Exact Satisfiabitity with Jokers*

Gordon Hoi[1], Sanjay Jain[2], Sibylle Schwarz[3] and Frank Stephan[4]

[1] School of Computing, National University of Singapore, 13 Computing Drive, Block COM1, Singapore 117417, Republic of Singapore, e0013185@u.nus.edu
[2] School of Computing, National University of Singapore, 13 Computing Drive, Block COM1, Singapore 117417, Republic of Singapore, sanjay@comp.nus.edu.sg
[3] Fakultät Informatik, Mathematik und Naturwissenschaften, Hochschule für Technik, Wirtschaft und Kultur Leipzig, Gustav-Freytag-Str. 42a, 04277 Leipzig, Germany sibylle.schwarz@htwk-leipzig.de
[4] Department of Mathematics and School of Computing, National University of Singapore, 10 Lower Kent Ridge Road, Block S17, Singapore 119076, Republic of Singapore, fstephan@comp.nus.edu.sg

Abstract. The XSAT problem asks for solutions of a set of clauses where for every clause exactly one literal is satisfied. The present work investigates a variant of this well-investigated topic where variables can take a joker-value (which is preserved by negation) and a clause is satisfied iff either exactly one literal is true and no literal has a joker value or exactly one literal has a joker value and no literal is true. While JX2SAT is in polynomial time, the problem becomes NP-hard when one searches for a solution with the minimum number of jokers used and the decision problem X3SAT can be reduced to the decision problem of the JX2SAT problem with a bound on the number of jokers used. JX3SAT is in both cases, with or without optimisation of the number of jokers, NP-hard and X3SAT can be reduced to it without increasing the number of variables. Furthermore, the general JXSAT problem can be solved in the same amount of time as variable-weighted XSAT and the obtained solution has the minimum amount of number of jokers possible.

1 Introduction

Consider a propositional formula φ with n variables, say $x_1, x_2, ..., x_n$. A well-studied question is which values for x_i from $\{0, 1\}$ one can take so that φ evaluates to 1. This problem is also known as the Boolean satisfiability problem, SAT, and was shown to be the first NP-complete problem.

Since then, many variants of Boolean satisfiability problem have been introduced. Some of these includes the 3SAT problem, where every clause must have exactly 3 literals. Another variant is the Exact Satisfiability problem, XSAT,

* S. Jain was supported in part by NUS grant C252-000-087-001; furthermore, S. Jain and F. Stephan have been supported in part by the Singapore Ministry of Education Academic Research Fund Tier 2 grant MOE2016-T2-1-019 / R146-000-234-112. Part of the work was done while S. Schwarz was on sabbatical leave to the National University of Singapore.

where we require that exactly 1 literal can only take on the value 1 and the other literals must take on 0. The problems X2SAT and X3SAT are special instances of XSAT where we restrict the clauses to 2 and 3 literals respectively. It is known that the decision problem of 2SAT, X2SAT are in the class **P**. On the other hand, the decision problems 3SAT [5], XSAT [12] and X3SAT [14] are known to be **NP**-complete. Furthermore, one can also deviate from the decision problems as mentioned above to talk about optimisation problems. If we wish to maximise the number of satisfied clauses in an 2SAT problem, also known as Max2SAT, the complexity of this problem increases and is now **NP**-hard as compared to its decision counterpart [1, 9]. Here Max2SAT asks on how many of (perhaps weighted) 2SAT-formulas can be satisfied in the best case and this problem has been widely studied [6, 8, 13]. Max2SAT can be generalised to Max-2-CSP where one has to satisfy as many constraints as possible over a domain, in this case a binary domain [3]. Chen [2] shows that various natural classes of CSP-problems have to be either **L**-complete, **NL**-complete, **P**-complete or **NP**-complete.

The Boolean satisfiability problem with two truth values are indeed well-studied and instead of asking, how many clauses can be satisfied, one can also ask how many special third values are needed in order to satisfy all clauses. So in this paper, we introduce another truth value, "J" for Joker, into our framework. The value "J" can be treated as an alternative to the truth value 1 with some slight changes. In this case, our satisfying condition can be either the values 1 or "J". The goal of this paper is to investigate if having more truth values can change the structural complexity of known existing problems. If one allows more truth values in the framework, will some existing **NP**-hard problem be brought down to **P** or vice versa.

We are in fact, not the first to introduce an additional truth value in our framework and study them. For example, Lardeux, Saubion and Hao, like us, introduce a new truth value into the system [7] and then study Boolean satisfiability from there. Their goal, unlike what we do, is to introduce a new truth value in order to define a local search procedure and to study the fundamental mechanisms behind it.

We show the following results. The decision problem JX2SAT is in the class **P** while the decision problem JX3SAT is **NP**-complete; in these problems we allow the usage of joker-values without making any statement on their number. The optimisation variant MinJX2SAT is shown to be **NP**-hard when asked to find the minimum amount of jokers or **NP**-complete when asked whether the number of jokers can be brought below a certain threshold. Finally, we give an exponential algorithm to solve MinJXSAT in $O(1.185^n)$ time.

2 The Underlying Logic and Joker Assignments

Given a set $V = \{x_1, \ldots, x_n\}$ of variables, propositional formulas are defined inductively with the usual connectives like $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$.

In this paper, we only use formulas in conjunctive normal form, that is, conjunctions of disjunctions (clauses) of literals. k-CNFs are formulas in CNF where every clause contains at most k literals. For XSAT-formulas and JXSAT-formulas, we make however the convention that a clause is satisfied if and only if exactly one literal in the clause is not 0; thus clauses like $x_1 \vee x_2 \vee \neg x_3$ have a different meaning in XSAT than in normal SAT, see below for more details.

The semantics of propositional formulas in XSAT problems with Joker values is defined by assignments $\beta : V \to \{0, 1, J\}$. Negation maps 0 to 1, 1 to 0 and J to J.

In SAT problems, an assignment $\beta : V \to \{0, 1, J\}$ satisfies a k-clause $l_1 \vee \cdots \vee l_k$ iff there is *at least one* $i \in \{1, \ldots, k\}$ such that $\beta(l_i) \in \{1, J\}$.

In XSAT problems, an assignment $\beta : V \to \{0, 1, J\}$ satisfies a k-clause $l_1 \vee \cdots \vee l_k$ iff there is *exactly one* $i \in \{1, \ldots, k\}$ such that $\beta(l_i) \in \{1, J\}$ and $\beta(l_j) = 0$ for all other literals.

