An efficient algorithm for Horn description

Jean-Jacques Hébrard, Bruno Zanuttini

An efficient algorithm for Horn description

Jean-Jacques Hébrard Bruno Zanuttini*†

June 20, 2003

Abstract

We give a new algorithm for computing a propositional Horn CNF formula given the set of its models. Its running time is $O(|R|n(|R| + n))$, where $|R|$ is the number of models and n that of variables, and the computed CNF contains at most $|R|n$ clauses. This algorithm also uses the well-known closure property of Horn relations in a new manner.

Keywords: Combinatorial problems; Algorithms; Description; Horn CNF

1 Introduction

This note addresses the problem of describing a Boolean relation by a propositional Horn CNF formula, in other words the problem of computing a Horn CNF given the set of its models. This process is part of the process of *structure identification*, which has been formalized by Dechter and Pearl and mainly studied in [3, 6, 9].

Converting a set of tuples into a Horn CNF can be thought of as a *knowledge compilation task* [3], where the input tuples may encode for instance the examples of a concept expressed over some attributes, and where we compute a formula-based definition ϕ of the concept. Such a task is not easy in general (see [6] for hardness results).

If the definition of the concept is Horn, then it allows for efficient reasoning, contrastingly with the general CNF case (remind that not every concept can be represented by a Horn CNF). Consequently, the process of Horn description can also be seen as the translation of an extensional constraint into a tractable set of clauses (see [8] for instance). Hopefully also, this process serves not only algorithmic purposes, but also allows to convert a set of tuples into a logically equivalent but smaller form; it is indeed known (see [9] for instance) that the CNF representation of a Boolean function is more compact (up to a polynomial) than its representation by a relation. Finally, a Horn CNF is more readable than

*Corresponding author.

†Département d'Informatique, Université de Caen, F 14032 Caen Cedex, France. *E-mail addresses:* {hebrard,zanutti}@info.unicaen.fr

a set of examples, since it can be read as a set of Prolog-like propositional rules for instance.

Several algorithms have been proposed in the literature for converting a Horn relation into a Horn CNF. For instance, [3] and [5] both describe the same *envelope-based* algorithm, with running time $O(|R|^2n^2)$, where $|R|$ is the number of input tuples and n the number of variables. Another algorithm is proposed in [2], even if it is meant to solve the corresponding problem in the framework of *exact identification with membership and equivalence queries*; this algorithm runs in time $O(|R|m^2n^2)$, where m is the minimal number of clauses in a Horn CNF describing R, and guarantees that the computed Horn CNF ϕ contains at most $m(n+1)$ clauses; since there always exists a Horn CNF describing R and containing at most $|R|n$ clauses (see [9]), the algorithm of [2] runs in time $O(|R|^3n^4)$. Finally, [9] gives an algorithm that runs in time $O(|R|^2n^2)$ but needs no previous testing of whether R is Horn; a CNF ϕ describing R is computed in all cases, and ϕ is Horn if and only if R also is (this is achieved by computing a *prime* CNF ϕ). Note that for all algorithms the resulting Horn CNF ϕ can be minimized to have at most $m(n-1)$ clauses in time quadratic in the length of ϕ [4], i.e., in time $O(|R|^2n^6)$ for the envelope-based algorithm, since it outputs a formula containing at most $|R|n^2$ clauses, and in time $O(|R|^2n^4)$ for the algorithm of [9], since it outputs a formula containing at most $|R|n$ clauses.

We give here a new algorithm, which runs in time $O(|R|n(|R|+n))$ and thus faster than all those previously known. This is achieved by exploiting in a new manner the well-known closure property of Horn relations. Moreover, the output formula is within the same size bounds as that of [9], i.e., it is never bigger than the number of variables times the size of R, but can be exponentially smaller. Note that the classical test for deciding whether there is a Horn description of R at all requires time $O(|R|^2n)$, and thus that our algorithm for *describing* R is not far from being as efficient.

The note is organized as follows. Section 2 reviews the useful notions and previous work. Section 3 defines the formula ϕ that we propose to compute. Finally, Section 4 discusses computational issues.

