# The Properties and Applications of Horn Clause Proofs

Bruce Spencer
Robin Cohen

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada N2L 3G1

**Research Report CS-89-64**

December, 1989

# The Properties and Applications of Horn Clause Proofs

Bruce Spencer and Robin Cohen
Department of Computer Science
University of Waterloo
Waterloo, Ontario, N2L 3G1
ebspencer@dragon.uwaterloo.ca and rcohen@dragon.uwaterloo.ca

**Abstract**

Reasoning systems based on clausal logic and resolution, commonly used in artificial intelligence research, may restrict the form of the clauses to simplify computation or to guarantee efficient reasoning. (Reasoning from general clauses is almost certainly intractable.) For example SLD resolution, the proof procedure underlying the programming language Prolog, reasons from definite clauses, clauses with exactly one positive literal. In this paper we weaken that restriction, and reason from Horn clauses, clauses with zero or one positive literal. We define HC proofs, which are proofs from Horn clauses, and show that they are sound and complete. Our treatment differs from other Horn clause reasoners in that it is easily implemented with a backtracking search strategy (we provide one), and its queries are unrestricted conjunctions. HC proofs can improve the efficiency of consistency-based non-monotonic reasoning systems when Horn clauses can describe the domain. For example, we have improved the performance of the Theorist system.

## 1 Introduction

Formal reasoning systems that use the clausal form of first order logic without equality are commonly used in artificial intelligence. But the general form of clauses acceptable by each system is often restricted, to simplify the reasoning procedure or to guarantee efficiency. For example SLD resolution, the simple proof procedure underlying the programming language Prolog, reasons from definite clauses. A definite clause is a disjunction of literals, exactly one of which is positive. Given a query which is a conjunction of positive literals, and a program which is a set of definite clauses, SLD resolution derives the instances of the query that are entailed by (in every model of) the program.

We consider a more general case, where the program is a set of Horn clauses and the query is a conjunction of positive and negative literals. A Horn clause is a disjunction of zero or one positive literals and zero or more negative literals. For this case, we define a Horn clause proof of a query, or HC proof for short, that indicates the provable instances of the query. The major result in this paper, stated informally, says that there is an HC proof of a query which indicates some instance is provable if and only if the instance is entailed by the set of clauses.

The case we consider is strictly simpler than the most general case, clauses which are disjunctions of, and queries which are conjunctions of any combination of positive and negative literals. Because our case is simpler, a procedure for building HC proofs involves less work than one that applies to general clauses, such as the MESON proof procedure[6].

Work similar to ours has been presented by Gallier and Raatz[4]. Their clauses are Horn clauses and their queries are disjunctions of negations of Horn clauses. Thus they can show the inconsistency of a set of Horn clauses. Our work is new in that it admits an unrestricted conjunction as a query, so we can show that a set of Horn clauses is inconsistent with a general clause. Also, Gallier and Raatz use a breadth-first search strategy. They encountered complications with backtracking search. Backtracking is important for minimizing the space required by the search procedure. We present a proof procedure for HC proofs that backtracks.

In the next section we define HC proofs of single literals and prove soundness and completeness for the propositional and first order cases. We extend HC proofs to a conjunction of literals. We use HC proofs to add logical negation to Prolog; this furnishes a backtracking procedure for building HC proofs. We compare this work with related literature and point out applications of HC proofs that improve the efficiency of consistency checking in non-monotonic reasoning systems, such as Theorist[8] and the ATMS[2].

## 2 Horn Clause Proofs

Informally, a Horn clause proof tree for a literal $g$ is a tree whose nodes contain instances of literals from the Horn clauses. A parent node contains a literal chosen from some clause, and its children contain the negation of each of the other literals in that clause. A leaf of the completed tree is either an instance of a singleton clause, so it spawns no new children, or an instance of the negation of $g$, the root of the tree.

HC proof trees differ from Prolog proof in two ways. First, any literal in a Horn clause can be put in the parent node. Prolog programs contain only definite clauses, and only instances of the positive literal in a clause can be put in the parent node. As a direct result, Prolog proof tree nodes only contain positive literals. Second, a leaf of a completed tree cannot be the negation of the root, since all nodes are positive.

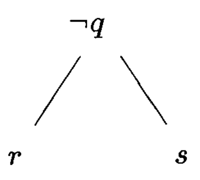
For this example Horn clause program

$$\begin{array}{l} \neg q \vee \neg r \vee \neg s \\ r \vee \neg q \\ s \end{array}$$

and the query $\neg q$, we build the tree as follows. First build a node that contains the literal to be proved:

$$\neg q$$

Next, select a clause which contains $\neg q$. In this example we select $\neg q \vee \neg r \vee \neg s$. Negate the other literals in the clause and put them as children of $\neg q$.



Now build a subtree for each of the children. For $s$ there is a clause that matches it, and it has no other disjuncts, so we have completed this branch. For $r$ we select the clause $r \vee \neg q$, negate $\neg q$ which becomes $q$, and make it a child of $r$.

$$\begin{tikzcd} & & \neg q \arrow[dl, dash] \arrow[dr, dash] & \\ & r \arrow[dl, dash] & & s \\ q & & & \end{tikzcd}$$

Now we apply the rule that has no Prolog counterpart. Since the $q$ is the negation of the topmost node, $\neg q$, we have proved it. There are no remaining nodes to be proved, so we have completed the tree.

It may appear contradictory that in this exercise we have proved both $q$ and $\neg q$. This is a proof by cases. Either $q$ or $\neg q$ is true. By building the proof tree we have established that if $q$ then $\neg q$, but this leads to a contradiction. So we reject $q$, leaving the case $\neg q$.

In the remainder of Section 2 we define HC proofs for propositional and first order logic. We begin with some background.

We assume the reader is familiar with first order logic as presented in, for example, [1, 5]. We shall represent formulas and sets of formulas with upper case italics $P$, literals with lower case italics, $g$, and substitutions in Greek, $\theta$. Let $P \setminus Q$ be the set of elements of the set $P$ that are not elements of the set $Q$. Let $\forall F$ represent the universal closure of $F$, where all free variables in $F$ are universally quantified. Let $\exists F$ be the existential closure.

Except where noted, we consider all formulas to be in Skolem standard form; all variables are implicitly universally quantified.

If $p$ is a literal then let $\overline{p}$ be the function applied to $p$ that negates it, adding or removing the $\neg$ operator as necessary. For examples, $\overline{\neg q} = q$, and $\overline{q} = \neg q$.

The result of replacing $g$ in a Skolem standard formula with $\overline{g}$ is still a Skolem standard formula since no quantifiers are negated. For example, replace $q$ with $\neg q$ in $\forall(p \vee q)$ to get $\forall(p \vee \neg q)$.

## 2.1 Propositional Horn Clause Proofs

In this section we introduce the propositional Horn clause proof, or HC proof, a proof of a propositional literal from a consistent set of propositional Horn clauses. The main result shows that a propositional literal is entailed by a set of propositional Horn clauses if and only if there is an HC proof of it.

**Definition 1** Let $P$ be a consistent set of propositional Horn clauses.

1.1 An **HC proof** of a propositional literal $g$ from $P$ is an HC1 proof of $g$ from $P \cup \{\overline{g}\}$.

1.2 An **HC1 proof** of a propositional literal $g$ from $P$ is
   (1) a clause $g_1 \vee \ldots \vee g_n$ selected from $P$ such that $g = g_i$ for some $i$, and
   (2) an HC1 proof of $\overline{g}_j$ from $P$ for each $j = 1, \ldots, i-1, i+1, \ldots, n$.

The following lemma shows that the additional clause $\overline{g}$ has no effect when a positive literal is being proved.

**Lemma 1** There is an HC proof of a positive literal $g$ from $P$ if and only if there is an HC1 proof of $g$ from $P$. Both proofs consist only of definite clauses.

