Horn functions and the AFP Algorithm

Rooholah Majdodin
Humboldt-Universität zu Berlin, Institut für Informatik
August 21, 2014

Abstract
It is described why multiple refinements with each negative counterexample does not improve the complexity of the AFP Algorithm. Also Canonical normal formulas for Horn functions are discussed.

1 Introduction
Please see [AFP92, AB11] for an introduction to Horn Functions. A learning algorithm for Horn Functions, that we call the AFP Algorithm, was first in [AFP92] presented (there it was called the Horn1 Algorithm). A normal form for Horn functions was presented in [AB11], and it was shown that it is canonical, that is every Horn function has only one formula in this normal form, up to the order of clauses. They also presented an algorithm that given a Horn Formula, outputs its canonical normal formula. Moreover they proved that the AFP Algorithm outputs the canonical normal form of the target function.

We have also independently discovered this canonical normal form and in section 2 we prove that the output of the AFP Algorithm is in this form. We believe that our presentation is briefer than [AB11].

It was presented in [Bal05, section 7] as a frequently asked unanswered question, that whether it improves the time and query complexities of the AFP Algorithm, if we change the algorithm, so that, after every negative counterexample, it makes all (and not just the first) possible refinements. Interestingly, [AFP92] had already briefly answered this question, negatively. Here we provide a detailed proof for that.

1.1 Notation
We follow the notation of [AB11]. Additionally we denote the variables with letters from the beginning of the alphabet, and the set of variables with \( \sigma \). Logical False will be denoted by \( F \). Subset and proper subset are denoted by \( \subseteq \) and \( \subset \) respectively. With function or formula we mean a Horn function or formula respectively. With \( a^n \) where \( a \in \{0, 1\} \), we mean a string of \( a \)'s of length \( n \).
2 Canonical normal form

The AFP Algorithm [AFP92], learns a function via a learning protocol.

\begin{algorithm}
\begin{algorithmtext}
1. AFP
2. $O \leftarrow ()$  \hfill *List of sets
3. $P \leftarrow []$  \hfill *Set of positive counterexamples
4. $H \leftarrow T$  \hfill *Set the $H$ equal to True.
5. while $\text{equal}(H, H^*) = (\text{"no"}, y)$ do
6. \hspace{1em} if $y \not\in H$ then  \hfill *a positive counterexample
7. \hspace{2em} $P \leftarrow P \cup \{y\}$
8. \hspace{1em} else  \hfill *a negative counterexample
9. \hspace{2em} for the first $s \in O$, such that $\text{member}(s \land y) = \text{"no"}$ and $[s \land y] \subset [s]$ do
10. \hspace{3em} $s \leftarrow s \land y$
11. \hspace{2em} if none is found then
12. \hspace{3em} Add $y$ as the last element in $O$
13. \hspace{2em} Set $H$ as the conjunction of all
14. \hspace{2em} $\{s \rightarrow a \mid s \in O, a \in \sigma \cup \{F\}$ And $\forall z \in P \ z \vdash s \rightarrow a\}$
15. output $H$
\end{algorithmtext}
\caption{the AFP Algorithm}
\end{algorithm}

The following properties define a Normal Form for formulas.

**Definition 1 (Normal Form).** A formula $H = \bigwedge_i (\alpha_i \rightarrow \beta_i)$ is in normal form, if

1. $\alpha_i \neq \alpha_j$ for $i \neq j$,
2. $\alpha_i \subset \beta_i$,
3. $\forall i, j \alpha_j \vdash (\alpha_i \rightarrow \beta_i)$.

Compare this Definition with [AB11]. Section 3, Definition 4.

**Theorem 1.** Each function has at least one representing formula in normal form.

**Proof.** By definition every function has a formula. We present a polynomial time algorithm that returns a normal form presentation of every input formula.

Repeat until no more changes are made:

a) Merge the clauses with the same antecedents.

b) For each clause $\alpha \rightarrow \beta$, if there is a clause $\kappa \rightarrow \gamma$ such that $\kappa \subset \alpha$, then replace $\alpha \rightarrow \beta$ with $\alpha \cup \gamma \rightarrow \beta$

At the end,

c) Delete all clauses $\alpha \rightarrow \beta$ that $\beta \subseteq \alpha$. 

