# Characterizing Boolean Satisfiability Variants

by

Ivan Tadeu Ferreira Antunes Filho

B.S. Computer Science and Electrical Engineering, and Mathematics, 2018

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 18 2019

Author . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Department of Electrical Engineering and Computer Science

Aug 23, 2019

Certified by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Erik Demaine

Professor

Thesis Supervisor

Accepted by . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

Katrina LaCurts

Master of Engineering Thesis Committee

THIS PAGE INTENTIONALLY LEFT BLANK

# Characterizing Boolean Satisfiability Variants

by

Ivan Tadeu Ferreira Antunes Filho

Submitted to the Department of Electrical Engineering and Computer Science
on Aug 23, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

## Abstract

We survey variants of the BOOLEAN SATISFIABILITY problem from over the years and organize them in what we believe to be the most comprehensive list of known results in SAT variants. We propose a new notation to specify them so that the problems can be compared with no ambiguities, and so that new problems can be more easily identified. We also show hardness of many new variants of SAT including a characterization of $S$-IN-$k$ SAT, hardness of variants of MAX PLANAR SAT, partial characterization of XNF SAT, consequences of the Ordered Planar Dichotomy, and hardness of NAE SAT variants with a bounded number of variable occurrences, in particular NAE E$k$SAT-$k$.

This is joint work with Aviv Adler, Leo Alcock, Anastasiia Alokhina, Joshua Ani, Jeffrey Bosboom, Erik Demaine, Yevhenii Diomidov, Jonathan Gabor, Yuzhou Gu, Linus Hamilton, Mirai Ikebuchi, Adam Hesterberg, Jayson Lynch, Zhezheng Luo, Xiao Mao, Kevin Sun, John Urschel, Yinzhan Xu, and Lillian Zhang.

Thesis Supervisor: Erik Demaine
Title: Professor

## Acknowledgments

I would like to thank Anne Hunter, MIT EECS' Undergraduate Academic Administrator, for her wonderful help with logistical issues, and support during stressful times; my thesis advisor, Prof. Erik Demaine, for introducing me to the world of hardness proofs, for helping me find a research topic about which I was really excited about, for all the encouragement, and for all the help with editing this thesis; and Jason Ku, with whom I worked as a TA during most of my M.Eng. years and who helped me discover my love for teaching.

I would like to thank Jayson Lynch, who helped me review and improve multiple parts of my thesis; Adam Hesterberg, for reviewing large portions of the thesis for typos and grammar mistakes and contextual errors; Jeffrey Bosboom, for finding some major errors in the original version of the proofs as well as helping me improve them; Jabari King, for helping me improve the organization and appearance of images and tables; David Kaufman and Hengameh Bagherian, for helping me review parts of my definitions.

I would like to thank the participants of the open problem sessions of the MIT class 6.892 Algorithmic Lower Bounds: Fun with Hardness Proofs, held in spring 2019.

Finally, a huge thanks to my friends and family for all of their support throughout the years!

# Contents

# List of Figures

# Chapter 1

# Introduction

Cook [Coo71], in his 1971 paper, and Levin [Lev73], in his 1973 paper, independently proved the existence of NP-complete problems, and, in particular, Cook showed that Boolean satisfiability (SAT), the problem of deciding if a Boolean formula is satisfiable, was one. Within a short time, SAT proved useful for reductions: in 1972, Karp [Kar72] showed that 0-1 Integer Programming, Clique, Set Cover, Directed Hamiltonian Cycle, Undirected Hamiltonian Cycle, Exact Cover, and among other problems, were NP-complete by reductions from SAT. Since then, extensive work in Boolean satisfiability has been done, and SAT variants have stayed a very important tool to analyze the complexity of other problems. Among some more recent results, $k$ 1s Positive Not-1-in-EU3SAT$_{\text{TS}}$ (see Chapter 2 for definitions) was used to analyze the complexity of $H$-free Edge Deletion [KW13]; and ∃∃! SAT to analyze Unique List Colorability [Mar08]. Other variants were used to analyze Defining Sets in Vertex Coloring [HM05], or even prove the complexity of multiple videogames [Alo+14]. One could say that SAT is the most useful problem to analyze the complexity of other problems. Figure 1-1 shows a timeline overview of the SAT problems that were studied throughout the years.

