Mach Learn (2011) 85:273–297
DOI 10.1007/s10994-011-5248-5

Construction and learnability of canonical Horn formulas

Marta Arias · José L. Balcázar

Received: 13 January 2010 / Accepted: 16 March 2011 / Published online: 3 June 2011

Abstract We describe an alternative construction of an existing canonical representation for definite Horn theories, the Guigues-Duquenne basis (or GD basis), which minimizes a natural notion of implicational size. We extend the canonical representation to general Horn, by providing a reduction from definite to general Horn CNF. Using these tools, we provide a new, simpler validation of the classic Horn query learning algorithm of Angluin, Frazier, and Pitt, and we prove that this algorithm always outputs the GD basis regardless of the counterexamples it receives.

Keywords Query learning · Horn formulas · Canonical representation

1 Introduction

The research area of Query Learning studies problems of learning target concepts, often modeled as Boolean functions, through learning algorithms that interact with the target concept through a variety of protocols. Positive results consist usually in algorithms that learn a given concept class, via a given protocol, within certain query or time resources.

The most successful protocol so far is via Membership and Equivalence queries. Three major algorithms for this model are the L* algorithm for learning regular sets in terms of deterministic finite automata (Angluin 1987); the algorithm that learns monotone DNF (closely related to its corresponding PAC version by Valiant 1984); and a sophisticated evolution of

Editor: Avrim Blum.

Work partially supported by MICINN projects SESAAME-BAR (TIN2008-06582-C03-01) and FORMALISM (TIN2007-66523).

M. Arias (✉)
LARCA Research Group, Departament LSI, Universitat Politècnica de Catalunya, Barcelona, Spain
e-mail: marias@lsi.upc.edu

J.L. Balcázar
Departamento de Matemáticas, Estadística y Computación, Universidad de Cantabria, Santander, Spain
e-mail: joseluis.balcazar@unican.es

the latter which allows for learning Horn formulas (Angluin et al. 1992) (the two variants of the algorithm were named HORN and HORN1 in Angluin et al. 1992 but most researchers chose to call it "AFP algorithm" after the initials of the authors' names).

Each of these has had a number of interesting applications. In particular, the AFP algorithm turned out to be crucial for developments in other areas such as Knowledge Compilation (Selman and Kautz 1996), Intuitionistic Logic Programming (Gaintzarain et al. 2005) or even Databases (Hermo and Lavín 2002; Kivinen and Mannila 1995), and similar algorithms exist for learning from entailment (Frazier and Pitt 1993) and for learning formulas in certain description logics (Frazier and Pitt 1996). Related research questions, with answers to a couple of them, are proposed in Balcázar (2005) and Hermo and Lavín (2002); one of these contributions is an alternative approach to proving the correctness of the AFP algorithm.

A number of questions remain open regarding the AFP algorithm. For example, is it optimal? We believe that it can be improved and simplified, and this is the main objective driving this work: we want a better algorithm that is able to learn conjunctions of Horn clauses, with provably better bounds (at least for certain subclasses of Horn CNF). This question remains open, but in this work we are able to shed some light into the following important fact about the dynamics of the AFP algorithm. Along a run of the algorithm, queries made by the algorithm depend heavily upon the counterexamples selected as answers to the previous queries. It is therefore natural to expect the outcome of the algorithm to depend on the answers received along the run. However, attempts at providing an example of such behavior consistently fail.

In this paper we prove that such attempts must in fact fail: we describe a canonical representation of Horn functions in terms of implications, and show that the AFP algorithm (Angluin et al. 1992) always outputs this particular representation. It turns out that this canonical representation (with minor differences) has been rediscovered in different fields in the past. To the best of our knowledge, the earliest description is due to Maier (1980) who in 1980 studied minimality of covers of sets of functional dependencies in the field of Theory of Databases. Later on in 1986, in the field of Formal Concept Analysis, the *Guigues-Duquenne basis* or GD basis (Guigues and Duquenne 1986; Wild 1994) was described. Finally, very recently, Bertet and Monjardet (2010) explores the history of this and other aspects that we study in the paper.

In order to prove this fact about the output of the AFP algorithm (Angluin et al. 1992), we provide first a new characterization of the GD basis, plus a number of properties related to it; then we provide a new correctness proof of the AFP algorithm, for the particular case of definite Horn targets. From the necessary invariants, we infer that, indeed, the algorithm is constructing the GD basis. This proof (as well as the algorithm) has the additional property that it can be translated into learning sublattices of an arbitrary lattice, so that it may apply to other concept classes that exhibit lattice structure. However, we use Horn CNF here, since this is more standard in the field of query learning theory; the translation to other lattice structures is almost immediate.

Since the GD basis is defined for definite Horn formulas only, to study the original learning algorithm we must extend the notion of GD basis to general Horn formulas. We do it by means of a reduction from general to definite Horn formulas. This reduction allows us to lift the characterization of the output of AFP as the generalized GD basis.

Some of the technical lemmas and theorems in this paper are based on previous concepts and results of Wild (1994), Guigues and Duquenne (1986), Maier (1980); we credit this fact appropriately throughout this presentation. As a general overview, we have adopted the following: the "bullet" operator ($^\bullet$) of Sect. 3.1 is directly taken from Wild (1994), the "star" operator $(^\star)$ is standard in the field of study of *Closure Spaces*, and the GD basis comes from the *Formal Concept Analysis* literature.

Most of the results in Sect. 4.1 were announced in Arias and Balcázar (2008); the remaining results are from Arias and Balcázar (2009).

2 Preliminaries

We start by reviewing basic definitions and results from Horn logic. We work within the standard framework in propositional logic, where one is given an indexable set of propositional variables of cardinality n, Boolean functions are subsets of the Boolean hypercube $\{0,1\}^n$, and these functions are represented by logical formulas over the variable set in the standard way. Binary strings of length n assign a Boolean value for each variable, and are therefore called assignments; given any Boolean function or formula H, the fact that assignment x makes it true (or "satisfies" it) is denoted $x \models H$. Following the standard overloading of the operator, $H \models H'$ means that, for every assignment x, if $x \models H$ then $x \models H'$. Assignments are partially ordered bitwise according to $0 \leq 1$ (the usual partial order of the hypercube); the notation is $x \leq y$. We employ the standard notation $\square$ as a formula representing the constant 0 (false) Boolean function; thus $x \not\models \square$ for every x.

A literal is a variable or its negation. A conjunction of literals is a *term*, and if none of the literals appears negated it is a *positive term*, also often referred to as a *monotone term* or *monotone conjunction*. We often identify positive terms and mere sets of variables; in fact, we switch back and forth between set-based notation and assignments. We denote terms, or equivalently subsets of variables, with Greek letters $(\alpha, \beta, \ldots)$ and assignments with letters from the end of the alphabet $(x, y, z, \ldots)$. We may abuse notation at times and it should be understood that if we use a subset α when an assignment is expected, it is to be interpreted as the assignment that sets to 1 exactly those variables in α. We denote this explicitly when necessary by $x = [\alpha]$. Similarly, if we use an assignment x where a subset of variables is expected, it is to be understood that we mean the set of variables that are set to 1 in x. We denote this explicitly by $\alpha = [x]$. Clearly, we have a bijection between sets of propositional variables and assignments, and $x = [\![x]\!]$ and $\alpha = [\![\alpha]\!]$ for all assignments x and variable sets α.

The following lemma is a direct consequence of the definition of satisfiability over positive conjunctions α:

Lemma 1 $x \models \alpha$ *iff* $\alpha \subseteq [x]$ *iff* $[\alpha] \leq x$.

2.1 Horn logic

In this paper we are only concerned with Horn functions, and their representations using conjunctive normal form (CNF). A Horn CNF formula is a conjunction of Horn clauses. A clause is a disjunction of literals. A clause is *definite Horn* if it contains exactly one positive literal, and it is *negative* if all its literals are negative. A clause is *Horn* if it is either definite Horn or negative. Since in this paper we are dealing with Horn functions only, we drop the "Horn" adjective frequently.

Horn clauses are generally viewed as implications where the negative literals form the antecedent of the implication (a positive term), and the singleton consisting of the positive literal, if it exists, forms the consequent of the clause. Note that both can be empty; if the consequent is empty, then we are dealing with a negative Horn clause, denoted $\alpha \rightarrow \square$.

This is equivalent to having $\square$ as one more literal in every negative Horn clause, which is harmless as the Boolean 0 is an identity element for disjunction, and then handling it (somewhat artificially) as a positive literal.

We stick to the wording "Horn CNF" even though we take the term CNF in a rather general meaning of "Boolean polynomial" or "conjunction of disjunctions", not literally the normal form of conjunction of maxterms; this widening of the term CNF is widespread in Horn query learning literature, and will lead to no ambiguity. In fact, we extend it a bit further by some degree of language abuse, as we allow our representations of Horn CNF to deviate slightly from the standard in that we represent clauses sharing the same antecedent together in one implication. Namely, an implication $\alpha \rightarrow \beta$, where both α and β are possibly empty sets of propositional variables is to be interpreted as the conjunction of definite Horn clauses $\bigwedge_{b\in\beta} \alpha \rightarrow b$ if $\beta \neq \emptyset$, and as the negative clause $\alpha \rightarrow \square$ if $\beta = \emptyset$.[1] A semantically equivalent interpretation is to see both sets of variables α and β as positive terms; the Horn formula in its standard form is obtained by distributivity over the variables of β. In analogy to the standard notation, we will continue to use the symbol $\square$ for an empty set in the conclusion of an implication. Of course, any result that holds for Horn formulas in implicational form with no other restrictions also holds for the clausal representation unless it explicitly depends of the implications proper, such as counting the number of implications, as we will do below.

Example 1 Let $(\neg a \vee b) \wedge (\neg a \vee c) \wedge (\neg a \vee \neg b)$ be a standard representation of a Horn function over the variable set $\{a, b, c\}$. In standard form this expression becomes $(a \rightarrow b) \wedge (a \rightarrow c) \wedge (ab \rightarrow \square)$. In our compact representation we write $(a \rightarrow bc) \wedge (ab \rightarrow \square)$. All these represent the same Boolean function. When clear from context, we drop the set representation in our examples.

We refer to our generalized notion of conjunction of clauses sharing the antecedent as *implication* as in the early paper by Guigues and Duquenne (1986). In their original paper, Angluin et al. (1992) refer to implications as *meta-clauses*, although in their intuitive description they hint at the additional condition that all clauses with the same antecedent should be forced into a single implication. We do allow, in principle, several implications of the same basis to share their antecedent. In our own previous work (Arias and Balcázar 2008) we have used the term *para-clauses*; and other works (Arias and Khardon 2002; Arias et al. 2007) refer to implications as *multi-clauses*. In fact, the term "implication" predates all these usages, except the earlier paper by Maier (1980) which, however, discussed its contribution in terms of functional dependencies, which share many of their properties with implications but are defined differently. The term *clause* retains its classical meaning (namely, a disjunction of literals). Notice that an implication may not be a clause, e.g. $(a \rightarrow bc)$ corresponds in classical notation to the formula $\neg a \vee (b \wedge c)$. Thus, $(a \rightarrow bc)$, $(ab \rightarrow c)$ and $(ab \rightarrow \square)$ are Horn implications but only the latter two are Horn clauses. Furthermore, we often use sets to denote conjunctions, as we do with positive terms, also at other levels: a generic (implicational) CNF $\bigwedge_i (\alpha_i \rightarrow \beta_i)$ is often denoted in this text by $\{(\alpha_i \rightarrow \beta_i)\}_i$. Parentheses are mostly optional and generally used for ease of reading.

