Requests For Hints That Return No Hints

Dana Angluin*, Yale University
YALEU/DCS/RR-647
September 1988

*Supported by the National Science Foundation, IRI-8718975
Requests For Hints That Return No Hints

Dana Angluin *
Yale University

September 1988

Abstract

We describe a simple algorithm that learns an arbitrary propositional Horn sentence in polynomial time using two types of queries: equivalence queries that return Horn clauses as counter-examples, and "derivation queries" that take a Horn clause and return one of the answers: "not implied", "subsumed", or "implied, but not subsumed". This improves the results in [1] in two respects: Arbitrary, rather than acyclic, propositional Horn sentences are learned, and derivation queries return strictly less information than request for hint queries. However, we argue that the algorithm of [1] is more reasonable in a practical sense.

1 Introduction

We refer the reader to [1] for definitions concerning propositional Horn clauses and sentences. We assume that there is a known set \( V \) of variables, and an unknown propositional Horn sentence \( \phi_* \) over \( V \) to be learned using two types of queries:

1. The input to an *equivalence query* is a propositional Horn sentence over \( V \). The answer is "yes" if \( \phi \) is logically equivalent to \( \phi_* \). Otherwise, the answer is "no" and a Horn clause \( C \) implied by \( \phi_* \) but not by \( \phi \) or vice versa. \( C \) is a *counter-example*.

2. The input to a *derivation query* is a Horn clause \( C \) over \( V \). If \( \phi_* \) does not imply \( C \), the answer is "not implied". If some clause of \( \phi_* \) subsumes \( C \), the answer is "subsumed". If no clause of \( \phi_* \) subsumes \( C \), but \( C \) is implied by \( \phi \), then the answer is "implied, but not subsumed".

Thus, equivalence queries are as in [1], but derivation requests are essentially request for hint queries that do not return a "hint" variable in the third case. The measure \( D(\phi) \) of how far \( \phi \) is from \( \phi_* \) is defined in [1]. The main result is:

**Theorem 1** The learning algorithm NIHL takes as input a propositional Horn sentence \( \phi_0 \) and uses equivalence queries and derivation requests to find a Horn sentence equivalent to an unknown propositional Horn sentence \( \phi_* \). It runs in time bounded by a polynomial in the sizes of \( \phi_0 \) and \( \phi_* \) and the number of variables, \( |V| \), and makes at most \( D(\phi_0) + 1 \) equivalence queries.

*Supported by NSF grant IRI-8718975*
2 The algorithm \textit{NIHL}

The algorithm \textit{NIHL} is based directly on the algorithm \textit{IHL} of [1]. It takes a Horn sentence $\phi$ over $V$ as input.

The procedure \textit{NIHL}($\phi$)

1. Make an equivalence query with $\phi$. If the reply is "yes", output $\phi$ and halt. Otherwise, the reply is a Horn clause $C$ that is a counterexample.

2. If $\phi \vdash C$ then let $C'$ be the clause returned by \textit{NFind-Incorrect}(C), remove $C'$ from $\phi$, and go to step 1.

3. If $\phi \not\vdash C$ then let $C'$ be the clause returned by \textit{NFind-Missing}(C), set $\phi = \phi \land C'$, and go to step 1.

\textbf{NFind-Incorrect.} The procedure \textit{NFind-Incorrect} takes as input a Horn clause $C$ that is implied by $\phi$ and not by $\phi_* \land \phi$ and returns a clause of $\phi$ that is not implied by $\phi_* \land \phi$. It runs in time and number of derivation queries bounded by a polynomial in the size of $\phi$ and the number of variables $|V|$. The method is to find a derivation of $C$ from $\phi$ and then make derivation queries to test each clause $C'$ of $\phi$ used in the derivation until one is found that is not implied by $\phi_* \land \phi$. This is essentially the same as the \textit{Find-Incorrect} procedure of [1], except that the notion of a derivation must be expanded to include non-positive clauses.

\textbf{NReduce.} The subprocedure \textit{NReduce}, takes as input a Horn clause $C$ that is subsumed by some clause of $\phi_* \land \phi$ and not implied by $\phi$, and returns a clause of $\phi_* \land \phi$ that is not implied by $\phi$. The running time and the number of derivation queries used by \textit{NReduce} is bounded by a polynomial in the size of its input clause. The method is a greedy search for a subset $C'$ of the clause $C$ such that some clause of $\phi_* \land \phi$ subsumes $C'$ but this is not true of any clause obtained by dropping one literal from $C'$. Such a $C'$ is actually a clause of $\phi_* \land \phi$, and is clearly not implied by $\phi$.

\textbf{NFind-Missing.} The procedure \textit{NFind-Missing} takes as input a Horn clause $C$ that is implied by $\phi_* \land \phi$, but not by $\phi$, and returns a Horn clause $C'$ that is in $\phi_* \land \phi$ but not implied by $\phi$. It runs in time and number of derivation queries bounded by a polynomial in the size of $\phi$ and the number of variables $|V|$.