An assignment $\beta : V \to \{0, 1, J\}$ satisfies a CNF φ iff it satisfies every clause in φ. The problem JSAT is defined as follows:

Instance: CNF φ;
Question: Is there an assignment $\beta : \mathrm{vars}(\varphi) \to \{0, 1, J\}$ such that β maps at least one literal in each clause of φ to 1 or J?

The problem JXSAT is defined as follows:

Instance: CNF φ;
Question: Is there an assignment $\beta : \mathrm{vars}(\varphi) \to \{0, 1, J\}$ such that β maps exactly one literal in each clause of φ to 1 or J and the other literals in the clause to 0?

For CNFs containing only positive literals, the problems JSAT and JXSAT coincide with SAT and XSAT. In our work we mainly concentrate on JXSAT, JX2SAT and JX3SAT as well as MinJXSAT, MinJX2SAT and MinJX3SAT.

Proposition 1. *For every negation-free CNF φ, $\varphi \in$ JXSAT iff $\varphi \in$ XSAT.*

Proof. We translate each assignment $\beta : \mathrm{vars}(\varphi) \to \{0, 1, J\}$ to the assignment $\beta' : \mathrm{vars}(\varphi) \to \{0, 1\}$ where

$$\forall x_i \in \mathrm{vars}(\varphi) : \quad \beta'(x_i) = \left\{ \begin{array}{l} 0 \text{ if } \beta(x_i) = 0 \\ 1 \text{ if } \beta(x_i) \in \{1, J\} \end{array} \right\}.$$

Then β' satisfies φ iff β satisfies φ. □

CNFs containing a variable x_i only negated can be translated to a CNF containing only positive literals by replacing each negative literal $\neg x_i$ by x'_i. Since the translated formula does not contain x_i, the number of variables in the formula does not increase.

Hence on the set of all CNFs containing each variable only positive or only negated, the problem JXSAT coincides with XSAT. Note that negation-free SAT is trivially in **P**, as one can make all literals true, since no literal comes in two opposed forms; thus also here the negation-free JSAT and SAT problems are eqivalent. So the interesting case is negation-free XSAT and that is as hard as normal XSAT, as one can translate every XSAT instance in polynomial time into a negation-free XSAT instance where the number of variables and the number of clauses does not increase. For that reason, JXSAT is at least as hard as XSAT. A direct proof, not relying on this fact, for the hardness of JXSAT can also be obtained by the direct transfer of the reduction of Theorem 7 from X3SAT to JX3SAT into a reduction from XSAT to JXSAT.

3 Encodings of truth Values and Clauses

3.1 Encoding of Truth Values as Pairs of Booleans

In the following, we will use the encoding $e_2 : \{0, 1, J\} \to \{0, 1\}^2$ of the truth values $x \in \{0, 1, J\}$ as a pair $(n, p) \in \{0, 1\}^2$ defined by

$$e_2(0) = (1, 0), \quad e_2(1) = (0, 1), \quad e_2(J) = (0, 0). \tag{1}$$

For each variable $x_i \in V$, n_i and p_i denote the first and second element of this pair, respectively. This encoding is a bijection between the sets $\{0, 1, J\}$ and $\{(1, 0), (0, 1), (0, 0)\}$. The inverse of e_2 is $e_2^{-1} : \{(1, 0), (0, 1), (0, 0)\} \to \{0, 1, J\}$.

$$e_2^{-1}(1, 0) = 0, \quad e_2^{-1}(0, 1) = 1, \quad e_2^{-1}(0, 0) = J. \tag{2}$$

3.2 Encoding of 2CNFs

Every 2-clause $l_i \vee l_j$ with variables in $\{x_1, \ldots, x_n\}$ is translated to a conjunction of two 2-clauses with variables in $\{n_1, \ldots, n_n, p_1, \ldots, p_n\}$ by

$$\begin{aligned}
e_{2c}(x_i \vee x_j) &= (n_i \vee n_j) \wedge (\neg n_i \vee \neg n_j), \\
e_{2c}(x_i \vee \neg x_j) &= (n_i \vee p_j) \wedge (\neg n_i \vee \neg p_j), \\
e_{2c}(\neg x_i \vee x_j) &= (p_i \vee n_j) \wedge (\neg p_i \vee \neg n_j), \\
e_{2c}(\neg x_i \vee \neg x_j) &= (p_i \vee p_j) \wedge (\neg p_i \vee \neg p_j).
\end{aligned} \tag{3}$$

To prevent the pair $(1, 1)$ that is not an encoding of a truth value from $\{0, 1, J\}$, we add a clause $\neg n_i \vee \neg p_i$ for each variable x_i to the resulting 2CNF.

This encoding transforms every 2CNF $\varphi = \bigwedge_{i \in \{1,\ldots,m\}} c_i$ to a 2CNF

$$e_{2c}(\varphi) = \bigwedge_{i \in \{1,\ldots,m\}} e_{2c}(c_i) \wedge \bigwedge_{i \in \{1,\ldots,n\}} (\neg n_i \vee \neg p_i) \tag{4}$$

with $2n$ variables and $2m + n$ clauses that will be interpreted as 2SAT instance.

Proposition 2. *For every set $\mathcal{C}$ of 2-clauses with variables from $\{x_1, \ldots, x_n\}$, there is an assignment*

$$\beta : \{x_1, \ldots, x_n\} \to \{0, 1, J\}$$

such that for each clause $c = (l_i \vee l_j)$ *in* $\mathcal{C}$ *exactly one of* $\beta(l_i), \beta(l_j)$ *is* 0 *iff there is an assignment*

$$\beta' : \{n_1, \ldots, n_n, p_1, \ldots, p_n\} \to \{0, 1\}$$

such that for each clause $c = (l_i \vee l_j)$ *in* $\mathcal{C}$*, at least one of* $\beta'(l_i), \beta'(l_j)$ *is* 1 *in each clause of* $e_{2c}(c)$ *as well as* $(\neg n_i \vee \neg p_i)$ *for every variable* x_i *occurring in* c*.*

Proof. ($\Rightarrow$) Given an assignment $\beta : V \to \{0, 1, J\}$, we define the assignment $\beta' : \{n_1, \ldots, n_n, p_1, \ldots, p_n\} \to \{0, 1\}$ for $e_{c2}(c)$ by the encoding of the truth values in $\{0, 1, J\}$ given in Equation 1, i.e., for each $i \in \{1, \ldots, n\}$, $\beta'(n_i)$ and $\beta'(p_i)$ are defined by

$$(\beta'(n_i), \beta'(p_i)) = e_2(\beta(x_i)).$$

Let β be a satisfying assignment for $\mathcal{C}$ which satisfies exactly one literal in each clause in $\mathcal{C}$. By definition of e_2 (Equation 1), we have $\neg n_i \vee \neg p_i$ for every variable x_i occurring in c.