Over time, however, the terminology and notation used to define such problems has not stayed constant. As an example, consider the term Planar. Lichtenstein [Lic82], in 1982, defined the graph of Planar SAT instances to be the bipartite incidence graph of clauses and variables, together with a Hamiltonian cycle through the variables, but Mansfield [Man83], in 1983, removed the variable cycle requirement when describing Planar SAT instances. Tippenhauer [Tip16], in 2016, defined Planar SAT to be a bipartite graph of literals and clauses, instead of variables and clauses, while in the same year Kazda et al. [KKR16] defined it to be a graph whose faces were clauses, and where the order of variables in clauses mattered. Other terms have had similar fates. Even classic problems, such as 1-in-3SAT, can sometimes refer to SAT problems where all the variables are forced to be positive and clauses contain exactly 3 variables, or it can refer to SAT problems where the variables can be either positive or negative, and the clauses contain 3 or fewer variables. This can make it confusing to compare different results.

To solve this, in this thesis we try to summarize the results on Boolean satisfiability from the past decades with consistent notation. Specifically, in Chapter 2, we present in Table 2.1 what we believe to be the most comprehensive list of known results in SAT variants. We focused mainly on results related to NP-completeness, P, and PSPACE, though the survey also covers some problems in different complexity classes. Related work has been done by Tippenhauer [Tip16], who surveyed Planar SAT problems; by Schaefer and Umans [SU02], who surveyed logic problems in the polynomial hierarchy; by Creignou, Khanna and

Sudan [CKS01], who worked in developing a framework for classifying the complexity of Boolean satisfiability; and by Schmidt [Sch10], who proposed a notation to specify constraints in formula length and in occurrences of variables in SAT problems.

In addition to the survey, we prove new results of our own. Among our new results are:

- **Characterization of $S$-in-E$k$SAT**: Given a conjunction of clauses each with $k$ variables, each of which is satisfied if the number of TRUE literals in the clause is in $S$, where $S$ is a subset of $\{0, 1, \ldots, k\}$. We characterize the NP-hardness for all $k \geq 3$.

- **NAE E$k$SAT-$k$ is in P**. We are given a conjunction of NAE clauses where each clause contains exactly $k$ literals and each variable occurs at most $k$ times. We reduce this problem to detecting minimally non 2-colorable hypergraphs that contain the same number of hyperedges and nodes, which is in P.

- **Results on Planar Max SAT**: We are given a value $k$, a conjunction of clauses, and a planar bipartite incidence graph of clauses and variables and the instance is satisfiable if at least $k$ clauses can be simultaneously satisfied. We show that multiple variants of PLANAR MAX SAT are NP-complete, and that their counting versions are #P complete, by providing reductions from PLANAR MAX 2SAT, and then from PLANAR EU3SAT to PLANAR MAX 2SAT .

We organize the thesis as follows. In Chapter 2, we present our survey results and define the notation used both on the survey and on the rest of the thesis. In Chapter 3, we present our new results on NAE SAT variants. In Chapter 4, we present results on $S$-IN-$k$SAT variants. In Chapter 5, we present results on MAX SAT variants, most of them in PLANAR MAX SAT. In Chapter 6, we present results on different variants of SAT that did not fit the other categories, such as XNF SAT. At last, in Chapter 7, we talk about open problems. Given the large number of terms defined in this thesis, we include at the end an index of terms to facilitate reading.

This work was initiated during an open problem session of the MIT class 6.892 Algorithmic Lower Bounds: Fun with Hardness Proofs, held in Spring 2019. This is joint work with Aviv Adler, Leo Alcock, Anastasiia Alokhina, Joshua Ani, Jeffrey Bosboom, Erik Demaine, Yevhenii Diomidov, Jonathan Gabor, Yuzhou Gu, Linus Hamilton, Mirai Ikebuchi, Adam Hesterberg, Jayson Lynch, Zhezheng Luo, Xiao Mao, Kevin Sun, John Urschel, Yinzhan Xu, and Lillian Zhang.