2
d) Change all clauses $\alpha \rightarrow \beta$ to $\alpha \rightarrow \alpha \cup \beta$.

As there are finite variables, and at each iteration only some clauses are merged or the size of some antecedents increases, the iteration (and algorithm) end in polynomial time. Properties 1, 2, and 3 of the normal form will be fulfilled by a, (c and d) and b respectively.

**Theorem 2.** Let $H^* = \bigwedge_i (\alpha_i \rightarrow \beta_i)$ be a normal form formula of the target function for the AFP Algorithm. While there are still antecedents of $H^*$ which are equal to no antecedent of $H$ (equivalently to no $s \in O$), there will be a negative counterexample.

**Proof.** Let $\alpha$ be such an antecedent, then $\alpha \not\models H^*$. If there are no clauses $s \rightarrow \gamma$ from $H^*$, that $s \subset \alpha$, then $\alpha \not\models H$ as a negative Counterexample. Else, on line 13, $\gamma \supseteq \bigcup_i \alpha_i \subseteq s \beta_i \setminus s$, and there are finite positive counterexamples until the equality for all such $s$ holds. And then, following the definition of normal form (definition 1-3), $\gamma \subset \alpha$, so $\alpha \models s \rightarrow \gamma$ and $\alpha \models H$ will be the negative counterexample. □

Let $H^* = \bigwedge_i (\alpha_i \rightarrow \beta_i)$ be a normal form formula for the target function. Then the list $O = (s_i)$ in the AFP Algorithm, has the property that

\[
\text{At each instant, } \forall i \exists k \ s_i \not\models \alpha_k \rightarrow \beta_k \text{ And } (s_j \not\models \alpha_k \rightarrow \beta_k \Rightarrow j \geq i) \quad (1)
\]

**Theorem 3** (Property of $O$). The above property holds.

Compare this with [AB11], Lemma 16.

**Proof.** When an $s_j \in O$ is added or after it is refined, it holds that $s_j$ violates at least one clause $\alpha \rightarrow \beta$ from $H^*$, such that $\forall j < i \ a \not\in s_j$. That holds at least untill the next time that $s_j$ is refined. □

**Theorem 4.** The AFP Algorithm returns a normal form formula for the target function.

**Proof.** Fix some normal formula $H^*$ of the target function. Via each negative counterexample, some members of a set in $O$ are removed (Line 10), or a new set is added to $O$ (Line 12). Theorem 3 implies that the size of $O$ is no more than the number of clauses of $H^*$. Following Theorem 2 while there are antecedents of $H^*$ that are not identical with some set in $O$, negative counterexamples will be given. As it is all finite, $O$ will be equal to set of the antecedents of $H^*$ and the conclusions will corrected via positive counterexamples. So the algorithm ends, and it then holds that $H = H^*$. □

**Theorem 5.** Each function has exactly one normal form, up to the order of clauses.

**Proof.** It follows from the free choice of $H^*$ in proof of theorem 4. □
More than one refinement with each negative counterexample

It seems an appealing question, that whether the AFP algorithm would be more efficient, in runtime and number of queries, had it tried to refine more that one set of $O$ with each negative counterexample, so far that [Bal05 section 7] considers it as a frequently asked unanswered question. Interestingly [AFP92] had already briefly answered this question: “Overzealous refinement may result in several examples in $O$ violating the same clause of $H^*$. To avoid this, whenever a new negative counterexample could be used to refine several examples in the sequence $O$, only the first among these is refined.”

Here we provide a proof for this answer.

Besides the AFP algorithm (Horn1), [AFP92] presents an improvement for it, Horn2, which is more efficient in determining the conclusions, but makes the same as Horn1 in finding the antecedents. We will show that the answer is negative for both versions of the algorithm (for worst-case runtime). Let AFP* be the same as AFP algorithm, but “the first” in line 9 be replaced by “for all”.