We show that β' satisfies at least one literal in each clause in $e_{2c}(c)$, for all $c \in \mathcal{C}$.

For every type $c = l_i \vee l_j$ of clauses we show that if β satisfies exactly one literal in c then β' satisfies $e_{2c}(c)$ defined in Equation 3.

- In $c = (x_i \vee x_j)$, exactly one of x_i, x_j is satisfied in each of the following cases
  - $\{\beta(x_i), \beta(x_j)\} = \{0, 1\}$
    Then we have $\{(\beta'(n_i), \beta'(p_i)), (\beta'(n_j), \beta'(p_j))\} = \{(1, 0), (0, 1)\}$.
  - $\{\beta(x_i), \beta(x_j)\} = \{0, J\}$
    Then we have $\{(\beta'(n_i), \beta'(p_i)), (\beta'(n_j), \beta'(p_j))\} = \{(1, 0), (0, 0)\}$.

  In both cases, we have $\{\beta'(n_i), \beta'(n_j)\} = \{0, 1\}$ and therefore β' satisfies at least one literal in each clause of $e_{2c}(c) = (n_i \vee n_j) \wedge (\neg n_i \vee \neg n_j)$.
- In $c = x_i \vee \neg x_j$, exactly one of $x_i, \neg x_j$ is satisfied in each of the following cases
  - $\beta(x_i) = \beta(x_j) \in \{0, 1\}$
    Then we have $(\beta'(n_i), \beta'(p_i)) = (\beta'(n_j), \beta'(p_j)) \in \{(1, 0), (0, 1)\}$,
  - $\beta(x_i) = 0$ and $\beta(x_j) = J$
    Then we have $(\beta'(n_i), \beta'(p_i)) = (1, 0)$ and $(\beta'(n_j), \beta'(p_j)) = (0, 0)$
  - $\beta(x_i) = J$ and $\beta(x_j) = 1$
    Then we have $(\beta'(n_i), \beta'(p_i)) = (0, 0)$ and $(\beta'(n_j), \beta'(p_j)) = (0, 1)$

  In all cases, we have $\{\beta'(n_i), \beta'(p_j)\} = \{0, 1\}$ and therefore β' satisfies at least one literal in each clause of $e_{2c}(c) = (n_i \vee p_j) \wedge (\neg n_i \vee \neg p_j)$
- The case $c = \neg x_i \vee x_j$ is symmetric to the previous case.
- In $c = \neg x_i \vee \neg x_j$, exactly one of $\neg x_i, \neg x_j$ is satisfied in each of the following cases
  - $\{\beta(x_i), \beta(x_j)\} = \{0, 1\}$
    Then we have $\{(\beta'(n_i), \beta'(p_i)), (\beta'(n_j), \beta'(p_j))\} = \{(1, 0), (0, 1)\}$.
  - $\{\beta(x_i), \beta(x_j)\} = \{1, J\}$
    Then we have $\{(\beta'(n_i), \beta'(p_i)), (\beta'(n_j), \beta'(p_j))\} = \{(0, 1), (0, 0)\}$.

  In both cases, we have $\{\beta'(p_i), \beta'(p_j)\} = \{0, 1\}$ and therefore β' satisfies at least one literal in each clause of $e_{2c}(c) = (p_i \vee p_j) \wedge (\neg p_i \vee \neg p_j)$.

($\Leftarrow$) Let $\beta' : \{n_1, \ldots, n_n, p_1, \ldots, p_n\} \rightarrow \{0, 1\}$ be an assignment such that at least one of $\beta'(l_i), \beta'(l_j)$ is 1 in each clause of $e_{2c}(c)$ and in $\neg n_i \vee \neg p_i$ for every variable x_i occurring in c, for each $c \in \mathcal{C}$.

We define $\beta : \{x_1, \ldots, x_n\} \rightarrow \{0, 1, J\}$ by the inverse encoding defined in Equation 2:

$$\beta(x_i) = e_{2c}^{-1}(\beta'(n_i), \beta'(p_i)).$$

For each clause $c = (l_i \vee l_j) \in \mathcal{C}$, we show that exactly one of $\beta(l_i), \beta(l_j)$ is 0.

- $c = x_i \vee x_j$
  Then β' satisfies $e_{2c}(c) = (n_i \vee n_j) \wedge (\neg n_i \vee \neg n_j)$, i.e., $\{\beta'(n_i), \beta'(n_j)\} = \{0, 1\}$.
  - $\beta'(n_i) = 0, \beta'(n_j) = 1$
    Then $(n_i, p_i) \in \{(0, 1), (0, 0)\}$ and $(n_j, p_j) = (1, 0)$ and hence $\beta(x_i) \in \{1, J\}$ and $\beta(x_j) = 0$
  - $\beta'(n_i) = 1, \beta'(n_j) = 0$ (symmetric)
    Then $(n_i, p_i) = (1, 0)$ and $(n_j, p_j) \in \{(0, 1), (0, 0)\}$ and hence $\beta(x_i) = 0$ and $\beta(x_j) \in \{1, J\}$.

  In each case, β maps exactly one of the literals x_i, x_j to 0.
- $c = x_i \vee \neg x_j$
  Then β' satisfies $e_{2c}(x_i \vee \neg x_j) = (n_i \vee p_j) \wedge (\neg n_i \vee \neg p_j)$, i.e., $\{\beta'(n_i), \beta'(p_j)\} = \{0, 1\}$.
  - $\beta'(n_i) = 0, \beta'(p_j) = 1$
    Then $(n_i, p_i) \in \{(0, 0), (0, 1)\}$ and $(n_j, p_j) = (0, 1)$ and hence $\beta(x_i) \in \{1, J\}$ and $\beta(x_j) = 1$, i.e., $\beta(\neg x_j) = 0$.
  - $\beta'(n_i) = 1, \beta'(p_j) = 0$
    Then $(n_i, p_i) = (1, 0)$ and $(n_j, p_j) \in \{(1, 0), (0, 0)\}$ and hence $\beta(x_i) = 0$ and $\beta(x_j) \in \{0, J\}$, i.e., $\beta(\neg x_j) \in \{1, J\}$.