[1] A reviewer suggested to point out that this differs from an alternative, older interpretation (Wang 1960), nowadays obsolete, in which $\alpha \rightarrow \beta$ represents the clause $(\neg x_1 \vee \cdots \vee \neg x_k \vee y_1 \vee \cdots \vee y_{k'})$, where $\alpha = \{x_1, \ldots, x_k\}$ and $\beta = \{y_1, \ldots, y_{k'}\}$. Though identical in syntax, the semantics are different; in particular, ours can only represent a conjunction of Horn clauses whereas the other represents a possibly non-Horn clause.

An assignment $x \in \{0,1\}^n$ satisfies the implication $\alpha \to \beta$, denoted $x \models \alpha \to \beta$, if it either falsifies the antecedent or satisfies the consequent, that is, $x \not\models \alpha$ or $x \models \beta$ respectively, where now we are interpreting both α and β as positive terms.

It is important to note that one has to be careful with the case $\beta = \emptyset$: as indicated above, we act as if $\square$ is present and handle it as a positive literal. Indeed, in this case the semantic interpretation is as follows: $x \models \alpha \to \emptyset$ if $x \not\models \alpha$ (since $x \not\models \square$ always by definition). Notice that if we were to translate directly $x \models \alpha \to \emptyset$, if $x \not\models \alpha$ or $x \models \emptyset$, we would get that this is always true since, by Lemma 1, we have that $x \models \emptyset$ for all x. This is why we prefer to stick to the notation $\alpha \to \square$ for negative implications.

Not all Boolean functions are Horn. The following semantic characterization is a well-known classic result of Horn (1956), McKinsey (1943), proved in the context of propositional Horn logic e.g. in Khardon and Roth (1996):

Theorem 1 *A Boolean function admits a Horn CNF basis if and only if the set of assignments that satisfy it is closed under bit-wise intersection.*

A Horn function admits several syntactically different Horn CNF representations; in this case, we say that these representations are equivalent. Such representations are also known as *theories* or *bases* for the Boolean function they represent. The *size* of a Horn function is the minimum number of clauses that a Horn CNF representing it must have. The *implication size* of a Horn function is defined analogously, but allowing formulas to have implications instead of clauses. Clearly, every clause is an implication, and thus the implication size of a given Horn function is always at most that of its standard size as measured in the number of clauses.

Horn CNF representations may as well include unnecessary implications. We will need to take this into account: an implication in a Horn CNF H is *redundant* if it can be removed from H without changing the Horn function represented. A Horn CNF is *irredundant* or *irreducible* if it does not contain any redundant implication. Notice that an irredundant H may still contain other sorts of redundancies, such as consequents larger than strictly necessary. Such redundancies are not contemplated in this paper.

2.2 Definite Horn functions

We describe the well-known method of *forward chaining* for definite Horn functions (Chang and Lee 1973). Notice that it directly extends to our compressed representation where consequents of clauses can contain more than one variable. Given a definite Horn CNF $H = \{\alpha_i \to \beta_i\}_i$ and a subset of propositional variables α, we construct chains of subsets of propositional variables $\alpha = \alpha^{(0)} \subset \alpha^{(1)} \subset \cdots \subset \alpha^{(k)} = \alpha^\star$. Each $\alpha^{(i)}$ with $i > 0$ is obtained from its predecessor $\alpha^{(i-1)}$ in the following way: if $[\alpha^{(i-1)}]$ satisfies all implications in H, then the process can stop with $\alpha^{(i-1)} = \alpha^\star$. If, on the other hand, $[\alpha^{(i-1)}]$ violates some implication $\alpha_j \to \beta_j \in H$, then $\alpha^{(i)}$ is set to $\alpha^{(i-1)} \cup \beta_j$.

Similarly, one can construct an increasing chain of assignments $x = x^{(0)} < x^{(1)} < \cdots < x^{(k)} = x^\star$ using our bijection $\alpha^{(i)} = [x^{(i)}]$ and $x^{(i)} = [\alpha^{(i)}]$ for all i.

Example 2 Let $H = \{a \to b, b \to c, ad \to e\}$. To compute $a^\star$ we do forward chaining: $a \subset ab \subset abc = a^\star$, since abc satisfies all clauses in H. Analogously, one could do this using assignments instead of variable subsets: $10000 < 11000 < 11100 = 10000^\star$.

See Chang and Lee (1973), Kleine Büning and Lettmann (1999) as a general reference for the following well-known results. Theorem 3(1) in particular refers to the fact that the forward chaining procedure is a sound and complete deduction method for definite Horn CNF.

Theorem 2 *The objects $x^\star$ and $\alpha^\star$ are well-defined and computed by the forward chaining procedure regardless of the order in which implications in H are chosen. The forward chaining process can be implemented to run in linear time in $|H|$ and n. Moreover, $x^\star$ and $\alpha^\star$ depend only on the underlying function being represented, and not on the particular choice of representation H; and for each $x^{(i)}$ and $\alpha^{(i)}$ along the way, we have that $(x^{(i)})^\star = x^\star$ and $(\alpha^{(i)})^\star = \alpha^\star$.*

Theorem 3 *Let h be a definite Horn function, let α be an arbitrary variable subset, and x an arbitrary assignment. Then,*

1. *$h \models \alpha \to b$ if and only if $b \in \alpha^\star$,*
2. *$x = x^\star$ if and only if $x \models h$, and*
3. *$x^\star = \bigwedge\{y | x \leq y$ and $y \models h\}$.*

Proof We do not prove the first part, as it is standard and appears in many places in the literature, e.g. Chang and Lee (1973), Kleine Büning and Lettmann (1999).

For the second part, it suffices to observe that, by construction, $x = x^\star$ if and only if it does not violate any clause of h.

To prove the third part notice that, from the first part of the theorem, we know that $h \models [x] \to [x^\star]$, and thus $y \models [x] \to [x^\star]$. We get that $y \geq x$ implies $y \geq x^\star$, for all such y, concluding that $\bigwedge\{y|x \leq y$ and $y \models h\} \geq x^\star$. It is not hard to see that $x^\star$ satisfies the conditions $x \leq x^\star$ and $x^\star \models h$, therefore $x^\star \in \{y|x \leq y$ and $y \models h\}$, and thus $x^\star \geq \bigwedge\{y|x \leq y$ and $y \models h\}$. Both inequalities together imply the required equality. □

Closure operator and equivalence classes It is easy to see that the $\star$ operator is extensive (that is, $x \leq x^\star$ and $\alpha \subseteq \alpha^\star$), monotonic (if $x \leq y$ then $x^\star \leq y^\star$, and if $\alpha \subseteq \beta$ then $\alpha^\star \subseteq \beta^\star$) and idempotent ($x^{\star\star} = x^\star$, and $\alpha^{\star\star} = \alpha^\star$) for all assignments x, y and variable sets α, β; that is, $\star$ is a closure operator (Arias and Balcázar 2008). Thus, we refer to $x^\star$ as the *closure* of x w.r.t. a definite Horn function f.

It should be always clear from the text with respect to what definite Horn function we are taking the closure, hence it is mostly omitted from the notation used. If we need to make this explicit, we may use the notation x^H to denote the closure w.r.t. the definite Horn CNF H, or x^h to denote the closure w.r.t. the definite Horn function h. An assignment x is said to be *closed* iff $x^\star = x$, and similarly for variable sets. By Theorem 3(2), closed assignments satisfy all implications, and assignments that are not closed must falsify some implication.

This closure operator induces a partition over the set of assignments $\{0, 1\}^n$ in the following straightforward way: two assignments x and y belong to the same class if $x^\star = y^\star$. This notion of equivalence class carries over as expected to the power set of propositional variables: the subsets α and β belong to the same class if $\alpha^\star = \beta^\star$. It is worth noting that each equivalence class consists of a possibly empty set of assignments that are not closed and a single closed set, its representative.

Moreover, the notion of equivalence classes carries over to implications by identifying an implication with its antecedent. Thus, two implications belong to the same class if their antecedents have the same closure. Thus, the class of an implication $\alpha \to \beta$ is, essentially, $\alpha^\star$.

Example 3 This example is taken from Guigues and Duquenne (1986). Let $H = \{e \to d, bc \to d, bd \to c, cd \to b, ad \to bce, ce \to ab\}$. Thus, the propositional variables are a, b, c, d, e, f. The following table illustrates the partition induced by the equivalence classes on the implications of H. The first column is the implication identifier, the second column is the implication itself, and the third column corresponds to the class of the implication. As one can see, there are three equivalence classes: one containing the first implication, another one containing implications 2, 3, and 4; and a final one containing implications 5 and 6.

1	$e \to d$	ed
2	$bc \to d$	bcd
3	$bd \to c$	bcd
4	$cd \to b$	bcd
5	$ad \to bce$	$abcde$
6	$ce \to ab$	$abcde$

3 The Guigues-Duquenne basis for definite Horn

In this section we characterize and show how to build a canonical basis for definite Horn functions that is of minimum implication size. It turns out that this canonical form is, in essence, the Guigues-Duquenne basis (the *GD basis*) which was introduced in Guigues and Duquenne (1986). Here, we introduce it in a form that is, to our knowledge, novel, although it is relatively close to the approaches of Maier (1980), Wild (1994).

Our construction is based on the notion of *saturation*, a notion that has been used already in the context of propositional Horn functions (Arias and Balcázar 2008; Arias et al. 2006), as well as in the context of Inductive Logic Programming (Rouveirol 1994). Informally, by *saturating* a representation we mean adding all the information that is missing from the representation but can be deduced from it. Let us define *saturation* more formally and then prove several interesting properties that serve as the basis for our work.

Definition 1 Let B be a definite Horn CNF. We say that an implication $\alpha \to \beta$ is *left-saturated* with respect to B if $[\alpha] \models B$ and $\alpha \not\models \alpha \to \beta$ (or, equivalently, $\beta \not\subseteq \alpha$). We say that an implication $\alpha \to \beta$ is *right-saturated* with respect to B if $\beta = \alpha^\star$, where the closure is taken with respect to B. Finally, an implication is *saturated* w.r.t. B if it is left- and right-saturated w.r.t. B.