2 Preliminaries

We assume basic knowledge about propositional logic. Let $x_1, x_2, \ldots, x_n$ be n Boolean variables. A *literal* is either a variable x_i (*positive literal*) or the negation $\neg x_i$ of one (*negative literal*). A clause is a finite set of literals, viewed as their disjunction, and a formula in *Conjunctive Normal Form* (CNF) is a finite set of clauses, viewed as their conjunction. A formula in CNF is *Horn* if each one of its clauses is Horn, i.e., contains at most one positive literal.

A n-place tuple $m \in \{0,1\}^n$ is viewed as a 0/1 assignment to $x_1, x_2, \ldots, x_n$, in this order; $m[i]$ denotes the ith component of m or, equivalently, the value assigned to x_i. A n-place relation $R \subseteq \{0,1\}^n$ is a set of n-place tuples; $|R|$ denotes the number of tuples in R. A tuple m is called a *model* of a CNF ϕ if it satifies ϕ (written $m \models \phi$), and a formula ϕ over $x_1, x_2, \ldots, x_n$ is said to

describe a n-place relation R if R is exactly the set of models of ϕ.

Let us recall from [7] that a relation R can be described by at least one Horn CNF if and only if for all $m, m' \in R$, the tuple $m \wedge m'$ (componentwise logical AND) is also in R; R is then said $\wedge$-*closed*. This yields a straightforward test for deciding whether a given n-place relation R can be described by a Horn CNF: first sort R with a lexicographic sort [1] in time $O(|R|n)$, represent it with a trie, then for all $m, m' \in R$ ($O(|R|^2)$ such pairs), decide whether $m'' = m \wedge m'$ is in R (in time $O(n)$ since R is represented by a trie); return 'NO' if such a test fails, 'YES' otherwise. The overall complexity of the algorithm is thus $O(|R|^2 n)$.

The problem we address in this note is the one of computing a Horn CNF ϕ describing a given n-place $\wedge$-closed relation R. Our algorithm solves it in time $O(|R|n(|R| + n))$ and outputs at most $|R|n$ clauses. Note that previously deciding whether R is $\wedge$-closed does not increase its asymptotical complexity.

3 The Horn formula

From now on, R denotes a fixed nonempty n-place $\wedge$-closed relation; note indeed that the case $R = \emptyset$ is obvious, since this relation is described by the Horn CNF $\{\emptyset\}$ whatever n is. We first define the Horn CNF $\phi(R)$ that our algorithm will output, and computational issues will be discussed in Section 4. As a running example, we define the 4-place relation

$$R_e = \{0000, 0001, 0100, 1000, 1001, 1110\}$$

It is easily seen that R_e is $\wedge$-closed.

We introduce a new symbol, '?', whose intuitive meaning is "0 or 1". For $t \in \{0, 1, ?\}^n$, recall that $t[i]$ denotes the ith component of t; we define $ext(t)$ to be the set of tuples $\{m \in \{0,1\}^n \mid \forall i = 1, \ldots, n, t[i] \neq ? \Rightarrow m[i] = t[i]\}$ and $cl(t)$ to be the clause $\{x_i \mid t[i] = 0\} \cup \{\neg x_i \mid t[i] = 1\}$. For instance, we have $ext(?10?) = \{0100, 0101, 1100, 1101\}$ and $cl(?10?) = \{\neg x_2, x_3\}$. Let us first remark that the clause $cl(t)$ is Horn if and only if t has at most one component equal to 0. Now for $T \subseteq \{0, 1, ?\}^n$, we define $ext(T)$ to be the set of tuples $\bigcup_{t \in T} ext(t)$, and $cnf(T)$ to be the CNF $\{cl(t) \mid t \in T\}$. Note that for $t \in \{0, 1, ?\}^n$, the models of $cl(t)$ are exactly the tuples in $\{0,1\}^n \backslash ext(t)$, and consequently that for $T \subseteq \{0,1\}^n$, the models of $cnf(T)$ are exactly the tuples in $\{0,1\}^n \backslash ext(T)$. Thus the following lemma is straightforward.