  In each case, β maps exactly one of the literals $x_i, \neg x_j$ to 0.
- $c = x_i \vee \neg x_j$ is symmetric to the previous case.
- $c = \neg x_i \vee \neg x_j$
  Then β' satisfies $e_{2c}(\neg x_i \vee \neg x_j) = (p_i \vee p_j) \wedge (\neg p_i \vee \neg p_j)$, i.e., $\{\beta'(p_i), \beta'(p_j)\} = \{0, 1\}$.
  - $\beta'(p_i) = 0, \beta'(p_j) = 1$
    Then $(\beta'(n_i), \beta'(p_i)) \in \{(0, 0), (1, 0)\}$, i.e., $\beta(x_i) \in \{0, J\}$ and $\beta(\neg x_i) \in \{1, J\}$, and $(\beta'(n_j), \beta'(p_j)) = (0, 1)$, i.e., $\beta(x_j) = 1$ and $\beta(\neg x_j) = 0$.
  - $\beta'(p_i) = 1, \beta'(p_j) = 0$ (symmetric)
    Then $(\beta'(n_i), \beta'(p_i)) = (0, 1)$ and, i.e., $\beta(x_i) = 1$, and $\beta(\neg x_i) = 0$, and $(\beta'(n_j), \beta'(p_j)) \in \{(1, 0), (0, 0)\}$, i.e., $\beta(x_j) \in \{0, J\}$ and $\beta(\neg x_j) \in \{1, J\}$

  In each case, β maps exactly one of the literals $\neg x_i, \neg x_j$ to 0.

This completes the proof. □

3.3 Encoding of Truth Values as Triples of Booleans

In the following, we will use the encoding $e_3 : \{0, 1, J\} \rightarrow \{0, 1\}^3$ of the truth values $x \in \{0, 1, J\}$ as a triple $(n, p, o) \in \{0, 1\}^3$ defined as by

$$e_3(0) = (1, 0, 0), \quad e_3(1) = (0, 1, 0), \quad e_3(J) = (0, 0, 1). \tag{5}$$

For each variable $x_i \in V$, n_i, p_i and o_i denote the first, second and third element of this triple, respectively. This encoding is a bijection between the sets $\{0,1,J\}$ and $\{(1,0,0),\ (0,1,0),\ (0,0,1)\}$. The inverse of e_3 is $e_3^{-1} : \{(1,0,0),\ (0,1,0),\ (0,0,1)\} \to \{0,1,J\}$:

$$e_3^{-1}(1,0,0) = 0, \quad e_3^{-1}(0,1,0) = 1, \quad e_3^{-1}(0,0,1) = J. \tag{6}$$

3.4 Encoding of 3CNFs

Every 3-clause $l_i \vee l_j \vee l_k$ with variables in $\{x_1, \dots, x_n\}$ is translated to a clause with variables in $\{n_1, \dots, n_n, p_1, \dots, p_n, o_1, \dots, o_n\}$. Because the semantic interpretation of $\vee$ is symmetric, it suffices to define e_{3c} in the following cases:

$$\begin{array}{ll} e_{3c}(x_i \vee x_j \vee x_k) & = (\neg n_i \vee \neg n_j \vee \neg n_k), \\ e_{3c}(\neg x_i \vee x_j \vee x_k) & = (\neg p_i \vee \neg n_j \vee \neg n_k), \\ e_{3c}(\neg x_i \vee \neg x_j \vee x_k) & = (\neg p_i \vee \neg p_j \vee \neg n_k), \\ e_{3c}(\neg x_i \vee \neg x_j \vee \neg x_k) & = (\neg p_i \vee \neg p_j \vee \neg p_k). \end{array} \tag{7}$$

For every variable x_i, we add the 3-clause $p_i \vee n_i \vee o_i$ to guarantee that every satisfying assignment maps each variable x_i to a unique truth value in $\{0,1,J\}$.

This encoding transforms every 3CNF $\varphi = \bigwedge_{i \in \{1,\dots,m\}} c_i$ to a 3CNF

$$e_{3c}(\varphi) = \bigwedge_{i \in \{1,\dots,m\}} e_{3c}(c_i) \wedge \bigwedge_{i \in \{1,\dots,n\}} (n_i \vee p_i \vee o_i) \tag{8}$$

with $3n$ variables and $m+n$ clauses that will be interpreted as X3SAT instance.

Proposition 3. *For every set of clauses $\mathcal{C}$, with variables from $\{x_1, \dots, x_n\}$, there is an assignment $\beta : \{x_1, \dots, x_n\} \to \{0,1,J\}$ such that exactly one of $\beta(l_i), \beta(l_j), \beta(l_k)$ is in $\{1,J\}$ for every clause $(l_i \vee l_j \vee l_k)$ in $\mathcal{C}$ iff there is an assignment $\beta' : \{n_1, \dots, n_n, p_1, \dots, p_n, o_1, \dots, o_n\} \to \{0,1\}$ such that exactly one of $\beta'(l_i), \beta'(l_j), \beta'(l_k)$ is 1 in each clause $(l_i \vee l_j \vee l_k)$ of $e_{3c}(c)$ for each $c \in \mathcal{C}$ as well as for each $(n_i \vee p_i \vee o_i)$ for each variable x_i occurring in clauses of $\mathcal{C}$.*

Proof. Note that $\beta(x_i) \in \{1,J\}$ iff $\beta'(\neg n_i) = 1$, and $\beta(\neg x_i) \in \{1,J\}$ iff $\beta'(\neg p_i) = 1$. The Proposition now follows from the definition of e_{3c}. □

4 The Decision Problem JX2SAT

Instance: 2CNF φ
Question: Is there an assignment $\beta : \mathrm{vars}(\varphi) \to \{0,1,J\}$ such that β maps exactly one literal in each clause of φ to 1 or J and the other literal to 0?

The encoding defined in Section 4, Equations 3 and 4, maps JX2SAT instances φ to 2SAT instances $e_{2c}(\varphi)$.

