Complexity of Satisfiability Problems with Symmetric Polynomial Clauses

NADIA CREIGNOU and MALIKA MORE, *Dépt. de Maths, Université de Caen, 14032 Caen, France.*
E-mail: {Malika.More,Nadia.Creignou}@math.unicaen.fr

Abstract

The problem SAT of CNF-Satisfiability is the prototype of NP-complete problems. In this problem the question is whether there exists a truth assignment such that each clause contains at least one true literal. Conversely, the problem 1-At-Most-SAT, in which the question is whether there exists a truth assignment so that each clause contains at most one true literal, is polynomial-time decidable. We define an infinite class of problems SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$), parametrized by symmetric polynomial-time decidable languages and generalizing usual satisfiability problems. For instance, the two problems previously considered are, respectively, parametrized by the languages corresponding to the regular expressions $0^*1(0+1)^*$ and $0^* + 0^*10^*$. We prove a dichotomy theorem for these satisfiability problems with symmetric polynomial clauses: SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is either polynomial or NP-complete. We also show a dichotomy result for the corresponding counting problems: #SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is either in FP or #P-complete. Besides, we provide a normal form characterization of the symmetric **regular** languages $\mathfrak{L}$ leading to a polynomial-time decidable problem SAT($\mathfrak{L}$). This characterization induces an algorithm to decide whether a given problem SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is polynomial-time decidable, provided that the languages $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ are symmetric and regular

Keywords: Computational complexity, NP-completeness, satisfiability, counting problems, #P-completeness, polynomial time, symmetric languages, regular languages.

1 Introduction

Since Cook's NP-completeness proof [1], the standard CNF-Satisfiability problem, SAT, has become a kind of canonical NP-complete problem. In this problem the question is whether there exists a truth assignment such that each clause contains at least one true literal. On the contrary, the problem 1-At-Most-SAT, in which the question is whether there exists a truth assignment so that each clause contains at most one true literal is polynomial-time decidable (even decidable in linear time since this problem is essentially equivalent to the well-known Horn renaming problem [8]). The aim of this paper is to study the complexity of a class of satisfiability problems SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$), parametrized by symmetric polynomial-time decidable languages, which contains these two particular problems as special cases. Let $\mathfrak{L} \subseteq \{0,1\}^*$, then SAT($\mathfrak{L}$) is the following problem: given a finite set $\{w_1, \ldots, w_p\}$ of words w_i over $\{x_1, \overline{x}_1, \ldots, x_n, \overline{x}_n\}$, is there a truth assignment s on the variables x_j for which $s(w_i) \in \mathfrak{L}$ for $1 \leq i \leq p$? Thus, the two previous problems are, respectively, parametrized by the regular expressions $0^*1(0+1)^*$ and $0^* + 0^*10^*$.

We show that every problem in this class is either polynomial-time decidable or NP-complete. Furthermore, we prove that our dichotomy proof can be adapted to the corresponding counting problems. Indeed, we show that the counting problem #SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is either in FP or $\#P$-complete. (Note that such results are rare in complexity theory and usu-

ally difficult to prove, see [14], [5] and [9]. Nevertheless, we remind the reader that all these 'dichotomies' break down if $P = NP$ or $\#P = FP$.) In addition, we present a normal form characterization of the symmetric **regular** (i.e. recognizable by a finite automaton) languages [13, p. 53] $\mathfrak{L}$ leading to a polynomial-time decidable problem SAT($\mathfrak{L}$) (resp. to a counting problem $\#$SAT($\mathfrak{L}$) in FP). This characterization yields an algorithm to decide whether a given problem SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) (resp. $\#$SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$)) is polynomial-time decidable (resp. is in FP), provided that the languages $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ are symmetric and regular.

This generalization of the standard CNF-Satisfiability problem is not the first one. In 1978, Schaefer [14] proposed to consider a class of satisfiability problems defined through a finite set of Boolean logical relations (an accurate and succinct statement of his results can be found in [6, p. 260, LO6]). He also got a dichotomy result in this way (later extended to the corresponding counting problems by Creignou and Hermann [3]) and our proof is adapted from his. However, the generalization he proposed imposed to restrain the problem to bounded-length clauses. This restriction may be artificial. The class we consider, defined through symmetric polynomial languages, naturally allows non-bounded length clauses (the assumption that the languages are symmetric, i.e. invariant by permutations, naturally appears when satisfiability problems are concerned since there is no ordering in clauses). It is worth pointing out that in Schaefer's version of the problem, the predicates take simple (unnegated) variables as arguments, but in the version considered here the arguments are literals. Our extensions of satisfiability, i.e. problems SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$), are worth being studied for they are widely expressive. Indeed, many NP-problems have a natural logical expression through short (i.e. linearly bounded) conjunctions of polynomial clauses. For example the Cubic Subgraph problem [6] can be expressed by SAT(3-Exactly) [7]. The particular satisfiability problem defined by the two languages $0^*1(0+1)^*$ (1-at-least) and $0^* + 0^*10^*$ (1-at-most), denoted by SAT' [7], has also been proved to be very powerful to express in a natural way a large number of classical NP-complete problems involving paths, partitions, matchings...(see [7] and [2]). Moreover, these problems are not too expressive since they are in NP and reducible in linear time on a deterministic Turing machine to the classical problem SAT (or 3SAT) as proved in [7].

Therefore, we prove two new dichotomy results concerning very expressive problems. Thus, we get an infinite collection of new NP-complete and #P-complete problems. For instance 2-At-Most-SAT is NP-complete whereas Even-SAT (even number of true literals) is in P. In particular, as all dichotomy results, this increases our knowledge about what makes problems difficult, and thus sheds some light on the very essence of complete problems.

2 Definitions and main result

2.1 Generalized satisfiability and polynomial languages

All through the paper we are interested in **symmetric** languages, defined over the binary alphabet $\{0, 1\}$. Such a language $\mathfrak{L}$ satisfies the following property :

$$w_1 \cdots w_n \in \mathfrak{L} \text{ iff } w_{\sigma(1)} \cdots w_{\sigma(n)} \in \mathfrak{L} \text{ for each permutation } \sigma \text{ of } \{1, \ldots, n\}.$$

Let $\mathfrak{L}$ be a language, and let $\mathcal{V}$ denote a set of variables. Let us recall that a **literal** is either a variable or a negated variable. If l is a literal, let $\bar{l}$ denote its negation. Note that $\bar{\bar{l}} = l$. An **$\mathfrak{L}$-clause** is an expression of the form $P_{\mathfrak{L}}(W)$, where $P_{\mathfrak{L}}$ is a symbol for the language $\mathfrak{L}$ and W is a word made up of literals, i.e. $W = l_1 \cdots l_n$ where l_i or $\bar{l}_i$ belongs to $\mathcal{V}$, $i = 1 \cdots n$.

For any truth assignment $s : \mathcal{V} \mapsto \{0,1\}$ (as usual, 1 represents truth and 0 falsity) we set :

$$s(W) = s(l_1) \cdots s(l_n) \in \{0,1\}^*.$$

We say that such an $\mathfrak{L}$-clause $P_{\mathfrak{L}}(W)$ is satisfied by s iff $s(W) \in \mathfrak{L}$. Note that the truth of a word in a symmetric language depends only on the number of 0s and 1s in the word.

Given symmetric polynomial languages $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ and a set of variables $\mathcal{V}$, the problem that we study is then defined as follows.

SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$)
Instance: Set of $\mathfrak{L}_j$-clauses $\{P_{\mathfrak{L}_{i_1}}(W_1) \cdots P_{\mathfrak{L}_{i_m}}(W_m)\}$ (for $r = 1 \ldots m, i_r \in \{1, \ldots, p\}$) over a set of variables $\mathcal{V}$.
Question: Is there a truth assignment $s : \mathcal{V} \mapsto \{0,1\}$ such that for each r, $P_{\mathfrak{L}_{i_r}}(W_r)$ is satisfied by s?