Lemma 1 *Let $T \subseteq \{0, 1, ?\}^n$. If $ext(T) = \{0,1\}^n \backslash R$, then $cnf(T)$ describes R.*

Example 2 (continued) *The complement of R_e in $\{0,1\}^4$ is the set of tuples $\{0010, 0011, 0101, 0110, 0111, 1010, 1011, 1100, 1101, 1111\}$. Accordingly, the CNF $cnf(\{0,1\}^4 \backslash R_e)$ is*

$$\{\{x_1, x_2, \neg x_3, x_4\}, \{x_1, x_2, \neg x_3, \neg x_4\}, \ldots, \{\neg x_1, \neg x_2, \neg x_3, \neg x_4\}\}$$

and describes R_e. Now let H_e be the set of $\{0, 1, ?\}$-tuples:

$$\{0?1?, ?1?1, 011?, 101?, 110?, 1111\}$$

It is easily seen that $ext(H_e) = \{0,1\}^4 \backslash R_e$, *and thus that the Horn CNF* $cnf(H_e)$:

$$\{\{x_1, \neg x_3\}, \{\neg x_2, \neg x_4\}, \ldots, \{\neg x_1, \neg x_2, \neg x_3, \neg x_4\}\}$$

describes R_e *as well.*

Our purpose is thus to define a set of tuples $H(R) \subseteq \{0,1,?\}^n$ such that $ext(H(R)) = \{0,1\}^n \backslash R$ and $\forall h \in H(R), cl(h)$ is Horn. The Horn CNF $\phi(R)$ describing R will then be defined to be $cnf(H(R))$.

For $t \in \{0,1\}^n \backslash R$, let $com(t)$ be the maximum integer in $\{1, \ldots, n\}$ such that $\exists m \in R, \forall i < com(t), m[i] = t[i]$. We define $H(R)$ to be the set of tuples $\{h(t) \mid t \in \{0,1\}^n \backslash R\}$, where $h(t) \in \{0,1,?\}^n$ is defined according to three different cases. Informally, $h(t)$ is obtained from t by replacing some of its components with '?', and at most one of its components is 0.

Let $t \in \{0,1\}^n \backslash R$.

- Case $t[com(t)] = 0$:

  We define $h(t)$ to be such that

  $$\begin{cases} \forall i < com(t) \text{ with } t[i] = 1, h(t)[i] = 1 \\ h(t)[com(t)] = 0 \\ otherwise, h(t)[i] = ? \end{cases}$$

  For instance, with $t = 1100 \in \{0,1\}^4 \backslash R_e$, we have $com(t) = 3$ and $t[com(t)] = 0$, thus $h(t)$ is 110?. The following claim ensures that every $m \in R$ satisfies $cl(h(t))$.

  Claim 3 *If* $t[com(t)] = 0$, *then* $R \cap ext(h(t)) = \emptyset$.

  Proof For sake of contradiction, let $\mu \in R$ with $\mu \in ext(h(t))$. Let $m \in R$ with $\forall i < com(t), m[i] = t[i]$. Since R is $\wedge$-closed, the tuple $m_0 = \mu \wedge m$ is in R; but $\forall i < com(t)$, if $t[i] = 1$ then (i) by definition of $h(t)$, $h(t)[i] = 1$ and thus $\mu[i] = 1$, since $\mu \in ext(h(t))$, and (ii) $m[i] = 1$ by definition of m; finally, $m_0[i] = 1$. By definition of m, we also get $\forall i < com(t)$, $t[i] = 0 \Rightarrow m[i] = 0 \Rightarrow m_0[i] = 0$; finally, $\forall i < com(t), m_0[i] = t[i] = m[i]$, and since $\mu \in ext(h(t))$, $\mu[com(t)] = 0$ and $m_0[com(t)] = 0 = t[com(t)]$. This contradicts the maximality of $com(t)$. □

Now we are left with the case where $t[com(t)] = 1$. Let $M(t) = \{m' \in R \mid \forall i \leq com(t), t[i] = 1 \Rightarrow m'[i] = 1\}$. We define $sim(t) \in \{0, 1, \ldots, n\}$ to be 0 if $M(t) = \emptyset$, and otherwise to be the greatest integer in $\{1, \ldots, n\}$ such that $\exists m' \in M(t), \forall i < sim(t), m'[i] = t[i]$. Note that by definition $sim(t) \leq com(t)$.