Lemma 4. *For every 2CNF φ we have*

$$\varphi \in \text{JX2SAT} \textit{ iff } e_{2c}(\varphi) \in \text{2SAT}.$$

Proof. Suppose φ is in JX2SAT as witnessed by assignment β to the variables. From the proof of Proposition 2, we know how to transform β to an assignment $\beta' : \{n_1, \ldots, n_n, p_1, \ldots, p_n\} \to \{0, 1\}$ that satisfies the corresponding 2SAT formula $e_{2c}(c)$, for each clause c in φ, as well as $\neg n_i \vee \neg p_i$ for every variable x_i occurring in c. Hence β' satisfies the 2SAT instance $e_{2c}(\varphi)$.

Suppose β' witnesses that $e_{2c}(\varphi) \in$ 2SAT. Then, from the proof of Proposition 2, we know how to transform β' to an assignment β such that β satisfies exactly one literal in each clause of φ. Thus, $\varphi \in$ JX2SAT. □

Theorem 5. *The decision problem* JX2SAT *is in* ***P****.*

Proof. By Lemma 4, the encoding in Equation 3 is a reduction from JX2SAT to 2SAT. Since 2SAT is in **P** and the reduction is linear, JX2SAT is also in the class **P**. □

5 The Optimisation Problem MinJX2SAT

MinJX2SAT is the problem to decide if 2CNF has a satisfying assignment with a given number of jokers.

Instance: (φ, h), where φ is a 2CNF and $h \in \mathbb{N}$

Question: Is there an assignment $\beta : \text{vars}(\varphi) \to \{0, 1, J\}$ such that $|\beta^{-1}(J)| = h$ and β maps exactly one literal in each clause of φ to 1 or J and the other literals in the clause to 0?

The minimum of jokers required to satisfy a 2CNF φ can be found by iteratively increasing h in (φ, h), beginning with $h = 0$. We note that the problem, as stated above, is even in **NP**; however, if one wants to determine the h by iterated solving of the problem, then the problem to get the optimal h from an JX2SAT instance is **NP**-hard, but there is no known method to reduce this numerical problem to a decision problem in **NP** with one query only.

Theorem 6. MinJX2SAT *is* ***NP****-hard.*

Proof. We will show this via a reduction of X3SAT to MinJX2SAT. Then we have **NP**-completeness of MinJX2SAT by **NP**-completeness of X3SAT [14].

First, we translate 3CNFs $\varphi = \bigwedge_{i \in \{1,\ldots,m\}} c_i$ with $\text{vars}(\varphi) = \{x_1, \ldots, x_n\}$, to MinJX2SAT instances (φ', m) where m is the number of clauses in φ. Then we show that $\varphi \in$ X3SAT iff $(\varphi', m) \in$ MinJX2SAT

$\varphi' = e_{32c}(\varphi)$ is defined by encoding each clause $c_i = \{l_{i1} \vee l_{i2} \vee l_{i3}\}$ in φ by a conjunction of six clauses

$$\begin{aligned}
e_{32c}(c_i) = \ & (l_{i1} \vee y_{i1}) && (c_{i1}) \\
& \wedge(l_{i2} \vee \neg y_{i1}) && (c_{i2}) \\
& \wedge(l_{i2} \vee y_{i2}) && (c_{i3}) \\
& \wedge(l_{i3} \vee \neg y_{i2}) && (c_{i4}) \\
& \wedge(l_{i3} \vee y_{i3}) && (c_{i5}) \\
& \wedge(l_{i1} \vee \neg y_{i3}) && (c_{i6})
\end{aligned} \tag{9}$$

$e_{32c}(\varphi)$ contains $n+3m$ variables $\text{vars}(e_{32c}(\varphi)) = \{x_1, \ldots, x_n\} \cup \bigcup_{i \in \{1,\ldots,m\}} \{y_{i1}, y_{i2}, y_{i3}\}$. The auxiliary variables y_{ij} guarantee that in every assignment that satisfies $e_{32c}(\varphi)$, exactly one literal is true in each clause of φ.

For every variable x_i, we add the clause $x_i \vee \neg x_i$ to avoid assignments mapping x_i to J.

We get the transformation of X3SAT instances $\varphi = \bigwedge_{i \in \{1,\ldots,m\}} c_i$ to a MinJX2SAT instance $(e_{32c}(\varphi), m)$ where

$$e_{32c}(\varphi) = \bigwedge_{i \in \{1,\ldots,m\}} e_{32c}(c_i) \wedge \bigwedge_{i \in \{1,\ldots,n\}} (x_i \vee \neg x_i) \tag{10}$$

We show that an assignment β satisfies exactly one literal in each clause of φ iff there is an assignment β' with $|\beta'^{-1}(J)| = m$ mapping exactly one literal in each clause of $e_{32c}(\varphi)$ to 1 or J and the others to 0.

($\Rightarrow$) Let $\beta : \{x_1, \ldots, x_n\} \to \{0, 1\}$ be an assignment that satisfies exactly one literal in each clause $l_{i1} \vee l_{i2} \vee l_{i3}$ in φ.

We extend the assignment β to an assignment

$$\beta' : \{x_1, \ldots, x_n\} \cup \{y_{ij} \mid i \in \{1, \ldots, m\}, j \in \{1, 2, 3\}\} \to \{0, 1, J\}$$

with $|\beta'^{-1}(J)| = m$ and show that β' satisfies the encoded 2CNF defined in Equation 9.

For every clause c_i in φ, we define the undirected graph $G(c_i) = (V, E)$ where

$$\begin{aligned} V &= \{l_{i1}, l_{i2}, l_{i3}, y_{i1}, y_{i2}, y_{i3}, \neg y_{i1}, \neg y_{i2}, \neg y_{i3}\} \text{ and} \\ E &= \{\{y_{ij}, \neg y_{ij}\} \mid j \in \{1, 2, 3\}\} \\ &\quad \cup \{\{l_{ij}, l\} \mid (l_{ij} \vee l) \text{ is a clause in } e_{32c}(c_i)\}; \end{aligned}$$

In this graph we have that two neighbouring nodes are either in a JX2SAT-clause or a pair of negated literals. Now $G(c_i) = l_{i1} - y_{i1} - \neg y_{i1} - l_{i2} - y_{i2} - \neg y_{i2} - l_{i3} - y_{i3} - \neg y_{i3} - l_{i1}$ is a circle of length nine and thus has no 2-colouring using 0 and 1. Hence every satisfying assignment β' for φ' maps at least one literal in $e_{32c}(c)$ to J. Because β' satisfies $\bigwedge_{i \in \{1,\ldots,n\}} (x_i \vee \neg x_i)$, it maps each x_i to 0 or 1. Therefore, $\beta'(y_{ij}) = J$ for one or more variables among y_{i1}, y_{i2}, y_{i3}.