**Proof** ($\Rightarrow$ by structural induction on the HC proof) By Definition 1 there is an HC1 proof of $g$ from $P \cup \{\neg g\}$. If that HC1 proof consists of one clause, it is $g$, a definite clause, so $g \in P$. This also serves as an HC1 proof of $g$ from $P$. Suppose the HC1 proof has $k+1$ clauses. Then there is a clause $C = g_1 \vee \ldots \vee g_n \in P$ where $g_i = g$ and an HC1 proof of each $\overline{g}_j$ for $j = 1, \ldots, i-1, i+1, \ldots, n$. Since $g$ is positive $C$ is a definite clause and each $\overline{g}_j$ is positive. By induction there is an HC1 proof of $\overline{g}_j$ from $P$. These proofs and $C$ comprise the HC1 proof of $g$ from $P$.

($\Leftarrow$) An HC1 proof from $P$ serves as an HC1 proof from any superset of $P$. A straightforward induction, similar to the previous case, ensures it consists of definite clauses.∎

HC proofs are sound with respect to model theory.

**Theorem 2** [Propositional Soundness] Let $P$ be a consistent set of propositional Horn clauses and $g$ a propositional literal. If there is an HC proof of $g$ from $P$ then $P \models g$.

**Proof** We show by induction that if we can build an HC1 proof of $g$ from $P \cup \{\overline{g}\}$ then $P \models g$. If the proof has one clause, it must be $g$. Since $g \in P$, we know that $P \models g$. Let there be an HC proof of $g$ that has $k+1$ clauses. Then without loss of generality assume the topmost clause is $g \vee g_1 \vee \ldots \vee g_n$. Then the HC proofs of $\overline{g}_i$ for $i = 1 \ldots n$ each have less than $k+1$ clauses, so by induction $P \models \overline{g}_i$ for $i = 1 \ldots n$. Since $g \vee g_1 \vee \ldots \vee g_n \in P$, we have $P \models g \vee g_1 \vee \ldots \vee g_n$, so we have $P \models g$ as required.∎

The next two lemmas will be required for the proof of completeness.

**Lemma 3** Let $P$ be a consistent set of propositional Horn clauses, and let $a$ and $b$ be positive propositional literals. Suppose $P \cup \{\neg a\}$ and $P \cup \{\neg b\}$ are each consistent. Then $P \cup \{\neg a\} \cup \{\neg b\}$ is consistent.

**Proof** Let $A = \{r_1, \ldots, r_n\}$ be the complete set of atoms in $P \cup \{\neg a\} \cup \{\neg b\}$. Then since $P \cup \{\neg a\}$ is consistent let $F_1$ be a model of $P \cup \{\neg a\}$. That is, $F_1$ is a function, $F_1 : A \rightarrow \{true, false\}$, and each clause in $P \cup \{\neg a\}$, evaluated by $F_1$ and the standard truth tables, evaluates to *true*. Similarly, since $P \cup \{\neg b\}$ is consistent, let $F_2$ be a model of $P \cup \{\neg b\}$. For each $r \in A$ we define $F_3(r) = F_1(r) \wedge F_2(r)$. We claim that $F_3$ is a model of $P \cup \{\neg a\} \cup \{\neg b\}$ so $P \cup \{\neg a\} \cup \{\neg b\}$ is consistent. Clearly $F_3(a)$ is a model of $\{\neg a\}$ since $F_1$ maps $a$ to *false*. Also $F_3(a)$ is a model of $\{\neg b\}$ since $F_2$ maps $b$ to *false*. Thus we only need to show $F_3$ is a model of $P$.

Suppose there exists a clause $C$ in $P$ such that $F_3(C) = false$. There are three cases.

(1) $C = c_0$ where $c_0$ is a positive literal. Either $F_1(c_0) = false$ or $F_2(c_0) = false$. Then either $F_1$ or $F_2$ is not a model of $C$. Contradiction.

(2) $C = \neg c_1 \vee \ldots \vee \neg c_n$, where each $c_i$ is positive. Then $F_3(c_1) = true, \ldots, F_3(c_n) = true$, so it must be that $F_1(c_1) = true, \ldots, F_1(c_n) = true$, so $F_1$ is not a model of $C$. Contradiction.

(3) $C = c_0 \vee \neg c_1 \vee \ldots \vee \neg c_n$, where each $c_i$ is positive. Then $F_1(c_1) = F_2(c_1) = true, \ldots, F_1(c_n) = F_2(c_n) = true$, and either $F_1(c_0) = false$ or $F_2(c_0) = false$. So either $F_1$ or $F_2$ is not a model of $C$. Contradiction.∎

**Lemma 4** Let $P$ be a consistent set of propositional Horn clauses and let $a_1, \ldots, a_n$ be positive propositional literals. If each of $P \cup \{\neg a_1\}, \ldots, P \cup \{\neg a_n\}$ is consistent, then $P \cup \{\neg a_1\} \cup \ldots \cup \{\neg a_n\}$ is consistent.

**Proof** We proceed by induction on $k$. The inductive hypothesis states that $P \cup \{\neg a_1\} \cup \ldots \cup \{\neg a_k\} \cup \{\neg a_{k+i}\}$ is consistent for each $i = 1 \ldots n-k$.

For $k = 0$, the inductive hypothesis is the premise of the lemma.

In the inductive step, we must show that the inductive hypothesis implies that $P \cup \{\neg a_1\} \cup \ldots \cup \{\neg a_{k+1}\} \cup \{\neg a_{k+1+i}\}$ is consistent for each $i = 1 \ldots n - k - 1$.

For each $j = 2 \ldots n-k$ we have

$$\begin{array}{rl} & P \cup \{\neg a_1\} \cup \ldots \cup \{\neg a_k\} \cup \{\neg a_{k+1}\} \text{ is consistent} \\ & P \cup \{\neg a_1\} \cup \ldots \cup \{\neg a_k\} \cup \{\neg a_{k+j}\} \text{ is consistent} \\ \text{Lemma 3} \Rightarrow & P \cup \{\neg a_1\} \cup \ldots \cup \{\neg a_k\} \cup \{\neg a_{k+1}\} \cup \{\neg a_{k+j}\} \text{ is consistent} \end{array}$$

Replace $j$ with $i+1$ in the above, and we have the conclusion of the inductive step. When $k$ becomes $n-1$ the conclusion of the inductive step is the result.∎

**Theorem 5** [Propositional Completeness] Let $P$ be a consistent set of propositional Horn clauses, and $g$ a propositional literal. If $P \models g$ then there is an HC proof of $g$ from $P$.

**Proof** Let $g$ be positive. We apply Lemma 1 so that we have only to build HC1 proof of $g$ from $P$, proceeding by induction on the number of clauses in $P$. If $P$ has no clauses then the theorem is satisfied trivially since $P \models g$ must be false. Let $P$ have $k+1$ clauses. We claim there is a clause $C = g \vee \neg g_1 \vee \ldots \vee \neg g_n, C \in P$ such that $P \setminus C \models g_i$ for all $i$. No generality is lost by assuming the positive literal appears first in $C$. The inductive hypothesis provides an HC1 proof of each $g_i$ from $P \setminus C$. Each of these is also an HC1 proof from $P$. These proofs, and $C$ comprise an HC1 proof of $g$.

We show the claim in three steps. (1) There must exist a clause $C = g \vee \neg g_1 \vee \ldots \vee \neg g_n, C \in P$. (2) $P \models g_i$ for all $i = 1 \ldots n$. (3) $P \setminus C \models g_i$ for all $i = 1 \ldots n$.