\textit{NFind-Missing} does a breadth-first search of the consequences of the antecedents of $C$ with respect to $\phi_*$ to find a Horn clause $C'$ that is subsumed by some clause of $\phi_* \land \phi$ but is not implied by $\phi$. It then returns \textit{NReduce}(C'). This breadth-first search makes the algorithm unwieldy in practice; it is now described in more detail.

If $A$ is a set of variables, for each nonnegative integer $i$ we define $Z_i(A)$ as follows. $Z_0(A) = A$. For each positive integer $i+1$, let $Z_{i+1}(A)$ be the set of all elements $x$ in

$$(V \cup \{\bot\}) - (Z_0(A) \cup \ldots \cup Z_i(A))$$

such that some clause of $\phi_*$ subsumes $(Z_0(A) \cup \ldots \cup Z_i(A) \rightarrow x)$. 

2
If \( x \) is in \( Z_i(A) \) then the shortest derivation of \((A \rightarrow x)\) from \( \phi_* \) takes \( i \) steps. Since \( V \cup \{\bot\} \) is finite, from some point on all the sets \( Z_i(A) \) will be empty. The sets \( Z_i(A) \) can be computed from \( A \) using derivation queries as follows. Assuming \( Z_0(A), \ldots, Z_i(A) \) have been computed, for each \( x \) in

\[(V \cup \{\bot\}) - (Z_0(A) \cup \ldots \cup Z_i(A))\]

make a derivation request with the clause

\[(Z_0(A) \cup \ldots \cup Z_i(A) \rightarrow x).\]

Then \( x \) is in \( Z_{i+1}(A) \) if and only if the reply is “subsumed”. This computation can be done in time and number of derivation queries bounded by a polynomial in \( |V| \).

As the sets \( Z_i(A) \) are generated, \textit{NFind-Missing} checks to see whether the clauses

\[C' = (Z_0(A) \cup \ldots \cup Z_i(A) \rightarrow x)\]

that are derived in one step from \( \phi_* \) are implied by \( \phi \). Once such a \( C' \) is found that is not implied by \( \phi \), \textit{NReduce} is called to reduce it to a clause in \( \phi_* \), which is then returned by \textit{NFind-Missing}.

Suppose \textit{NFind-Missing} is called with a Horn clause \( C = (A \rightarrow z) \) such that \( \phi_* \vdash C \) and \( \phi \not\vdash C \). \((z\) may be \( \bot \) or a variable.\) If any value is returned, then \( C' \) is a clause that is not implied by \( \phi \) and is subsumed by some clause of \( \phi_* \), by the definition of \( Z_{i+1}(A) \), so by the correctness of \textit{NReduce}, the value returned will be a clause of \( \phi_* \) that is not implied by \( \phi \).

To see that the procedure must terminate, note that if for all nonnegative integers \( i \), if every \( x \in Z_{i+1} \) is such that

\[C' = (Z_0(A) \cup \ldots \cup Z_i(A) \rightarrow x)\]

is implied by \( \phi \), then every clause with antecedents \( A \) implied by \( \phi_* \) is also implied by \( \phi \), contrary to the input assumption that \( C = (A \rightarrow z) \) is implied by \( \phi_* \) but not by \( \phi \). Hence an appropriate \( i \) and \( x \) must be found. The running time and the number of derivation queries is bounded by a polynomial in the size of \( \phi \) and the number of variables \( |V| \).

**Proof of Theorem 1.** To see that Theorem 1 is true, it suffices to note that each iteration of the main loop reduces the value of \( D(\phi) \) by at least one, and when \( D(\phi) = 0 \), \( \phi \) must be logically equivalent to \( \phi_* \). Hence, termination with a correct output is guaranteed after at most \( D(\phi_0) \) iterations of the loop, and at most \( D(\phi_0) + 1 \) equivalence queries. The bounds on the time and number of derivation queries of the subprocedures \textit{NFind-Incorrect} and \textit{NFind-Missing} establish the bounds in Theorem 1.

3 Why \textit{NIHL} is more impractical than \textit{IHL}

The chief difference between \textit{IHL} and \textit{NIHL} is in the procedure that takes a clause \( C \) implied by \( \phi_* \) but not by \( \phi \) and returns a clause \( C' \) of \( \phi_* \) that is not implied by \( \phi \). In the case of \textit{IHL}, if the teacher answers requests for hints using one derivation of \( C \) from \( \phi_* \), the number of queries will be bounded by the size of that derivation. In the case of \textit{NIHL}, the queries amount to uncontrolled forward-chaining from the antecedents \( A \) of \( C \) with respect to \( \phi_* \), which does not seem promising. Hence \textit{NIHL} seems to be primarily of theoretical interest.
References