The following construction shows that it suffices to map exactly one of the y_{ij} to J for each clause $c_i = l_{i1} \vee l_{i3} \vee l_{i3}$ in φ. Let $\beta'(x_i) = \beta(x_i)$.

Without loss of generality, we can assume $\beta(l_{i1}) = 1$ and $\beta(l_{i2}) = \beta(l_{i3}) = 0$, because the semantic interpretation of $\vee$ is symmetric.

Since $\beta'(l_{i1}) = \beta(l_{i1}) = 1$, we have $\beta'(y_{i1}) = 0$ and $\beta'(y_{i3}) = 1$ to satisfy clauses c_{i1} and c_{i6} in Equation 9. Since $\beta'(l_{i2}) = \beta'(l_{i3}) = 0$, clauses c_{i2} and c_{i5} are satisfied by β' as well.

The remaining clauses c_{i3} and c_{i4} contain the variable y_{i2} as positive and negative literal, respectively. Because $\beta(l_{i2}) = \beta(l_{i3}) = 0$, neither y_{i2} nor $\neg y_{i2}$ can be mapped to 0 by a satisfying assignment. To satisfy both clauses, β' maps y_{i2} to J.

Applied to each clause in the original formula φ, this proves that β' satisfies $e_{32c}(\varphi)$ using exactly one joker for each clause of the original formula φ.

We have defined a construction of an exact satisfying assignment β' for $e_{32c}(\varphi)$, given an exact satisfying assignment β for φ. Hence we have shown that exact satisfiability of φ implies exact satisfiability of $e_{32c}(\varphi)$.

($\Leftarrow$) Let $\beta' : \{x_1, \ldots, x_n\} \cup \bigcup_{i \in \{1,\ldots,m\}} \{y_{i1}, y_{i2}, y_{i3}\} \to \{0, 1, J\}$ be an assignment that satisfies exactly one literal of each clause in $e_{32c}(\varphi)$.
We show that the restriction of β' to the domain $\{x_1, \ldots, x_n\}$ is an assignment $\beta : \{x_1, \ldots, x_n\} \to \{0, 1\}$ that maps exactly one literal of each clause $l_{i1} \vee l_{i2} \vee l_{i3}$ in φ to 1.

Because $e_{32c}(\varphi)$ contains a clause $x_i \vee \neg x_i$ for each original variable $x_i \in \{x_1, \ldots, x_n\}$, there is no $x_i \in \{x_1, \ldots, x_n\}$ such that $\beta(x_i) = \beta'(x_i) = J$.

For each clause $c_i = l_{i1} \vee l_{i2} \vee l_{i3}$ in φ we know that β' satisfies the encoding $e_{32c}(c_i)$ defined in Equation 9.

We show that β maps exactly one of the literals l_{ij} to 1 and the others to 0.

1. We show that β maps at least one of the literals l_{ij} to 1.
   Assume $\beta(l_{i1}) = \beta(l_{i2}) = \beta(l_{i3}) = 0$.
   Then all clauses $c_{i1}, \ldots, c_{i6}$ can only be satisfied if $\beta'(y_{i1}) = \beta'(y_{i2}) = \beta'(y_{i3}) = J$. But in this case, β' does not satisfy the condition $|\beta'^{-1}(J)| = m$ since the encoding of each clause requires at least one J.
   A contradiction to the assumption; thus β maps at least one of the literals l_{ij} to 1.
2. We show that β maps at most one of the literals l_{ij} to 1.
   Assume there are distinct k and k' in $\{1, 2, 3\}$ such that $\beta(l_{ik}) = \beta(l_{ik'}) = 1$. Without loss of generality, we can assume $k = 1$ and $k' = 2$, e.g., $\beta(l_{i1}) = \beta(l_{i2}) = 1$.
   Then $e_{32c}(c_i)$ contains the pair of clauses $(l_{i1} \vee y_{i1})$ and $(l_{i2} \vee \neg y_{i1})$. There is no value $\beta'(y_{i1}) \in \{0, 1, J\}$ such that exactly one literal in each clause of c_i is satisfied.
   A contradiction to the assumption; thus β maps at most one of the literals l_{ij} to 1.

We have shown that the restriction of β' to $\{x_1, \ldots, x_n\}$ satisfies exactly one literal in each clause of φ and therefore φ. Therefore exact satisfiability of $e_{32c}(\varphi)$ implies exact satisfiability of φ, and we have shown both directions of
$\varphi \in$ X3SAT iff $e_{32c}(\varphi) \in$ MinJX2SAT. □

6 The Decision Problem JX3SAT

Instance: 3CNF φ
Question: Is there an assignment $\beta : \mathrm{vars}(\varphi) \to \{0, 1, J\}$ such that β maps exactly one literal in each clause of φ to 1 or J and all other literals to 0?

Theorem 7. *The decision problem* JX3SAT *is* ***NP****-hard.*

Proof. We show this by reduction of X3SAT to JX3SAT. Then JX3SAT is **NP**-hard because X3SAT is **NP**-hard [14].

X3SAT instances φ are translated to JX3SAT instances φ' by adding clauses $x_i \vee \neg x_i$ for each variable occurring in φ. This translates each 3CNF φ with n variables and m clauses to a 3CNF

$$\varphi' = \varphi \wedge \bigwedge_{x_i \in \text{vars}(\varphi)} (x_i \vee \neg x_i)$$

with n variables and $m + n$ clauses. We show that $\varphi \in$ X3SAT iff $\varphi' \in$ JX3SAT.

($\Rightarrow$) Assume $\beta : \{x_1, \ldots, x_n\} \to \{0, 1\}$ satisfies exactly one literal in each clause of φ. Because β maps every variable x_i to 0 or 1, it satisfies exactly one of the literals in $x_i \vee \neg x_i$ for each variable x_i. Therefore it satisfies exactly one literal in each clause of φ'.