(1) Suppose there is no clause $C = g \vee \neg g_1 \vee \ldots \vee \neg g_n$ in $P$. Then every clause in $P$ that mentions $g$ mentions it negatively. $P$ is consistent, so from any model of $P$ construct a new interpretation that differs, possibly, from

the model only in that it maps $g$ to false. This new interpretation is also a model of $P$ because it does not affect the value of any clause that does not mention $g$, and any clause that does mention $g$ mentions it negatively, so it must evaluate to *true*. Since there is a model of $P$ that is not a model of $g$, we have a contradiction.

(2) Suppose $r$ clauses $C_j \in P$ contain a positive $g$. Let $C_j = g \vee \neg g_{j1} \vee \ldots \vee \neg g_{jn_j}$. Suppose for all $j = 1 \ldots r$ there is $i_j$ such that $P \not\models g_{ji_j}$. Then for each $j$, $P \cup \{\neg g_{ji_j}\}$ is consistent. By Lemma 4, $P \cup \{\neg g_{1i_1}\} \cup \ldots \cup \{\neg g_{ri_r}\}$ is consistent. From any model of this construct a new interpretation that differs, possibly, from the model only in that it maps $g$ to false. This new interpretation is also a model of $P$ because every clause that contains a positive $g$ also contains some $\neg g_{ji_j}$ that evaluates to *true*. Thus we have a model of $P$ that is not a model of $g$. Contradiction.

(3) Suppose for some $i$, $P \setminus C \not\models g_i$. Then there is a model of $P \setminus C$ that maps $g_i$ to *false*. This is also a model of $C$, so it is a model of $P$. This contradicts $P \models g_i$, step (2) above.

This completes the demonstration of completeness for a positive goal.

We now let $g = \neg h$, a negative literal, and build an HC proof for $\neg h$, given that $P \models \neg h$. We proceed by induction on the number of negative clauses in $P$, clauses that contain only negative literals.

If $P$ has no negative clauses then $P \models \neg h$ is impossible because $P \cup \{h\}$ always has a model. That model can be built by assigning every atom to *true*. This completes the base case. Let $P$ have $k+1$ negative clauses. Select a negative clause $C = \neg c_0 \vee \ldots \vee \neg c_n$. If $P \setminus C \cup \{h\}$ is inconsistent, then by the inductive hypothesis there is an HC proof of $\neg h$ from $P \setminus C$. This serves as an HC proof of $\neg h$ from $P$, so in this case, the demonstration is complete.

In the remaining case, $P \setminus C \cup \{h\}$ is consistent. We know that $P \setminus C \cup \{h\} \models c_i$ for all $i = 0 \ldots n$, because otherwise $P \setminus C \cup \{h\} \cup \{c_i\}$ is consistent, and a model of this is a model of $P \cup \{h\}$ which cannot exist. By the demonstration of completeness for positive literals, there is an HC1 proof of $c_i$ from $P \setminus C \cup \{h\}$ for all $i = 0 \ldots n$. We claim that there exists a $c_i$ such that $P \setminus C \not\models c_i$. If the claim is false then $P \setminus C \models c_0 \wedge \ldots \wedge c_n$, so $P \setminus C \models \neg C$. By monotonicity of $\models$, $P \models \neg C$. Since $C \in P$, $P \models C$. This means $P$ is inconsistent, which contradicts our premise, so the claim must be true. Without loss of generality, suppose $P \setminus C \not\models c_0$. Then by Theorem 2 there is no HC proof of $c_0$ from $P \setminus C$. But since $P \setminus C \cup \{h\} \models c_0$ there is

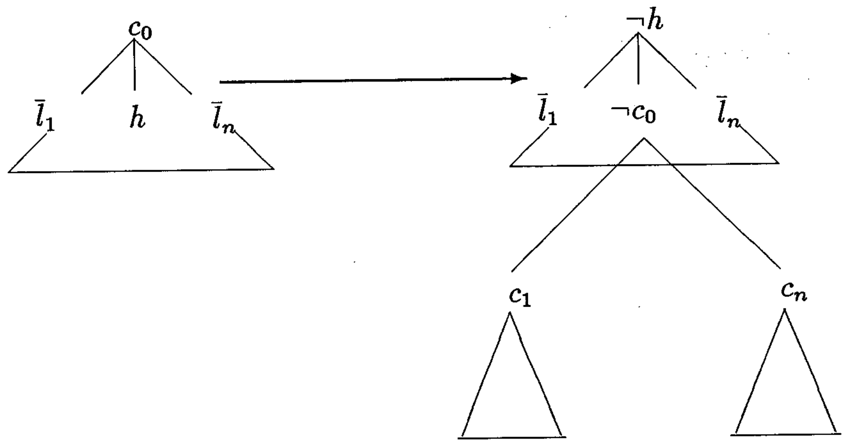


Figure 1: Construction of an HC1 proof of $\neg h$

an HC proof of $c_0$ from $P \setminus C \cup \{h\}$. This HC proof must use $h$.

We build the HC1 proof of $\neg h$ bottom up. (See Figure 1.) Starting with the HC1 proofs of $c_i$ from $P \setminus C \cup \{h\}$ for all $i = 1 \ldots n$, add the clause $C$ on top. This makes an HC1 proof of $\neg c_0$ from $P \cup \{h\}$. We proceed by turning the HC1 proof of $c_0$ "on its head" and putting it on top of the proof of $\neg c_0$, which gives us our proof of $\neg h$.

Since the HC1 proof of $c_0$ from $P \setminus C \cup \{h\}$ uses $h$, without loss of generality that proof's topmost clause is $c_0 \vee l_1 \vee \ldots \vee l_m$ such that for some $j$ either $\bar{l}_j = h$ or the HC1 proof of $\bar{l}_j$ uses $h$. If $\bar{l}_j = h$, as shown in Figure 1, we put the clause $c_0 \vee l_1 \vee \ldots \vee l_{j-1} \vee \neg h \vee l_{j+1} \vee \ldots \vee l_m$ on top of the proof of $\neg c_0$, noting that there is an HC1 proof of each $\bar{l}_k, k = 1 \ldots j-1, j+1 \ldots m$. This completes the proof of $\neg h$. If there is no $j$ such that $\bar{l}_j = h$ then for some $j$ the HC proof of $\bar{l}_j$ uses $h$. The topmost clause of this either contains $h$ or a literal for which the HC proof uses $h$. Repeat the construction until $\neg h$ is reached. This completes the construction of the HC1 proof of $\neg h$.∎

## 2.2 First Order Horn Clause Proofs

We define Horn clause proofs of literals in a first order logic without equality. The definition is generalized from the propositional case. The main result of this section is the generalization of soundness and completeness from the propositional case.

Variables in the literals impose a type of communication between subproofs. When a substitution is made for a variable in one child node, all other children that use that variable must now use the new instance. To control this communication we do the subproofs one at a time. The order is determined by a choosing function. A different choosing function results in a different proof. We call the choosing function R, and the proof that results an RHC proof. Each proof also produces a substitution for the variables in the clauses and the goal. That substitution also depends on R, so it is called the R-computed answer substitution. When there is no confusion we will refer to the R-computed answer substitution simply as the answer substitution.

**Definition 2** A **choosing function** R is a function from a sequence of literals to a literal.

**Definition 3** Let $P$ be a consistent set of Horn clauses.

3.1 An **RHC proof** of a literal $g$ from $P$ is an RHC1 proof of $g$ from $P$ retaining $\overline{g}$.

3.2 An **RHC1 proof** of a literal $g$ from $P$ retaining $t$ is either

(case a) nothing, if there is a most general unifier $\theta$ of $g$ and $t$, (In this case the **R-computed answer substitution** is $\theta$.), or

(case b) (1) a clause $g_1 \vee \ldots \vee g_n$ selected from $P$ and given new, unique variables, such that there is a $g_i$ and a most general unifier $\theta_i$ of $g$ and $g_i$, and

(2) a RHC1 proof sequence for $\overline{g}_1\theta_i, \ldots, \overline{g}_{i-1}\theta_i, \overline{g}_{i+1}\theta_i, \ldots, \overline{g}_n\theta_i$ from $P$ retaining $t\theta_i$, which has an R-computed answer substitution $\theta_{seq}$.