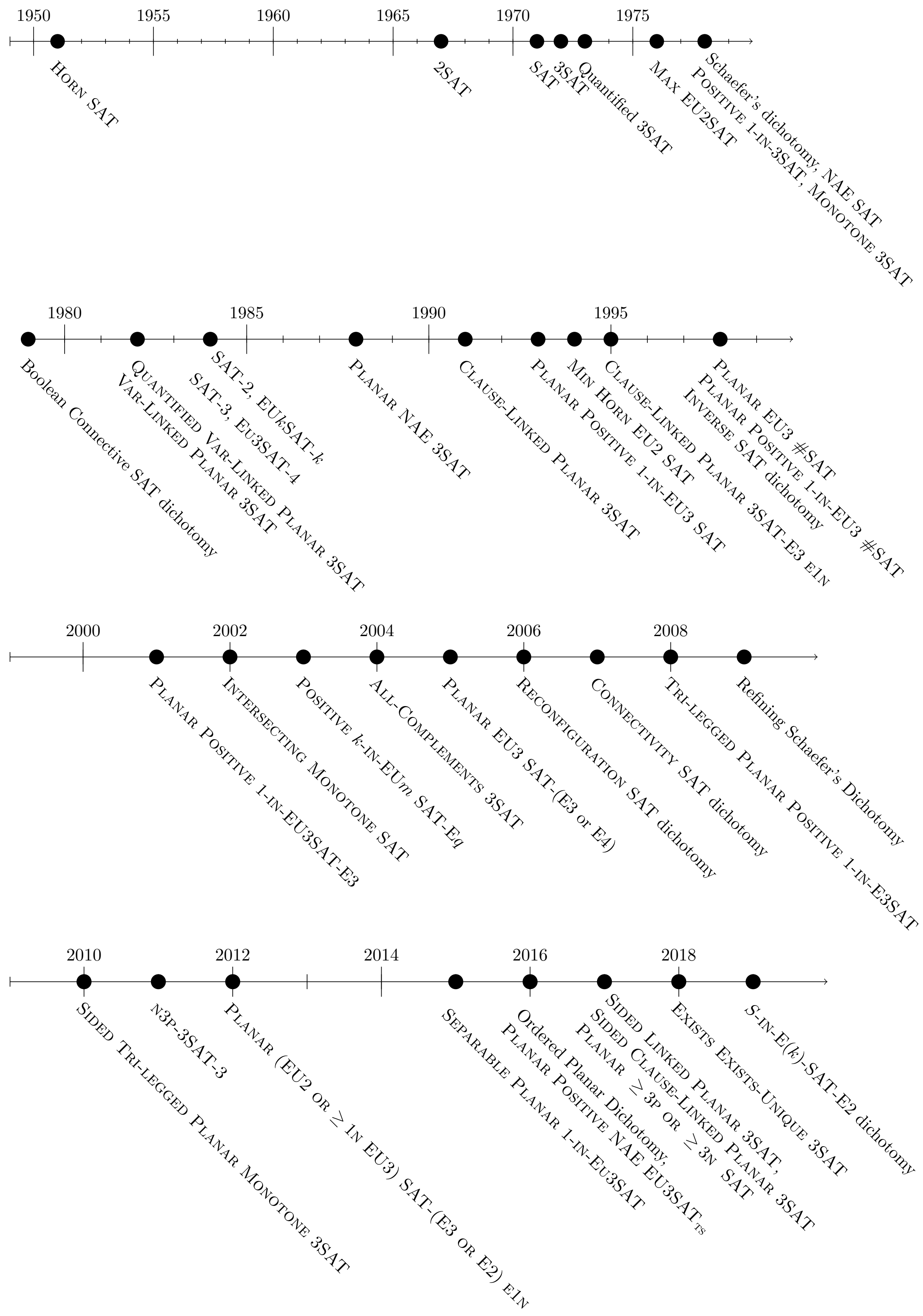


Figure 1-1: Timeline overview of SAT problems.

# Chapter 2

# Survey of SAT variants

In this chapter we first present our survey, followed by our definitions, and then show how it differs from previous notation and terminology. Section 2.1 shows a summary of both past work as well as the results presented in this thesis. Section 2.2 first explains how the name of the multiple SAT variants are specified, followed by the meaning of the multiple terms used when specifying it. It also introduces the terms that are used on our proofs. At last, Section 2.3 gives specific examples on how our notation and terminology differs from some previous papers, to exemplify how our notation clears ambiguities and to make it easier for someone knowledgeable of those problems to understand our new definitions.