A variant of this problem is SAT$_c$($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$), in which we allow the constants 0 and 1 to appear in the words given as instance. The word $s(W)$ is then obtained by replacing the literals by their truth value and letting the constants unchanged.

EXAMPLE 2.1
The classical satisfiability problem SAT can be expressed as SAT($\mathfrak{L}$) where $\mathfrak{L}$ is the language defined by the regular expression $0^*1(0+1)^*$. In the instance each clause is considered as an $\mathfrak{L}$-clause $P_{\mathfrak{L}}(W)$, where W is a word made up of literals.

REMARK 2.2
Observe that SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is in NP since the languages involved are polynomial.

2.2 *Logical preliminaries*

If W is a finite word made up of literals, let $\mathcal{V}ar(W)$ denote the set of variables occurring in W. Let s, s_1, s_2 be truth assignments.

- $s = s_1 \oplus s_2$ is defined by $s(x) = 1$ iff $s_1(x) \neq s_2(x)$,
- $s = s_1 \cap s_2$ (resp. $s_1 \cup s_2$) is defined by $s(x) = 1$ (resp. 0) iff $s_1(x) = s_2(x) = 1$ (resp. 0).

If W is a word made up of literals, v a variable or a constant (0 or 1), and l a literal or a constant (0 or 1), then $W[l/v]$ denotes the word formed from W by replacing each occurrence of v by l (and each occurrence of $\bar{v}$ by $\bar{l}$). If V is a set of variables, then $W[l/V]$ denotes the result of substituting l for every occurrence of every variable in V.

Let $\mathfrak{L}$ be a language. Let us denote by $\mathfrak{L}(k)$ the set of words of length k in $\mathfrak{L}$:

$$\mathfrak{L}(k) = \{W : W \in \mathfrak{L} \text{ and } |W| = k\}.$$

$\mathfrak{L}(k)$ is a subset of $\{0,1\}^k$ and can be seen as a logical relation of arity k (in the terminology used by Schaefer [14]). Such a relation is definable by a CNF-formula (the words W correspond to the truth assignments satisfying the formula).

EXAMPLE 2.3
Let $\mathfrak{L}$ be the language represented by the expression $0^*1(0+1)^*$. The relation $\mathfrak{L}(2) = \{01, 10, 11\}$ is definable by the formula $(x \vee y)$.

EXAMPLE 2.4
Let $\mathfrak{L}'$ be the language represented by the expression 0^*10^*. The relation $\mathfrak{L}'(3) = \{001, 010, 100\}$ is definable by the formula $((x \vee y \vee z) \wedge (\bar{x} \vee \bar{y}) \wedge (\bar{y} \vee \bar{z}) \wedge (\bar{z} \vee \bar{x}))$.

Now, we define the notions of affine and bijunctive languages which are due to Schaefer [14].

DEFINITION 2.5
A language $\mathfrak{L}$ is affine if for each k, $\mathfrak{L}(k)$ is the set of solutions of some system of linear equations over the field IF_2.

EXAMPLE 2.6
Let

$$\mathfrak{L}^5 = \left\{ W \in \{0,1\}^* \ : \ |W|_1 \equiv 0 \ mod \ 2 \right\}$$

(where $|W|_1$ denotes the number of 1s in the word W) corresponding to the expression $0^*(10^*10^*)^*$ (words containing an even number of 1). For each k,

$$\mathfrak{L}^5(k) = \left\{ W = w_1 \cdots w_k \in \{0,1\}^k \ : \ w_1 \oplus \cdots \oplus w_k \equiv 0 \right\}$$

(where $\oplus$ denotes addition in the field IF_2). Thus, $\mathfrak{L}^5$ is affine.

DEFINITION 2.7
A language $\mathfrak{L}$ is bijunctive if for each k, $\mathfrak{L}(k)$ is definable by a 2-CNF-formula.

EXAMPLE 2.8
Let $\mathfrak{L}^7$ be defined by the expression $0^* + 0^*10^*$ (words containing at most one 1). It is easy to see that for each k,

$$\mathfrak{L}^7(k) = \left\{ W = w_1 \cdots w_k \in \{0,1\}^k \ : \ \bigwedge_{\substack{i,j=1 \\ i \neq j}}^{k} (\bar{w}_i \vee \bar{w}_j) \right\}.$$

Thus, $\mathfrak{L}^7$ is bijunctive.

2.3 *Main result*

THEOREM 2.9
Let $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ be symmetric polynomial languages. If every language $\mathfrak{L}_i$, $1 \leq i \leq p$, is affine or if every language $\mathfrak{L}_i$, $1 \leq i \leq p$, is bijunctive then SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is in P, otherwise SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is NP-complete.

REMARK 2.10
Although stated for unbounded clauses, this result is of interest even for fixed-length clauses.

3 The polynomial case

This section is devoted to the proof of the polynomial case of Theorem 2.9. It relies on a characterization of affine or bijunctive symmetric languages using generic examples.

EXAMPLE 3.1
Let $\mathfrak{L}^0 = \emptyset$ and let $\mathfrak{L}^1, \mathfrak{L}^2, \mathfrak{L}^3, \mathfrak{L}^4, \mathfrak{L}^5, \mathfrak{L}^6, \mathfrak{L}^7, \mathfrak{L}^8$ be the languages, respectively, denoted by the regular expressions $0^*, 1^*, 0^*+1^*, (0+1)^*, 0^*(10^*10^*)^*, 0^*10^*(10^*10^*)^*, 0^*+0^*10^*, 1^* + 1^*01^*$. The languages $\mathfrak{L}^0$ to $\mathfrak{L}^8$ are regular and *a fortiori* polynomial. The languages $\mathfrak{L}^0$ to $\mathfrak{L}^4$ are symmetric, affine and bijunctive, whereas $\mathfrak{L}^5$ (words containing an even number of 1) and $\mathfrak{L}^6$ (odd number of 1) are symmetric and affine but not bijunctive and $\mathfrak{L}^7$ (at most one 1) and $\mathfrak{L}^8$ (at most one 0) are symmetric and bijunctive but not affine (see Lemma 4.3). The problems SAT($\mathfrak{L}^0$) to SAT($\mathfrak{L}^4$) are obviously polynomial, and SAT($\mathfrak{L}^5$), SAT($\mathfrak{L}^6$) (linear systems over IF_2), SAT($\mathfrak{L}^7$) and SAT($\mathfrak{L}^8$) (see [8]) are also known to be polynomial-time decidable. Note that for each fixed k, the linear system over IF_2 (resp. the 2-CNF-formula) defining $\mathfrak{L}^5(k)$ or $\mathfrak{L}^6(k)$ (resp. $\mathfrak{L}^7(k)$ or $\mathfrak{L}^8(k)$) can be written in time polynomial in k.

We show below that if $\mathfrak{L}$ is an affine or bijunctive symmetric language, then, for any $k \geq 3$, there exists one and only one i, $0 \leq i \leq 8$, such that $\mathfrak{L}(k) = \mathfrak{L}^i(k)$. The cases of $k = 1$ and $k = 2$ are different for then the sets $\mathfrak{L}^i(k)$ are not pairwise distinct. It is easy but tedious to deal with these particular cases. Therefore, for sake of readibility, we assume that $\mathfrak{L}$ does not contain any word of length less than or equal to 2.

DEFINITION 3.2
A set C separates the sets $A_1, \ldots A_s$ iff the sets $A_1 \cap C, \ldots, A_s \cap C$ are pairwise different.

Let us show that the generic examples of affine or bijunctive symmetric languages are easy to discriminate, using the notion of separating set.