(In this case the **R-computed answer substitution** for the RHC1 proof is $\theta_i\theta_{seq}$ restricted to variables in $g$.)

3.3 An **RHC1 proof sequence** for the sequence $g_1, \ldots, g_n$ from $P$ retaining $t$ is (case a) the empty sequence, if $n = 0$. (The **R-computed answer substitution** for the empty sequence is $\epsilon$, the identity substitution.), or

(case b) (1) an RHC1 proof from $P$ retaining $t$ of $g_i$ with R-computed answer substitution $\theta_i$, where $g_i = R(g_1, \dots, g_n)$ and

(2) an RHC1 proof sequence for $g_1\theta_i, \dots, g_{i-1}\theta_i, g_{i+1}\theta_i, \dots, g_n\theta_i$ from $P$ retaining $t\theta_i$ with R-computed answer substitution $\theta_{seq}$.

(In this case the **R-computed answer substitution** for the whole sequence is $\theta_i\theta_{seq}$.)

Soundness is the first important property of RHC proofs we demonstrate.

**Theorem 6** [First Order Soundness] Let $P$ be a consistent set of Horn clauses, and $g$ a literal. If there is a choosing function R and an RHC proof of $g$ from $P$ with R-computed answer substitution $\theta$ then $P \models g\theta$.

**Proof** To show that RHC proofs are sound, we need to show RHC1 proofs are sound. We claim that if there is an RHC1 proof of $g$ from $P$ retaining $t$ with answer substitution $\theta$ then $P \cup \{t\theta\} \models g\theta$. Given this we know that an RHC1 proof of $g$ from $P$ retaining $\overline{g}$ with answer substitution $\theta$ means that $P \cup \{\overline{g}\theta\} \models g\theta$. This means $P \cup \{\overline{g}\theta\} \cup \{\neg(g\theta)\}$ is unsatisfiable, or in other words $P \cup \{\forall\overline{g}\theta\} \cup \{\exists\overline{g}\theta\}$ is unsatisfiable. If there is a model of $P \cup \{\exists\overline{g}\theta\}$ then this is a can be made a model of $P \cup \{\forall\overline{g}\theta\} \cup \{\exists\overline{g}\theta\}$ by restricting the domain to elements such that $\overline{g}\theta$ is true. But we know this set is unsatisfiable, so $P \cup \{\exists\overline{g}\theta\}$ is unsatisfiable. Hence $P \models \neg(\exists\overline{g}\theta)$, which means $P \models g\theta$.

We show the claim by induction on the number of clauses in the RHC1 proof of $g$. If there are no clauses in the RHC1 proof then there is a most general unifier $\theta$ of $g$ and $t$, which is the answer substitution. Since $g\theta = t\theta$ we know that $P \cup \{t\theta\} \models g\theta$. This completes the base case. Suppose the result holds for proofs with up to $k$ clauses. Let there be a proof with $k+1$ clauses. Then without loss of generality there is (1) a clause $g_1 \vee \dots \vee g_n$ such that $\theta_1$ is a most general unifier of $g$ and $g_1$, and (2) an RHC1 proof sequence for $\overline{g}_1\theta_1, \dots, \overline{g}_n\theta_1$ retaining $t\theta_1$ with answer substitution $\theta_{seq}$. Also $\theta = \theta_1\theta_{seq}$.

We introduce an inner induction to show if there is an RHC1 proof sequence of $h_1, \dots, h_n$ retaining $t$ with answer substitution $\sigma$ where each RHC1 proof uses up to $k$ clauses then $P \cup \{t\sigma\} \models h_1\sigma, \ldots, P \cup \{t\sigma\} \models h_n\sigma$. For $n = 0$ there is nothing to show since the sequence is empty. Suppose the inner induction result holds for sequences up to $n$. Let there be an RHC1 proof sequence for $h_1, \ldots, h_{n+1}$ retaining $t$. Without loss of generality suppose R selects $h_{n+1}$. Then there is an RHC1 proof of $h_{n+1}$ retaining $t$ with answer substitution $\sigma_{n+1}$, and an RHC1 proof sequence for $h_1\sigma_{n+1}, \ldots, h_n\sigma_{n+1}$ retaining $t\sigma_{n+1}$ with answer substitution $\sigma_{seq}$. By the outer induction hypothesis, $P \cup \{t\sigma_{n+1}\} \models h_{n+1}\sigma_{n+1}$, so $P \cup \{t\sigma_{n+1}\sigma_{seq}\} \models h_{n+1}\sigma_{n+1}\sigma_{seq}$. By the inner induction hypothesis, $P \cup \{t\sigma_{n+1}\sigma_{seq}\} \models h_1\sigma_{n+1}\sigma_{seq}, \ldots, P \cup \{t\sigma_{n+1}\sigma_{seq}\} \models h_n\sigma_{n+1}\sigma_{seq}$. The inner induction argument is complete.

Apply the inner induction result to (2) by replacing $h_j$ in the inner result with $\overline{g}_j\theta_1$ and $\sigma$ with $\theta_{seq}$. Then $P \models \overline{g}_1\theta_1\theta_{seq}, \ldots, P \models \overline{g}_n\theta_1\theta_{seq}$. Since $(g_1 \vee \ldots \vee g_n) \in P$, $P \models (g_1\theta_1\theta_{seq} \vee \ldots \vee g_n\theta_1\theta_{seq})$, so $P \models g_1\theta_1\theta_{seq}$. Since $g_1\theta_1\theta_{seq} = g\theta_1\theta_{seq}$, and $\theta = \theta_1\theta_{seq}$, we have $P \models g\theta$.∎

We will also show that RHC proofs are complete, in the sense that any correct answer substitution is an instance of some computed answer substitution. We first show a version of the lifting lemma. Informally, it says given a proof of a goal from a set of clauses we can lift this to a proof of a more general goal from a set of more general clauses. The lifting lemma itself requires the following lemma.

**Lemma 7** Let $g_1$ and $g_2$ be literals, and $\phi_1$ and $\phi_2$ be substitution. Suppose that $\phi_1$ binds no variables in $g_2$ or in $phi_2$, that $\phi_2$ binds no variables in $g_1$ or in $phi_1$. If $\theta$ is a unifier of $g_1\phi_1$ and $g_2\phi_2$ then there is a most general unifier $\theta'$ of $g_1$ and $g_2$ and a substitution $\gamma$ such that $\phi_1\theta = \theta'\gamma$ restricted to the variables in $g_1$ and $\phi_2\theta = \theta'\gamma$ restricted to the variables in $g_2$.

**Proof** $g_2\phi_2\phi_1 = g_2\phi_2$ since $\phi_1$ binds no variables in $phi_2$. $g_1\phi_2 = g_1$ since $\phi_2$ binds no variables in $g_1$. $\theta$ unifies $g_1\phi_1$ and $g_2\phi_2$ so $g_1\phi_1\theta = g_2\phi_2\theta$. Therefore $g_1\phi_2\phi_1\theta = g_2\phi_2\phi_1\theta$. In other words $\phi_2\phi_1\theta$ unifies $g_1$ and $g_2$. Therefore there is a most general unifier, $\theta'$ of $g_1$ and $g_2$, which means there is a substitution $\gamma$ such that $\phi_2\phi_1\theta = \theta'\gamma$.