## 2.1 Survey Table

Table 2.1: Summary of the complexity of SAT problems. A star ($*$) denotes a result obtained in this thesis. A star with a question mark ($*$?) is a result that seemed to be assumed to be known in the literature, but that we found no paper proving. We provide proofs of the latter results here. To keep the list shorter, if the same paper proved results on multiple new constrained versions of a SAT problem, we just include the most constrained one, since it implies the other results.

| SAT type | Complexity |
|---|---|
| Classic SAT problems (pre 1980) | |
| SAT | NP-complete [Coo71; Lev73] |
| CNF-SAT | NP-complete [Coo71] |
| NOT-TAUTOLOGY | NP-complete [Coo71] |
| 2SAT | P [Kro67] NL-complete [Pap94] |
| 3SAT | NP-complete [Kar72] |
| Monotone 3SAT | NP-complete [Gol78] |
| Quantified 3SAT | PSPACE-complete [SM73] |
| Positive 1-in-U3 SAT | NP-complete [Sch78] |
| Positive Not-1-in-U3 SAT$_t$ | P-complete [Sch78] |
| NAE 3SAT | NP-complete [Sch78] |

Continued on next page

Continued from previous page

| SAT type | Complexity |
|---|---|
| Horn SAT | P [Hor51] P-complete [Pap94] |
| Dual Horn SAT | P-complete [Sch78] |
| Max EU2SAT | NP-complete [GJS76] |
| $S$ SAT (Set of all satisfiable $S$-formulas) | Schaefer's dichotomy [Sch78] [1] |
| Every relation in $S$ is satisfied when all variables are 0 | P |
| Every relation in $S$ is satisfied when all variables are 1. | P |
| Every relation in $S$ is Horn. | P |
| Every relation in $S$ is Dual-Horn. | P |
| Every relation in $S$ is affine. | P |
| Every relation in $S$ is bijunctive. | P |
| If none of the above. | NP-complete |
| $S$ SAT$_c$ (Satisfiable $S$-formulas with one 0, and one 1 variables) | Schaefer's dichotomy [Sch78] |
| Every relation in $S$ is satisfied when all variables are 0. | P |
| Every relation in $S$ is satisfied when all variables are 1. | P |
| Every relation in $S$ is Horn. | P |
| Every relation in $S$ is Dual-Horn. | P |
| If none of the above. | NP-complete |
| Quantified $S$ SAT$_c$ ($S$ QSAT$_c$) | Schaefer's dichotomy [Sch78] |
| Every relation in $S$ is Horn. | P |
| Every relation in $S$ is Dual-Horn. | P |
| Every relation in $S$ is affine. | P |
| Every relation in $S$ is bijunctive. | P |
| If none of the above. | PSPACE complete |
| $S$ SAT, where $S$ is a finite set of Boolean connectives | Dichotomy [Lew79] |
| There is an $S$-formula equivalent to $X \wedge \neg Y$ | NP-complete |
| Otherwise | P |
| Circuit SAT | NP-complete [Lad75] |
| Circuit Monotone SAT | NP-complete [Gol77] |
| Planar Circuit SAT | NP-complete [Gol77] |
| $(2k)$-Quantified DNF SAT | $\Sigma_k^P$-complete [Wra76; Sto76] |
| $(2k+1)$-Quantified CNF SAT | $\Sigma_k^P$-complete [Wra76; Sto76] |
| $2k$-Quantified CNF SAT, with $\forall$ and $\exists$ switched | $\Pi_k^P$-complete [Wra76; Sto76] |
| $2k+1$-Quantified DNF SAT, with $\forall$ and $\exists$ switched | $\Pi_k^P$-complete [Wra76; Sto76] |
| Planar SAT | |
| Planar ($\geq$ 3p or $\geq$ 3n) SAT | Always Satisfiable ($O(n^2)$) [Pil17] |
| Planar $(\geq 4) SAT$ | Always Satisfiable ($O(n^{1.5})$) [Pil17] |

Continued on next page

[1] "Deciding which side of the dichotomy a SAT problem falls in is itself NP-hard, when the relations are given in CNF (Theorem 6.5.1). If the relations are given as a set of tuples specifying the satisfiable assignment of literals, then, it is in P [Che09].

