Approximating Propositional Knowledge with Affine Formulas

Bruno Zanuttini [1]

Abstract. We consider the use of affine formulas, i.e., conjonctions of linear equations modulo 2, for approximating propositional knowledge. These formulas are very close to CNF formulas, and allow for efficient reasoning ; moreover, they can be minimized efficiently. We show that this class of formulas is identifiable and PAC-learnable from examples, that an affine least upper bound of a relation can be computed in polynomial time and a greatest lower bound with the maximum number of models in subexponential time. All these results are better than those for, e.g., Horn formulas, which are often considered for representing or approximating propositional knowledge. For all these reasons we argue that affine formulas are good candidates for approximating propositional knowledge.

1 INTRODUCTION

Affine formulas correspond to one of the only six classes of relations for which the generalized satisfiability problem is tractable [9]. These formulas consist in conjunctions (or, equivalently, systems) of linear equations modulo 2, and are very close to usual CNF formulas. Indeed, in some sense usual disjunction inside the clauses is simply replaced with addition modulo 2, and as well as, e.g., Horn formulas, affine formulas are stable under conjunction. Intuitively, while Horn clauses represent *causal* relations, linear equations represent *parity* relations between variables (with, as a special case, equations over only two variables specifying either that they must be equal or that they must be different). Moreover, most of the notions that are commonly used with CNF formulas (such as prime implicants/ates) can be transposed straightforwardly to them. Finally, a great deal of reasoning tasks that are intractable with general CNF formulas are tractable with affine formulas : e.g., satisfiability or deduction. It is also true of problems that are intractable even with Horn formulas, although these formulas are often considered for representing or approximating knowledge : e.g., counting of models, minimization.

Nevertheless, not many authors have studied this class of formulas ; mainly Schaefer [9], Kavvadias, Sideri and Stavropoulos [6, 8] and Zanuttini and Hébrard [12]. Moreover, none of them has really studied them as a candidate for representing or approximating propositional knowledge. We believe however that they are good candidates for approximation, for instance in the sense of [10] : given a knowledge base (KB), the idea is to compute several approximations of it with better computational properties, and to use later these approximations for helping to answer queries that are asked to it. Most of the time, the approximations will give the answers to these queries, and in case they do not, since the approximations have good computational properties, only a small amount of time will have been lost and the query will be asked directly to the KB. Note also that some KBs can be represented *exactly* by a formula with good properties ; in this case, the formula can give the answer to any query. To summarize, approximations can help saving a lot of time when answering queries (for instance in an on-line framework), especially if they can be reasoned with efficiently and if their size is reasonable.

Not many classes of formulas satisfy these requirements. Horn formulas are often considered for approximating knowledge (see for instance [10]), but they have some limits : e.g., the shortest Horn approximation of a knowledge base may be exponentially larger than its set of models, and some problems are not tractable with Horn formulas : counting the models, abduction, minimization... Affine formulas satisfy these requirements quite better : on one hand they all can be made very small, which guarantees that an affine approximation can almost never be bigger than the original KB, and on the other hand, they have very good computational properties for reasoning.

We focus here on the *acquisition* of affine formulas from relations, with a computational point of view ; in other words, we are interested in the complexity of computing affine approximations of knowledge bases represented as sets of vectors. We first present (Section 2) several simple technical results about vector spaces that will be useful. Then we consider (Section 3) the *identification* of an affine structure in a relation [4], which corresponds to the special case when the knowledge base can be represented exactly by an affine formula ; it is well-known that affine formulas are identifiable, but we recall the proof for sake of completeness. Then we study (Section 4) the process of *approximating* a relation with affine formulas [10] : we show that the affine least upper bound of a relation can be computed in polynomial time, and that an affine greatest lower bound with the maximum number of models can be computed in subexponential time. Finally (Section 5), we consider the problem of *PAC-learning* these formulas [11], which corresponds to the case when the relation is affine but the algorithm has a limited access to it ; we show that affine formulas are PAC-learnable from examples only.

We wish to emphasize that these results are better than the corresponding ones for Horn formulas. Although they are also identifiable, the problem of approximation with Horn formulas is intractable : the Horn least upper bound of a relation may be exponentially larger than it, and computing a Horn greatest lower bound with the maximum number of models is NP-hard. Finally, the question is still open whether Horn formulas are PAC-learnable from examples. We also wish to emphasize here that we consider the class of affine formulas mainly for *approximating* propositional knowledge, independently of the knowledge that they can represent *exactly*.