Restricting substitutions to the variables in $g_1$, $\phi_2\phi_1 = \phi_1$ since $\phi_2$ does not bind any variables in $g_1$ and $\phi_1$ does not bind any variables in $\phi_2$. Therefore, restricted to variables in $g_1$, $\phi_1\theta = \theta'\gamma$.

Restricted to the variables in $g_2$, $\phi_2\phi_1 = \phi_2$ since $\phi_1$ binds no variables in $\phi_2$. Therefore, restricted to variables in $g_2$, $\phi_2\theta = \theta'\gamma$.∎

**Lemma 8** [Lifting Lemma] Let $P$ be a consistent set of Horn clauses, and $g$ a literal. Let $P^+$ be a set of instances of clauses of $P$, defined as $P^+ = \{C\pi \mid C \in P, \pi \text{ is some substitution}\}$. For every substitution $\tau$ such that there is a choosing function R and an RHC proof of $g\tau$ from $P^+$ with R-computed answer substitution $\theta$, there is a choosing function $R_1$ and an $R_1$HC proof of $g$ from $P$ with $R_1$-computed answer substitution $\theta'$ and a substitution $\gamma$ such that $\tau\theta = \theta'\gamma$.

**Proof** We need to show that if there is an RHC1 proof of $g\tau$ from $P^+$ retaining $t\tau$ with R-computed answer substitution $\theta$ then there is a choosing function $R_1$ and an $R_1$HC1 proof of $g$ from $P$ retaining $t$ with an $R_1$ computed answer substitution $\theta'$ and a substitution $\gamma$ such that $\tau\theta = \theta'\tau$. We proceed by induction on the number of clauses in the RHC1 proof. If there are no clauses then there is a most general unifier $\theta$ of $g\tau$ and $t\tau$. Then $\tau\theta$ is a unifier of $g$ and $t$, so there is a most general unifier $\theta'$ of $g$ and $t$, which means there exists a $\gamma$ such that $\theta'\gamma = \tau\theta$. This constitutes an $R_1$HC1 proof of $g$ from $P$ retaining $t$ with answer substitution $\theta'$, where any choosing function will do for $R_1$. This completes the base case. Suppose the lemma holds for a proof with up to $k$ clauses. (See Figure 2.) Let there be an RHC1 proof of $g\tau$ from $P$ retaining $t\tau$ with $k+1$ clauses. Without loss of generality, there is (1) a clause $(g_1 \vee \ldots \vee g_n)\phi \in P^+$ such that a substitution $\theta_1$ unifies $g\tau$ and $g_1\phi$, and (2) there is an RHC1 proof sequence for $\overline{g}_2\phi\theta_1, \ldots, \overline{g}_n\phi\theta_1$ from $P^+$ retaining $t\phi\theta_1$ with answer substitution $\theta_{seq}$.

We introduce a nested induction to show that if there is an RHC1 proof sequence of $h_1\delta, \ldots h_n\delta$ from $P^+$ retaining $t\delta$ involving proofs with up to $k$ clauses and with answer substitution $\sigma$, then there is a choosing function $R_1$ and an $R_1$HC1 proof sequence of $h_1, \ldots, h_n$ from $P$ retaining $t$ with answer substitution $\sigma'$ and a substitution $\rho$ such that $\delta\sigma = \sigma'\rho$. We proceed by induction on the length of the sequence. If the sequence is empty then the answer substitution for the proof sequence is $\epsilon$. Let the new proof sequence be empty, and its answer substitution be $\epsilon$. To have $\delta\sigma = \sigma'\rho'$, we let $\rho = \delta$, since $\sigma = \sigma' = \epsilon$. This complete the base case. Suppose the induction step holds for sequence of length $n$. Let there be a proof sequence for $h_1\delta, \ldots, h_{n+1}\delta$

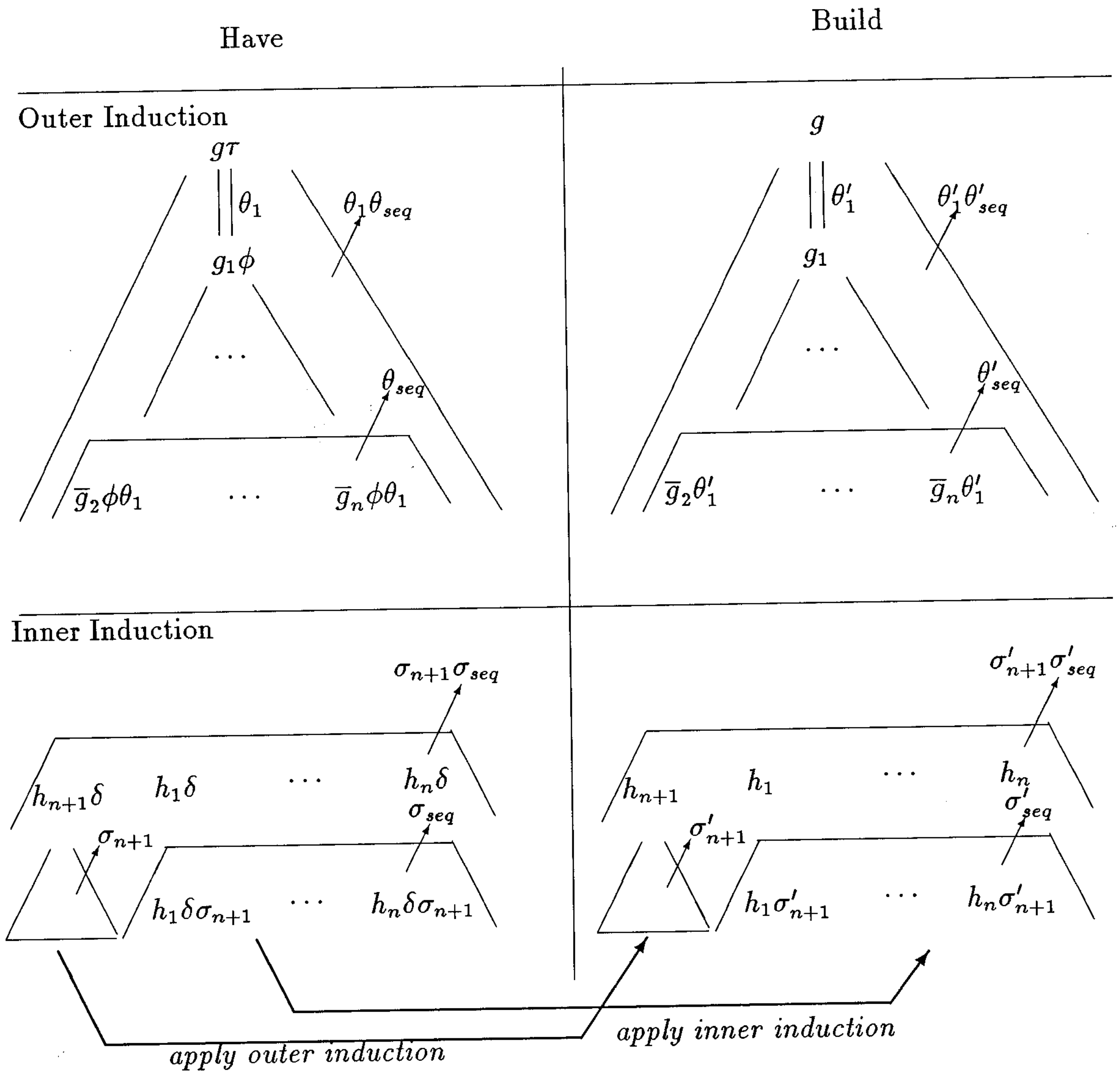


Figure 2: Lifting an RHC1 proof of $g\tau$ from $P^+$ to an R$_1$HC1 proof of $g$ from $P$.