LEMMA 3.3
For any $k \geq 3$, the set $S(k) = \{0^k, 1^k, 10^{k-1}, 110^{k-2}\}$
(resp. $S'(k) = \{0^k, 1^k, 10^{k-1}, 01^{k-1}\}$) separates the languages $\mathfrak{L}^0(k), \ldots, \mathfrak{L}^6(k)$
(resp. $\mathfrak{L}^0(k), \ldots, \mathfrak{L}^4(k), \mathfrak{L}^7(k), \mathfrak{L}^8(k)$).

PROOF. Let us check the intersections : $\mathfrak{L}^0(k) \cap S(k) = \mathfrak{L}^0(k) \cap S'(k) = \emptyset$;
$\mathfrak{L}^1(k) \cap S(k) = \mathfrak{L}^1(k) \cap S'(k) = \{0^k\}$ and $\mathfrak{L}^2(k) \cap S(k) = \mathfrak{L}^2(k) \cap S'(k) = \{1^k\}$;
$\mathfrak{L}^3(k) \cap S(k) = \mathfrak{L}^3(k) \cap S'(k) = \{0^k, 1^k\}$;
$\mathfrak{L}^4(k) \cap S(k) = S(k)$ and $\mathfrak{L}^4(k) \cap S'(k) = S'(k)$;
$\mathfrak{L}^5(k) \cap S(k) = \{0^k, 1^k, 110^{k-2}\}$ if k even and $\{0^k, 110^{k-2}\}$ if not;
$\mathfrak{L}^6(k) \cap S(k) = \{10^{k-1}\}$ if k even and $\{1^k, 10^{k-1}\}$ if not;
$\mathfrak{L}^7(k) \cap S'(k) = \{0^k, 10^{k-1}\}$ and $\mathfrak{L}^8(k) \cap S'(k) = \{1^k, 01^{k-1}\}$. ∎

Now, let us state the key-result of this section. In the case $\mathfrak{L}$ is affine, it means that, for each k, $\mathfrak{L}(k)$ coincides with the words of length k of some generic affine language $\mathfrak{L}^0$ to $\mathfrak{L}^6$.

LEMMA 3.4
A symmetric language $\mathfrak{L}$ is affine (resp. bijunctive) iff for all $k \geq 3$, $\mathfrak{L}(k) \in \{\mathfrak{L}^0(k), \ldots, \mathfrak{L}^6(k)\}$ (resp. $\{\mathfrak{L}^0(k), \ldots, \mathfrak{L}^4(k), \mathfrak{L}^7(k), \mathfrak{L}^8(k)\}$).

PROOF. **Affine Case.** For all $k \geq 3$, let us examine the different possible forms of the linear system $\Sigma_k(\mathfrak{L})$ defining the logical relation $\mathfrak{L}(k)$. If $\mathfrak{L}(k)$ is non-trivial, we can assume that all equations in $\Sigma_k(\mathfrak{L})$ are of the form $x_{i_1} \oplus \ldots \oplus x_{i_p} = \epsilon$ with $1 \leq p \leq k$ and $\epsilon \in \{0, 1\}$. If

$\Sigma_k(\mathfrak{L})$ can be reduced to $x_1 \oplus \ldots \oplus x_k = 0$ (resp. $x_1 \oplus \ldots \oplus x_k = 1$), then $\mathfrak{L}(k) = \mathfrak{L}^5(k)$ (resp. $\mathfrak{L}(k) = \mathfrak{L}^6(k)$). If $\Sigma_k(\mathfrak{L})$ contains an equation $x_{i_1} \oplus \ldots \oplus x_{i_p} = \epsilon$ with $1 \leq p \leq k-1$, since $\mathfrak{L}$ is symmetric, the sum of any p variables is always ϵ. Now, if $p \geq 2$ for any $i \neq j \in \{1, \ldots, k\}$, consider $\{x_{i_1}, \ldots, x_{i_{p-1}}\}$ such that $\{x_{i_1}, \ldots, x_{i_{p-1}}\} \cap \{x_i, x_j\} = \emptyset$. Then $x_i \oplus x_{i_1} \oplus \ldots \oplus x_{i_{p-1}} = \epsilon$ and $x_j \oplus x_{i_1} \oplus \ldots \oplus x_{i_{p-1}} = \epsilon$, hence $x_i \oplus x_j = 0$, i.e. $x_i = x_j$. Consequently, $\mathfrak{L}(k) \subset \{0^k, 1^k\}$.

Bijunctive Case. For all $k \geq 3$, let us examine the different possible forms of the 2-CNF formula $F_k(\mathfrak{L})$ defining the logical relation $\mathfrak{L}(k)$. Since $\mathfrak{L}$ is symmetric, if x_i appears in some clause, then all the clauses obtained by replacing x_i by any other variable x_j can be assumed to belong to $F_k(\mathfrak{L})$. Consequently, the form of $F_k(\mathfrak{L})$ can be assumed to be (if non-trivial):

$$\bigwedge_{i \in P}(x_i) \wedge \bigwedge_{i \in N}(\bar{x}_i) \wedge \bigwedge_{(i,j) \in PN}(x_i \vee \bar{x}_j) \wedge \bigwedge_{(i,j) \in PP}(x_i \vee x_j) \wedge \bigwedge_{(i,j) \in NN}(\bar{x}_i \vee \bar{x}_j)$$

where the sets P and N are either $\emptyset$ or $\{1, \ldots, k\}$ and the sets PN, PP and NN are either $\emptyset$ or $\{1, \ldots, k\}^2 \setminus \{(1,1), \ldots, (k,k)\}$. The different $\mathfrak{L}^i(k)$ are obtained depending on the combinations of these cases. The non-trivial cases $\mathfrak{L}^8(k)$ and $\mathfrak{L}^7(k)$ are, respectively, obtained when $P = N = PN = NN = \emptyset$ and $P = N = PN = PP = \emptyset$. (i.e. correspond to $\bigwedge_{i \neq j}(x_i \vee x_j)$ and $\bigwedge_{i \neq j}(\bar{x}_i \vee \bar{x}_j)$). ∎

This characterization of affine or bijunctive symmetric languages induces the polynomial case of Theorem 2.9 as an easy corollary.

COROLLARY 3.5
Let $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ be polynomial symmetric and affine (resp. bijunctive) languages. Then SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is polynomial-time decidable.

PROOF. **Affine case.** Let $\{P_{\mathfrak{L}_{i_1}}(W_1) \cdots P_{\mathfrak{L}_{i_m}}(W_m)\}$ be an instance of the problem SAT $(\mathfrak{L}_1, \ldots, \mathfrak{L}_p)$. Let us describe a polynomial-time algorithm determining a linear system Σ $(\{P_{\mathfrak{L}_{i_1}}(W_1) \cdots P_{\mathfrak{L}_{i_m}}(W_m)\})$ such that there exists a truth assignment $s : \mathcal{V} \mapsto \{0,1\}$ satisfying each $P_{\mathfrak{L}_{i_r}}(W_r)$ (i.e. $s(W_r) \in \mathfrak{L}_{i_r}$ for $r = 1 \ldots m$) iff the set of solutions of the system $\Sigma(\{P_{\mathfrak{L}_{i_1}}(W_1) \cdots P_{\mathfrak{L}_{i_m}}(W_m)\})$ is not empty.

For $r = 1 \ldots m$

1. Determine $\alpha_r = |W_r|$ and denote $W_r = l_{r,1} \ldots l_{r,\alpha_r}$ (Recall that $\alpha_r \geq 3$).
2. Determine $\mathfrak{L}_{i_r} \cap S(\alpha_r)$ (Recall that $\mathfrak{L}_{i_r}$ is polynomial). Deduce $j \in \{0, \ldots, 6\}$ such that $\mathfrak{L}_{i_r}(\alpha_r) = \mathfrak{L}^j(\alpha_r)$.
3. Determine the system $\Sigma(\alpha_r)$ with variables $x_1 \ldots x_{\alpha_r}$ defining $\mathfrak{L}^j(\alpha_r)$.
4. For $j = 1 \ldots \alpha_r$, if $l_{r,j}$ is a variable (say x), then replace each occurrence of x_j in $\Sigma(\alpha_r)$ by x and if $l_{r,j}$ is a negated variable (say $\bar{x}$), then replace each occurrence of x_j in $\Sigma(\alpha_r)$ by $1 \oplus x$. Note $\Sigma(P_{\mathfrak{L}_{i_r}}(W_r))$ the system obtained.