2 PRELIMINARIES AND TECHNICAL TOOLS

We assume a countable number of propositional variables $x_1, x_2, \ldots$. A *linear equation (modulo 2)* is an equation of the form $x_{i_1} \oplus x_{i_2} \oplus$

[1] GREYC Université de Caen, bd Mal Juin, 14032 Caen Cedex, France

$\cdots \oplus x_{i_k} = a$, $a \in \{0,1\}$, where $x_{i_1} \oplus \cdots \oplus x_{i_k}$ stands for $x_{i_1} + \cdots + x_{i_k}$ (mod 2). An *affine formula* is a finite conjunction of linear equations ; e.g., the formula ϕ :

$$(x_1 \oplus x_3 \oplus x_4 = 1) \wedge (x_2 \oplus x_3 = 0)$$

is affine. A n-place vector $m \in \{0,1\}^n$, seen as a $0/1$ assignment to the variables $x_1, x_2, \ldots, x_n$, is a *model* of an affine formula ϕ over the same variables (written $m \models \phi$) if m satisfies all the equations of ϕ. We denote by $m[i]$ the ith component of m, and for $m_1, m_2 \in \{0,1\}^n$, we write $m_1 \oplus m_2$ for the n-place vector m such that $\forall i = 1, \ldots, n, m[i] = m_1[i] \oplus m_2[i]$.

A set of vectors $R \subseteq \{0,1\}^n$ is called an *n-place relation*, and an n-place relation R is said *affine* if it is the set of all the models of an affine formula ϕ over the variables $x_1, x_2, \ldots, x_n$; ϕ is then said to *describe* R. For instance, the 5-place relation R :

$$\{00010, 00011, 01100, 01101, 10000, 10001, 11110, 11111\}$$

is affine and is described by the formula ϕ above. The number of vectors in a relation R is written $|R|$.

It is a well-known fact that the satisfiability problem is polynomial for affine formulas [9] ; indeed, it corresponds to deciding whether a given system of equations modulo 2 has a solution, and thus can be solved by gaussian elimination [3, Section 8][2]. Thus this problem can be solved in time $O(k^2 n)$ for an affine formula of k equations over n variables. Deduction of clauses, i.e., the problem of deciding $\phi \models \alpha$ where ϕ is an affine formula and α is a clause (finite disjunction of negated and unnegated variables), is polynomial too ; indeed, it corresponds to deciding whether the affine formula $\phi \wedge \bigwedge_{x_i \in \alpha}(x_i = 0) \wedge \bigwedge_{\neg x_i \in \alpha}(x_i = 1)$ is unsatisfiable, which requires time $O((k+\ell)^2 n)$ for a clause of length ℓ. Minimizing an affine formula or counting its models can also be performed efficiently by putting ϕ in *echelon form* [3, Section 8], which again requires time $O(k^2 n)$ with gaussian elimination.

We now introduce the parallel that we will use between affine relations and vector spaces over the two-element field (*vector spaces* for short). For R a relation and $m \in R$, let R_m denote the relation $\{\mu \oplus m, \mu \in R\}$; for ϕ an affine formula and $m \models \phi$, let ϕ_m denote the affine formula obtained from ϕ by replacing x_i with $x_i \oplus 1$ for every i such that $m[i] = 1$, and simplifying. Let us first remark that for all R, m, ϕ, $(R_m)_m = R$ and $(\phi_m)_m = \phi$. Now suppose that R is affine and that ϕ describes it. Then for any model m of ϕ (i.e., for any $m \in R$), it is easily seen that ϕ_m describes R_m and that R_m is a vector space ; conversely, if R is a relation such that for any $m \in R$, R_m is a vector space, then R is affine (see [3, Theorems 8.9 and 9.1]). This correspondence allows us to use the usual notions of linear algebra, and especially the notion of basis of a vector space.

Let us first recall that the cardinality of a vector space over the two-element field is always a power of 2. A *basis* B of a vector space V is a set of $\log_2 |V|$ vectors of V that are linearly independent, i.e., such that none is a linear combination of the others, and that generate V in the sense that their linear combinations are all and only the elements of V ; let us also recall that two different linear combinations of linearly independent vectors give two different vectors (which yields $|V| = 2^{|B|}$). For more details we refer the reader to [3].