- Case $t[com(t)] = 1$ and $sim(t) = 0$:

  We define $h(t)$ to be such that

  $$\begin{cases} \forall i \leq com(t) \text{ with } t[i] = 1, h(t)[i] = 1 \\ \text{otherwise }, h(t)[i] = ? \end{cases}$$

For instance, with $t = 0101 \in \{0,1\}^4 \backslash R_e$, we have $com(t) = 4$, and it can be checked that $\forall m' \in R_e, \exists i \leq 4, m'[i] = 0$ and $t[i] = 0101[i] = 1$. Thus $sim(t) = 0$, and $h(t)$ is ?1?1. Note that in the general case, by definition of $sim(t)$ we have $\forall m' \in R, \exists i \leq com(t), m'[i] = 0$ and $t[i] = 1$, and the following claim follows.

Claim 4 *If $t[com(t)] = 1$ and $sim(t) = 0$, then $R \cap ext(h(t)) = \emptyset$.*

- Case $t[com(t)] = 1$ and $sim(t) \neq 0$:

  By definition of $sim(t)$ we know that $\exists m' \in R, \forall i \leq com(t), t[i] = 1 \Rightarrow m'[i] = 1$. We show that $t[sim(t)] = 0$. Assume indeed $t[sim(t)] = 1$, and let $m' \in M(t)$ with $\forall i < sim(t), m'[i] = t[i]$. By definition of $M(t)$ and since $sim(t) \leq com(t)$, we have $m'[sim(t)] = 1$. Thus $\forall i < sim(t) + 1, m'[i] = t[i]$, which contradicts the maximality of $sim(t)$. Thus $t[sim(t)] = 0$; we then define $h(t)$ to be such that

  $$\begin{cases} \forall i \leq com(t) \text{ with } t[i] = 1, h(t)[i] = 1 \\ h(t)[sim(t)] = 0 \\ \text{otherwise }, h(t)[i] = ? \end{cases}$$

  For instance, with $t = 0010 \in \{0,1\}^4 \backslash R_e$, we have $com(t) = 3$, and $1110 \in R_e$ is such that $\forall i \leq 3, 0010[i] = 1 \Rightarrow 1110[i] = 1$. Thus we have $sim(t) \neq 0$ and find $sim(t) = 1$, which yields $h(t) = 0?1?$.

  Claim 5 *If $t[com(t)] = 1$ and $sim(t) \neq 0$, then $R \cap ext(h(t)) = \emptyset$.*

  Proof For sake of contradiction, let $\mu \in R$ with $\mu \in ext(h(t))$, and let $m' \in M(t)$ be such that $\forall i < sim(t), m'[i] = t[i]$. Since R is $\wedge$-closed, the tuple $m_0 = \mu \wedge m'$ is in R; but as for Claim 3, we have $\forall i < com(t), t[i] = 1 \Rightarrow m_0[i] = 1$, and $\forall i < sim(t), t[i] = 0 \Rightarrow m'[i] = 0 \Rightarrow m_0[i] = 0$; finally, we get $\forall i < sim(t), m_0[i] = t[i]$. But since $\mu \in ext(h(t))$, we also get $\mu[sim(t)] = 0$ and thus $m_0[sim(t)] = 0$. Finally, $\forall i < sim(t) + 1, m_0[i] = t[i]$, which contradicts the maximality of $sim(t)$. □

Finally, we define $H(R)$ to be the set of tuples $\{h(t) \mid t \in \{0,1\}^n \backslash R\}$ and the CNF $\phi(R)$ to be $cnf(H(R))$. For instance, with R_e we get the CNF

$$\phi(R_e) = \{\{x_1, \neg x_3\}, \{\neg x_2, \neg x_4\}, \{x_1, \neg x_2, \neg x_3\}, \{\neg x_1, x_2, \neg x_3\}, \{\neg x_1, \neg x_2, x_3\}, \{\neg x_1, \neg x_2, \neg x_3, \neg x_4\}\}$$

Then the following result holds.