Let $\Sigma(\{P_{\mathfrak{L}_{i_1}}(W_1) \cdots P_{\mathfrak{L}_{i_m}}(W_m)\})$ be $\bigcup_{r=1}^{m} \Sigma(P_{\mathfrak{L}_{i_r}}(W_r))$. The resolution of this linear system is obviously polynomial (e.g. using Gaussian elimination).

Bijunctive case. Use the polynomial Davis–Putnam algorithm [4] to decide whether the similarly obtained 2-CNF formula is satisfiable. ∎

4 The NP-complete case

This section is devoted to the proof of Theorem 2.9 for the NP-complete case. The proof is modeled on the proof of Schaefer [14] with necessary changes being made to fit the changed assumptions of this paper.

We show that if $\mathfrak{L}_1$ and $\mathfrak{L}_2$ are polynomial languages over $\{0,1\}$ such that $\mathfrak{L}_1$ is not affine and $\mathfrak{L}_2$ is not bijunctive then SAT($\mathfrak{L}_1$, $\mathfrak{L}_2$) is NP-complete. (Then, the more general result for SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_m$) can easily be deduced from the proofs.) The proof consists in the construction of a polynomial-time reduction from the problem 1-Exactly-3SAT, which is well-known to be NP-complete [14], to SAT($\mathfrak{L}_1$, $\mathfrak{L}_2$) using SAT$_c$($\mathfrak{L}_1$, $\mathfrak{L}_2$) as an intermediate problem.

PROPOSITION 4.1
Let $\mathfrak{L}_1$ and $\mathfrak{L}_2$ be languages and assume that $\mathfrak{L}_1$ is not affine and $\mathfrak{L}_2$ is not bijunctive. Then SAT$_c$($\mathfrak{L}_1$, $\mathfrak{L}_2$) is NP-hard.

We provide a reduction from 1-Exactly-3SAT that obeys the following principle. We encode the clause 1-Exactly-(x,y,z) (which is true iff exactly one of the variables x, y and z is true) by a word W^1 (or several words) and a word W^2 (or several words) made up of the literals x, $\bar{x}$, y, $\bar{y}$, z, $\bar{z}$, possibly some other literals and the constants 0 and 1. Roughly speaking, these words W^1 and W^2 are such that a truth assignment s satisfies the initial clause 1-Exactly-(x,y,z) iff $s(W^1)$ belongs to $\mathfrak{L}_1$ and $s(W^2)$ belongs to $\mathfrak{L}_2$. For this purpose let us introduce a new definition which is adapted from the notion of 'representable relations' introduced by Schaefer [14]. Let us emphasize that this definition reflects the basic idea of the reduction.

DEFINITION 4.2
Let $\mathcal{R}ep_c(\mathfrak{L}_1, \mathfrak{L}_2)$ be the set of all formulas representable by $\mathfrak{L}_1$ and $\mathfrak{L}_2$. A Boolean formula F belongs to $\mathcal{R}ep_c(\mathfrak{L}_1, \mathfrak{L}_2)$ iff there exists a set of finite words $\{W_1^1, \ldots, W_p^1, W_1^2, \ldots, W_q^2\}$ made up of literals and constants such that $\mathcal{V}ar(F) \subset \bigcup_{i=1}^{p} \mathcal{V}ar(W_i^1) \cup \bigcup_{i=1}^{q} \mathcal{V}ar(W_i^2)$ and such that for all truth assignment s: $\mathcal{V}ar(F) \mapsto \{0,1\}$, s satisfies F iff s can be extended to $\bigcup_{i=1}^{p} \mathcal{V}ar(W_i^1) \cup \bigcup_{i=1}^{q} \mathcal{V}ar(W_i^2)$ so that

$$\{s(W_1^1), \ldots, s(W_p^1)\} \subset \mathfrak{L}_1 \text{ and } \{s(W_1^2), \ldots, s(W_q^2)\} \subset \mathfrak{L}_2.$$

We will then say that the set of words $\{W_1^1, \ldots, W_p^1, W_1^2, \ldots, W_q^2\}$ represents the formula F.

Note that, $\mathcal{R}ep_c(\mathfrak{L})$ is naturally defined in setting $\mathfrak{L}_1 = \mathfrak{L}_2 = \mathfrak{L}$, and that $F \in \mathcal{R}ep_c(\mathfrak{L}_1)$ implies $F \in \mathcal{R}ep_c(\mathfrak{L}_1, \mathfrak{L}_2)$. In order to prove that 1-Exactly-3SAT can be reduced to SAT$_c$ ($\mathfrak{L}_1$, $\mathfrak{L}_2$) it is clear that it is sufficient to show that the clause 1-Exactly(x,y,z) belongs to $\mathcal{R}ep_c(\mathfrak{L}_1, \mathfrak{L}_2)$. Studying representable formulas requires the following characterization of affine and bijunctive languages.

LEMMA 4.3
For all n, let $X_n = x_1 \cdots x_n$ be a word made up of n distinct variables. A language $\mathfrak{L}$ is affine (resp. bijunctive) iff for all n and for all truth assignments s_0, s_1, s_2 such that $s_0(X_n)$, $s_1(X_n)$ and $s_2(X_n)$ are in $\mathfrak{L}$, the word $s_0 \oplus s_1 \oplus s_2(X_n)$ also belongs to $\mathfrak{L}$ (resp. the word $[(s_0 \cup s_1) \cap (s_1 \cup s_2) \cap (s_2 \cup s_0)](X_n)$).

PROOF. $\mathfrak{L}$ is affine (resp. bijunctive) iff for all n, the logical relation corresponding to $\mathfrak{L}(n)$ is affine (resp. bijunctive). Now, the characterization given in the lemma corresponds exactly to the characterization of affine (resp. bijunctive) relations due to Schaefer [14, Lemmas 3.1.a and 3.1.b and Note p. 222]. ∎

REMARK 4.4
It is worth noticing that $(s_0 \cup s_1) \cap (s_1 \cup s_2) \cap (s_2 \cup s_0)$ is just the 2-of-3 majority function $MAJ(s_0, s_1, s_2)$.

LEMMA 4.5
Let $\mathfrak{L}$ be a non-affine language. Then, $\mathcal{R}ep_c(\mathfrak{L})$ contains $(\bar{x} \vee \bar{y})$.

PROOF. It suffices to show that $\mathcal{R}ep_c(\mathfrak{L})$ contains one of the clause $(x \vee y)$, $(\bar{x} \vee y)$ or $(\bar{x} \vee \bar{y})$ since the clause $(\bar{x} \vee \bar{y})$ can be obtained from the two first ones by negating the appropriate variables. Using Lemma 4.3, let X be a finite word made up of variables and let s_0, s_1 and s_2 be truth assignments such that $s_0(X), s_1(X)$ and $s_2(X)$ belong to $\mathfrak{L}$ but $s_0 \oplus s_1 \oplus s_2(X)$ does not. Form X' from X by negating all occurrences of the variables in the set $\{v \in \mathcal{V}ar(X) : s_0(v) = 1\}$. Define $s'_i = s_i \oplus s_0$, for $i = 0, 1, 2$ (s'_0 is the all-zero assignment, $\vec{0}$). Observe that an assignment s verifies $s(X) \in \mathfrak{L}$ iff $s \oplus s_0(X') \in \mathfrak{L}$. Thus, $s'_0(X')$, $s'_1(X')$ and $s'_2(X')$ belong to $\mathfrak{L}$, but $s'_0 \oplus s'_1 \oplus s'_2 = s'_1 \oplus s'_2(X')$ does not. For $i, j = 0, 1$, let $V_{i,j} = \{v \in \mathcal{V}ar(X') : s'_1(v) = i \wedge s'_2(v) = j\}$ and let

$$W := X'[0/V_{0,0}, x/V_{0,1}, y/V_{1,0}, z/V_{1,1}].$$

Evidently, $W \in \{x, y, z, \bar{x}, \bar{y}, \bar{z}, 0\}^*$. Let us denote every assignment t: $\{x, y, z\} \mapsto \{0, 1\}$ by the triple $(t(x), t(y), t(z))$. By construction of W we have:

- let $t_0 = (0, 0, 0)$, then $t_0(W) \in \mathfrak{L}$,
- let $t_1 = (0, 1, 1)$, then $t_1(W) \in \mathfrak{L}$,
- let $t_2 = (1, 0, 1)$, then $t_2(W) \in \mathfrak{L}$,
- $t_1 \oplus t_2 = (1, 1, 0)$ and $t_1 \oplus t_2(W) \notin \mathfrak{L}$.