Continued from previous page

| SAT type | Complexity |
|---|---|
| PLANAR EU3SAT | NP-complete [Man83], #P-complete [III+98] |
| PLANAR (EU2 OR EU3) SAT-3 E1N $\geq$ 1P | NP-complete [KMT10] |
| PLANAR (EU2 OR $\geq$ 1N EU3) SAT-(E3 OR E2) E1N | NP-complete [Día+12] |
| PLANAR EU3 SAT-4 | NP-complete [JM95] |
| PLANAR EU3 SAT-(E3 OR E4) | NP-complete [LMM05] |
| PLANAR MONOTONE E3SAT-E4 | NP-complete [DDD16] |
| PLANAR MONOTONE (EU2 OR EU3)SAT-E3 | NP-complete [DDD16] |
| PLANAR MONOTONE ((E2N EU2) OR (E3P EU3))SAT-E4 E1N | NP-complete [DDD16] |
| 2-CONNECTED PLANAR MONOTONE EU3SAT-E5 | NP-complete [DDD16] |
| 3-CONNECTED PLANAR EU3SAT-4 | NP-complete [Kra91] |
| PLANAR 1-IN-EU3SAT | NP-complete [DF86] |
| SEPARABLE PLANAR 1-IN-EU3SAT | NP-complete [Wu15] |
| PLANAR POSITIVE 1-IN-EU3 SAT | NP-complete [Lar93], #P complete [III+98] |
| PLANAR POSITIVE 1-IN-EU3SAT-E3 | NP-complete [MR01] |
| PLANAR POSITIVE 1-IN-EU3SAT-E3 is ASP-hard? | Open |
| PLANAR $\{0, 1, k-1, k\}$-IN-EU$k$SAT | P. Thm:6.7.9 $*$ |
| PLANAR NAE 3SAT | P [Mor88] |
| SEPARABLE PLANAR NAE 3SAT | P [Mor88] |
| PLANAR NAE SAT | P [Alo+09] |
| PLANAR (NAE OR ALL EQUAL) SAT | P Thm:6.7.8$*$ |
| PLANAR NAE 3SAT$_{\text{TS}}$ | NP [Deh15] |
| PLANAR POSITIVE NAE EU3SAT$_{\text{TS}}$ | NP-complete [Deh16] |
| $\forall\exists$PLANAR EU3SAT-3 | $\Pi_2^P$ [Gut96] |
| ORDERED PLANAR ($S$) SAT | Ordered Planar Dichotomy [KKR16] |
| $S$ SAT is in P | P |
| $S$ only contains self-complementary relations such that $dR$ is an even $\Delta$-matroid | P |
| Otherwise | NP-complete |
| LINKED PLANAR SAT | |
| VAR-LINKED PLANAR 3SAT | NP-complete [Lic82] |
| VAR-LINKED SEPARABLE PLANAR 3SAT | NP-complete [Lic82] |
| VAR-LINKED PLANAR EU3SAT | NP-complete [Pil17] |
| VAR-LINKED PLANAR U3SAT-E3 E2P E1N | NP-complete [MG08] |
| VAR-LINKED PLANAR POSITIVE 1-IN-EU3SAT | NP-complete [MR08] |
| QUANTIFIED VAR-LINKED PLANAR 3SAT | PSPACE-complete [Lic82] |
| SIDED VAR-LINKED PLANAR MONOTONE 3SAT | NP-complete [BK10] |
| TRI-LEGGED PLANAR POSITIVE 1-IN-E3SAT | NP-complete [MR08] |