($\Leftarrow$) Assume $\beta' : \{x_1, \ldots, x_n\} \to \{0, 1, J\}$ maps exactly one literal in each clause of φ' to 1 or J. φ' contains a clause $x_i \vee \neg x_i$ for each variable x_i and β' satisfies φ', e.g. β' maps exactly one literal in each clause of φ to 1 or J.

Assume $\beta'(x_i) = J$ for a variable $x_i \in \text{vars}(\varphi)$. Then $\beta'(\neg x_i) = J$ and β' satisfies more than one literal in $x_i \vee \neg x_i$. Therefore β' maps every variable x_i to a value in $\{0, 1\}$ and satisfies exactly one literal in each clause of φ. $\square$

7 An Exponential Algorithm to solve MinJXSAT

Earlier, we defined the optimisation problem MinJX2SAT and showed that it is **NP**-hard. Similarly, we define the **NP**-hard optimisation problem MinJXSAT.

Instance: (φ, h), where φ is in CNF and $h \in \mathbb{N}$.
Question: Is there an assignment $\beta : \text{vars}(\varphi) \to \{0, 1, J\}$ such that $|\beta^{-1}(J)| = h$ and β maps exactly one literal in each clause of φ to 1 or J and the other literals in the clause to 0?

Note that since the MinJXSAT problem is more general than MinJX2SAT, having an algorithm to solve it would also mean having an algorithm to solve MinJXSAT. Here, we first reduce the problem to the MinXSAT problem, where it is also known to be **NP**-hard. Before presenting the reduction, we define the MinXSAT problem.

Instance: (φ, ω), where φ is in CNF and ω are weights assigned to variables when they take on certain values.
Question: Is there an assignment satisfying φ such that exactly one literal in each clause is true and the rest are false? In addition, the weights obtained in this assignment is the least among all other such satisfying assignments for this instance.

Theorem 8. MinJXSAT *instances can be reduced to* MinXSAT *instances, using at most the same number of variables and clauses.*

Proof. Consider a MinJXSAT instance P having variables $x_1, \ldots, x_n$. Now, construct a MinXSAT instance G_0 as follows:

(a) Variables of G_0 are u_i, v_i, w_i for $i \in \{1, 2, \ldots, n\}$, where the intended meaning is the following: $u_i = 1$ iff $x_i = J$; $v_i = 1$ iff $x_i = 0$; $w_i = 1$ iff $x_i = 1$.

(b) Clauses of G_0 are given by (b.1) and (b.2).

(b.1) Clauses $(u_i \vee v_i \vee w_i)$, for $i = 1, \ldots, n$ Note that this ensures exactly one of u_i, v_i, w_i is one. Thus, we can have our intended interpretation of u_i, v_i, w_i with respect to the values of x_i.

(b.2) For each clause $(x_{i_1} \vee x_{i_2} \vee \ldots \vee x_{i_k} \vee \neg x_{j_1} \vee \neg x_{j_2} + \ldots \vee \neg x_{j_r})$ in P, the following clause is in G_0:
$(\neg v_{i_1} \vee \neg v_{i_2} \vee \ldots \vee \neg v_{i_k} \vee \neg w_{j_1} \vee \neg w_{j_2} \vee \ldots \vee \neg w_{j_r})$.

(c) Weights of u_i are 1 and weights of v_i, w_i are 0.

Note that if exactly one of $x_{i_1}, x_{i_2}, \ldots, x_{i_k}, \neg x_{j_1}, \neg x_{j_2}, \ldots, \neg x_{j_r}$ is non-zero then using the values for u_i, v_i, w_i as in (a), exactly one of $\neg v_{i_1}, \neg v_{i_2}, \ldots, \neg v_{i_k}, \neg w_{j_1}, \neg w_{j_2}, \ldots, \neg w_{j_r}$ is non-zero. Similarly, if exactly one of $\neg v_{i_1}, \neg v_{i_2}, \ldots, \neg v_{i_k}, \neg w_{j_1}, \neg w_{j_2}, \ldots, \neg w_{j_r}$ is non-zero then using the substitution of values as in (a), exactly one of $x_{i_1}, x_{i_2}, \ldots, x_{i_k}, \neg x_{j_1}, \neg x_{j_2}, \ldots, \neg x_{j_r}$ is non-zero.

Thus, any solution for G_0 as a XSAT instance with weight w gives a solution for P as a JXSAT instance using w jokers (using the interpretation in (a)). Similarly, any solution for P as an JXSAT instance using w jokers gives a solution for G_0 using weight w.

Note that G_0 has $3n$ variables. Below we will progressively construct $G_1, G_2, \ldots, G_n$ to reduce the number of variables. The following invariants are maintained for $k = 0, 1, \ldots, n$:

(I): In G_k, for $i = 1, \ldots, k$, at most one of u_i, v_i, w_i appears in any clause. If u_i does not appear: then any solution for G_k is a solution for G_{k-1} by fixing $u_k = 0$ and $v_i = \neg w_i$. Furthermore, the minimal weight solution for G_{k-1} has $u_i = 0$, and any solution for G_{k-1} with $u_i = 0$ is also a solution for G_k.

(II): Except for the clauses $(u_i \vee v_i \vee w_i)$, $k < i \leq n$, the variables v_j, w_j only appear as $\neg v_j, \neg w_j$ in the clauses in G_k. In G_k, for $i > k$, $u_i, \neg u_i$, only appear in the clause $(u_i \vee v_i \vee w_i)$.

(III): If $k > 0$, the number of clauses in G_k is at most the number of clauses in G_{k-1}. The variables used in G_k are a subset of the variables used in G_{k-1}.

We now describe how to obtain G_k from G_{k-1} for $k = 1, \ldots, n$:

(1) Suppose all the clauses which have $\neg v_k$ or $\neg w_k$ in them are $\neg v_k \vee \alpha_i$ for $i \in \{1, \ldots, s\}$ and $\neg w_k \vee \beta_j$ for $j \in \{1, \ldots, r\}$, where α_i, β_j are disjunctive formula over variables.

Without loss of generality assume that no clause has both $\neg v_k$ and $\neg w_k$ (as otherwise, for the clause $(\neg v_k \vee \neg w_k \vee \alpha)$ along with $(u_k \vee v_k \vee w_k)$ we trivially have that exactly one v_k and w_k is 1 and the other 0; thus u_k must be 0 and all literals in α must be 0; we can thus simplify the clauses in G_{k-1}, and drop the clause $(\neg v_k \vee \neg w_k \vee \alpha)$).