Observe that at least two of the variables x, y, and z actually occur in W. Indeed, note that if $V_{0,1} = V_{1,0} = \emptyset$ then $s'_1 = s'_2$. Thus, $s'_1 \oplus s'_2 = \vec{0} = s'_0$, contradicting the fact according to which $s'_0(X')$ belongs to $\mathfrak{L}$ but $s'_1 \oplus s'_2(X')$ does not. In the same way, if $V_{0,1} = V_{1,1} = \emptyset$ (respectively $V_{1,0} = V_{1,1} = \emptyset$) then $s'_2 = \vec{0}$ (respectively $s'_1 = \vec{0}$), thus $s'_1 \oplus s'_2 = s'_1$ (respectively $s'_1 \oplus s'_2 = s'_2$), a contradiction. Hence, there are two cases to consider:

Case 1: Only the variables x and y occur in W (resp. x and z or y and z). All assignments to two variables occur as restrictions of t_0, t_1, t_2 and $t_1 \oplus t_2$. Thus, the representability of $(\bar{x} \vee \bar{y})$ (resp. $(\bar{x} \vee z)$ or $(\bar{y} \vee z)$) can easily be checked. Therefore, $(\bar{x} \vee \bar{y})$ belongs to $\mathcal{R}ep_c(\mathfrak{L})$.

Case 2: All three variables x, y, and z occur in W. We are interested in the truth assignments t such that $t(W) \in \mathfrak{L}$. Let us denote by $\mathcal{S}$ the set of these assignments. As we observed above if $t = (0, 0, 0)$, $(0, 1, 1)$, or $(1, 0, 1)$ then $t(W)$ belongs to $\mathfrak{L}$ and if $t = (1, 1, 0)$ then $t(W)$ does not. Hence, $\mathcal{S}$ contains $\{(0, 0, 0), (0, 1, 1), (1, 0, 1)\}$, let us call this subset the *base*, and does not contain $(1, 1, 0)$. We do not know whether $\mathcal{S}$ contains $(0, 0, 1)$, $(0, 1, 0)$, $(1, 0, 0)$, and $(1, 1, 1)$. Therefore there are 16 cases to analyse. Figure 1 shows that the word $\tilde{W}$ (obtained from W by possibly replacing some variable by a constant) can always represent one of the clauses $(x \vee y)$, $(\bar{x} \vee y)$ or $(\bar{x} \vee \bar{y})$ (for example, if S is formed by the base and $(0, 0, 1)$ then the word $\tilde{W}$ obtained from W in replacing the variable z by the constant 1 represents the clause $(\bar{x} \vee \bar{y})$). Hence, in all cases $\mathcal{R}ep_c(\mathfrak{L})$ necessarily contains one of these clauses, thus completing the proof of Lemma 4.5. ∎

if $\mathcal{S}=$ base and	$W=$	represents	if $\mathcal{S}=$ base and	$W=$	represents
$\emptyset$	W	$\bar{x}\vee\bar{y}$	$(0,1,0),(1,0,0)$	$W[0/x]$	$y\vee\bar{z}$
$(0,0,1)$	$W[1/z]$	$\bar{x}\vee\bar{y}$	$(0,1,0),(1,1,1)$	$W[1/z]$	$x\vee y$
$(0,1,0)$	$W[0/x]$	$y\vee\bar{z}$	$(1,0,0),(1,1,1)$	$W[1/z]$	$x\vee y$
$(1,0,0)$	$W[0/y]$	$x\vee\bar{z}$	$(0,0,1),(0,1,0),(1,0,0)$	$W[0/z]$	$\bar{x}\vee\bar{y}$
$(1,1,1)$	$W[1/z]$	$x\vee y$	$(0,0,1),(0,1,0),(1,1,1)$	$W[0/y]$	$\bar{x}\vee z$
$(0,0,1),(0,1,0)$	$W[1/z]$	$\bar{x}\vee\bar{y}$	$(0,0,1),(1,0,0),(1,1,1)$	$W[0/x]$	$\bar{y}\vee z$
$(0,0,1),(1,0,0)$	$W[1/z]$	$\bar{x}\vee\bar{y}$	$(0,1,0),(1,0,0),(1,1,1)$	$W[1/z]$	$x\vee y$
$(0,0,1),(1,1,1)$	$W[0/x]$	$\bar{y}\vee z$	$(0,0,1),(0,1,0),(1,0,0),(1,1,1)$	$W[0/z]$	$\bar{x}\vee\bar{y}$

FIG. 1. Analysis of 16 cases for Lemma 4.5

LEMMA 4.6
If $\mathfrak{L}_2$ is not bijunctive and if $(\bar{x}\vee\bar{y})$ belongs to $\mathcal{R}ep_c(\mathfrak{L}_1)$ then 1-Exactly(x,y,z) belongs to $\mathcal{R}ep_c(\mathfrak{L}_1,\mathfrak{L}_2)$.

PROOF. Let $\mathfrak{L}_2$ be a non-bijunctive language. Using Lemma 4.3, let X be a finite word made up of variables and let s_0, s_1 and s_2 be truth assignments such that $s_0(X)$, $s_1(X)$ and $s_2(X)$ belong to $\mathfrak{L}_2$ but $[(s_0\cup s_1)\cap(s_1\cup s_2)\cap(s_2\cup s_0)](X)$ does not. Form X' from X by negating all occurrences of the variables in the set $\{v\in\mathcal{V}ar(X)\ :\ s_0(v)=1\}$. Observe that for all truth assignment s, $s(X)\in\mathfrak{L}_2$ iff $s\oplus s_0(X')\in\mathfrak{L}_2$. Define $s'_i=s_i\oplus s_0$, for $i=0,1,2$ (s'_0 is the all-zero assignment, $\vec{0}$). Observe that $s_0\oplus[(s_0\cup s_1)\cap(s_1\cup s_2)\cap(s_2\cup s_0)]=s'_1\cap s'_2$. Hence, from the remark above, $s'_0(X')$, $s'_1(X')$ and $s'_2(X')$ belong to $\mathfrak{L}_2$, but $s'_1\cap s'_2(X')$ does not.

