardness of distribution-specific learning. In *STOC '93: Proceedings of the twenty-fifth annual ACM symposium on Theory of computing*, pages 372–381, New York, NY, USA, 1993. ACM Press.

[12] N. Linial, Y. Mansour, and N. Nisan. Constant depth circuits, Fourier transform, and learnability. *Journal of the ACM*, 40(3):607–620, 1993.

[13] J. Naor and M. Naor. Small-bias probability spaces: Efficient constructions and applications. In *ACM Symposium on Theory of Computing*, pages 213–223, 1990.

[14] R. L. Rivest and R. Sloan. A formal model of hierarchical concept learning. *Inf. Comput.*, 114(1):88–114, 1994.

[15] G. Seroussi and N. H. Bshouty. Vector sets for exhaustive testings of logic circuits. In *IEEE Transactions on Information Theory*, volume 34, pages 513–522, May 1988.

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