Proposition 6 *The CNF $\phi(R)$ is Horn and describes R.*

Proof By construction $\phi(R)$ is Horn. Now Claims 3, 4 and 5 ensure that $R \cap ext(H(R)) = \emptyset$, i.e., $ext(H(R)) \subseteq \{0,1\}^n \backslash R$. Conversely, it is easily seen that for every $t \in \{0,1\}^n \backslash R$ we have $t \in ext(h(t))$, which yields $\{0,1\}^n \backslash R \subseteq ext(H(R))$. Finally $\{0,1\}^n \backslash R = ext(H(R))$, and Lemma 1 concludes. □

4 The algorithm

Now we show that ϕ can be computed in time $O(|R|n(|R|+n))$ given R. The first point is to express $H(R)$ in terms of the elements of R, since R is the input of the problem; for the moment it is expressed in terms of the elements of $\{0,1\}^n \backslash R$.

For $m \in R$, let $p(m)$ (resp. $s(m)$) denote the length of the longest common prefix of m and its predecessor (resp. successor) $m' \in R$ with respect to the lexicographic order, or -1 if m' does not exist. Now for $t \in \{0,1\}^n \backslash R$ with $t[com(t)] = 0$ (resp. $t[com(t)] = 1$), let $\mu(t)$ be the least (resp. greatest) element of R with respect to the lexicographic order, such that $\forall i < com(t), t[i] = \mu(t)[i]$; it is immediate that $p(\mu(t)) + 1 < com(t)$ (resp. $s(\mu(t)) + 1 < com(t)$).

Example 7 (continued) *We go on with the relation R_e. The predecessor of $m = 0001$ in R_e with respect to the lexicographic order is 0000, thus $p(0001) = 3$, and its successor is 0100, thus $s(0001) = 1$. Similarly, the predecessor of $m = 1110$ in R_e is 1001, thus $p(1110) = 1$, and it has no successor, thus $s(1110) = -1$. Now we have seen (Section 3) that with $t = 1100 \in \{0,1\}^4 \backslash R_e$, we have $com(t) = 3$ and consequently $t[com(t)] = 0$; we obtain $\mu(1100) = 1110$. Similarly, with $t = 0101$ we have seen that $com(t) = 4$ and $t[com(t)] = 1$, and we obtain $\mu(t) = 0100$.*

It is easily seen from the definition of $h(t)$ that if $\mu(t) = \mu(t')$ and $com(t) = com(t')$, then $h(t) = h(t')$, since by definition of $\mu(t)$ we have $\forall i < com(t), t[i] = \mu(t)[i]$; thus we can define $h(t)$ in terms of the elements of R. Indeed, let $j \in \{1,\ldots,n\}$, $m \in R$ and $t \in \{0,1\}^n \backslash R$ such that $\mu(t) = m$ and $com(t) = j$. If $j > p(m)+1$ and $m[j] = 1$, or $j > s(m)+1$ and $m[j] = 0$, we define $h(m,j)$ to be $h(t)$. From the definition of $p(m)$ (resp. $s(m)$) and from $p(\mu(t)) + 1 < com(t)$ (resp. $s(\mu(t)) + 1 < com(t)$), as remarked above, it is easily seen that for such m and j a convenient t always exists, and from $(\mu(t) = \mu(t'), com(t) = com(t')) \Rightarrow h(t) = h(t')$ we conclude that $H(R) = \{h(m,j) \mid m \in R$ and $(p(m)+1 < j, m[j] = 1)$ or $(s(m) + 1 < j, m[j] = 0)\}$.

Example 8 (continued) *We have seen that $m = 1110 \in R_e$ is such that $m = \mu(1100)$, and that $com(1100) = 3$. Since $m[3] = 1$ and $p(m) = 1$ (see Example 7), thus $3 > p(m) + 1$, it follows that $h(m,3) = h(1100) = 110$? (see Section 3). Similarly, we have seen that $m = 0100 = \mu(0101)$ and $com(0101) = 4$. Since $m[4] = 0$ and $s(m) = 0$ (its successor in R_e is 1000), we get $h(m,4) = h(0101) =$?1?1.*

Now for $m \in R$ and convenient $j \in \{1,\ldots,n\}$ we can express $h(m,j)$ in terms of m and j. The following is nothing more than the translation of the definition of $h(t)$ into these terms. Let $m \in R$, $j \in \{1,\ldots,n\}$ and $t \in \{0,1\}^n$ such that $\mu(t) = m$ and $com(t) = j$.