from $P^+$ retaining $t\delta$ with answer substitution $\sigma$. Then without loss of generality, suppose R selects $h_{n+1}\delta$. There is an RHC1 proof of $h_{n+1}\delta$ from $P^+$ retaining $t\delta$ with answer substitution $\sigma_{n+1}$, and a proof sequence of $h_1\delta\sigma_{n+1}, \ldots, h_n\delta\sigma_{n+1}$ from $P^+$ retaining $t\delta\sigma_{n+1}$ with answer substitution $\sigma_{seq}$. Then define the selection function $\text{R}_1$ so that $\text{R}_1(h_1, \ldots, h_{n+1}) = h_{n+1}$. By the outer induction hypothesis, there is an $\text{R}_1$HC1 proof of $h_{n+1}$ from $P$ retaining $t\delta$ with answer substitution $\sigma'_{n+1}$ and there is a substitution $\rho_{n+1}$ such that $\delta\sigma_{n+1} = \sigma'_{n+1}\rho_{n+1}$. Since $\sigma'_{n+1}\rho_{n+1} = \delta\sigma_{n+1}$ a proof sequence for $h_1\delta\sigma_{n+1}, \ldots, h_n\delta\sigma_{n+1}$ retaining $t\delta\sigma_{n+1}$ is a proof sequence for $h_1\sigma'_{n+1}\rho_{n+1}, \ldots, h_n\sigma'_{n+1}\rho_{n+1}$ retaining $t\sigma'_{n+1}\rho_{n+1}$. By the inner induction hypothesis, (replace $\delta$ in the statement of the hypothesis with $\rho_{n+1}$, and the $h_i$ with $h_i\sigma'_{n+1}$) there is an $\text{R}_1$HC1 proof sequence for $h_1\sigma'_{n+1}, \ldots, h_n\sigma'_{n+1}$ from $P$ retaining $t\sigma'_{n+1}$ with answer substitution $\sigma'_{seq}$ and a substitution $\rho_{seq}$ such that $\rho_{n+1}\sigma_{seq} = \sigma'_{seq}\rho_{seq}$. Thus we have an $\text{R}_1$HC1 proof sequence of $h_1, \ldots, h_{n+1}$ from $P$ retaining $t$ with substitution $\sigma'_{n+1}\sigma'_{seq}$. Let $\sigma' = \sigma'_{n+1}\sigma'_{seq}$ and let $\rho = \rho_{seq}$. Then $\delta\sigma = \delta\sigma_{n+1}\sigma_{seq} = \sigma'_{n+1}\rho_{n+1}\sigma_{seq} = \sigma'_{n+1}\sigma'_{seq}\rho_{seq} = \sigma'\rho$. This completes the inner induction.

Returning to the outer induction, we have (1) the clause $(g_1 \vee \ldots \vee g_n)\phi \in P^+$, and a substitution $\theta_1$ such that $g\tau\theta_1 = g_1\phi\theta_1$. Since the clause has unique variables, we can be sure that $\tau$ binds no variables in $g_1$ or in $\phi$. Likewise we can assume that $\phi$ binds no variables in $g$ or in $\tau$. By Lemma 7 there is a most general unifier $\theta'_1$ of $g$ and $g_1$ and a substitution $\gamma_1$ such that $\tau\theta_1 = \theta'_1\gamma_1$ restricted to the variables in $g$, and $\phi\theta_1 = \theta'_1\gamma_1$ restricted to the variables in the $g_1$. Given (2) the RHC1 proof sequence for $\overline{g}_2\phi\theta_1, \ldots, \overline{g}_n\phi\theta_1$ from $P^+$ retaining $t\phi\theta_1$ with answer substitution $\theta_{seq}$, this is also a proof sequence for $\overline{g}_2\theta'_1\gamma_1, \ldots, \overline{g}_n\theta'_1\gamma_1$ from $P^+$ retaining $t\theta'_1\gamma_1$. Apply the inner induction result, letting the $h$'s in the induction claim be the $\overline{g}_i\theta'_1$, $\delta$ be $\gamma_1$ and $\sigma$ be $\theta'_{seq}$, to get an $\text{R}_1$HC1 proof sequence for $\overline{g}_2\theta'_1, \ldots, \overline{g}_n\theta'_1$ from $P$ retaining $t\theta'_1$ with answer substitution $\theta'_{seq}$ and a substitution $\gamma_{seq}$ such that $\gamma_1\theta_{seq} = \theta'_{seq}\gamma_{seq}$. This produces an $\text{R}_1$HC1 proof of $g$ from $P$ retaining $t$ with answer substitution $\theta'_1\theta'_{seq}$, so $\theta' = \theta'_1\theta'_{seq}$. Let $\gamma = \gamma_{seq}$. Then $\tau\theta = \tau\theta_1\theta_{seq} = \theta'_1\gamma_1\theta_{seq} = \theta'_1\theta'_{seq}\gamma_{seq} = \theta'\gamma$.∎

**Theorem 9** [First Order Completeness] Let $P$ be a consistent set of Horn clauses, and $g$ a literal. For every substitution $\omega$ such that $P \models g\omega$, there is a choosing function R and an RHC proof of $g$ from $P$ with R-computed answer substitution $\omega'$ and a substitution $\gamma$ such that $\omega = \omega'\gamma$.

**Proof** Let $\{x_1, \ldots, x_r\}$ be the complete set of variables in $g\omega$. Then let $\{c_1, \ldots, c_r\}$ be a set of constants not appearing in $P$ or $g\omega$. Define $\phi = \{x_1/c_1, \ldots, x_r/c_r\}$. Then $g\omega\phi$ is ground. Since $P \models g\omega\phi, P \cup \{\overline{g}\omega\phi\}$ is an unsatisfiable set of clauses in Skolem standard form. Herbrand's Theorem[1](p. 61) states that there is an unsatisfiable set of clauses if and only if there is a finite unsatisfiable set of ground instances of these clauses. Let $S$ be the finite unsatisfiable set of ground instances of clauses in $P \cup \{\overline{g}\omega\phi\}$. Let $P^*$ be the set of instances of clauses from $P$ that appear in $S$. Then again by Herbrand's Theorem, $P^*$ is satisfiable since $P$ is. So $\overline{g}\omega\phi \in S$, since otherwise $S = P^*$ and one of them is satisfiable while the other is not. Thus $S = P^* \cup \{\overline{g}\omega\phi\}$. Since $S$ is unsatisfiable, $P^* \models g\omega\phi$. We may consider all ground literals to be propositions, so by Theorem 5, there is an HC proof of $g\omega\phi$ from $P^*$. Observe that this is an RHC proof of $g\omega\phi$ from $P^*$ with answer substitution $\epsilon$. Any choosing function will do for R; the order of subproofs does not matter since no variables are bound. By textually replacing all instances of $c_i$ with the corresponding $x_i$ in $P^*$ then we have built a new set $P^+$ of clauses that are instances of the clauses in $P$. By doing the same replacement of $c_i$ with $x_i$ in the RHC proof of $g\omega\phi$ from $P^*$ with answer substitution $\epsilon$ we have built an RHC proof of $g\omega$ from $P^+$ with answer substitution $\epsilon$. Now apply the lifting lemma, letting $\tau$ in the lemma be $\omega$, $\theta'$ be $\omega'$, and $\theta$ be $\epsilon$.∎

## 3 General Queries

Until now we have restricted ourselves to HC proofs of single literals. Here we define an HC proof sequence, for proving a sequence of any number of positive and negative literals.

**Definition 4** An RHC **proof sequence** for the sequence $g_1, \ldots, g_n$ from $P$ retaining $t$ is

(case a) the empty sequence, if $n = 0$. (The **R-computed answer substitution** for the empty sequence is $\epsilon$, the identity substitution.), or

(case b) (1) an RHC proof from $P$ of $g_i$ with R-computed answer substitution $\theta_i$, where $g_i = R(g_1, \ldots, g_n)$ and

(2) an RHC proof sequence for $g_1\theta_i, \ldots, g_{i-1}\theta_i, g_{i+1}\theta_i, \ldots, g_n\theta_i$ from $P$ with R-computed answer substitution $\theta_{seq}$.