Example 1 *We go on with the relation R above. Since R is affine and $01100 \in R$, the relation R_{01100} :*

$$\{00000, 00001, 01110, 01111, 10010, 10011, 11100, 11101\}$$

is a vector space, and its subset $\{00001, 01110, 10010\}$ is one of its bases.

We end this section by giving four simple complexity results concerning bases and linearly independent sets of vectors. The rest of the paper uses no result from linear algebra but these ones. The proofs are given in appendix.

Proposition 1 *Let $B \subseteq \{0,1\}^n$ and $m \in \{0,1\}^n$. Deciding whether B is a set of linearly independent vectors, or whether m is linearly independent from B can be performed in time $O(|B|^2 n)$.*

Proposition 2 *Given a relation R over n variables, finding a linearly independent subset of R that is maximal for set inclusion requires time $O(|R|n^3)$.*

Proposition 3 *Given a basis B of a vector space $V \subseteq \{0,1\}^n$, computing an affine formula ϕ describing V requires time $O(n^4)$, and ϕ contains at most n equations.*

Proposition 4 *Given an n-place relation R and a linearly independent set of vectors $B \subseteq R$, deciding whether the vector space V generated by B is included in R requires time $O(|R|n)$.*

3 IDENTIFICATION

The problem of *structure identification* was formalized by Dechter and Pearl [4]. It consists in some kind of knowledge compilation, where a formula is searched with required properties and that admits a given set of models. In our framework, it corresponds to checking whether some knowledge given as a relation can be represented *exactly* by an affine formula before trying to *approximate* it by such a formula. Identifying an affine structure in a relation R means discovering that R is affine, and computing an affine formula ϕ describing it.

It is well-known from linear algebra (see also [9, 6]) that affine structures are *identifiable*, i.e., that there exists an algorithm that, given a relation R, can either find out that R is the set of models of no affine formula over the same variables, or give such a formula, in time polynomial in the size of R.

The algorithm is the following. We first transform the problem into one of vector spaces, by choosing any $m \in R$ and computing the relation R_m. The problem has now become that of deciding whether R_m is a vector space. Then we compute a subset B_m of R_m that is linearly independent and maximal for set inclusion (Proposition 2) ; we know by maximality of B_m that all the vectors in R_m are linearly dependent from B_m, i.e., that R_m is included in the vector space generated by B_m. Thus if $|R_m| = 2^{|B_m|}$, we can conclude that R_m is exactly this vector space, and we can compute from B_m an affine formula ϕ_m describing R_m (Proposition 3) ; the formula $\phi = (\phi_m)_m$ will describe $(R_m)_m = R$. Otherwise, if $|R_m| \neq 2^{|B_m|}$, we can conclude that R_m is not a vector space, i.e., that R is not affine.

Proposition 5 (identification) *Affine structures are identifiable in time $O(|R|n^3 + n^4)$, where R is the relation and n the number of variables.*

Proof. Computing R_m from R requires time $O(|R|n)$, computing B_m, $O(|R|n^3)$ (Proposition 2), computing ϕ_m from B_m, $O(n^4)$ (Proposition 3) and finally, computing ϕ requires time $O(|\phi_m|) \subseteq O(n^2)$. □

For sake of completeness, we also mention the approach in [12] for proving the identifiability of affine structures ; this approach exhibits and uses a syntactic link between usual CNFs and affine formulas instead of results from linear algebra.

[2] Most of the results we will use from [3] are given for equations with real or complex coefficients and unknowns, but can be applied straightforwardly to our framework with the same proofs.

4 APPROXIMATION

We now turn our attention to the problem of *approximation* itself. Approximating a relation R by an affine formula means computing an affine formula ϕ whose set of models is as close as possible to R ; thus this process takes place naturally when R cannot be represented *exactly* by an affine formula. Many measures of closeness can be considered, but we will focus on the two notions explored by Selman and Kautz in [10].