- Case $j > p(m) + 1$ and $m[j] = 1$:

Then $t[j] = t[com(t)] = 0$ and we get:

$$\begin{cases} \forall i < j \text{ with } m[i](= t[i]) = 1, h(m,j)[i] = 1 \\ h(m,j)[j] = 0 \\ otherwise, h(m,j)[i] = ? \end{cases}$$

Now if $j > s(m) + 1$ and $m[j] = 0$, then $t[j] = t[com(t)] = 1$; remark that $\forall i < j = com(t), m[i] = t[i]$, but $m[j] = 0$ and $t[j] = 1$. Let $sim(m,j) = sim(t)$; as for $h(t)$ we define $h(m,j)$ according to the value of $sim(m,j)$.

- Case $j > s(m) + 1$, $m[j] = 0$ and $sim(m,j) = 0$:

  Then we define $h(m,j)$ to be such that:

$$\begin{cases} \forall i < j \text{ with } m[i](= t[i]) = 1, h(m,j)[i] = 1 \\ h(m,j)[j] = 1 \text{ (since } m[j] = 0 \text{ but } t[j] = 1) \\ \text{otherwise }, h(m,j)[i] = ? \end{cases}$$

- Case $j > s(m) + 1$, $m[j] = 0$ and $sim(m,j) \neq 0$:

  Then we define $h(m,j)$ to be such that:

$$\begin{cases} \forall i < j \text{ with } m[i] = 1, h(m,j)[i] = 1 \\ h(m,j)[j] = 1 \\ h(m,j)[sim(m,j)] = 0 \\ \text{otherwise }, h(m,j)[i] = ? \end{cases}$$

Now let us remark that if $p(m)$, $s(m)$ and $sim(m,j)$ are known for all $m \in R$ and all $j \in \{1, \ldots, n\}$ such that $s(m) + 1 < j$ and $m[j] = 0$, then computing $H(R) = \bigcup_{m,j} h(m,j)$ requires time $O(|R|n^2)$. If R is sorted then $p(m)$ and $s(m)$ are easily computed, thus the only difficult point is the computation of $sim(m,j)$.

Let us recall that with $M(m,j) = \{m' \in R \mid m'[j] = 1 \text{ and } \forall i < j, m[i] = 1 \Rightarrow m'[i] = 1\}$, $sim(m,j)$ is by definition 0 if $M(m,j) = \emptyset$, and otherwise the greatest integer in $\{1, \ldots, n\}$ such that there exists $m' \in M(m,j)$ with $\forall i < sim(m,j), m'[i] = m[i]$. For given m and j it is thus easy to compute $sim(m,j)$ in time $O(|R|n)$, but this would yield time $O(|R|^2 n^2)$ for computing all $sim(m,j)$'s. We show that for a fixed $m \in R$, one can compute in time $O(|R|n)$ the value of $sim(m,j)$ for all convenient j's, which yields time $O(|R|^2 n)$ for computing all $sim(m,j)$'s. This can indeed be achieved by the algorithm on Figure 1. The principle of the algorithm is simply to identify, for given m and j and for a $m' \in R$, whether $m' \in M(m,j)$. Indeed, if for a given j we have $m[j] = 1$ and $m'[j] = 0$, then by definition of $M(m,j')$ we have $\forall j' > j, m' \notin M(m,j')$. Thus the algorithm reads m' as long as $m' \in M(m,j)$ for the current j, and updates $sim(m,j)$ accordingly, and then stops reading m', i.e., lets the computed value of $sim(m,j')$ unchanged for every $j' > j$. Thus for a given $m \in R$ the algorithm of Figure 1 computes $sim(m,j)$ for all convenient j's in time $O(|R|n)$.

Finally, we can establish the running time of the full algorithm, which is summarized on Figure 2.