Definition 2 Let $B = \{\alpha_i \to \beta_i\}_i$ be a definite Horn CNF. Then, B is *left-saturated* if every implication $\alpha_i \to \beta_i$ of B is left-saturated w.r.t. $B \setminus \{\alpha_i \to \beta_i\}$. This is equivalent to the following conditions:

1. $[\alpha_i] \not\models \alpha_i \to \beta_i$, for all i;
2. $[\alpha_i] \models \alpha_j \to \beta_j$, for all $i \neq j$.

Notice that the case where B contains a single implication $\alpha \to \beta$, the notion of left-saturation is reduced to irredundancy or, equivalently, the fact that $\beta \not\subseteq \alpha$.

Definition 3 Let $B = \{\alpha_i \to \beta_i\}_i$ be a definite Horn CNF. Then, B is *right-saturated* if $\beta_i = \alpha_i^\star$ for all i, i.e., all of its implications are right-saturated w.r.t. itself. Accordingly, we denote right-saturated bases with $\{\alpha_i \to \alpha_i^\star\}_i$.

Definition 4 We say that a definite Horn CNF B is *saturated* if it is left- and right-saturated.

Example 4 Let $H = \{a \to b, b \to c, ad \to e\}$.

– H is not left-saturated: for example, the antecedent of $ad \to e$ is such that $[ad] \not\models a \to b$. One can already see that by including b in the antecedent of the third clause, one avoids this particular violation.
– H is not right-saturated because $a^\star = abc$ and, for example, the implication $a \to b$ is missing ac from its right-hand-side.
– The equivalent $H' = \{a \to abc, b \to bc, abcd \to abcde\}$ is saturated.

Lemma 2 *If B is left-saturated then B is irredundant.*

Proof If $|B| = 1$ left-saturation reduces to irredundancy by definition. Otherwise, suppose that B, where $|B| \geq 2$ is left-saturated but redundant, and so it contains a redundant implication $\alpha \to \beta$. By definition of left-saturation, we have that $[\alpha] \not\models \alpha \to \beta$ and thus $[\alpha] \not\models B$, and that $[\alpha] \models \alpha' \to \beta'$ for implications $\alpha' \to \beta'$ other than $\alpha \to \beta$. Therefore $[\alpha] \models B \setminus \{(\alpha \to \beta)\}$, contradicting the fact that $\alpha \to \beta$ is redundant in B since $B \not\equiv B \setminus \{\alpha \to \beta\}$. □

Lemma 3 *Let $B = \{\alpha_i \to \beta_i\}_i$ be an irredundant definite Horn CNF. Then, $[\alpha_i] \not\models \alpha_i \to \beta_i$ for all i.*

Proof We are dealing with definite implications, therefore no β_i can be empty. Moreover, the fact that B is irredundant implies that some variable in β_i must be from outside α_i, otherwise the implication would be redundant. Thus we have that $[\alpha_i] \models \alpha_i$ but $[\alpha_i] \not\models \beta_i$ because the variable of β_i outside of α_i is set to 0 in $[\alpha_i]$ by definition. □

Lemma 4 *Let $B = \{\alpha_i \to \alpha_i^\star\}_i$ be an irredundant and right-saturated definite Horn CNF. Then, the following hold for all $i \neq j$*

1. $\alpha_i \neq \alpha_j$;
2. $\alpha_i \subseteq \alpha_j \;\Rightarrow\; \alpha_i \subset \alpha_j$.

Proof Follows simply from the fact that if $\alpha_i = \alpha_j$ with $i \neq j$, then the implications $\alpha_i \to \alpha_i^\star$ and $\alpha_j \to \alpha_j^\star$ would, in fact, be the same implication, and thus B is redundant. The second statement is a direct consequence of the first. □

Lemma 5 *Let $B = \{\alpha_i \to \alpha_i^\star\}_i$ be an irredundant, right-saturated definite Horn CNF. Then, B is left-saturated if and only if the following implication is true for all $i \neq j$:*

$$\alpha_i \subset \alpha_j \quad \Rightarrow \quad \alpha_i^\star \subseteq \alpha_j.$$

Moreover, in that case a stronger statement holds:

$$\alpha_i \subset \alpha_j \quad \Rightarrow \quad \alpha_i^\star \subset \alpha_j.$$

Proof Assume that the implication $\alpha_i \subset \alpha_j \;\Rightarrow\; \alpha_i^\star \subseteq \alpha_j$ holds for all $i \neq j$. We show that B must be left-saturated, namely, that the following equivalence must hold: $i = j \;\Leftrightarrow\; [\alpha_i] \not\models \alpha_j \to \alpha_j^\star$. If $i = j$, Lemma 3 guarantees that $[\alpha_i] \not\models \alpha_i \to \alpha_i^\star$, since B is assumed to be irredundant. For the other direction, assume by way of contradiction that $[\alpha_i] \not\models \alpha_j \to \alpha_j^\star$, for some $i \neq j$. This happens if $[\alpha_i] \models \alpha_j$ (namely, $\alpha_j \subseteq \alpha_i$ by Lemma 1 and thus $\alpha_j \subset \alpha_i$ by Lemma 4) but $[\alpha_i] \not\models \alpha_j^\star$ (namely, $\alpha_j^\star \not\subseteq \alpha_i$). But this contradicts the implication $\alpha_i \subset \alpha_j \Rightarrow \alpha_i^\star \subseteq \alpha_j$.

Conversely, assume that $\alpha_i \subset \alpha_j$ for some $i \neq j$, and that B is left-saturated, which implies by definition that $[\alpha_j] \models \alpha_i \to \alpha_i^\star$: then $\alpha_i^\star \subseteq \alpha_j$ follows. Furthermore, as B is irredundant, α_j cannot be closed, but $\alpha_i^\star$ is, so that they cannot coincide and the inclusion is proper. □

The following Lemma is a variant of a result of Wild (1994) translated into our notation. We include the proof which is, in fact, missing from Wild (1994).

Lemma 6 *Let $B = \{\alpha_i \to \alpha_i^\star\}_i$ be a saturated definite Horn CNF. Then for all i and γ it holds that $(\gamma \subseteq \alpha_i$ and $\gamma^\star \subset \alpha_i^\star) \Rightarrow \gamma^\star \subseteq \alpha_i$.*

Proof Let us assume that the conditions of the implication are true, namely, that $\gamma \subseteq \alpha_i$ and $\gamma^\star \subset \alpha_i^\star$. We proceed by cases: if γ is closed, then $\gamma^\star = \gamma$ and the implication is trivially true. Otherwise, γ is not closed. Let $\gamma = \gamma^{(0)} \subset \gamma^{(1)} \subset \cdots \subset \gamma^{(k)} = \gamma^\star$ be one of the possibly many series of elements constructed by the forward chaining procedure described in Sect. 2.2. We argue that if $\gamma^{(l)} \subseteq \alpha_i$ and $\gamma^{(l)} \subset \gamma^\star$, then $\gamma^{(l+1)} \subseteq \alpha_i$ as well. By repeatedly applying this fact to all the elements along the chain, we arrive at the desired conclusion, namely, $\gamma^\star \subseteq \alpha_i$. Let $\gamma^{(l)}$ be such that $\gamma^{(l)} \subseteq \alpha_i$ and $\gamma^{(l)} \subset \gamma^\star$. Let $\alpha_k \to \alpha_k^\star$ be the implication of B that this chain triggers in order to get the next element $\gamma^{(l+1)}$ of the chain, that is, $\gamma^{(l+1)} = \gamma^{(l)} \cup \alpha_k^\star$. The following inequalities hold: $\alpha_k \subseteq \gamma^{(l)}$ because $\gamma^{(l)} \not\models \alpha_k \to \alpha_k^\star$, $\gamma^{(l)} \subseteq \alpha_i$ by assumption; hence $\alpha_k \subseteq \alpha_i$. Using Lemma 5, and noticing the fact that, actually, $\alpha_k \subset \alpha_i$ since $\gamma^{(l)} \subset \alpha_i$ (otherwise we could not have $\gamma^\star \subset \alpha_i^\star$), we conclude that $\alpha_k^\star \subset \alpha_i$. We have that $\alpha_k^\star \subset \alpha_i$ and $\gamma^{(l)} \subset \alpha_i$ so that $\gamma^{(l+1)} = \gamma^{(l)} \cup \alpha_k^\star \subseteq \alpha_i$ as required. □

The next result characterizes our version of the canonical basis based on saturation.

Theorem 4 *Definite Horn functions have at most one saturated basis.*

Proof Let B_1 and B_2 be two equivalent and saturated bases. Let $\alpha \to \alpha^\star$ be an arbitrary implication in B_1. We show that $\alpha \to \alpha^\star \in B_2$ as well. By symmetry, this implies that $B_1 = B_2$.

By Lemma 3, we have that $[\alpha] \not\models B_1$ and thus $[\alpha]$ must violate some implication $\beta \to \beta^\star \in B_2$. Hence, it must hold that $\beta \subseteq \alpha$ but $\beta^\star \not\subseteq \alpha$. If $\beta = \alpha$, $\alpha \to \alpha^\star \in B_2$ as required. The rest of the proof shows that other case $\beta \subset \alpha$ is not possible.

Let us assume then that $\beta \subset \alpha$ so that, by monotonicity, $\beta^\star \subseteq \alpha^\star$. If $\beta^\star \subset \alpha^\star$, then we can use Lemma 6 with $\alpha_i = \alpha$ and $\gamma = \beta$ and conclude that $\beta^\star \subseteq \alpha$, contradicting the fact that $[\alpha] \not\models \beta \to \beta^\star$. Thus, it should be that $\beta^\star = \alpha^\star$. Now, consider $\beta \to \alpha^\star \in B_2$. Clearly β is negative (otherwise, $\beta = \beta^\star$, and then $\beta \to \beta^\star$ is redundant) and thus it must violate some implication $\gamma \to \gamma^\star \in B_1$, namely, $\gamma \subseteq \beta$ but $\gamma^\star \not\subseteq \beta$. If $\gamma = \beta$, then we have $\alpha \to \alpha^\star \in B_1$ and $\gamma \to \gamma^\star$ with $\gamma \subset \alpha$ and $\gamma^\star = \beta^\star = \alpha^\star$ contradicting the fact that B_1 is irredundant. Thus, $\gamma \subset \beta$ and so $\gamma^\star \subseteq \beta^\star$. If $\gamma^\star \subset \beta^\star$ then we use Lemma 6 as before but with $\alpha_i = \beta$ and we conclude that $\gamma^\star \subseteq \beta$. Again, this means that $\beta \models \gamma \to \gamma^\star$ contradicting the fact that β violates this implication. So the only remaining case is $\gamma^\star = \beta^\star$ but this means that we have the implications $\alpha \to \alpha^\star \in B_1$ and $\gamma \to \gamma^\star \in B_1$ with $\gamma \subset \alpha$ but $\alpha^\star = \gamma^\star$ which again makes B_1 redundant. □

In fact, as we shall argue shortly, definite Horn functions have exactly one saturated basis.