The first way we can approximate R is by finding an affine formula ϕ_{lub} whose set of models R_{lub} is a superset of R, but minimal for set inclusion. Then ϕ_{lub} is called an affine *least upper bound* (LUB) of R [10, 4]. The second notion is dual to this one : we now search for an affine formula ϕ_{glb} whose set of models R_{glb} is a subset of R, but maximal for set inclusion. The formula ϕ_{glb} is then called an affine *greatest lower bound* (GLB) of R [10]. Remark that if R is affine, then ϕ_{lub} and ϕ_{glb} both describe it.

Example 2 (continued) *We consider the non-affine relation* $R = \{00011, 01101, 10000, 11110, 11111\}$. *It is easily seen that* $\phi = (x_1 \oplus x_3 \oplus x_4 = 1) \wedge (x_2 \oplus x_3 = 0)$ *is its (unique) affine LUB (with* 8 *models), and that the formula* $\phi \wedge (x_3 \oplus x_4 \oplus x_5 = 0)$ *is its affine GLB with the maximum number of models (*4*).*

Selman and Kautz suggest to use these bounds in the following manner. If R is a knowledge base, store it as well as an affine LUB ϕ_{lub} and an affine GLB ϕ_{glb} of it. When R is asked a deductive query α, i.e., when it must be decided $R \models \alpha$ where α is a clause, first decide $\phi_{lub} \models \alpha$: if the answer is positive, then conclude $R \models \alpha$. On the other hand, if the answer is negative, then decide $\phi_{glb} \models \alpha$: if the answer is negative, then you can conclude $R \not\models \alpha$. In case it is positive, then you must query R itself. In the case of affine (or Horn) approximations, since deduction is tractable, either the answer will have been found quickly with the bounds or only a small amount of time will have been lost, under the condition that the size of the approximation is comparable to or less than the size of R ; but we have seen that, contrary to Horn formulas, affine formulas can always be made very small.

We study here these two notions of approximation with affine formulas.

4.1 Affine LUBs

We first consider affine *LUBs* of relations. Let R be a relation. Once again we transform the problem of computing an affine LUB of R into a problem of vector spaces, by choosing $m \in R$ and considering the relation R_m. Since R_m is a vector space if and only if R is affine, we consider the closure V_m of R_m under linear combinations, i.e., the unique smallest vector space including R_m, and the associated affine relation $V = (V_m)_m$. It is easily seen that V is uniquely defined (whatever $m \in R$ has been chosen) and is the smallest affine relation including R. It follows that the affine LUB ϕ_{lub} of a relation R is unique up to logical equivalence, and that its set of models is exactly V (see also [9, 6]).

Now we must compute an affine formula ϕ_m describing V_m, given the relation R_m ; we will then set $\phi_{lub} = (\phi_m)_m$. The idea is the same as for identification : compute a basis B_m of V_m, and then use Proposition 3 for computing ϕ_m. But we have seen that V_m is the closure of R_m under linear combination, and thus any maximal (for set inclusion) linearly independent subset of R_m is a basis of V_m. Finally, we get the following result.

Proposition 6 (LUB) *Let R be a n-place relation. The affine LUB ϕ_{lub} of R is unique up to logical equivalence and can be computed in time $O(|R|n^3 + n^4)$.*

Proof. We must first choose $m \in R$ and compute R_m, in time $O(|R|n)$. Then we must compute a maximal linearly independent subset B_m of R_m, in time $O(|R|n^3)$ (Proposition 2). Finally, we must compute ϕ_m from B_m and set $\phi_{lub} = (\phi_m)_m$, which requires time $O(n^4)$ (Proposition 3). □

4.2 Affine GLBs

Contrary to the case of LUBs, the affine GLB of a relation is not unique up to logical equivalence in general, and there is even no reason for two affine GLBs of a relation to have the same size. What is most interesting then is to search for an affine GLB $\phi_{max-glb}$ with the maximum number of models over all affine GLBs. The associated decision problem is NP-hard for Horn GLBs (see [7]), but we show here that there exists a subexponential algorithm for the affine case ; remark that while NP-hard problems can be considered intractable, subexponential algorithms can stay reasonable in practice.

We still work with the relation R_m for a given $m \in R$. What we must do is find a vector space V_m included in R_m and with maximum cardinality, and then to compute an affine formula ϕ_m describing V_m ; we will then set $\phi_{max-glb} = (\phi_m)_m$. We proceed by searching the maximal k for which there exists k linearly independent vectors $m_1, \ldots, m_k \in R_m$ that generate a vector space V_m included in R_m. Since k can only range between 1 and $\log_2 |R_m| = \log_2 |R|$, we get a subexponential algorithm.