For $i,j=0,1$, let $V_{i,j}=\{v\in\mathcal{V}ar(X')\ :\ s'_1(v)=i\wedge s'_2(v)=j\}$ and let

$$W^2=X'[0/V_{0,0},x/V_{0,1},y/V_{1,0},z/V_{1,1}].$$

Clearly, $W^2\in\{x,y,z,\bar{x},\bar{y},\bar{z},0\}^*$. Moreover, the three variables x, y and z actually occur in W^2. Indeed, note that if $V_{1,1}=\emptyset$ then $s'_1\cap s'_2=s'_0$, contradicting the fact that $s'_0(W)$ belongs to $\mathfrak{L}$ and $s'_1\cap s'_2(W)$ does not. In the same way if $V_{1,0}=\emptyset$ (respectively $V_{0,1}=\emptyset$) then $s'_1\cap s'_2=s'_1$ (respectively s'_2), a contradiction. Let us denote every assignment t: $\{x,y,z\}\mapsto\{0,1\}$ by the triple $(t(x),t(y),t(z))$. By construction of W^2:

- let $t_0=(0,0,0)$, then $t_0(W^2)\in\mathfrak{L}_2$,
- let $t_1=(0,1,1)$, then $t_1(W^2)\in\mathfrak{L}_2$,
- let $t_2=(1,0,1)$, then $t_2(W^2)\in\mathfrak{L}_2$,
- $t_1\cap t_2=(0,0,1)$ and $t_1\cap t_2(W^2)\notin\mathfrak{L}_2$.

Let $\tilde{W}^2$ be the word obtained from W^2 by negating all occurrences of the variable z. On account of the above statement, if $t=(0,0,1)$, $(0,1,0)$ or $(1,0,0)$ then $t(\tilde{W}^2)$ belongs to $\mathfrak{L}_2$, but if $t=(0,0,0)$ then $t(\tilde{W}^2)$ does not. By assumption $(\bar{x}\vee\bar{y})$ is representable by $\mathfrak{L}_1$. Without loss of generality let us suppose that $(\bar{x}\vee\bar{y})$ is represented by a single word $W^1_{x,y}$ (according to the proof of Lemma 4.5 this is effectively the case in our context). This means that a truth assignment $s:\{x,y\}\mapsto\{0,1\}$ satisfies $(\bar{x}\vee\bar{y})$ iff s can be extended to $\mathcal{V}ar(W^1_{x,y})$ so that $s(W^1_{x,y})$ belongs to $\mathfrak{L}_1$.

Let us now consider the following set of words:

$$\{\tilde{W}^2,W^1_{x,y},W^1_{y,z},W^1_{z,x}\}.$$

Let $\mathcal{V}$ be the set of all the variables occurring in these words. It is easy to see that for all truth assignment $t : \{x, y, z\} \mapsto \{0, 1\}$, t satisfies the clause 1-Exactly(x, y, z) iff $t(\bar{W}^2) \in \mathfrak{L}_2$ and $\{t(W^1_{x,y}), t(W^1_{y,z}), t(W^1_{z,x})\} \subset \mathfrak{L}_1$. (Note that $\{t(W^1_{x,y}), t(W^1_{y,z}), t(W^1_{z,x})\} \subset \mathfrak{L}_1$ iff at most one of the variables x, y and z is true). Finally, we have proved that if $\mathfrak{L}_2$ is non-bijunctive and if $\mathcal{R}ep_c(\mathfrak{L}_1)$ contains $(\bar{x} \vee \bar{y})$ then the special clause 1-Exactly(x, y, z) is representable by $\mathfrak{L}_1$ and $\mathfrak{L}_2$, thus completing the proof of Lemma 4.6. ∎

We have now assembled all the information necessary to prove Proposition 4.1.

PROOF. of Proposition 4.1 Suppose that $\mathfrak{L}_1$ is not affine. It follows from Lemma 4.5 that $\mathcal{R}ep_c(\mathfrak{L}_1)$ contains $(\bar{x} \vee \bar{y})$. Now, suppose that $\mathfrak{L}_2$ is not bijunctive. Then, the clause 1-Exactly(x, y, z) belongs to $\mathcal{R}ep_c(\mathfrak{L}_1, \mathfrak{L}_2)$ from Lemma 4.6. This clearly enables us to provide a polynomial-time reduction from 1-Exactly-3SAT to SAT$_c(\mathfrak{L}_1, \mathfrak{L}_2)$, thus proving the NP-hardness of this last problem. ∎

REMARK 4.7
As was pertinently pointed out by an anonymous referee, Proposition 4.1 could be deduced from Schaefer's results in showing how our simplifying assumptions (symmetric languages and literals instead of variables) lead to vanish two cases of Schaefer's theorem [14, Theorem 3.0, cases a) and b)]. However, we have chosen to give a detailed proof, in particular to develop the proof of Lemma 4.5, in order to get a more explicit reduction that can be easily shown to be parsimonious (i.e. preserving the number of solutions). This was not explicit in Schaefer's proof for he allowed existential quantifiers to represent formulas. So, our more precise reduction will be of use in the next section for the corresponding counting problems.

To prove results on SAT$(\mathfrak{L}_1, \mathfrak{L}_2)$ it remains to show that there exists a reduction from each satisfiability problem with constants SAT$_c(\mathfrak{L}_1, \mathfrak{L}_2)$ to the corresponding problem without constants SAT$(\mathfrak{L}_1, \mathfrak{L}_2)$. This is the purpose of the following proposition.

PROPOSITION 4.8
SAT$_c(\mathfrak{L}) \leq$ SAT$(\mathfrak{L})$.

The more general result for SAT$_c(\mathfrak{L}_1, \mathfrak{L}_2)$ is easy to deduce. To prove the proposition we need one more definition, which is also due to Schaefer [14].

DEFINITION 4.9
A language $\mathfrak{L}$ is complementive iff for all word X in $\{0, 1\}^*$, $X \in \mathfrak{L}$ iff $\bar{X} \in \mathfrak{L}$. (where $\bar{X}$ is obtained from X by replacing 0 by 1 and conversely).

PROOF. of Proposition 4.8 **Case 1:** If $\mathfrak{L}$ is not complementive. Then, there exists a word X in $\{0, 1\}^*$ such that $X \in \mathfrak{L}$ and $\bar{X} \notin \mathfrak{L}$. Let $W = X[\bar{\alpha}/0, \alpha/1]$ where α is a reserved variable. For each truth assignment s, $s(W)$ belongs to $\mathfrak{L}$ iff $s(\alpha) = 1$ by definition of X. Let $\{P_{\mathfrak{L}}(W_1), \ldots, P_{\mathfrak{L}}(W_p)\}$ be a set of $\mathfrak{L}$-clauses given as instance for SAT$_c(\mathfrak{L})$. The words $W_1, \ldots, W_p$ are made up of literals (different from α and $\bar{\alpha}$) and of constants 0 and 1. Let us define $W'_i := W_i[\bar{\alpha}/0, \alpha/1]$, $i = 1, \ldots, p$. Then, let us consider the set $\{P_{\mathfrak{L}}(W), P_{\mathfrak{L}}(W'_1), \ldots, P_{\mathfrak{L}}(W'_p)\}$ as the corresponding instance for SAT$(\mathfrak{L})$.

Case 2: If $\mathfrak{L}$ is complementive. The two constants play a symmetric role in $\mathfrak{L}$. To the set of $\mathfrak{L}$-clauses $\{P_{\mathfrak{L}}(W_1), \ldots, P_{\mathfrak{L}}(W_p)\}$ it suffices to associate the set $\{P_{\mathfrak{L}}(W'_1), \ldots, P_{\mathfrak{L}}(W'_p)\}$. If there is a truth assignment s with $\{s(W'_1), \ldots, s(W'_p)\} \subset \mathfrak{L}$ that assigns α to 0, then $\{(1-s)(W'_1), \ldots, (1-s)(W'_p)\} \subset \mathfrak{L}$ too since the language is complementive, and $(1-s)(\alpha) = 1$. Thus, we can suppose $s(\alpha) = 1$. Then, $\{s(W_1), \ldots, s(W_p)\} \subset \mathfrak{L}$ as required. ∎

We are now in a position to prove the NP-complete case of Theorem 2.9. Since 1-Exactly-3SAT [14] is NP-complete, it follows immediately from Propositions 4.1 and 4.8 that if $\mathfrak{L}_1$ is not affine and if $\mathfrak{L}_2$ is not bijunctive (possibly $\mathfrak{L}_1 = \mathfrak{L}_2$) then SAT($\mathfrak{L}_1, \mathfrak{L}_2$) is NP-hard. Moreover, SAT($\mathfrak{L}_1, \mathfrak{L}_2$) is in NP (since $\mathfrak{L}_1$ and $\mathfrak{L}_2$ are polynomial), thus SAT($\mathfrak{L}_1, \mathfrak{L}_2$) is NP-complete.