(2) Case 1: $s > 0$ and $r > 0$, then form G_k from G_{k-1} as follows:
   (a) Drop $\neg v_k \vee \alpha_i$ for $i \in \{1, \ldots, s\}$,
   (b) Drop $\neg w_k \vee \beta_j$ for $j \in \{1, \ldots, r\}$,
   (c) Drop $(u_k \vee v_k \vee w_k)$,
   (d) Add $(\alpha_i \vee u_k \vee \beta_1)$ for $i \in \{1, \ldots, s\}$,
   (e) Add $(\alpha_1 \vee u_k \vee \beta_j)$ for $j \in \{1, \ldots, r\}$.

   Note that the old clauses of type (a), (b) and (c) imply $v_k \equiv \alpha_1 \equiv \ldots \equiv \alpha_s$ and $w_k \equiv \beta_1 \equiv \ldots \equiv \beta_r$ and that the new clauses of types (d) and (e) imply $\alpha_1 \equiv \alpha_2 \equiv \ldots \equiv \alpha_s$ and $\beta_1 \equiv \beta_2 \equiv \ldots \equiv \beta_r$. The above modification removes the variables v_k and w_k. Each solution for G_{k-1} is also a solution for G_k as we must have $v_k \equiv \alpha_i$, $w_k \equiv \beta_j$ in G_{k-1}. Furthermore, any solution for G_k gives a solution for G_{k-1}, by taking $v_k \equiv \alpha_1$, $w_k \equiv \beta_1$ in G_{k-1}.

(3) Case 2: At most one of s, r is 0.
   Then, fix $u_k = 0$ and let $v_k \equiv \neg w_k$, and drop the clause $(u_k \vee v_k \vee w_k)$ from G_{k-1} to form G_k.
   Note that any solution of G_k is also a solution for G_{k-1} by taking $u_k = 0$ and $v_k \equiv \neg w_k$. Also, any solution for G_{k-1} with $u_k = 0$ is also a solution for G_k. Note here that in the minimal weight solution of G_{k-1}, u_k must be 0.

From the above analysis, it follows that MinXSAT solution for G_n gives a MinJXSAT solution for the P and vice versa. ☐

Porschen [10] provided an algorithm which solves variable-weighted XSAT in time $O(1.185^n)$. Although Porschen and Pagge [11] provided a faster algorithm for the special case of variable-weighted X3SAT, this algorithm cannot be used above, as the elimination of variables increases the clause-length and therefore also JX2SAT and JX3SAT are only translated to weighted XSAT. For Corollary 9 (b), note that one can use the elimination procedure of the previous theorem to eliminate from any XSAT-instance variables occurring both positive and negative without increasing the number of clauses and then use Proposition 1.

Corollary 9. (a) MinJXSAT *can be solved in time* $O(1.185^n)$.
(b) JXSAT *has the same time complexity as* XSAT *when measured in dependence of the number of variables and the number of clauses.*

8 Future work and Summary

In this paper, we introduced an additional truth value into our framework and study their structural properties with regards to known existing variants of Boolean satisfiability problems. In fact, our results show that their structural properties remain the same. The decision problem JX2SAT is shown to be in the class **P** by reducing it to 2SAT. The decision problem JX3SAT and the optimisation problem MinJX2SAT are both shown to be **NP**-hard. This was done by giving a reduction from X3SAT to the respective problems. Finally, we also gave an exponential algorithm to solve MinJXSAT in $O(1.185^n)$.

Here are some possible direction for future work. We have just studied how it relates to well-studied classes like **P** and **NP**. One question might be whether probabilistic algorithms can accomodate jokers in the same way as the deterministic weighted XSAT algorithm. Furthermore, it remains open whether one can find a translation from JX2SAT or JX3SAT into weighted X3SAT without an increase of variables, which might improve on the complexity bound for the corresponding algorithm. The current method does not do this.

References

1. Rafael Accorsi. *MAX2SAT is NP-complete.* WINS-ILLC, University of Amsterdam, 1999.
2. Hubie Chen. A Rendezvous of Logic, Complexity, and Algebra SIGACT News (Logic Column). ACM, 2006.
3. Serge Gaspers and Gregory B. Sorkin. Separate, measure and conquer: faster polynomial-space algorithms for Max 2-CSP and counting dominating sets. *ACM Transactions on Algorithms (TALG)*, 13(4):44:1–36, 2017.
4. Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms.* Springer, October 2010.
5. Richard M. Karp. Reducibility among combinatorial problems. *Complexity of computer computations*, pp. 85–103. Springer, Boston, MA, 1972.
6. Arist Kojevnikov and Alexander S. Kulikov. A new approach to proving upper bounds for MAX-2-SAT. *Proceedings of the seventeenth Annual ACM–SIAM Symposium on Discrete Algorithms*, SODA 2006, ACM, pages 11–17, 2006.
7. Frédéric Lardeux, Frédéric Saubion and Jin-Kao Hao. Three truth values for the SAT and MAX-SAT problems. *International Joint Conference on Artificial Intelligence*, IJCAI 2005, vol. 19, p. 187, Lawrence Erlbaum Associates Ltd, 2005.
8. Rolf Niedermeier and Peter Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, 36:63–88, 2000.
9. Christos Harilaos Papadimitriou. *Computational Complexity.* Addison-Wesley, USA, 1994.
10. Stefan Porschen. *On variable-weighted exact satisfiability problems.* Annals of Mathematics and Artificial Intelligence, 51(1):27–54, 2007.
11. Stefan Porschen and Galyna Plagge. Minimizing variable-weighted X3SAT. *Proceedings of the International Multiconference of Engineers and Computer Scientists*, IMECS 2010, 17–19 March 2010, Hongkong, Volume 1, pages 449–454, 2010.
12. Stefan Porschen and Tatjana Schmidt. On some SAT-variants over linear formulas. In International Conference on Current Trends in Theory and Practice of Computer Science (pp. 449-460). Springer, Berlin, Heidelberg, 2009.
13. Daniel Raible and Henning Fernau. A new upper bound for Max-2-SAT: A graph-theoretic approach. *Proceedings of the Thirty-Third International Symposium on Mathematical Foundations of Computer Science* (MFCS 2008), Springer Lecture Notes in Computer Science, 5162:551–562, 2008.
14. Thomas J. Schaefer. The Complexity of Satisfiability Problems. *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC 1978, pages 216–226, 1978.