3.1 Constructing the GD basis

So far, our definition of saturation only tests whether a given basis is actually saturated; we study now a saturation process to obtain the GD basis. New definitions are needed. Let H be any definite Horn CNF, and α any variable subset. Let $H(\alpha)$ be those implications of H whose antecedents fall in the same equivalence class as α, namely, $H(\alpha) = \{\alpha_i \to \beta_i | \alpha_i \to \beta_i \in H \text{ and } \alpha^\star = \alpha_i^\star\}$.

Given a definite Horn CNF H and a variable subset α, we introduce a new operator $\bullet$ that we define as follows: $\alpha^\bullet$ is the closure of α *with respect to the subset of implications* $H \setminus H(\alpha)$. That is, in order to compute $\alpha^\bullet$ one does forward chaining starting with α but one is not allowed to use implications in $H(\alpha)$.

This operator has been used in the past in related contexts. As far as we know, the earliest description of this operator can be found in the work of Maier (1980), where he calls it *direct determination* and denotes it with $\overset{\bullet}{\longrightarrow}$. Maier's definition is, in fact, the closest definition to our own; he uses this operator as part of a procedure for minimizing a certain type of representation for functional dependencies in relational databases, which is closely related to our own representation for definite Horn functions. A few years later, and independently, Guigues and Duquenne (1986) uses this same operator in their work, which is used again in Wild (1994) also in the context of simplifying definite Horn representations.

Example 5 Let $H = \{a \to b, a \to c, c \to d\}$. Then, $(ac)^\star = abcd$ but $(ac)^\bullet = acd$ since $H(ac) = \{a \to b, a \to c\}$ and we are only allowed to use the implication $c \to d$ when computing $(ac)^\bullet$.

The following simple lemmas are going to be useful throughout the paper:

Lemma 7 *Let H be any definite Horn CNF, and let $H' \subseteq H$. Then, for any variable b and variable subset α with closure $\alpha^\star$ (w.r.t. H), it must hold that implications $\beta \to \beta'$ involved in derivations for $H' \models \alpha \to b$ are such that $\beta^\star \subseteq \alpha^\star$.*

Proof In order to apply $\beta \to \beta'$, we need to get from α to a term that includes β, so that $\beta \subseteq \alpha^\star$, and $\beta^\star \subseteq \alpha^\star$ as well. □

Lemma 8 *Let H be any definite Horn CNF. For any variable subsets α and β such that $\beta^\star \subset \alpha^\star$, it must be that $H \setminus H(\alpha) \models \beta \to \beta^\star$.*

Proof Clearly, $H \models \beta \to \beta^\star$, since the closure is taken with respect to H. If in a forward chain derivation of $\beta \to \beta^\star$ from H, an implication $\gamma \to \gamma'$ such that $\gamma^\star = \alpha^\star$ was used, we would have that $\gamma^\star = \alpha^\star \subseteq \beta^\star$, contradicting the assumption that $\beta^\star \subset \alpha^\star$. Therefore, all implications $\gamma \to \gamma'$ from H involved in any derivation of $\beta \to \beta^\star$ from H must be such that $\gamma^\star \subset \alpha^\star$ and, hence, $\beta \to \beta^\star$ can also be derived from $H \setminus H(\alpha)$ as required. □

Next, we prove that the bullet operator does not depend on the particular syntactic representation used to compute it, but it is an operator that only depends on the underlying Boolean function in a similar way that the star operator does not depend syntactically on the representation used to compute the closures by forward chaining, but on the function being represented.

Theorem 5 *Given a definite Horn function f represented by a definite Horn CNF H and a variable subset α, the operator $\alpha^\bullet$ does not depend on the representation H, but only on the underlying definite Horn function f represented by H.*

Proof Given two equivalent definite Horn representations for f, H and G, we need to show that the closure of α with respect to $H' = H \setminus H(\alpha)$ and with respect to $G' = G \setminus G(\alpha)$ is the same. Essentially, we need to show that, for all variables b, it holds that $H' \models \alpha \to b$ if and only if $G' \models \alpha \to b$. By symmetry, it is sufficient to show one of the directions. Assume, then, that $G' \models \alpha \to b$, and take any forward chain derivation of $\alpha \to b$ from G'. All implications involved in this derivation must be of the form $\beta \to \beta' \in G'$ such that $\beta^\star \subseteq \alpha^\star$ (see Lemma 7). The case $\beta^\star = \alpha^\star$ is not possible since it would imply $\beta \to \beta' \in G(\alpha)$ and thus $\beta \to \beta' \notin G'$. Therefore, all implications used in the derivation for $G' \models \alpha \to b$ must have antecedents β such that $\beta^\star \subset \alpha^\star$. Let G'' be the set of all the implications involved in this derivation. Now, for all implications $\beta \to \beta' \in G''$ we have that $H \models \beta \to \beta'$ since $G'' \subseteq G' \subseteq G \equiv H$. Moreover, Lemma 8 guarantees that $H' = H \setminus H(\alpha) \models \beta \to \beta'$, since $\beta^\star \subset \alpha^\star$. We conclude that $H' \models G'' \models \alpha \to b$ as required. □

The following presents several useful facts involving the star and bullet operators:

Lemma 9 *Let H be any definite Horn CNF, and let α and β be arbitrary variable subsets. The following facts hold*:

1. $\alpha \subseteq \alpha^\bullet \subseteq \alpha^\star$
2. $\alpha^{\bullet\star} = \alpha^\star$
3. $\beta^\bullet \subseteq \alpha^\bullet \Rightarrow \beta^\star \subseteq \alpha^\star$
4. $\beta^\bullet = \alpha^\bullet \Rightarrow \beta^\star = \alpha^\star$

Proof In the following, we use the fact that $\star$ is a closure operator.

1. Follows from the already stated fact that to get from α to $\alpha^\bullet$ we do forward chaining with the implications in $H \setminus H(\alpha)$, but to get to $\alpha^\star$ we are allowed to use all implications in H.
2. From Property 1 of this lemma, we know that $\alpha \subseteq \alpha^\bullet \subseteq \alpha^\star$. Thus, by monotonicity and idempotency of the star operator it holds that $\alpha^\star \subseteq \alpha^{\bullet\star} \subseteq \alpha^{\star\star} = \alpha^\star$ and the equality follows.
3. By monotonicity, we have that $\beta^\bullet \subseteq \alpha^\bullet$ implies $\beta^{\bullet\star} \subseteq \alpha^{\bullet\star}$. Using Property 2, we obtain the desired fact $\beta^\star \subseteq \alpha^\star$.
4. Just apply Property 3 twice. □

Computing the GD basis of a definite Horn H The algorithm for constructing the GD basis of an arbitrary definite Horn CNF H is presented in Fig. 1 and works as follows. First, saturate every implication $C = \alpha \to \beta$ in H by replacing it with the implication $\alpha^\bullet \to \alpha^\star$. Then, all that is left to do is to remove redundant implications. This can be done, for example, by doing the following syntactic check over the saturated version of H: (1) remove implications s.t. $\alpha^\bullet = \alpha^\star$, (2) remove duplicate implications, and (3) remove subsumed implications, i.e., implications $\alpha^\bullet \to \alpha^\star$ for which there is another implication $\beta^\bullet \to \beta^\star$ s.t. $\alpha^\star = \beta^\star$ but $\beta^\bullet \subset \alpha^\bullet$.

Let us denote with GD(H) the implicational definite Horn CNF obtained by applying this procedure to input H. Note that this algorithm is designed to work when given a definite Horn CNF both in implicational or standard form. That is, if the input is in standard form,

```
GD(H)
1 for every implication $C \in H$        ▷ /* $C = \alpha \to \beta$ */
2       do $\beta \leftarrow \alpha^\star$                  ▷ /* right-saturate C */
3          $\alpha \leftarrow \alpha^\bullet$                  ▷ /* left-saturate C */
4 remove from H redundant implications
5 return H
```

Fig. 1 Constructing the GD basis for definite Horn CNF

the step in which the original clauses are right-saturated will convert these clauses into implications. The redundancy removal step at the end of the process will make sure that of the clauses that could be grouped together because they share an antecedent, only one implication will survive.

Observe also that this algorithm never adds new implications, so that the size of $\mathrm{GD}(H)$ (in number of implications) is at most that of H; and that the changes all along will always preserve the semantics, so that, as functions, $H = \mathrm{GD}(H)$.

The procedure can be implemented to run in quadratic time, since finding the closures of antecedent and consequent of each implication can be done in linear time w.r.t. the size of the initial Horn CNF H, as well as detecting redundancy of a given implication.

Example 6 Following our previous example, let $H = \{a \to b, a \to c, c \to d\}$. Then, $a^\bullet = a$, $a^\star = abcd$, $c^\bullet = c$, and $c^\star = cd$. Thus our H after the for loop is $\{a \to abcd, a \to abcd, c \to cd\}$, and the final GD basis is $\mathrm{GD}(H) = \{a \to abcd, c \to cd\}$.

Example 7 Let $H = \{a \to b, a \to c, ad \to e, ab \to e\}$. First, we compute antecedents' closures to compute the right-hand-sides of the implications and to figure out how clauses partition into equivalence classes:

- $a^\star = abce$,
- $(ad)^\star = abcde$, and
- $(ab)^\star = abce$.

The resulting partition is illustrated in the following table. Horizontal lines correspond to class boundaries (there are only two classes: $abce$ and $abcde$); the first column is the implication index in H, the second column is the implication itself, and the third column contains the closed class representative (that is, the antecedent's closure w.r.t. H).

	Clause	*Right-sat.* ($\alpha^\star$)	*Class*
1	$a \to b$	$abce$	$abce$
2	$a \to c$	$abce$	$abce$
4	$ab \to e$	$abce$	$abce$
3	$ad \to e$	$abcde$	$abcde$

Next, we compute the left-saturations of the implications. In order to compute $H(a)$ we need to select out of all the implications of H those that belong to the same equivalence class of $a^\star$, namely $abce$. These implications are: $H(a) = \{a \to b, a \to c, ab \to e\}$. Similarly, to compute $H(ad)$, we select those implications that fall into the equivalence class of $(ad)^\star = abcde$, which only results in the single implication $H(ad) = \{ad \to e\}$. Finally, to compute

$H(ab)$, we observe that ab and a belong to the same class (their closures coincide: $a^\star = (ab)^\star = abce$), and therefore $H(ab) = H(a)$. Thus,

- To compute $a^\bullet$, compute the closure of a with respect to $H \setminus H(a) = \{ad \to e\}$; hence $a^\bullet = a$.
- To compute $(ad)^\bullet$, compute the closure of ad with respect to $H \setminus H(ad) = \{a \to b, a \to c, ab \to e\}$; hence $ad^\bullet = abcde$.
- To compute $(ab)^\bullet$, compute the closure of ab with respect to $H \setminus H(ab) = \{ad \to e\}$; hence $ab^\bullet = ab$.

The following table contains a summary of the saturation process done so far.