REMARK 4.10
The assumption that the language is symmetric is not necessary in this section. However, the proof of Lemma 3.4 in the polynomial case strongly depends on this assumption.

5 Corresponding counting problems

There is also a dichotomy result for the corresponding satisfiability counting problems, #SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$), in which we are not interested only in the existence of a solution but ask more precisely for the number of distinct solutions (see [11]).

5.1 Counting class #P

Using the fact that the decision problem SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) is in NP it is easy to see that the corresponding counting problem #SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) belongs to the class #P. This complexity class was introduced first by Valiant [15, 16] (see also [12]). The #P-complete problems are the most difficult problems in the class #P. The reductions in the #P-hardness proofs must preserve the number of solutions. Therefore the necessity to look for parsimonious or weakly parsimonious reductions that we define below.

DEFINITION 5.1 (Kozen [12])
Let Σ and Γ be non-empty alphabets. Let $w\colon \Sigma^* \to \mathcal{P}(\Gamma^*)$, where $\mathcal{P}(\Gamma^*)$ denotes the power set of Γ^*, and let $x \in \Sigma^*$. We refer to the elements of $w(x)$ as *witnesses* for x and to w as *witness function*.

Let $w\colon \Sigma^* \to \mathcal{P}(\Gamma^*)$ and $v\colon \Pi^* \to \mathcal{P}(\Delta^*)$ be counting problems. A *polynomial many-one counting (or weakly parsimonious) reduction* from w to v consists of a pair of polynomial-time computable functions $\sigma\colon \Sigma^* \to \Pi^*$ and $\tau\colon N \to N$ such that $|w(x)| = \tau(|v(\sigma(x))|)$. When such a reduction exists we say that w *reduces to* v and we denote $w \leq_! v$. A counting reduction σ, τ is *parsimonious* if τ is the identity function .

Note that a composition of (weakly) parsimonious reductions is a (weakly) parsimonious reduction.

DEFINITION 5.2
A counting problem w is **#P-hard** if $v \leq_! w$ for all problems $v \in \#P$. A counting problem w is **#P-complete** if it is #P-hard and $w \in \#P$.

If w is #P-complete and there is a weakly parsimonious reduction from w to v then v is #P-hard. The following counting problem is #P-complete and used in the proof of Theorem 5.3 to prove #P-hardness.

#Monotone-2Sat (proved #P-complete by Valiant [16])
Instance: Set V of Boolean variables, a Boolean formula $\mathcal{B}$ over V in conjunctive normal form where each clause of $\mathcal{B}$ has exactly two negative literals.
Question: How many truth assignments for V satisfy $\mathcal{B}$?

5.2 Dichotomy result in #P

THEOREM 5.3
Let $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ be symmetric polynomial languages. If $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ are all affine then $\#\mathrm{SAT}(\mathfrak{L}_1, \ldots, \mathfrak{L}_p)$ is in FP, otherwise $\#\mathrm{SAT}(\mathfrak{L}_1, \ldots, \mathfrak{L}_p)$ is #P-complete.

PROOF. Suppose first that the languages $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ are all affine. Each instance of the problem $\#\mathrm{SAT}(\mathfrak{L}_1, \ldots, \mathfrak{L}_p)$ can be transformed in a linear system of equations as is proposed in the proof of Corollary 3.5. The rank r of this system can be obtained in polynomial time (e.g. using Gaussian elimination). The number of truth assignments satisfying the initial instance is then 2^{n-r}, where n is the number of distinct variables occurring in the clauses, hence is polynomial-time calculable.

Now, for sake of readibility let us prove that if $\mathfrak{L}$ is not affine then $\#\mathrm{SAT}(\mathfrak{L})$ is #P-complete. The proof of Theorem 5.3 can then easily be deduced.

Observe that if $\mathfrak{L}$ is not affine, then Lemma 4.5 provides a parsimonious reduction from the #P-complete problem #Monotone-2SAT to $\#\mathrm{SAT}_c(\mathfrak{L})$ (observe in Figure 1 that there is a one-to-one correspondance between the truth assignments s such that $s(\tilde{W})$ belongs to $\mathfrak{L}$ and the ones that satisfy $(\bar{x} \vee \bar{y})$). Similarly, Proposition 4.8 provides a weakly parsimonious reduction (with a multiplicative factor 2 if $\mathfrak{L}$ is complementive, case 2) from $\#\mathrm{SAT}_c(\mathfrak{L})$ to $\#\mathrm{SAT}(\mathfrak{L})$. Hence, by transitivity $\#\mathrm{SAT}(\mathfrak{L})$ is #P-hard. Moreover, $\#\mathrm{SAT}(\mathfrak{L})$ is clearly in #P. Therefore, if $\mathfrak{L}$ is not affine then $\#\mathrm{SAT}(\mathfrak{L})$ is #P-complete. ∎

6 Decision algorithm for affine or bijunctive regular symmetric languages

As an additional result, let us describe a normal form for affine or bijunctive **regular** symmetric languages that yields a decision algorithm for membership of both classes of languages. All through this section, we make use of several properties of finite automata and regular languages. If necessary, the reader is invited to refer to [10] for a precise statement and a proof of these properties. First, we need the following definition.

DEFINITION 6.1
The length closure of a language $\mathfrak{L}$ is $\lceil\mathfrak{L}\rceil = \{W \in \{0,1\}^* : \text{ there exists } W_0 \in \mathfrak{L}, |W| = |W_0|\}$. We say that the language $\mathfrak{L}$ is length-closed if $\mathfrak{L} = \lceil\mathfrak{L}\rceil$.

REMARK 6.2
Note that if $\mathfrak{L}$ is regular, then $\lceil\mathfrak{L}\rceil$ is also regular. For instance, a finite non-deterministic automaton recognizing $\lceil\mathfrak{L}\rceil$ is obtained from a finite deterministic automaton $\mathcal{A}$ recognizing $\mathfrak{L}$ by replacing any move $\delta(q,a) = q'$ (where $a \in \{0,1\}$ and q, q' are states) of the transition function δ of $\mathcal{A}$ by the two moves $\delta(q,0) = q'$ and $\delta(q,1) = q'$. Note that this algorithm works in polynomial time.

PROPOSITION 6.3
A regular symmetric language $\mathfrak{L}$ is affine (resp. bijunctive) iff there are pairwise disjoint length-closed regular languages $\mathfrak{A}_1(\mathfrak{L}), \ldots, \mathfrak{A}_6(\mathfrak{L})$ (resp. $\mathfrak{A}_1(\mathfrak{L}), \ldots, \mathfrak{A}_4(\mathfrak{L}), \mathfrak{A}_7(\mathfrak{L}), \mathfrak{A}_8(\mathfrak{L})$) such that : $\mathfrak{L} = \bigcup_{i=1}^{6} (\mathfrak{L}^i \cap \mathfrak{A}_i(\mathfrak{L}))$ (resp. $\bigcup_{i \in \{1,\ldots,4,7,8\}} (\mathfrak{L}^i \cap \mathfrak{A}_i(\mathfrak{L}))$).

PROOF. Note that $\mathfrak{A}_i(\mathfrak{L})$ intuitively represents the (unary representation of the) integers k for which $\mathfrak{L}(k) = \mathfrak{L}^i(k)$. Besides, if the $\mathfrak{A}_i(\mathfrak{L})$ are regular, then $\mathfrak{L}$ is obviously regular.