Input: $m \in R$
Output: $sim(m,j)$ for all $j \in \{1,\ldots,n\}$ such that $j > s(m)+1$ and $m[j]=0$

Begin
  For $j = 1,\ldots,n$ **do** $sim(m,j) \leftarrow 0$ **endfor**;
  For every $m' \in R$ **do**
    $j_0 \leftarrow$ the maximum index with $\forall j < j_0, m'[j] = m[j]$;
    $j \leftarrow j_0$;
    While $m'[j]=1$ or $m[j]=0$ **do**
      If $j > s(m)+1$, $m[j]=0$ and $m'[j]=1$ **then**
        $sim(m,j) \leftarrow \max(sim(m,j), j_0)$;
      Endif;
      $j \leftarrow j+1$;
    Endwhile;
  Endfor;
End;

Figure 1: Computation of $sim(m,j)$ for a given m and all convenient j's

Input: a $\wedge$-closed relation $R \subseteq \{0,1\}^n$
Output: a Horn CNF $\phi(R)$ describing R;

Begin
  $H(R) \leftarrow \emptyset$;
  Sort R by lexicographic order: $R = (m_1, m_2, \ldots, m_{|R|})$;
  For $k = 1,2,\ldots,|R|$ **do**
    compute $p(m_k)$ and $s(m_k)$;
    compute $sim(m_k,j)$ for all convenient j's with the algorithm of Figure 1;
    For $j \in \{1,\ldots,n\}$ with $j > p(m_k)+1, m_k[j]=1$, or $j > s(m_k)+1, m_k[j]=0$ **do**
      $H(R) \leftarrow H(R) \cup \{h(m_k,j)\}$;
    Endfor;
  Endfor;
  return $\phi(R) = cnf(H(R))$;
End;

Figure 2: Computation of $\phi(R)$ from R

Proposition 9 *Let R be a n-place $\wedge$-closed relation. A Horn CNF ϕ describing R and containing at most $|R|n$ clauses can be computed in time $O(|R|n(|R|+n))$.*

Proof Sorting R requires time $O(|R|n)$ with a lexicographic sort [1]. Now the body of the main *For* loop is executed $|R|$ times; for a given $m_k \in R$, computing $p(m_k)$ and $s(m_k)$ requires time $O(n)$ by reading m_{k-1} and m_{k+1}, computing $sim(m_k, j)$ for all convenient j's at a time requires time $O(|R|n)$, and finally the body of the inner *For* loop is executed n times, with a running time $O(n)$ for a given j. Thus the main *For* loop requires time $O(|R|(n + |R|n + n^2))$. Finally, computing $cnf(H(R))$ from $H(R)$ consists in reading once $H(R)$. Now the number of computed clauses is obviously bounded by the number of possible pairs $\{m_k, j\}$, i.e., $|R|n$, which completes the proof. □

References

[1] A. Aho, J. Hopcroft, and J. Ullman. *The design and analysis of computer algorithms.* Addison-Wesley, 1974.

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

[3] R. Dechter and J. Pearl. Structure identification in relational data. *Artificial Intelligence*, 58:237–270, 1992.

[4] P.L. Hammer and A. Kogan. Optimal compression of propositional Horn knowledge bases: complexity and approximation. *Artificial Intelligence*, 64:131–145, 1993.

[5] D. Kavvadias, C.H. Papadimitriou, and M. Sideri. On Horn envelopes and hypergraph transversals (extended abstract). In *Proc. 4th International Symposium on Algorithms And Computation (ISAAC'93)*, number 762 in Springer Lecture Notes in Computer Science, pages 399–405. Springer, 1993.

[6] D. Kavvadias and M. Sideri. The inverse satisfiability problem. *SIAM Journal on Computing*, 28(1):152–163, 1998.

[7] J. McKinsey. The decision problem for some classes of sentences without quantifiers. *Journal of Symbolic Logic*, 8:61–77, 1943.

[8] T.J. Schaefer. The complexity of satisfiability problems. In *Proc. 10th Annual ACM Symposium on Theory Of Computing (STOC'78)*, pages 216–226. ACM Press, 1978.

[9] B. Zanuttini and J.-J. Hébrard. A unified framework for structure identification. *Information Processing Letters*, 81(6):335–339, 2002.