Continued on next page

Continued from previous page

| SAT type | Complexity |
|---|---|
| Tri-legged Planar $S$ 3SAT | Equivalent to Var-Linked Planar $S$ 3SAT [Tip16] |
| Sided Tri-legged Planar $S$ 3SAT | Equivalent to Sided Var-Linked Planar $S$ 3SAT. Thm:6.8.1 * |
| Clause-Linked Planar 3SAT | NP-complete [KLN91] |
| Clause-Linked Planar EU3SAT | NP-complete [Pil17] |
| Clause-Linked Planar Positive 1-in-EU3SAT | NP-complete [Cha+16] |
| Clause-Linked Planar Monotone 3SAT | NP-complete [Pil17] |
| Clause-Linked Planar (e3p EU3 or U2)SAT-E3 e1n | NP-complete [Fel+95] |
| Sided Clause-Linked Planar 3SAT | NP-complete [Pil17] |
| Quantified Clause-Linked Planar 3SAT | Open |
| Linked Planar 3SAT | NP/ASP/#P-complete [Pil17] |
| Linked Planar EU3SAT | NP-complete [Pil17] |
| Linked Planar Positive 1-in-E3SAT | NP-complete [Pil17] |
| Linked Planar Monotone 3SAT | NP-complete [Pil17] |
| Sided Linked Planar 3SAT | NP/ASP/#P-complete [Pil17] |
| Quantified Linked Planar 3SAT | PSPACE-complete. Thm:6.3.1 $*$ |
| Max Planar SAT | |
| Max Planar 2SAT | NP/ASP/#P-complete. Thm:5.2.3 $*$ |
| Min Planar EU2SAT | NP-complete. Thm:5.2.5 |
| Max Planar Xor E2SAT | P. Thm:5.3.1 $*$ |
| Max Planar Xnor E2SAT | P. Corollary:5.3.2 $*$ |
| Max Planar Xor 2SAT | NP/ASP/#P-complete. Thm:5.4.1 $*$ |
| Max Planar Xnor 2SAT | NP/ASP/#P-complete. Corollary:5.4.2 $*$ |
| Max Planar Xor EU3SAT | NP/ASP/#P-complete. Thm:5.5.1 $*$ |
| Max Planar Xnor EU3SAT | NP/ASP/#P-complete. Corollary:5.5.2 $*$ |
| Max Planar Horn 2SAT | NP/ASP/#P-complete. Thm:5.6.1 $*$ |
| Max Planar Dual Horn 2SAT | NP/ASP/#P-complete. Corollary:5.6.2 $*$ |
| Max SAT | |
| Max $S$ SAT, where $S$ is finite | Dichotomy [Cre95]. Either MAXSNP-complete, or in P |
| Max 1-in-$k$SAT-2 | P. Corollary:4.1.2 $*$ |
| Max All Equal EU3SAT | NP-complete. Thm:5.1.1 $*$ |
| Max 2SAT-5 | MAXSNP-complete [Pap94] |
| Max 3SAT-3 | MAXSNP-complete [Pap94] |
| Max NAE SAT | MAXSNP-complete [Pap94] |
| Quantified Max 3SAT. Approximate within ratio $\epsilon \in (0, 1)$ | PSPACE-complete [Con+97] |
| Min Horn EU2SAT | NP-complete [KKM94] |

Continued on next page

Continued from previous page

| SAT type | Complexity |
|---|---|
| MINMAX E3SAT . Approximate within ratio $0 < \epsilon < 1$ | [HRT07] |
| Each existential variable occurs at most 3 times, and universal variables occurs at most 2. | $\Pi_2^P$-complete. |
| Each existential variable occurs at most 2 times. | coNP. |
| Each universal variable occurs at most 1 time each. | NP. |
| ALMOST PLANAR SAT | |
| BOUNDED-GENUS ORIENTABLE SURFACE NAE 3SAT | P (FPT in genus). Thm:3.9.1 $*$ |
| 1-PLANAR POSITIVE NAE 3SAT | NP-complete. Thm:3.6.1$*$ |
| CROSSING NUMBER $n^\epsilon$ NAE 3SAT | NP-complete. Thm:3.7.1 $*$ |
| CROSSING NUMBER $O(\log n)$ NAE 3SAT | P Thm:3.8.1 * |
| CNF SAT | |
| $k$-SAT-2, for all $k$ | P [Tov84] |
| 3SAT-3 | NP-complete [Tov84] |
| N3P-3SAT-3 | NP-complete [DFZ11] |
| MONOTONE 3SAT-3 | NP-complete. Thm:6.2.1 $*$? |
| EU3SAT-4 | NP-complete [Tov84] |
| MONOTONE E3SAT-4 | NP-complete [DD16] |
| EU$k$SAT-$k$ | P [Tov84] |
| NAE SAT | |
| NAE SAT-2 | P. Thm:3.1.1 $*$ |
| NAE-3SAT$_c$ | ASP-complete [BLS12] |
| POSITIVE NAE 3SAT-3 | NP. Thm:3.4.1 $*$? |
| NAE E$k$SAT-$k$ | P. Thm:3.3.1 $*$ |
| NAE E3SAT-3 + 1 extra clause | Open |
| NAE E3SAT$_C$-3 | Open |
| POSITIVE NAE E3SAT-4 | NP. Thm:3.5.1 $*$ |
| POSITIVE NAE EU$k$SAT$\frac{2^{k-3}}{k}$ | Always Satisfiable. Thm:3.2.2 $*$ |
| POSITIVE NAE EU$k$SAT-E$k$ | Always Satisfiable. Thm:3.2.1 $*$ |
| $\forall\exists$ NAE 3SAT | $\Pi_2^P$ [EG96] |
| $S$-IN-$k$ SAT | |
| 1-IN-$k$SAT-2 | P. Thm:4.1.1 $*$ |
| POSITIVE $k$-IN-EU$m$ SAT-E$q$}, $q \geq 3$,$m \geq 3$, and $1 \leq k \leq m-1$ | NP-complete [Kra03] |
| POSITIVE $k$-IN-EU$m$ SAT-E$q$, $q \leq 3$, or $m \leq 3$, or $k = 0$, or $k = m$ | P [Kra03] |
| $\{0, 1, k-1, k\}$-IN-EU$k$SAT | NP-complete. Thm:4.4.1 * |
| $S$-IN-E$k$ SAT, for $k \geq 3$ | Characterization. Thm:4.2.2 $*$? |
| $\{0, k\}$-IN-E$k$ SAT | P |
| $\{0, 1, 2, \ldots, k\}$-IN-E$k$ SAT | P |
| $\emptyset$-IN-E$k$ SAT | P |