Proposition 7 (maximum GLB) *Let R be an n-place relation. An affine GLB $\phi_{max-glb}$ of R with the maximum number of models can be computed in time $O(|R|n(\log\log|R|)2^{(\log|R|)^2})$.*

Proof. We search the maximal k by dichotomy. Begin with $k = \log|R|/2$. For a given k, compute all the $\binom{|R_m|}{k}$ subsets of R_m of k vectors, and for each one of them, test whether it is linearly independent (in time $O(k^2 n)$ with Proposition 1) and whether the vector space it is a basis for is included in R_m (in time $O(|R|n)$ with Proposition 4). If it is the case for at least one subset of size k, then increase k (by dichotomy) and go on, otherwise decrease k and go on. Finally, since k is always bounded by $\log_2|R|$, at most $\log_2\log_2|R|$ different k's will have been tried, and we get the time complexity

$$O((\log\log|R|) \times \binom{|R|}{\log|R|} \times ((\log|R|)^2 n + |R|n))$$

which is less than $O((\log\log|R|)|R|^{\log|R|}(|R|n))$, which in turn equals $O(|R|n(\log\log|R|)2^{(\log|R|)^2})$. □

5 PAC-LEARNING

We finally turn our attention to the problem of *learning* affine formulas from examples. The main difference with the other problems considered so far is that the algorithm has not access to the entire relation R. It must compute an affine approximation of an affine relation R by asking as few informations as possible about R. Nevertheless, learning is a rather natural extension of approximation, since it corresponds in some sense to introducing a dynamical aspect in it : the algorithm is supposed to improve its result when it is allowed more time for asking informations about R.

We consider here the *PAC-learning* framework of Valiant [11, 1], with examples only. In this framework, we wish an algorithm to be able to compute a function ϕ of n variables (in our context, an affine formula) by asking only a polynomial number of vectors of an affine relation R, such that ϕ approximates with high probability the relation R rather closely (Probably Approximately Correct learning).

More precisely, an affine n-place relation R is given, as well as an error parameter ε. The algorithm must compute an affine formula ϕ over the variables $x_1, \ldots, x_n$ such that ϕ approximates R with an error controlled by ε ; we will authorize here only one-sided errors, i.e., the models of ϕ must form a subset of R. At any time, the algorithm can ask a vector $m \in R$ to an oracle, but the number of these calls must be polynomial in n and ε, as well as the work performed with each vector[3]. Note that in a first time we assume that the algorithm knows n, while this is not the case in Valiant's framework, but we will see at the end of the section how to deal with this problem.

To be as general as possible, a probability distribution D over the vectors $m \in R$ is fixed, for two purposes : (i) when asked a vector of R, the oracle outputs $m \in R$ with probability $D(m)$, independently of the previously output vectors (ii) the error corresponding to the affine formula ϕ computed by the algorithm is defined as $E(\phi) = \sum_{m \in R, m \neq \phi} D(m)$, and ϕ is said to be a *correct approximation* of R if $E(\phi) \leq 1/\varepsilon$.

Finally, the class of affine formulas will be said *PAC-learnable from examples only* if there exists an algorithm that, for a fixed affine n-place relation R and a real number ε, can compute in time polynomial in n and ε, and with a polynomial number of calls to the oracle, an affine formula ϕ that with probability at least $1 - 1/\varepsilon$ is a correct approximation of R. We exhibit here such an algorithm[4].

The idea is first to treat R as R_m, where m is the first vector obtained from the oracle, i.e., to replace each obtained vector μ with $\mu \oplus m$; once again this is done for tranforming the problem into one of vector spaces. The idea is then to obtain a certain number of vectors of R_m from the oracle and to maintain a maximal linearly independent subset B_m of them. When enough vectors have been asked, the algorithm can compute an affine formula ϕ_m from this set (Proposition 3) and output $\phi = (\phi_m)_m$; since $B_m \subseteq R_m$ and R_m is closed under linear combination, the models of ϕ_m will always form a subset of R_m, as required.