	Original clause	*Right-sat.* ($\alpha^\star$)	*Left-sat.* ($\alpha^\bullet$)	*Saturated implication* ($\alpha^\bullet \to \alpha^\star$)
1	$a \to b$	$abce$	a	$a \to abce$
2	$a \to c$	$abce$	a	$a \to abce$
4	$ab \to e$	$abce$	ab	$ab \to abce$
3	$ad \to e$	$abcde$	$abcde$	$abcde \to abcde$

After saturation of every implication in H, we obtain $H' = \{a \to abce, a \to abce, abcde \to abcde, ab \to abce\}$. It becomes clear that we should only keep the first implication; the rest are either identical, trivial, or subsumed by it. Hence, $\mathrm{GD}(H) = \{a \to abce\}$.

In the remainder of this section we show that the given algorithm computes the unique saturated representation of its input. First, we need a couple of simple lemmas:

Lemma 10 *Let H be any definite Horn CNF. For any variable subsets α and β, the following holds: if $\beta \subseteq \alpha^\bullet$ but $\beta^\star \subset \alpha^\star$, then $\beta^\star \subseteq \alpha^\bullet$.*

Proof By Lemma 8, we have that $H \setminus H(\alpha) \models \beta \to \beta^\star$, Moreover, $\beta \subseteq \alpha^\bullet$ implies $\beta \to \beta^\star \models \alpha^\bullet \to \beta^\star$ and thus $H \setminus H(\alpha) \models \alpha^\bullet \to \beta^\star$. By definition of the bullet operator we also have that $H \setminus H(\alpha) \models \alpha \to \alpha^\bullet$, and concatenating the two we get that $H \setminus H(\alpha) \models \alpha \to \beta^\star$. By construction, everything that can be derived from α using implications in $H \setminus H(\alpha)$ must be included in $\alpha^\bullet$ and therefore, $\beta^\star \subseteq \alpha^\bullet$ as required. □

Lemma 11 *The algorithm computing* GD(H) *outputs the GD basis of H for any definite Horn formula H.*

Proof Let H be the input to the algorithm, and let $H' = \mathrm{GD}(H)$ be its output. First we observe that H' must be irredundant, since all redundant clauses have been removed in step 4 of the GD(H) procedure (see Fig. 1).

Next, we show that H' must be saturated. Because of the initial saturation process of implications of H in steps 1–3 of the procedure GD(H), implications of H' have the form $\alpha^\bullet \to \alpha^\star$. From Lemma 9(2) we know that $\alpha^{\bullet\star} = \alpha^\star$, and thus H' must be right-saturated.

It remains to show that H' is left-saturated. From Lemma 9(1) we know that $\alpha^\bullet \subseteq \alpha^\star$, and the fact that H' contains no redundant implications guarantees that $\alpha^\bullet \subset \alpha^\star$. Thus, $[\alpha^\bullet] \not\models \alpha^\bullet \to \alpha^\star$ so that Condition 1 of left-saturation is satisfied. Now let $\gamma^\bullet \to \gamma^\star$ be any other implication in H'. We need to show that $[\alpha^\bullet] \models \gamma^\bullet \to \gamma^\star$. Assume by way of contradiction that this is not so, and $[\alpha^\bullet] \models \gamma^\bullet$ but $[\alpha^\bullet] \not\models \gamma^\star$. That is, $\gamma^\bullet \subseteq \alpha^\bullet$ but $\gamma^\star \not\subseteq \alpha^\bullet$.

By Lemma 9(4), $\gamma^\bullet = \alpha^\bullet$ implies $\gamma^\star = \alpha^\star$, contradicting the fact that H' is irredundant while containing both implications. Thus, $\gamma^\bullet \subset \alpha^\bullet$, and therefore using Lemma 9(3) $\gamma^\star \subseteq \alpha^\star$ must hold as well. If $\gamma^\star = \alpha^\star$ then $\alpha^\bullet \to \alpha^\star$ is redundant in H'. Thus, it can only be that $\gamma^\bullet \subset \alpha^\bullet$ and $\gamma^\star \subset \alpha^\star$. But in this case Lemma 10 (setting $\beta = \gamma^\bullet$) guarantees that $\gamma^\star = \gamma^{\bullet\star} \subseteq \alpha^\bullet$, which contradicts our assumption that $\gamma^\star \nsubseteq \alpha^\bullet$. □

It is clear that GD(H) has at most as many implications as H. Thus, if H is of minimum size, then so is GD(H). This, together with Theorem 4 stating that the GD basis is unique, implies:

Theorem 6 (Guigues and Duquenne 1986) *The GD basis of a definite Horn function is of minimum implicational size.*

4 The GD basis and query learning Horn CNF

As stated in the Introduction, we apply the results on the GD basis to better understand a learning algorithm for Horn CNFs. The learning protocol is as follows: a learning algorithm is allowed to interact with a Horn CNF target via membership and equivalence queries. In a membership query, the learner submits an assignment, and receives as answer whether the assignment satisfies the target; we refer to a positive or negative membership query according to the answer. In an equivalence query, the learner submits a Horn CNF, and receives as answer either an affirmative answer, meaning that the queried Horn CNF is equivalent to the target and the learning process has succeeded and must stop, or a counterexample, that is, an assignment that falsifies the target but satisfies the query (negative counterexample) or vice versa (positive counterexample).

The classic query learning algorithm by Angluin et al. (1992) is able to learn Horn CNF with membership and equivalence queries. It was proved in Angluin et al. (1992) that the outcome of the algorithm is always equivalent to the target concept. However, the following questions remain open: (1) which of the Horn CNF, among the many logically equivalent candidates, is output? And (2) does this output depend on the specific counterexamples given to the equivalence queries? Indeed, each query depends on the counterexamples received so far, and intuitively the final outcome should depend on that as well.

Our main result from this section is that, contrary to our first intuition, the output is always the same Horn CNF: namely, the GD basis of the target Horn function. This section assumes that the target is definite Horn, further sections in the paper lift the "definite" constraint.

4.1 The AFP algorithm for definite Horn CNF

We recall some aspects of the learning algorithm as described in Arias and Balcázar (2008), which bears only slight, inessential differences with the original in Angluin et al. (1992). The algorithm maintains a set P of all the positive examples seen so far. The fact that the target is definite Horn allows us to initialize P with the positive example 1^n. The algorithm maintains also a sequence $N = (x_1, \dots, x_t)$ of representative negative examples (these become the antecedents of the clauses in the hypotheses). The argument of an equivalence query is prepared from the list $N = (x_1, \dots, x_t)$ of negative examples combined with the set P of positive examples. The query corresponds to the following intuitive bias: everything is assumed positive unless some (negative) $x_i \in N$ suggests otherwise, and everything that

```
AFP()
1  $N \leftarrow ()$          ▷ /* empty list */
2  $P \leftarrow \{1^n\}$     ▷ /* top element */
3  $t \leftarrow 0$
4  while $EQ(H(N, P)) = (\text{"no"}, y)$      ▷ /* $y$ is the counterexample */
5      do if $y \not\models H(N, P)$
6          then add $y$ to $P$
7          else find the first $i$ such that     ▷ /* $N = (x_1, \ldots, x_t)$ */
8              $x_i \wedge y < x_i$, and          ▷ /* that is, $x_i \not\leq y$ */
9              $x_i \wedge y$ is negative         ▷ /* use membership query */
10           if found
11             then $x_i \leftarrow x_i \wedge y$     ▷ /* replace $x_i$ by $x_i \wedge y$ in $N$ */
12             else $t \leftarrow t + 1$; $x_t \leftarrow y$ ▷ /* append $y$ to end of $N$ */
13 return $H(N, P)$
```

Fig. 2 The AFP learning algorithm for definite Horn CNF

some x_i suggests negative is assumed negative unless some positive example $y \in P$ suggests otherwise. This is exactly the intuition in the hypothesis constructed by the AFP algorithm.

For the set of positive examples P, denote $P_x = \{y \in P \mid x \leq y\}$. The hypothesis to be queried, given the set P and the list $N = (x_1, \ldots, x_t)$, is denoted $H(N, P)$ and is defined as $H(N, P) = \{[x_i] \to [\bigwedge P_{x_i}] | x_i \in N\}$.

Intuitively, the query is constructed by considering $\bigwedge P_x$ as our current candidate to x^*. By Theorem 3(3), if P_x does contain all the positive examples above x, then P_x and x^* would coincide; however, this is not necessary in general, and P_x may become equal to x^* much before.

A positive counterexample is treated just by adding it to P. A negative counterexample y is used to either refine some x_i into a smaller negative example, or to add x_{t+1} to the list. Specifically, let

$$i := \min(\{j \mid MQ(x_j \wedge y) \text{ is negative, and } x_j \wedge y < x_j\} \cup \{t + 1\})$$

and then refine x_i into $x_i' = x_i \wedge y$, in case $i \leq t$, or else make $x_{t+1} = y$, subsequently increasing t. The value of i is found through membership queries on all the $x_j \wedge y$ for which $x_j \wedge y < x_j$ holds.

The AFP algorithm is described in Fig. 2. As is customary in the presence of equivalence queries, the partial correctness is guaranteed: if it stops, then the output is indeed equivalent to the target. We will prove termination of the algorithm by bounding the number t of negative examples x_i in use. We will infer also from the necessary lemmas our claim that its output is always the GD basis, namely, the unique saturated definite Horn formula that is equivalent to the target. First, we discuss separately the following easy properties:

Lemma 12 *Let P be a set of positive examples, and let x be a negative example for some Horn function f. Consider the implication $[x] \to [\bigwedge P_x]$, and assume that y satisfies it and that $x \leq y$. Then, $x^\star \leq \bigwedge P_x \leq y$, where the closure $x^\star$ is taken with respect to f.*

Proof We have stated in Theorem 3(3) that $x^* = \bigwedge\{y|h(y) = 1, x \leq y\}$. Clearly $P_x \subseteq \{y|h(y) = 1, x \leq y\}$ which proves the first inequality. For the second, by assumption $y \models [x] \to [\bigwedge P_x]$ whereas $x \leq y$ means $y \models [x]$, thus $y \models [\bigwedge P_x]$, that is, $\bigwedge P_x \leq y$. □

Iterating the same argument to the successive counterexamples received, we obtain:

Lemma 13 *Along any run of the AFP algorithm, at the point of issuing the equivalence query, for every x_i and x_j in N with $i < j$ and $x_i \le x_j$, it holds that $x_i^\star \le \bigwedge P_{x_i} \le x_j$.*

Proof Lemma 12 guarantees that $x_i^\star \le \bigwedge P_{x_i}$. To show $\bigwedge P_{x_i} \le x_j$, consider the evolution of x_i and x_j during an arbitrary execution of the algorithm. At the time x_j is created, we know it is a negative counterexample to the current hypothesis, for which it must be therefore positive. That hypothesis includes the implication $[x_i] \to [\bigwedge P_{x_i}]$, and x_j must satisfy it, and then $x_i \le x_j$ implies $\bigwedge P_{x_i} \le x_j$. From that point on, further positive examples may enlarge P_{x_i} and thus reduce $\bigwedge P_{x_i}$, keeping the inequality. Further negative examples y may reduce x_i, again possibly enlarging P_{x_i} and keeping the inequality; or may reduce x_j into $x_j \wedge y$. If $x_i \not\le x_j \wedge y$ anymore, then there is nothing left to prove. Finally, if $x_i \le x_j \wedge y$, then $x_i \le y$, and y is again a negative counterexample that must satisfy the implication $[x_i] \to [\bigwedge P_{x_i}]$ as before, so that $\bigwedge P_{x_i} \le x_j \wedge y$ also for the new value of x_j. □

The essential intuition of our contributions in this section is captured in the following lemma:

Lemma 14 *Along the running of the AFP algorithm, at the point of issuing the equivalence query, for every x_i and x_j in N with $i < j$ there exists a positive example z such that $x_i \wedge x_j \le z \le x_j$.*

Proof We argue inductively along the successive queries and refinements. We need to establish the fact at the time of creating a new element of N, that is, for each $i \le t$ with respect to x_{t+1}, and we need to argue that the refinements that take place at line 11 of the algorithm maintain the fact stated. Both are argued similarly. If a positive counterexample is received, no element of N changes; thus we only need to consider a negative counterexample y.