Throughout this section, let $H^* = \bigwedge_i (\alpha_i \rightarrow \beta_i)$ be the canonical normal form of the target function, such that antecedents with smaller size, have smaller index. We say that the sequence of counterexamples in a run of the AFP Algorithm is ordered (relative to $H^*$) if it is as follows.

The sequence of counterexamples is a succession of $m$ subsequences, where $m$ is the number of clauses of $H^*$. Let $(z_j)$ be the $i$th subsequence. Then each $z_j$ is a superset of $\alpha_j$ but not a superset of $\beta_j$ and not a superset of any $\alpha_k$ such that $\alpha_k \not\subset \alpha_j$; moreover $z_j \cap z_j \subset z_j$, and the last element of $(z_j)$ is $\alpha_i$. Note that in this definition there is no restriction on positive counterexamples or their order relative to negative counterexamples.

It is straightforward to show that for every target function, there exists at least one ordered sequence of counterexamples. An example will be given in the proof of theorem 7.

Lemma 6. For any Horn Formula, if the (negative) counterexamples are ordered, then the AFP and AFP* algorithms perform exactly the same operations.

Proof. By induction we show that after round $i$, $s_i = \alpha_i$ and will not be changed. So at round $i + 1$, for any of the negative counterexamples $z_k$ if the test of line 9 is true for $s_j = \alpha_j$, $j < i$, then it should be refined, but then as the index of clauses of $H^*$ is ordered by the size of antecedents, Property of $O$ (theorem 4) cannot be satisfied.

Therefore during round $i$ only $s_i$ can be added or refined and as the last negative counterexample in round $i$ is $\alpha_i$, with a similar argument, $s_i$ will be refined to $\alpha_i$. □

Theorem 7. The worst-case time, equivalence and membership query complexities of the AFP* algorithm is not better than that of the AFP algorithm.

Proof. For the target functions from the class $\{f_n\}$ defined below, if the counterexamples are given as described below, by lemma 6 the AFP* Algorithm and AFP algorithm

1 Here each subsequence simply consists of consecutive parts of the sequence.
perform exactly the same operations, because the (negative) counterexamples are ordered. But the AFP algorithm with this setting will reach its worst-case time, equivalence and membership query complexities on general input (Compare with [AFP92] theorem 2, we do not repeat that argument here). The result follows.

The function $f_n$, is defined over the set of $2n + 1$ variables $\sigma = \{a_1, \ldots, a_{2n+1}\}$, and its canonical normal form has $m = n$ clauses:

$$f_n = \bigwedge_{1 \leq i \leq n} (a_i \rightarrow a_{2n+1})$$

The $i$th subsequence of negative counterexamples are

$$y_1 = 0^{i-1}10^{n-i}1^n0$$
$$y_j = 0^{i-1}10^{n-j+1}0^{j-1}0$$
$$y_{n+1} = 0^{i-1}10^n0$$

After any $y_j$ a sequence of positive counterexamples ($w_k$) will be given:

$$w_k = \sigma - \{d_k\}$$

where $d_k$ is the $k$th variable such that $d_k \notin y_j$ and $d_k \neq a_{2n+1}$. □

It is straightforward to give classes of functions and counterexamples for which the AFP has its general worse-case runtime and query complexities while the AFP* makes substantially worse (roughly speaking, because property of $O$ (theorem 3) no more holds). But we find theorem 7 enough and more interesting. By a similar argument one can get similar results for the Horn2 Algorithm.

References

[AB11] Marta Arias and José L Balcázar. Construction and learnability of canonical horn formulas. *Machine Learning*, 85(3):273–297, 2011.

[AFP92] Dana Angluin, Michael Frazier, and Leonard Pitt. Learning conjunctions of horn clauses. *Machine Learning*, 9(2-3):147–164, 1992.

[Bal05] Jose L Balcázar. Query learning of horn formulas revisited. *Giulia Battilotti, Paola Zizzi Logical Characterizations of PK and NPK over an Arbitrary Structure K*. 30 Olivier Bournez, Felipe Cucker, Paulin Jacobé de Naurois, Jean-Yves Marion Fuzzy Interval-valued Processes Algebra.........................44, page 1, 2005.