Continued on next page

Continued from previous page

| SAT type | Complexity |
|---|---|
| $\{\text{even}\}$ and $\{\text{odd}\}$-IN-E$k$ SAT | P |
| $\{k-1, k\}$-IN-E$k$ SAT | P |
| $\{0, 1\}$-IN-E$k$ SAT | P |
| $\{\text{Other Sets}\}$-IN-E$k$ SAT | NP-complete |
| $S$-IN-4SAT-$O(1)$ | Multiple results. Thm:4.3.1 $*$ |
| $\{1\}$-IN-E4SAT-4 | NP-complete |
| $\{2\}$-IN-E4SAT-5 | NP-complete |
| $\{3\}$-IN-E4SAT-4 | NP-complete |
| $\{0, 2\}$-IN-E4SAT-3 | NP-complete |
| $\{1, 2\}$-IN-E4SAT-4 | NP-complete |
| $\{2, 3\}$-IN-E4SAT-4 | NP-complete |
| $\{0, 1, 3\}$-IN-E4SAT-4 | NP-complete |
| $\{0, 1, 4\}$-IN-E4SAT-6 | NP-complete |
| $\{0, 2, 3\}$-IN-E4SAT-3 | NP-complete |
| $\{0, 3, 4\}$-IN-E4SAT-6 | NP-complete |
| $\{1, 2, 3\}$-IN-E4SAT-5 | NP-complete |
| $\{0, 1, 3, 4\}$-IN-E4SAT-5 | NP-complete |
| $\{0, 2, 3, 4\}$-IN-E4SAT-5 | NP-complete |
| $\{1, 2, 3, 4\}$-IN-E4SAT-5 | NP-complete |
| POSITIVE $S$-IN-EU$(2k)$-SAT-E2. | Dichotomy [Kol+19] |
| $S$ is empty; always not satisfiable | P |
| There is an even number in $S$ | P |
| $k \in S$ | P |
| $\{k-1, k+1\} \subset S$ | P |
| $\exists a, b$ such that $S = \{a, a+2, a+4, \dots, b\}$ | P |
| Otherwise | NP-complete |
| POSITIVE $S$-IN-EU$(2k+1)$-SAT-E2. | Dichotomy [Kol+19] |
| $S$ is empty; always not satisfiable | P |
| It is trivial, i.e. if it is always satisfiable | P |
| $\exists a, b$ such that $S = \{a, a+2, a+4, \dots, b\}$ | P |
| Otherwise | NP-complete |
| When is POSITIVE $S$-IN-EU$(2k+1)$-SAT-E2 always satisfiable? | Open |
| $\{1, 4\}$-IN-EU5-SAT-E2 | Open |
| General SAT | |
| $S$ SAT$_{\text{CS}}$-$k$, for $k \geq 3$ | Equivalent to $S$ SAT$_{\text{CS}}$ [DF03, Theorem 2.3] |
| $S$-SAT$_{\text{CS}}$-E2. | Partial Dichotomy [KKR16; Fed01] |
| If $S$-SAT$_{\text{C}}$ is in P | P |
| If not all relations in $S$ are $\Delta$-matroids | NP-complete |
| If all relations in $S$ are even $\Delta$-matroids | P |