(In this case the **R-computed answer substitution** for the whole sequence is $\theta_i\theta_{seq}$.)

HC proof sequences are sound and complete.

**Lemma 10** Let $P$ be a set of Horn clauses and $G$ a sequence of literals. If there is an RHC proof sequence of $G$ from $P$ with answer substitution $\theta$ then, treating $G$ as a conjunction, $P \models G\theta$.

**Proof** (by induction on the length of $G$) If $G$ is empty then treated as a conjunction, $G = \mathit{true}$, and there is nothing to show. If $G = g_1, \ldots, g_{n+1}$ then without loss of generality suppose there is a RHC proof of $g_{n+1}$ with answer substitution $\theta_{n+1}$, and an RHC proof sequence of $g_1\theta_{n+1}, \ldots, g_n\theta_{n+1}$ with answer substitution $\theta_{seq}$. Then $\theta = \theta_{n+1}\theta_{seq}$. By the soundness of RHC proofs, $P \models g_{n+1}\theta_{n+1} \models g_{n+1}\theta_{n+1}\theta_{seq}$, and by induction $P \models g_1\theta_{n+1}\theta_{seq} \wedge \ldots \wedge g_n\theta_{n+1}\theta_{seq}$. Combining these we obtain $P \models G\theta$. ∎

**Lemma 11** Let $P$ be a set of Horn clauses and $G$ a sequence of literals. Treating $G$ as a conjunction, if there is a substitution $\theta$ such that $P \models G\theta$ then there is an RHC proof sequence of $G$ from $P$ with answer substitution $\theta'$ and a substitution $\gamma$ such that $\theta = \theta'\gamma$.

**Proof** (by induction on the length of $G$) If $G$ is empty then, treating $G$ as a conjunction $G = \mathit{true}$. Let $\gamma = \theta$, let the proof sequence be the empty sequence, and let $\theta' = \epsilon$. If $G = g_1, \ldots, g_n$ then (1)$P \models g_{n+1}\theta$ and (2)$P \models (g_1 \wedge \ldots \wedge g_n)\theta$. By completeness of RHC proofs, from (1) there is an RHC proof of $g_{n+1}$ with answer substitution $\theta'_{n+1}$ and a substitution $\gamma_{n+1}$ such that $\theta = \theta'_{n+1}\gamma_{n+1}$. From (2) we have $P \models (g_1\theta'_{n+1} \wedge \ldots \wedge g_n\theta'_{n+1})\gamma_{n+1}$. By induction there is an RHC proof sequence of $g_1\theta'_{n+1}, \ldots, g_n\theta'_{n+1}$ with answer substitution $\gamma'_{n+1}$ and a substitution $\gamma_{seq}$ such that $\gamma_{n+1} = \gamma'_{n+1}\gamma_{seq}$. Given this RHC proof sequence, and the RHC proof of $g_{n+1}$ we have an RHC proof sequence of $g_1, \ldots, g_{n+1}$ with answer substitution $\theta' = \theta'_{n+1}\gamma'_{n+1}$. Let $\gamma = \gamma_{seq}$. Then $\theta = \theta'_{n+1}\gamma_{n+1} = \theta'_{n+1}\gamma'_{n+1}\gamma_{seq} = \theta'\gamma$. ∎

Building an HC proof sequence of $g_1, \ldots, g_n$ corresponds to showing that a general clause $\overline{g}_1 \vee \ldots \vee \overline{g}_n$ is inconsistent with a set of Horn clauses. In example 9 from [7], these clauses are inconsistent:

$$p \lor q \tag{1}$$
$$\neg p \lor q \tag{2}$$
$$p \lor \neg q \tag{3}$$
$$\neg p \lor \neg q \tag{4}$$

We can show they are inconsistent by building an HC proof sequence. Since clause (1) is non-Horn, we negate it and build a proof sequence of the negated literals, $\neg p$ and $\neg q$.

$$\begin{tikzcd}
\neg p \arrow[d, dash, "(2)"] & \neg q \arrow[d, dash, "(3)"] \\
\neg q \arrow[d, dash, "(3)"] & \neg p \arrow[d, dash, "(4)"] \\
p & q
\end{tikzcd}$$

## 4 Logical Negation in Prolog

By making two changes to the Prolog interpreter, we can convert it to an HC interpreter, a procedure for building Horn clause proof trees (modulo the occur-check problem).

By Lemma 1, there is no difference between the Prolog interpreter and the HC interpreter for positive goals.

Given a negative goal, the HC interpreter must retain a literal which is the positive form of that goal, and apply substitutions to it as they arise from unification. If a new goal arises that unifies with the retained literal, that goal is considered proved.

The other change affects the database of clauses. Prolog clauses are definite clauses, and they only need to be referenced by their positive literals. The HC interpreter must have access to negative clauses, clauses that contain only negative literals. And it must be able to access a clause by any literal in it, positive or negative.

To illustrate, we give two Prolog programs. The first, a familiar one provided for reference, builds Prolog proofs.

```
:- op(300, xfx, (←)).

% proof(G) is true if there is a proof of G from definite clauses
proof(G) :-
        definite_clause(G ← L),
        proof_seq(L).

% proof_seq(L) is true if L is a list of literals and
% there is a proof for each one.
proof_seq([ ]).
proof_seq([L1 | Ls]) :-
        proof(L1),
        proof_seq(Ls).

% definite_clause(G ← L) means G ← L is a definite clause
% given new variables where G is the head, positive literal, and
% L is a list of literals, in their positive form.
```

The second builds HC proofs. It needs to access the clauses by all literals. One way to do this, used here, is to form the contrapositives. For a clause $g_1 \vee \ldots \vee g_n$ there are $n$ contrapositive forms. For each $i = 1 \ldots n$ the contrapositive form is $g_i \leftarrow \overline{g}_1, \ldots, \overline{g}_{i-1}, \overline{g}_{i+1}, \ldots, \overline{g}_n$.

```
:- op(50, fx, (¬)).
:- op(300, xfx, (←)).

% hc_proof(G) is true if there is an HC proof of G.
hc_proof(G) :-
        negate(G, Neg_G),
        hc1_proof(G, Neg_G).

% hc1_proof(G, T) is true if there is an HC1 proof of G retaining T
hc1_proof(T, T).
hc1_proof(G, T) :-
        contrapositive_horn_clause(G ← L),
        hc1_proof_seq(L, T).
```

```
hcl_proof_seq([ ], _).
hcl_proof_seq([L | Ls], T) :-
        hcl_proof(L, T),
        hcl_proof_seq(Ls, T).

% Clauses database routines:
% contrapositive_horn_clause(G ← L) means G ← L is a Horn
% clause in one of its contrapositive forms, given unique variables.
```

To see that this program correctly builds RHC proofs it is necessary only to compare Definition 3 with the program. The program is considerably simpler because the underlying Prolog interpreter deals with the variable substitutions. Except for this, the program is a translation of Definition 3 where the selection function R always chooses the leftmost literal.

## 5 Comparison with Related Work

Gallier and Raatz also consider reasoning from Horn clauses, in [4]. They describe HORNLOG, a proof procedure that applies when the program is a set of Horn clauses and the query is a disjunction of negations of Horn clauses. Our work is new in two ways.

HC proofs apply in a slightly more general case than HORNLOG. Restated as refutation, HORNLOG can detect the unsatisfiability of a set of Horn clauses. Because HC proof sequences can prove an arbitrary conjunction of literals, they can refute a set of clauses, one of which is a general clause and the rest are Horn clauses.