The point is that only a polynomial number of vectors are needed for ϕ_m to be with high probability a correct approximation of R_m. To show this, we will use the function $L(\varepsilon, n)$ defined in [11] ; the value $L = L(\varepsilon, n)$ is the smallest integer such that in L independent Bernoulli trials $T_1, \ldots, T_L$ each with probability $P_i \geq 1/\varepsilon$ of success (the P_i's being not necessarily equal), the probability of having at least n successes is at least $1 - 1/\varepsilon$. Valiant shows that $L(\varepsilon, n)$ is almost linear in ε and n (more precisely, $\forall n \geq 1, \forall \varepsilon > 1, L(\varepsilon, n) \leq 2\varepsilon(n + \log_e \varepsilon)$). We show below that $L(\varepsilon, n)$ vectors of R_m are enough for ϕ_m to be correct.

Proposition 8 (PAC-learning) *The class of affine formulas is PAC-learnable from $L(\varepsilon, n)$ examples, where ε is the error parameter and n the number of variables involved.*

Proof. We have to show that if the algorithm presented above has obtained $L(\varepsilon, n)$ vectors (remind that each vector μ is replaced with $\mu \oplus m$, where m is the first vector obtained) and kept a maximal linearly independent subset B_m of them, then an affine formula ϕ_m describing the vector space generated by B_m is a correct approximation of R_m. We have seen that the set of models of ϕ_m is a subset of R_m, as required. Now we have to show that with probability at least $1 - 1/\varepsilon$, $E(\phi) \leq 1/\varepsilon$. We thus consider the event $E(\phi) > 1/\varepsilon$, and show that its probability is less than $1/\varepsilon$. For this purpose, we associate to each call to the oracle a trial T_i, which is considered a success if and only if the vector obtained is linearly independent from the current independent set of vectors B_m maintained by the algorithm. Since B_m can only increase during the process, the probability P_i of success of T_i is always at least $E(\phi)$. Now since there are $k \leq n$ linearly independent vectors in R_m and ϕ_m is not correct ($E(\phi) > 1/\varepsilon$), the algorithm has obtained less than k successes ; finally, since the calls to the oracle are independent Bernoulli trials and $L(\varepsilon, n) (\geq L(\varepsilon, k))$ such calls have been made, the definition of $L(\varepsilon, k)$ guarantees that this can happen with probability less than $1/\varepsilon$. Thus the learning algorithm is correct. To complete the proof, it suffices to remark that the work performed by the algorithm with each vector requires only polynomial time, since it corresponds to deciding the linear independence of a vector m from the current set B_m, and $|B_m| \leq n$; thus Proposition 1 concludes. □

To conclude the section, we consider the case when the algorithm does not know in advance the number of variables on which the relation R is built. Then the vectors output by the oracle are built on $t \geq n$ variables, but are not necessarily total ; in case a partial vector m is output, it means that all total vectors matching m match one vector in R. But it is easily shown that if R really depends on a variable x_i and is affine, then all partial vectors like above must assign a value to x_i, since a model assigning a to x_i cannot be a model any more if the value of x_i becomes $\overline{a}$; indeed, if $\{x, x_{i_1}, \ldots, x_{i_k}\}$ satisfies a linear equation depending on x_i, $\{\overline{x} = x \oplus 1, x_{i_1}, \ldots, x_{i_k}\}$ necessarily falsifies it. Thus the algorithm needs only take into account the variables that are defined in all the vectors output by the oracle, and the result stays the same (with $L(\varepsilon, t)$ calls to the oracle).

6 CONCLUSION

We have presented affine formulas as good candidates for approximating propositional knowledge. Indeed, we have seen that these formulas admit very good computational properties for reasoning tasks (in particular satisfiability, deduction, counting of models) and are guaranteed to be very short : their size can always be minimized efficiently to $O(n^2)$, where n is the number of variables involved.

Then we have shown that these formulas can easily be acquired from examples. Indeed, this class is *identifiable*, which means that given a relation R, an affine formula ϕ with R as its set of models can be computed, if it exists, in polynomial time. When such a formula does not exist, an affine *least upper bound* of R can be computed with roughly the same algorithm, and an affine *greatest lower bound* of R with the maximal number of models can be computed in subexponential time. Finally, we have shown that affine formulas are *PAC-learnable* from examples only.