First note the easiest case whereby x_i gets refined into $x_i' = x_i \wedge y$. This leaves x_j untouched, and brings down $x_i \wedge x_j$ into $x_i' \wedge x_j$; the same value of z will do: $x_i' \wedge x_j \le x_i \wedge x_j \le z \le x_j$.

Now consider the case in which x_j is refined into $x_j' = x_j \wedge y$. We assume as inductive hypothesis that a corresponding z exists before the refinement: $x_i \wedge x_j \le z \le x_j$.

We establish first the following auxiliary claim: there is a positive example z' for which $x_i \wedge y \le z' \le y$. To find such z', observe that x_i came before x_j but was not chosen for refinement; either $x_i \wedge y$ is itself positive, and we can simply choose $z' = x_i \wedge y$, or $x_i \le y$. In this case we use Lemma 12 as follows. Since y was a negative counterexample, it must satisfy the query, which includes the clause $[x_i] \to [\bigwedge P_{x_i}]$; x_i is a negative example, and $x_i \le y$, hence Lemma 12 applies: $x_i^\star \le y$, whence $x_i \wedge y \le x_i \le x_i^\star \le y$. We pick $z' = x_i^\star$, which is of course positive.

At this point, we have the already existing z fulfilling $x_i \wedge x_j \le z \le x_j$, and the z' just explained for which $x_i \wedge y \le z' \le y$. Observe the following: $x_i \wedge x_j' = x_i \wedge x_j \wedge y = x_i \wedge x_i \wedge x_j \wedge y = (x_i \wedge x_j) \wedge (x_i \wedge y)$. The first half of this last expression is bounded above by z, and the second half is bounded above by z', therefore $x_i \wedge x_j' \le z \wedge z' \le x_j \wedge y = x_j'$. Moreover, both z and z' being positive, and the target being closed under intersection (Theorem 1), ensures that $z \wedge z'$ is positive.

The induction basis case of creation of $x_{t+1} = y$ is handled in the same way: the positive z' obtained in the same manner fulfills directly the condition $x_i \wedge y \le z' \le y$, which is what we need. □

Our key lemma for our next main result is:

Lemma 15 *All hypotheses $H(N,P)$ output by the AFP learning algorithm in equivalence queries are saturated.*

Proof Recall that $H(N,P) = \{[x_i] \to [\bigwedge P_{x_i}] \mid x_i \in N\}$, where $P_{x_i} = \{y \in P \mid x_i \leq y\}$. Let $\alpha_i = [x_i]$ and $\beta_i = [\bigwedge P_{x_i}]$ for all i so that $H(N,P) = \{\alpha_i \to \beta_i \mid 1 \leq i \leq t\}$.

First we show that $H(N,P)$ is left-saturated. To see that $x_i \not\models \alpha_i \to \beta_i$ it suffices to note that $x_i < \bigwedge P_{x_i}$ since x_i is negative but $\bigwedge P_{x_i}$ is positive by Theorem 1, being an intersection of positive examples; thus, these two assignments must be different.

Now we show that $x_i \models \alpha_j \to \beta_j$, for all $i \neq j$. If $x_i \not\models \alpha_j$, then clearly $x_i \models \alpha_j \to \beta_j$. Otherwise, $x_i \models \alpha_j$ and therefore $x_j \leq x_i$. If $i < j$, then by Lemma 14 we have that $x_i \wedge x_j \leq z \leq x_j$ for some positive z. Then, $x_i \wedge x_j = x_j \leq z \leq x_j$, so that $x_j = z$, contradicting the fact that x_j is negative whereas z is positive. Otherwise, $j < i$. We apply Lemma 13: it must hold that $\bigwedge P_{x_j} \leq x_i$. Thus, in this case, $x_i \models \alpha_j \to \beta_j$ as well because $x_i \models \beta_j = [\bigwedge P_{x_j}]$.

It is only left to show that $H(N,P)$ is right-saturated. Clearly, $H(N,P)$ is consistent with N and P, that is, $x \not\models H(N,P)$ for all $x \in N$ and $y \models H(N,P)$ for all $y \in P$. Take any $x \in N$ contributing the implication $[x] \to [\bigwedge P_x]$ to $H(N,P)$. We show that it is right-saturated, i.e., $\bigwedge P_x = x^\star$, where the closure is taken with respect to $H(N,P)$. We note first that $H(N,P) \models [x] \to ([x])^\star$ since the closure is taken w.r.t. implications in $H(N,P)$. By the construction of $H(N,P)$, all examples $y \in P_x$ must satisfy it, hence they must satisfy the implication $[x] \to ([x])^\star$ as well. Therefore, since $y \models [x]$ we must have that $y \models ([x])^\star$, or equivalently, that $x^\star \leq y$. This is true for every such y in P_x and thus $x^\star \leq \bigwedge P_x$. On the other hand, it is obvious that $\bigwedge P_x \leq x^\star$ since the implication $[x] \to [\bigwedge P_x]$ of $H(N,P)$ guarantees that all the variables in $\bigwedge P_x$ are included in the forward chaining process in the final $x^\star$. So we have $x^\star \leq \bigwedge P_x \leq x^\star$ as required. □

Putting Theorem 4 and Lemma 15 together, we obtain that AFP, run on a definite Horn target, if it stops, always outputs the GD basis of the target concept. It remains to show that it does stop.

4.2 Termination

We explain now how the lemmas just proved guarantee that t, the cardinality of N, is bounded by a parameter of the target. Essentially, the proof says that the elements of N, together with their corresponding values of z considered pairwise as per Lemma 14, establish a lower bound on the number of implications of the target. The argument is as in Angluin et al. (1992), somewhat adapted to our previous results.

Lemma 16 *Let H be any definite Horn formula equivalent to the target, cast as a conjunction of implications. Consider two different negative examples x_i and x_j in N. Each of them falsifies some implication in H, but it cannot be the same implication.*

Proof Consider an implication $\alpha \to \beta \in H$ such that $x_i \not\models \alpha \to \beta$ and, likewise, $x_j \not\models \alpha \to \beta$, for $i < j$. Consider also the positive example z s.t. $x_i \wedge x_j \leq z \leq x_j$ whose existence is guaranteed by Lemma 14. That is, $[\alpha] \leq x_i$, and $[\alpha] \leq x_j$, which implies that $[\alpha] \leq x_i \wedge x_j \leq z$; however, z is a positive example, and must satisfy $z \models \alpha \to \beta$. Then, from $[\alpha] \leq z$ immediately follows $[\beta] \leq z \leq x_j$, and therefore $x_j \models \alpha \to \beta$ actually. □

It is straightforward to implement in polynomial time the algorithm. Let the target be a Boolean function on n variables, and let the GD basis of the target have m implications; by Theorem 6, all definite Horn formulas equivalent to the target have at least m implications. Each negative counterexample either increases the cardinality of N, which can happen at most m times, or refines some x_i, which can happen at most n times each, for a total of at most $n \times m$ negative counterexamples. Likewise, each positive counterexample added to P must change at least one of the implications in the query, by reducing $\bigwedge P_x$, which again can happen at most n times for each $x \in N$: this accounts for $n \times m$ equivalence queries leading to positive counterexamples. Finally, between counterexamples, at most m membership queries are asked, and termination is guaranteed. Therefore, we can state:

Theorem 7 (Angluin et al. 1992) *AFP, run on a definite Horn target on n variables, and whose GD basis has m implications, always outputs the GD basis of the target concept in polynomial time, with at most* $2nm + 1$ *equivalence and* nm^2 *membership queries, accounting for a total of* $O(nm^2)$ *queries. All queries consist of at most m implications.*

That is, $O(nm^2)$ queries are sufficient, also for the case of General Horn discussed below. The combinatorial notion of certificate size has been used to prove lower bounds on query complexity for many algorithms, including nonlearnability proofs (Hellerstein et al. 1996). Several analyses along this line are provided in Arias et al. (2006); for the particular case of algorithms that do not query hypotheses essentially larger than the target, a refined certificate size bound appears in Arias and Balcázar (2008), which is related to the number of implications as we employ here.

As a consequence, the following is known: in the case $m \leq n$, $\Omega(m^2)$ queries are necessary. For $m \geq n$, only a weaker lower bound of $\Omega(mn)$ exists. There is room for potential improvements of either the query complexity of the algorithm or the certificate size lower bound.

5 A canonical basis for general Horn

Naturally, we wish to extend the notion of saturation and GD basis to general Horn functions. We do this via a a prediction-with-membership reduction (Angluin and Kharitonov 1995) from general Horn to definite Horn, and use the corresponding intuitions to define a GD basis for general Horn. We use this reduction to generalize our AFP algorithm to general Horn CNF, and as a consequence one obtains that the generalized AFP always outputs a saturated version of the target function. Indeed, for the generalized AFP it is also the case that the output is only dependent on the target, and not on the counterexamples received along the run.

5.1 Reducing general Horn CNF to definite Horn CNF

In this section we describe the intuition of the representation mapping, which we use in the next section to obtain a canonical basis for general Horn functions.

For any general Horn CNF H over n propositional variables, e.g. $X = \{x_i | 1 \leq i \leq n\}$, we construct a definite Horn H' over the set of $n+1$ propositional variables $X' = X \cup \{\mathbf{f}\}$, where $\mathbf{f}$ is a new "dummy" variable; in essence $\mathbf{f}$ represents the *false* (that is, empty) consequent of the negative clauses in H. The relationship between the assignments for H and H' are as follows: for assignments of $n + 1$ variables xb where x assigns to the variables in X and b

is the truth value assigned to **f**, $x0 \models H'$ if and only if $x \models H$, whereas $x1 \models H'$ if and only if $x = 1^n$. Define the implication $C_{\mathbf{f}}$ as $\mathbf{f} \rightarrow X'$.