HORNLOG is a proof procedure, based on graph rewriting, and it uses a particular search method, breadth-first search. It is reported that backtracking search methods introduce complications to the graph rewriting procedure; previous graphs are difficult to recover. On the other hand, we have shown how to extend Prolog to build HC proofs, thus they can be found by depth-first search with backtracking. Backtracking implementations need significantly less space, because a stack is used to store different branches of the search space at different times in the same space.

One final difference occurs in the first order case. HORNLOG is a complete procedure with respect to indefinite answers, whereas HC proofs are only complete for definite answers. To explain, if there is no HC proof of $g$ from $P$ then there is no substitution $\omega$ such that $P \models g\omega$. However there may be a set of substitutions $\{\omega_1, \ldots, \omega_n\}$ such that $P \models g\omega_1 \vee \ldots \vee g\omega_n$. For example, consider $P$ a single clause $\neg p(a) \vee \neg p(b)$, and the query $\neg p(X)$. Then there is no Horn clause proof of $\neg p(X)$ from $P$. This seems to indicate there is no $X$ such that $p(X)$. However $P$ guarantees there is such an $X$, and either $X = a$ or $X = b$. Poole [10] gives a method for generating such an answer. That method leads to a new definition of RHC proofs. We will present it, and a proof of its completeness in a future paper.

Another proof procedure, closely related to HC proofs, is the MESON (Model Elimination Subgoal OrieNted) procedure [6]. It is complete for general clauses, and is the basis for [11] and [12] and the theorem prover in [8]. The negative ancestor rule for model elimination is the new rule needed to extends the procedure's coverage from definite to general clauses. Quoting [11]:

> (In the proof tree) If the current goal matches the complement of one of its ancestor goals. then apply the matching substitution and treat the current goal as if it were solved.

This rule requires each literal in the proof tree to be compared to each of its ancestors. Contrast this with Horn clause proof trees where a literal only needs to be compared to one ancestor, the top-level one. In this example,

$$\begin{gathered} b \vee \neg a \\ c \vee \neg b \\ d \vee \neg c \\ \neg a \vee \neg d \end{gathered}$$

the proof tree for $\neg a$ is

$$\begin{tikzcd}
\neg a \arrow[d, no head] \\
\neg b \arrow[d, no head] \\
\neg c \arrow[d, no head] \\
\neg d \arrow[d, no head] \\
a
\end{tikzcd}$$

To prove $a$, the negative ancestor rule asks us to compare $a$ with each of $\neg d, \neg c, \neg b$ and finally $\neg a$. Our procedure only compares $a$ with the $\neg a$.

In other circumstances we can avoid this checking altogether. By Lemma 1 when the topmost goal is positive, no negative goal can arise, so no match can ever succeed.

## 6 HC Proofs for Non-Monotonic Reasoners

HC proofs can show that a conjunction of literals is inconsistent with a set of Horn clauses in the following manner: if $P$ is a set of Horn clauses then $P \cup \{a_1 \wedge \ldots \wedge a_n\}$ is inconsistent if and only if $P \cup \{a_1\} \cup \ldots \cup \{a_{n-1}\} \models \neg a_n$, which can be found with an HC proof.

The propositional version of this is one task performed by de Kleer's assumption-based truth maintenance system (ATMS[2]). The ATMS checks it conjunctions, called environments, for consistency by generating all minimal inconsistent conjunctions, called nogoods. An environment is rejected if it is a superset of some nogood. But generating all minimal nogoods is more work than is required, especially in ATMS applications characterized by many minimal nogoods and few environments that need to be tested for consistency, or in ATMS applications that do not need all environments for each literal. In these cases the cost of calculating all possible nogoods is not justified by the need to test a few environments for consistency. Horn clause proof trees offer an alternative test for consistency that does not generate unrelated inconsistencies.

Theorist[8, 9] is a reasoning system based on first order logic, that explains observations by producing theories. Certain predicates are designated as defaults, and these may be assumed true if doing so does not lead to inconsistency. To explain an observation, Theorist selects a set of defaults, that together with the known facts, entail the observation. This selected set is called the theory. Theorist rejects any theory that is inconsistent.

Theorist implementations to date[10] use general clauses, and a proof procedure based on the MESON proof procedure to check theories for consistency. In cases where Horn clauses are enough to encode the domain facts, Theorist may take advantage of HC proofs. For diagnosing a full adder circuit, a Theorist interpreter using HC proofs was 25% faster than a interpreter based on the negative ancestor rule, but otherwise identical. We will describe our implementation in another paper.

## 7 Summary

A pure Prolog program is a set of first order definite clauses. Each definite clause is a disjunction of one positive literal and any number of negative literals. There can be no clause with only negative literals in a Prolog program, so Prolog programs cannot assert or entail a negative literal. In this paper we considered reasoning from a set of Horn clauses. A Horn clause is either a definite clause or a clause with only negative literals. With Horn clauses it is possible to assert and entail negative literals. We defined HC proofs for first order logic without equality, and showed that they are sound and complete with respect to model theory. A sequence of HC proofs can show that a set of Horn clauses is inconsistent with a general clause.

The idea behind HC proofs is this:

> To prove a literal $g$ build a proof tree with $g$ as the top level goal, but assume that the negation of $g$ is already proved.

HC proofs can be used to add logical negation to Prolog, without also adding disjunction. We described the changes to a Prolog interpreter that extend it to build HC proofs, and illustrated with a Prolog program. The HC proof procedure involves less work than the MESON[6] proof procedure, a procedure for reasoning from general clauses.

We also applied HC proofs to the checking consistency in Theorist, a logic-based system for hypothetical reasoning. A Horn Theorist interpreter diagnosed a full adder circuit 25% faster than a comparable Theorist implementation based on the negative ancestor rule.

## Acknowledgements

Special thanks to Fahiem Bacchus for useful comments on a earlier draft. This work has benefited from comments and discussions by Paul Kates, Peter van Beek and Fei Song.

## References

[1] Chin-Liang Chang and Richard Char-Tung Lee, Symbolic Logic and Mechanical Theorem Proving, *Academic Press* (1973).

[2] Johan de Kleer, An assumption-based Truth Maintenance System, *Artificial Intelligence* 28:127-162 (1986).

[3] W. P. Dowling and J. H. Gallier, Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae, *Journal of Logic Programming* 1(3):267-284 (1984).

[4] Jean Gallier and Stan Raatz, HORNLOG: A Graph-Based Interpreter for General Horn Clauses, *Journal of Logic Programming* 4(2):119-155 (1987).

[5] John Wylie Lloyd, Foundations of Logic Programming, *Springer-Verlag* (1984).

[6] D. W. Loveland, *Automated Theorem Proving: A Logical Basis*, North Holland, Amsterdam (1978).

[7] Francis Jeffry Pelletier, Seventy-Five Problems for Testing Automated Theorem Provers, *Journal of Automated Reasoning* 2:191-216 (1986).

[8] David Poole, Randy Goebel, and Romas Aleliunas, Theorist: a logical reasoning system for defaults and diagnosis, in *The Knowledge Frontier:*

*Essays in the Representation of Knowledge*, N. Cercone and G. McCalla (Eds.), Springer Verlag, New York, 331-352 (1987).

[9] David Poole, A Logical Framework for Default Reasoning, *Artificial Intelligence*, 36:27-47 (1988).

[10] David Poole, Compiling A Default Reasoning System into Prolog, *Research Report CS-88-01*, Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada (1988).

[11] M. E. Stickel, A Prolog Technology Theorem Prover, *Journal of Automated Reasoning* 4:353-380 (1989).

[12] Zerksis D. Umrigar and Vijay Pitchumani, An Experiment in Programming with Full First-Order Logic, in Symposium of Logic Programming, IEEE Computer Society Press, Washington D. C. (1985).