Affine case. For $i = 1 \ldots 6$, let $\mathfrak{A}_i(\mathfrak{L})$ be the language $\{W : |W| = k$ and $k \geq 3$ and $\mathfrak{L}(k) = \mathfrak{L}^i(k)\}$. These languages are length-closed and, due to Lemma 3.3, pairwise disjoint. Moreover, for all $k \geq 3$ and for all $i = 1 \ldots 6$, we have $(\mathfrak{L}^i \cap \mathfrak{A}_i(\mathfrak{L}))(k) = \mathfrak{L}^i(k)$ if $\mathfrak{L}(k) = \mathfrak{L}^i(k)$ and $\emptyset$ if not. Hence, we only have to prove that if $\mathfrak{L}$ is regular then, for $i = 1 \ldots 6$, $\mathfrak{A}_i(\mathfrak{L})$ is also regular.

We make use of the set $S(k) = \{0^k, 1^k, 10^{k-1}, 110^{k-2}\}$ that separates the generic affine and symmetric languages $\mathfrak{L}^0(k), \ldots, \mathfrak{L}^6(k)$ (see Lemma 3.3). We have $\mathfrak{L}(k) = \mathfrak{L}^i(k)$ iff $\mathfrak{L}(k) \cap S(k) = \mathfrak{L}^i(k) \cap S(k)$, due to Lemma 3.4. For instance, $\mathfrak{L}(k) = \mathfrak{L}^5(k)$ iff $110^{k-2} \in \mathfrak{L}(k)$ and $10^{k-1} \notin \mathfrak{L}(k)$. So, $\mathfrak{A}_i(\mathfrak{L})$ can be computed as follows. For sake of readability, we confuse the regular expressions and the languages they denote.

$\mathfrak{A}_4(\mathfrak{L}) = (\lceil \mathfrak{L} \cap 0^* \rceil \cap \lceil \mathfrak{L} \cap 1^* \rceil \cap \lceil \mathfrak{L} \cap 10^* \rceil \cap \lceil \mathfrak{L} \cap 110^* \rceil)$
$\mathfrak{A}_5(\mathfrak{L}) = \lceil \mathfrak{L} \cap 110^* \rceil \setminus \lceil \mathfrak{L} \cap 10^* \rceil$ and $\mathfrak{A}_6(\mathfrak{L}) = \lceil \mathfrak{L} \cap 10^* \rceil \setminus \lceil \mathfrak{L} \cap 110^* \rceil$
$\mathfrak{A}_1(\mathfrak{L}) = \lceil \mathfrak{L} \cap 0^* \rceil \setminus (\lceil \mathfrak{L} \cap 1^* \rceil \cup \lceil \mathfrak{L} \cap 110^* \rceil)$
$\mathfrak{A}_2(\mathfrak{L}) = \lceil \mathfrak{L} \cap 1^* \rceil \setminus (\lceil \mathfrak{L} \cap 0^* \rceil \cup \lceil \mathfrak{L} \cap 10^* \rceil)$
$\mathfrak{A}_3(\mathfrak{L}) = (\lceil \mathfrak{L} \cap 0^* \rceil \cap \lceil \mathfrak{L} \cap 1^* \rceil) \setminus \lceil \mathfrak{L} \cap 110^* \rceil$.

Bijunctive case. There are similar formulas using $S'(k)$ instead of $S(k)$. ∎

The announced decision algorithm is now an easy corollary.

COROLLARY 6.4
Let $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ be regular symmetric languages, defined by deterministic finite automata. The property 'Are $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ all affine (resp. bijunctive)?' is decidable.

PROOF. Let us describe a decision algorithm. For $j = 1 \ldots p$ and $i = 1 \ldots 6$ (resp. $i \in \{1, \ldots, 4, 7, 8\}$), compute a (non-deterministic) finite automaton accepting $\mathfrak{A}_i(\mathfrak{L}_j)$. This can be done from a deterministic finite automata accepting $\mathfrak{L}_j$ using the characterization of $\mathfrak{A}_i(\mathfrak{L}_j)$ provided in the proof of Proposition 6.3. Then, compute a non-deterministic finite automaton accepting $\mathfrak{L}'_j = \bigcup_{i=1}^{6} (\mathfrak{L}^i \cap \mathfrak{A}_i(\mathfrak{L}_j))$ (resp. $\bigcup_{i \in \{1,\ldots,4,7,8\}} (\mathfrak{L}^i \cap \mathfrak{A}_i(\mathfrak{L}_j))$). Note that there are polynomial algorithms achieving the combinatorial operations involved (union, intersection, complementation, length closure) over finite automata. Finally, compute a deterministic finite automaton accepting $\mathfrak{L}'_j$ (this is not polynomial in general) and verify that $\mathfrak{L}_j = \mathfrak{L}'_j$ (there is a polynomial-time algorithm to decide whether two deterministic finite automata accept the same language).
Note that the overall complexity of this algorithm is not polynomial. ∎

REMARK 6.5
The property 'Is SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) polynomial-time decidable?' is decidable using the same algorithm, when the languages $\mathfrak{L}_1, \ldots, \mathfrak{L}_p$ are symmetric and regular. (Note that given a finite automaton, it is easy to decide whether the language it recognizes is symmetric. It suffices to verify that the minimum state equivalent automaton has a symmetric transition function δ, i.e. $\hat{\delta}(q, ab) = \hat{\delta}(q, ba)$ for all state q and all input symbols a and b, see for example [10].)

REMARK 6.6
The algorithm stated in the proof of Corollary 6.4 shows that the property 'Is #SAT($\mathfrak{L}_1, \ldots, \mathfrak{L}_p$) in FP?' is decidable.

Acknowledgement

We are greatly indebted to Etienne Grandjean for suggesting the problem and for many stimulating conversations. We also thank Pascal Weil for helpful references concerning regular languages.

References

[1] S. A. Cook. The complexity of theorem-proving procedures. In *Third Annual ACM Symposium on Theory of Computing*. pp. 151–158. 1971.

[2] N. Creignou. The class of problems that are linearly equivalent to satisfiability or a uniform method for proving NP-completeness. *Theoretical Computer Science*, **145**, 111–145, 1995.

[3] N. Creignou and M. Hermann. Complexity of generalized satisfiability counting problems. To appear in *Information and Computation*, **125**, 1996.

[4] M. Davis and H. Putnam. Computing program for quantification theory. *JACM*, **7**, 201–215, 1960.

[5] S. Fortune, J. Hopcroft and J. Wyllie. The directed subgraph homeomorphism problem. *Theoretical Computer Science*, **10**, 112–121, 1980.

[6] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., New York, NY, 1979.

[7] E. Grandjean. Sorting, linear time and Satisfiability problem. To appear in *Annals of Mathematics and Artificial Intelligence*, M. Nivat and S. Grigorieff, eds., Special Issue. J. C. Baltzer Scientific Publishing Co. (Suisse), 1996.

[8] J. J. Hebrard. A linear algorithm for renaming a set of clauses as a Horn set. *Theoretical Computer Science*, **124**, 343–350, 1994.

[9] P. Hell and J. Nešetřil. On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, **48**, 92–110, 1990.

[10] J. E. Hopcroft and J. D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Reading, MA, 1979.

[11] D. S Johnson. A catalog of complexity classes. In *Handbook of Theoretical Computer Science, Volume A: Algorithms and Complexity*, J. van Leeuwen, ed. pp. 67–161. North-Holland, Amsterdam, 1990.

[12] D. C. Kozen. *The Design and Analysis of Algorithms*, Counting problems and #P, pp. 138–143. Springer-Verlag, New York, NY, 1992.

[13] J.-E. Pin. *Variétés de langages formels*. Masson, Etudes et recherches en informatique, 1984.

[14] T. J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the 10th STOC, San Diego, CA*, pp. 216–226. ACM, New York, NY, 1978.

[15] L. G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, **8**, 189–201, 1979.

[16] L. G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, **8**, 410–421, 1979.

Received 16 May 1995