Lemma 17 *For any general Horn CNF H over variables X, let $X' = X \cup \{\mathbf{f}\}$, where $\mathbf{f} \notin X$; then the implication $\mathbf{f} \rightarrow X'$ is saturated.*

Proof Right-saturation is because there is absolutely no variable left out of the right-hand side. Left-saturation is due to the fact that no other implication from H can be applied to the left-hand side. □

Let H_d be the set of definite Horn clauses in H, and $H_n = H \setminus H_d$ the negative ones. Define the mapping g as

$$g(H) = H_d \cup \{\neg C \rightarrow X' | C \in H_n\} \cup \{C_{\mathbf{f}}\}.$$

That is, $g(H)$ includes the definite clauses of H, the special implication $C_{\mathbf{f}}$, and the clauses C that are negative are made definite by forcing all the positive literals, including **f**, into them. Clearly, the resulting $g(H)$ is definite Horn. Observe that the new implication $C_{\mathbf{f}}$ is saturated (see Lemma 17) and the ones coming from H_n are right-saturated. Observe also that g is injective: given $g(H)$, we recover H by removing the implication $C_{\mathbf{f}}$, and replacing by $\square$ all the right-hand sides containing **f**. Clearly, $g^{-1}(g(H)) = H$, since g^{-1} is removing all that g adds.

5.2 Constructing a GD-like basis for general Horn CNF

The notion of left-saturation translates directly into general Horn CNF:

Definition 5 Let $B = \{\alpha_i \rightarrow \beta_i\}_i$ be a basis for some general Horn function. Notice that now β_i can possibly be empty (it is empty for the negative clauses). Then, B is *left-saturated* if the following two conditions hold:

1. $[\alpha_i] \not\models \alpha_i \rightarrow \beta_i$, for all i;
2. $[\alpha_i] \models \alpha_j \rightarrow \beta_j$, for all $i \neq j$.

Notice that now in the second condition, if $\beta_j = \square$, i.e., we are dealing with a negative clause, then $[\alpha_i] \models \alpha_j \rightarrow \beta_j$ translates directly into $[\alpha_i] \not\models \alpha_j$ (equivalently, $\alpha_i \not\supseteq \alpha_j$) since it could never happen that $[\alpha_i] \models \alpha_j$ but $[\alpha_i] \models \square$, where $\square$ is the (unsatisfiable) empty clause. The conditions can be more explicitly stated as follows:

1. $[\alpha_i] \not\models \alpha_i \rightarrow \beta_i$, for all i;
2. $[\alpha_i] \models \alpha_j \rightarrow \beta_j$, for all $i \neq j$ such that $\beta_j \neq \square$;
3. $\alpha_j \not\subseteq \alpha_i$, for all $i \neq j$ such that $\beta_j = \square$.

The third condition guarantees that no negative clause is a subset of any other clause. If this were not so, we clearly would have a redundant clause.

For a definite Horn CNF H, right-saturating a clause $\alpha \rightarrow \beta$ essentially means that we add to its consequent everything that is implied by its antecedent, namely $\alpha^\star$. This can no longer be done in the case of general Horn CNF, since we need to take special care of the negative clauses. If $\beta = \square$, we cannot set β to $\alpha^\star$ without changing the underlying Boolean function being represented. The closure $x^\star$ of an assignment x is defined as the closure with respect to all definite clauses in the general Horn CNF. It is useful to continue to partition assignments x in the Boolean hypercube according to their closures $x^\star$; however, in the general Horn case, we distinguish a new class (the negative class) of closed assignments that are actually negative, that is, it is possible now that $x^\star \not\models H$. These assignments are exactly those that satisfy all definite clauses of H but violate negative ones. Based on this, the negative clauses (those with antecedent α such that $[\alpha^\star] \not\models B$) should be left unmodified, and the definite clauses (those whose antecedents α are such that $[\alpha^\star] \models B$) should be right-saturated. Thus, the definition is:

Definition 6 Let $B = \{\alpha_i \to \beta_i\}_i$ be a basis for some general Horn function. Then, B is *right-saturated* if, for all i, $\beta_i = \square$ or $\beta_i = \alpha_i^\star$.

Observe that, clearly, $\beta_i = \square$ corresponds exactly to those implications where $\alpha_i^\star \not\models B$. As for the definite case, "saturated" means that the general Horn CNF in question is both left- and right-saturated. We must see that this is the "correct" definition in some sense:

Lemma 18 *A basis H is saturated iff $H = g^{-1}(\mathrm{GD}(g(H)))$.*

Proof First let us note that the expression $g^{-1}(\mathrm{GD}(g(H)))$ is well-defined. We can always invert g on $\mathrm{GD}(g(H))$, since saturating $g(H)$ does not modify $C_{\mathbf{f}}$ (already saturated) and it does not touch the positive literals of implications containing $\mathbf{f}$ since these are right-saturated. Therefore, we can invert it since the parts added by g are left untouched by the construction of $\mathrm{GD}(g(H))$.

We prove first that if H is saturated then $H = g^{-1}(\mathrm{GD}(g(H)))$. Assume, then, that H is saturated but $H \neq g^{-1}(\mathrm{GD}(g(H)))$. Applying g, which is injective, this can only happen if $\mathrm{GD}(g(H)) \neq g(H)$, namely, $g(H)$, as a definite Horn CNF, differs from its own GD basis and, hence, it is not saturated: it must be because some implication other than $C_{\mathbf{f}}$ is not saturated, since this last one is saturated by construction. Also the ones containing $\mathbf{f}$ in their consequents are right-saturated, so no change happens in the right-hand-sides of these implications when saturating $g(H)$. This means that when saturating we must add a literal different from $\mathbf{f}$ to the right-hand-side of an implication not containing $\mathbf{f}$ or to the left-hand-side of an implication. In both cases, this means that the original H could not be saturated either, contradicting our assumption.

It is only left to show that an H such that $H = g^{-1}(\mathrm{GD}(g(H)))$ is indeed saturated. By way of contradiction, assume that H is not saturated but $H = g^{-1}(\mathrm{GD}(g(H)))$. Applying g to both sides, we must have that $g(H) = \mathrm{GD}(g(H))$ so that $g(H)$ is actually saturated. Notice that the only difference between H and $g(H)$ is in the implication $C_{\mathbf{f}}$ and the right-hand-sides of negative clauses in H; $g(H)$ being left-saturated means that so must be H since the left-hand-sides of H and $g(H)$ coincide exactly (ignoring $C_{\mathbf{f}}$ naturally). Therefore, H is left-saturated as well. It must be that H is not right-saturated, that is, it is either missing some variable in some non-empty consequent, or some clause that should be negative is not. In the first case, then $g(H)$ is missing it, too, and it cannot be saturated. In the second case, then there is a redundant clause in H contradicting the fact that H is left-saturated (see Lemma 2). In both cases we arrive at a contradiction, thus the lemma follows. □

Notice that this last lemma also gives us a way to compute the saturation (that is, the GD basis) of a given general Horn CNF:

Theorem 8 *General Horn functions have a unique saturated basis. This basis, which we denote $\mathrm{GD}(H)$, can be computed by $\mathrm{GD}(H) = g^{-1}(\mathrm{GD}(g(H)))$.*

Proof If H is saturated then $H = g^{-1}(\mathrm{GD}(g(H)))$. The uniqueness of such an H follows from the following facts: first, $g(H)$ and $g(H')$ are equivalent whenever H and H' are equivalent; second, $\mathrm{GD}(g(H))$ is unique for the function represented by H (Theorem 4) and third, g^{-1} is uniquely defined since g is injective. □

Example 8 Let H be the general Horn CNF $\{a \to b, a \to c, abc \to \square\}$. Then,

- $g(H) = \{a \to b, a \to c, abc \to abc\mathbf{f}, \mathbf{f} \to abc\mathbf{f}\}$;
- $\mathrm{GD}(g(H)) = \{a \to abc\mathbf{f}, \mathbf{f} \to abc\mathbf{f}\}$;
- $\mathrm{GD}(H) = g^{-1}(\mathrm{GD}(g(H))) = \{a \to \square\}$.

Similarly to the case of definite Horn functions, $\mathrm{GD}(H)$ does not increase the number of new implications, and therefore if H is of minimum size, $\mathrm{GD}(H)$ must be of minimum size as well. This, together with the uniqueness of saturated representation implies that:

Theorem 9 *The GD basis of a general Horn function is of minimum implicational size.*

5.3 The AFP algorithm for general Horn CNF

We study now the AFP algorithm operating on general Horn CNF, by following a detour: we obtain it via reduction to the definite case.

We consider, therefore, an algorithm that, for target a general Horn function H, simulates the version of AFP algorithm from Fig. 2 on its definite transformation $g(H)$, where g is the representation transformation from Sect. 5.1. It has to simulate the membership and equivalence oracles for definite Horn CNF that the underlying algorithm expects, by using the oracles that it has for general Horn.

Initially, we set $P = \{1^{n+1}\}$, and $N = (0^n 1)$ since we know that $g(H)$ is definite and must contain the implication $\mathbf{f} \to X \cup \{\mathbf{f}\}$ by construction. In essence, the positive assignment $1^{n+1} = \mathbf{f}^\star$ and the negative assignment $0^n 1 = \mathbf{f}^\bullet$ guarantee that the implication $C_\mathbf{f}$ is included in every hypothesis $H(N, P)$ that the simulation outputs as an equivalence query. The resulting algorithm for general Horn CNF is described in Fig. 4.

In order to deal with the queries, we use two transformations: we must map examples over the $n + 1$ variables, asked as membership queries, into examples over the original example space over n variables, although in some cases we are able to answer the query directly as we shall see. Upon asking $x0$ as membership query for $g(H)$, we pass on to H the membership query about x. Membership queries of the form $x1$ are answered always negatively, except for 1^{n+1} which is answered positively (in fact query 1^{n+1} never arises anyway, because that example is in P from the beginning). Conversely, n-bit counterexamples x from the equivalence query with H are transformed into $x0$. The equivalence queries themselves are transformed according to g^{-1}. It is readily checked that all equivalence queries belong indeed to the image set of g since $C_\mathbf{f} \in H(N, P)$.

All together, these functions constitute a prediction-with-membership (pwm) reduction from general Horn to definite Horn, in the sense of Angluin and Kharitonov (1995).