We have argued that all these results made affine formulas an interesting class for approximating knowledge, by comparing them to the corresponding ones for Horn formulas, which are often considered for representing or approximating propositional knowledge. Indeed, Horn formulas are identifiable as well as affine formulas and with a comparable time complexity [4, 12] ; on the other hand, the Horn LUB of a relation can be exponentially bigger than it [5, Theorem 6] while the affine LUB of a relation can always be computed

[3] In the framework of [11], the running time can also be polynomial in the size of the shortest affine description of R, but it will be useless here.

[4] Following [11] and for sake of simplicity, we use only one parameter ε for bounding both the probability of success of the algorithm and the correctness of ϕ ; but two parameters ε_1 and ε_2 could be used with the same complexity results.

in polynomial time, and computing a Horn GLB of a relation with the maximum number of models is a NP-hard problem [7], while it is only subexponential for affine formulas. Then, affine formulas are PAC-learnable from examples only while the problem is still open for Horn formulas ; [2] only gives an algorithm for learning Horn formulas with access to an equivalence oracle. All these results show that *acquisition* of affine formulas from examples is in general easier than acquisition of Horn formulas. But we also emphasize that *working* with affine formulas is also easier in general than with Horn formulas. For instance, mininmizing or counting the models of an affine formula is polynomial, while it is intractable with Horn. In a forthcoming paper, we study more deeply the properties of affine formulas for reasoning, as well as their semantics, i.e., the natural pieces of knowledge that they can really represent.

ACKNOWLEDGEMENTS

I wish to thank Jean-Jacques Hébrard for his very important help in improving the redaction of this paper.

REFERENCES

[1] D. Angluin, 'Computational learning theory : survey and selected bibliography', in : *Proc. 24th Annual ACM Symposium on Theory Of Computing (STOC'92)* (Victoria, Canada), New York : ACM Press, 319–342, (1992).

[2] D. Angluin, M. Frazier and L. Pitt, 'Learning conjunctions of Horn clauses (extended abstract)', in : *Proc. 31st Annual Symposium on Foundations of Computer Science* (St Louis, USA), Los Alamitos : IEEE Computer Society, 186–192, (1990).

[3] C.W. Curtis, *Linear algebra. An introductory approach*, Springer-Verlag, 1984.

[4] R. Dechter and J. Pearl, 'Structure identification in relational data', *Artificial Intelligence*, **58**, 237–270 (1992).

[5] H. Kautz, M. Kearns and B. Selman, 'Horn approximations of empirical data', *Artificial Intelligence*, **74**, 129–145, (1995).

[6] D. Kavvadias and M. Sideri, 'The inverse satisfiability problem', *SIAM J. Comput.*, **28 (1)**, 152–163, (1998).

[7] 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)*, Springer Lecture Notes in Computer Science, **762**, 399–405, (1993).

[8] D. Kavvadias, M. Sideri and E.C. Stavropoulos, 'Generating all maximal models of a Boolean expression', *Inform. Process. Lett.*, **74**, 157–162, (2000).

[9] T.J. Schaefer, 'The complexity of satisfiability problems', in : *Proc. 10th Annual ACM Symposium on Theory Of Computing (STOC'78)* (San Diego, USA), ACM Press, New York 216–226 (1978).

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

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

[12] B. Zanuttini and J.-J. Hébrard, 'A unified framework for structure identification', *Inform. Process. Lett.*, **81**, 335–339, (2002).

APPENDIX

We give here the proofs of the propositions given in Section 2.

Proposition 1 *Let $B \subseteq \{0,1\}^n$ and $m \in \{0,1\}^n$. Deciding whether B is a set of linearly independent vectors, or whether m is linearly independent from B can be performed in time $O(|B|^2 n)$.*

Proof. For the first point, transform B into a set of non-zero vectors B' in echelon form with gaussian elimination, in time $O(|B|^2 n)$, and check whether $|B'| = |B|$ [3, Theorem 6.16]. For the second point, still transform B into B', then transform $B' \cup \{m\}$ into a set B'' in echelon form, and check whether $|B''| = |B'|$. □

Proposition 2 *Given a relation R over n variables, finding a linearly independent subset of R that is maximal for set inclusion requires time $O(|R|n^3)$.*