It is interesting to note that if we unfold the simulation, we end up with the original algorithm by Angluin et al. (1992) (obviously, with no explicit reference to our "dummy" **f**). We include the algorithm from Angluin et al. (1992) in Fig. 3 so that readers can verify the equivalence of our reduction and the original algorithm. In fact, the only difference between our algorithm in Fig. 4 is in the construction of hypotheses: ours uses the reduction described in Sect. 5.1 and the construct $H(N, P)$, whereas the algorithm AFP of Angluin et al. (1992)

```
ORIGINALAFP()
1   $N \leftarrow ()$            ▷ /* empty list */
2   $P \leftarrow \{\}$          ▷ /* set of positive counterexamples */
3   $H \leftarrow \{\}$          ▷ /* hypothesis */
4   while EQ($H$) = (“no”, $y$)                 ▷ /* $y$ is the counterexample */
5       do if $y \not\models H$
6             then $P \leftarrow P \cup \{y\}$
7             else find the first $i$ such that   ▷ /* $N = (x_1, \dots, x_t)$ */
8                     $x_i \wedge y < x_i$, and          ▷ /* that is, $x_i \not\leq y$ */
9                     $x_i \wedge y$ is negative        ▷ /* use membership query */
10                 if found
11                    then $x_i \leftarrow x_i \wedge y$          ▷ /* replace $x_i$ by $x_i \wedge y$ in $N$ */
12                         $H \leftarrow H \setminus clauses(x_i) \cup clauses(x_i \wedge y)$
13                    else $t \leftarrow t+1$; $x_t \leftarrow y$ ▷ /* append $y$ to end of $N$ */
14                         $H \leftarrow H \cup clauses(y)$
15          remove from $H$ clauses violated by positive examples in $P$
16  return $H$
```

Fig. 3 The original AFP learning algorithm from Angluin et al. (1992) for general Horn CNF

```
GENAFP()
1   $N \leftarrow (0^n 1)$
2   $P \leftarrow \{1^{n+1}\}$
3   ▷ /* simulate AFP */
4   if AFP asks an equivalence query with $H(N, P)$
5      then pass $g^{-1}(H(N, P))$ to our equivalence oracle
6           if answer is “Yes”
7              then return $g^{-1}(H(N, P))$ and stop
8           elseif answer is “No” and $x$ is counterexample received
9              then pass $f(x) = x0$ as a counterexample to AFP
10  elseif AFP asks a membership query $MQ(x')$
11     then if $h(x') = 0$              ▷ /* membership query transformation */
12             then answer the query with a “No”
13          elseif $h(x') = 1$
14             then answer the query with a “Yes”
15          elseif $h(x') = x \notin \{0, 1\}$
16             then answer query using our own membership oracle on $x$
```

Fig. 4 The AFP learning algorithm for general Horn CNF

directly builds their hypotheses from the negative x_i and positive examples received. These negative examples, coincide exactly for both algorithms, as the mechanism that maintains N is the same. Next, we show that, in fact, the hypotheses constructed by our reduction and the original AFP (Angluin et al. 1992) are equivalent and almost syntactically identical (up to very minor, irrelevant differences as we explain below).

AFP (Angluin et al. 1992) builds the hypotheses in the following way. Let X be the set of propositional variables. Each $x_i \in N$ generates a set of possible clause candidates

$clauses(x_i)$ that is included in the hypothesis, where[2]

$$clauses(x_i) = \{[x_i] \rightarrow b | b \in X \setminus [x_i] \cup \{\square\}\}.$$

Positive examples are used to remove from this set those clauses that are falsified by the positive examples. If no positive counterexample y received is such that $y \geq x_i$, then the whole set $clauses(x_i)$ is included in the hypothesis, including the clause $[x_i] \rightarrow \square$ that is, in fact, subsuming the rest of $clauses(x_i)$.

If a positive counterexample y is seen s.t. $y \geq x_i$, then all clauses in the candidate set $clauses(x_i)$ such that $y \not\models [x_i] \rightarrow b$ are removed. This certainly removes $[x_i] \rightarrow \square$. In essence, the ones surviving are those $[x_i] \rightarrow b$, where b is set to 0 in x_i but set to 1 in *all* positive counterexamples y such that $y \geq x_i$.

To see that $H(N, P)$ contains the same implications, notice that if there is no $y \geq x_i$, then the implication included in $H(N, P)$ in our version is $[x_i] \rightarrow X \cup \{\mathbf{f}\}$ (because 1^{n+1} is used to construct the right-hand-side). When constructing the inverse image g^{-1} all the literals in the consequent are removed and thus the corresponding clause is $[x_i] \rightarrow \square$, which is logically equivalent to the set $clauses(x_i)$ generated by Angluin et al. (1992), since $[x_i] \rightarrow \square$ subsumes all the others. On the other hand, if there are positive y s.t. $y \geq x_i$, then our algorithm constructs the implication $[x_i] \rightarrow [\bigwedge P_{x_i}]$. Notice that since $P_{x_i} = \{y | y \geq x_i, y \in P\}$, variables in $[\bigwedge P_{x_i}]$ are precisely those that are set to 1 in all $y \in P_{x_i}$. Thus, the only difference between our clauses and the ones in Angluin et al. (1992) is the fact that ours are right-saturated (the ones in Angluin et al. 1992 are only missing the variables in $[x_i]$ from their right-hand-sides).

Therefore, the outcome of AFP on a general Horn target H comes univocally determined by the outcome of AFP on the corresponding definite Horn function $g(H)$; combining this fact with Theorems 7 and 8 leads to:

Theorem 10 *The AFP algorithm always outputs the GD basis of the target concept.*

6 Conclusions and future work

This paper makes several contributions towards a better understanding of learning Horn formulas. On the canonical representation side, we were able to understand, interpret, and adapt a result from the field of Formal Concept Analysis, namely the existence of the Guigues-Duquenne basis for definite Horn theories (Guigues and Duquenne 1986; Wild 1994). We were able to generalize this result to Horn theories in general, thus lifting the restriction of definiteness. To the best of our knowledge, the existence of a canonical, minimum representation for general Horn theories is not well known at all in the field of learning Boolean functions, and, in our opinion, it is of intrinsic interest. This paper presents this canonical representation in such a form that, hopefully, researchers in our field can understand and benefit from.

It is not unusual that learning algorithms exploit the fact that unique, minimum representations exist. For instance, this is the case for monotone DNF formulas, where the learning algorithm directly exploits the fact that any monotone DNF has a unique, minimum representation as the disjunction of its minterms, and the algorithm's task is to discover these minterms one at a time.

[2]The original paper (Angluin et al. 1992) uses **F** instead of $\square$; here we prefer to stick to our notation for consistency.

The second contribution of this paper is to provide a new analysis of the classic algorithm that learns conjunctions of Horn clauses (Angluin et al. 1992). Our proof is quite different from the one provided in the original paper, and brings to the fore the tight relation of the AFP algorithm and the canonical GD representation. In fact, we were able to prove that AFP always outputs the GD representation of the target concept, regardless of which counterexamples are received. We feel that we have greatly learned from doing this work, and that this new knowledge will prove crucial to continue to progress in the understanding of learning Horn functions; we still hope to obtain a new algorithm that improves on the one by Angluin et al. (1992).

Acknowledgements We thank the reviewers for their thorough comments that greatly helped improve this paper.

References

Angluin, D. (1987). Learning regular sets from queries and counterexamples. *Information and Computation*, *75*(2), 87–106.

Angluin, D., & Kharitonov, M. (1995). When won't membership queries help? *Journal of Computer and System Sciences*, *50*(2), 336–355.

Angluin, D., Frazier, M., & Pitt, L. (1992). Learning conjunctions of Horn clauses. *Machine Learning*, *9*, 147–164.

Arias, M., & Balcázar, J. L. (2008). Query learning and certificates in lattices. In *LNAI: Vol. 5254*. *ALT 2008* (pp. 303–315).

Arias, M., & Balcázar, J. L. (2009). Canonical Horn representations and query learning. In *LNAI: Vol. 5809*. *ALT 2009* (pp. 156–170).

Arias, M., & Khardon, R. (2002). Learning closed Horn expressions. *Information and Computation*, *178*(1), 214–240.

Arias, M., Feigelson, A., Khardon, R., & Servedio, R. A. (2006). Polynomial certificates for propositional classes. *Information and Computation*, *204*(5), 816–834.

Arias, M., Khardon, R., & Maloberti, J. (2007). Learning Horn expressions with LogAn-H. *Journal of Machine Learning Research*, *8*, 587.

Balcázar, J. L. (2005). Query learning of Horn formulas revisited. In *Computability in Europe conference*.

Bertet, K., & Monjardet, B. (2010). The multiple facets of the canonical direct unit implicational basis. *Theoretical Computer Science*, *411*(22–24), 2155–2166.

Chang, C. L., & Lee, R. (1973). *Symbolic logic and mechanical theorem proving*. Orlando: Academic Press.

Frazier, M., & Pitt, L. (1993). Learning from entailment: an application to propositional Horn sentences. In *Proceedings of the international conference on machine learning* (pp. 120–127), Amherst, MA. San Mateo: Morgan Kaufmann.

Frazier, M., & Pitt, L. (1996). CLASSIC learning. *Machine Learning*, *25*, 151–193.

Gaintzarain, J., Hermo, M., & Navarro, M. (2005). On learning conjunctions of Horn$^{\supset}$ clauses. In *Computability in Europe conference*.

Guigues, J. L., & Duquenne, V. (1986). Familles minimales d'implications informatives resultants d'un tableau de données binaires. *Mathématiques Et Sciences Humaines*, *95*, 5–18.

Hellerstein, L., Pillaipakkamnatt, K., Raghavan, V., & Wilkins, D. (1996). How many queries are needed to learn? *Journal of the ACM*, *43*(5), 840–862.

Hermo, M., & Lavín, V. (2002). Negative results on learning dependencies with queries. In *International symposium on artificial intelligence and mathematics*.

Horn, A. (1956). On sentences which are true of direct unions of algebras. *The Journal of Symbolic Logic*, *16*, 14–21.

Khardon, R., & Roth, D. (1996). Reasoning with models. *Artificial Intelligence*, *87*(1–2), 187–213.

Kivinen, J., & Mannila, H. (1995). Approximate inference of functional dependencies from relations. *Theoretical Computer Science*, *149*(1), 129–149.

Kleine Büning, H., & Lettmann, T. (1999). *Propositional logic: deduction and algorithms*. Cambridge: Cambridge University Press.

Maier, D. (1980). Minimum covers in relational database model. *Journal of the ACM*, *27*, 664–674.

McKinsey, J. C. C. (1943). The decision problem for some classes of sentences without quantifiers. *The Journal of Symbolic Logic*, *8*, 61–76.

Rouveirol, C. (1994). Flattening and saturation: two representation changes for generalization. *Machine Learning*, *14*(1), 219–232.

Selman, B., & Kautz, H. (1996). Knowledge compilation and theory approximation. *Journal of the ACM*, *43*(2), 193–224.

Valiant, L. G. (1984). A theory of the learnable. *Communications of the ACM*, *27*(11), 1134–1142.

Wang, H. (1960). Toward mechanical mathematics. *IBM Journal for Research and Development*, *4*, 2–22.

Wild, M. (1994). A theory of finite closure spaces based on implications. *Advances in Mathematics*, *108*, 118–139.