Proof. The subset B of R is built step by step. First initialize it with any vector $m_0 \in R$ not identically 0. During the process, pick any vector $m \in R$ not yet in B, and check whether it is linearly independent from B (Proposition 1). If yes, add it to B, otherwise eliminate it from R. Since there cannot be more than n linearly independent vectors in $\{0,1\}^n$ [3, Theorem 5.1], the number of vectors in B can never exceed n, and each vector of R is considered only once, yielding the time complexity $O(|R||B|^2 n) \subseteq O(|R|n^3)$. □

Proposition 3 *Given a basis B of a vector space $V \subseteq \{0,1\}^n$, computing an affine formula ϕ describing V requires time $O(n^4)$, and ϕ contains at most n equations.*

Proof. First complete the basis $B = \{m_1, \ldots, m_k\}$ with $n-k$ vectors $m_1, \ldots, m_n \in \{0,1\}^n$ such that $\{m_1, \ldots, m_n\}$ is a basis for the vector space $\{0,1\}^n$; this can be done in time $O(|B|^2 n + n^2) \subseteq O(n^3)$ by putting B in echelon form. Then associate the linear equation $E_j = (\bigoplus_{i=1,\ldots,n} c_{ij} x_i = 0)$ to m_j for $j = k+1, \ldots, n$, where the c_{ij}'s are uniquely determined for a given $j \in \{k+1, \ldots, n\}$ by the system

$$S_j = \left\{ \begin{array}{ccccccccc} & & & & & & & & \\ m_1[1] & c_{1j} & \oplus & \cdots & \oplus & m_1[n] & c_{nj} & = & 0 \\ \cdots & & & & & & & & \\ m_{j-1}[1] & c_{1j} & \oplus & \cdots & \oplus & m_{j-1}[n] & c_{nj} & = & 0 \\ m_j[1] & c_{1j} & \oplus & \cdots & \oplus & m_j[n] & c_{nj} & = & 1 \\ m_{j+1}[1] & c_{1j} & \oplus & \cdots & \oplus & m_{j+1}[n] & c_{nj} & = & 0 \\ \cdots & & & & & & & & \\ m_n[1] & c_{1j} & \oplus & \cdots & \oplus & m_n[n] & c_{nj} & = & 0 \end{array} \right.$$

Then the affine formula $\phi = \bigwedge_{j=k+1}^{n} E_j$ describes V. Indeed, by construction of S_j, for $i = 1, \ldots, k$, m_i satisfies E_j, thus every linear combination of $\{m_1, \ldots, m_k\}$ satisfies every E_j, thus V is included in the set of models of ϕ. On the other hand, if $m \in \{0,1\}^n$ is not in V, then it is the linear combination of some vectors of $\{m_1, \ldots, m_n\}$, among which at least one m_j with $j > k$; write $m = m_j \oplus \bigoplus_{h=1}^{\ell} m_{i_h}$; then

$$\begin{aligned} \bigoplus_{i=1}^{n} c_{ij} m[i] &= \bigoplus_{i=1}^{n} c_{ij} (m_j[i] \oplus \bigoplus_{h=1}^{\ell} m_{i_h}[i]) \\ &= \bigoplus_{i=1}^{n} c_{ij} m_j[i] \oplus \bigoplus_{h=1}^{\ell} (\bigoplus_{i=1}^{n} c_{ij} m_{i_h}[i]) \end{aligned}$$

Since $\bigoplus_{i=1}^{n} c_{ij} m_{i_h}[i] = 0$ for all h and $\bigoplus_{i=1}^{n} c_{ij} m_j[i] = 1$ (by construction of S_j), we get $\bigoplus_{i=1}^{n} c_{ij} m[i] = 1$, i.e., m does not satisfy E_j, and thus does not satisfy ϕ. There are $n-k$ systems S_j to solve, each one in time $O(n^3)$ with gaussian elimination (n equations and n unknowns), thus the total time complexity of the process is $O((n-k)n^3) \subseteq (n^4)$. □

Proposition 4 *Given an n-place relation R and a linearly independent set of vectors $B \subseteq R$, deciding whether the vector space V generated by B is included in R requires time $O(|R|n)$.*

Proof. It suffices to generate all the linear combinations of vectors of B, and to answer 'no' as soon as one is not in R, or 'yes' if all are in R. Since two different linear combinations of linearly independent vectors are different, each vector of R can be found at most once, and deciding $m \in R$ requires time $O(n)$ if R is sorted (in time $O(|R|n)$ with a radix sort), which completes the proof. □