Continued on next page

Continued from previous page

| SAT type | Complexity |
|---|---|
| Otherwise | Open |
| $S$ SAT, for $S$ SAT $\in P$ | Full characterization [All+09] |
| The full characterization can be seen in "Refining Schaefer's Theorem" [All+09] | $L$, $NL$, $AC^0$, $\oplus L$ or $P$ complete |
| XNF SAT | |
| $S$ XNF SAT$_C$ | Partial characterization. Thm:6.6.4 * |
| Each relation in $S$ is the disjunction or the complement of a disjunction of a polynomial number of mutually exclusive AND clauses. | RP |
| Otherwise | Open |
| $S$ XnorNF SAT$_C$ | Partial characterization. Corollary:6.6.5 * |
| Each relation in $S$ is the disjunction or the complement of a disjunction of a polynomial number of mutually exclusive AND clauses. | RP |
| Otherwise | Open |
| $S$ XNF U SAT$_C$ | Partial characterization. Thm:6.6.7 * |
| Each relation in $S$ is multilinear. | RP |
| If $S$ XNF SAT$_C$ is in RP | RP |
| Otherwise | Open |
| Other SAT | |
| Renamable Horn SAT (Horn after complementation of some variables) | P [Cha+90] |
| $k$ 1s Positive Not-1-in-EU3SAT$_{ts}$ | NP-complete [KW13] |
| Not-1-in-EU3SAT-3 | NP-complete. Thm:6.1.2 * |
| Circuit Not Implies SAT | NP-complete. Thm:6.4.1 * |
| Planar Not Implies Circuit SAT | Open |
| Inverse $S$ SAT, where $S$ is finite | Dichotomy [KS98] |
| Every relation in $S$ is Horn. | P |
| Every relation in $S$ is Dual-Horn. | P |
| Every relation in $S$ is affine. | P |
| Every relation in $S$ is bijunctive. | P |
| If neither of the above. | coNP-complete |
| Exists Exists-Unique 3SAT ($\exists$ $\exists!$ 3SAT) | $\Sigma_2^P$-complete [Mar08] |
| Exists Exists-Unique NAE 3SAT ($\exists$ $\exists!$ NAE 3SAT) | $\Sigma_2^P$-complete. Thm:3.10.2 * |
| Exists Exists-Unique given Proper Partial Assignment 3SAT ($\exists x$ $\exists!_t y$ 3SAT) | $\Sigma_2^P$-complete [HM05] |
| Minimum Defining Set 3SAT | $\Sigma_2^P$-complete [HM05] |
| Minimum Defining Set of Solution 3SAT | $\Sigma_2^P$-complete [HM05] |
| Unique given Solution 3SAT | coNP-complete [HM05] |
| Unique SAT | US-complete [Pap94; BG82] |

Continued on next page

Continued from previous page

| SAT type | Complexity |
|---|---|
| Critical SAT | DP-complete [Pap94] |
| All-Complements 3SAT | $\Pi_2^P$-complete [Sze04] |
| All-Complements 2SAT | P [Sze04] |
| 3SAT-UNSAT | DP-complete [Pap94] |
| #SAT | Dichotomy [CLX09] |
| #Planar SAT | Dichotomy [CLX10] |
| Reconfiguration $S$ SAT | Dichotomy [Gop+06] |
| $S$ SAT is in P | P |
| $S$ is tight | P |
| Otherwise | PSPACE-complete |
| Connectivity $S$ SAT | Dichotomy [Gop+06; MTY07] |
| Every relation in $S$ is affine. | P |
| Every relation in $S$ is bijunctive. | P |
| Every relation in $S$ is Horn. | coNP-complete |
| Every relation in $S$ is Dual-Horn. | coNP-complete |
| $S$ is tight, but not one of the previous cases | coNP-complete |
| Otherwise | PSPACE-complete |
| Intersecting Monotone SAT | co-TRANS-HYP-complete [EG02] |
| Intersecting Monotone $k$SAT | P [EG02] |
| Intersecting Positive NAE SAT | co-TRANS-HYP-complete [EG02] |

## 2.2 Definitions

For the name of our SAT problems, we use a notation and structure intended to minimize the ambiguity while still being succinct and relatively easy to parse. An example of our notation can be seen in Figure 2-1, where it shows a fictitious SAT problem.

$$\underbrace{k\ 1\text{S}}_{\text{Satisfiability Constraint}}\ \overbrace{\text{MAX}}^{\text{Satisfiability Condition}}\ \underbrace{\text{VAR-LINKED}}_{\text{Modifiers to Graph Constraint}}\ \overbrace{\text{PLANAR}}^{\text{Graph Constraint}}\ \underbrace{\text{MONOTONE}}_{\text{Constraint on Clauses}}\ \overbrace{\text{NAE}}^{\text{Type of Clauses}}\ \underbrace{\text{EU3}}_{\text{Number and Uniqueness of Variables in Clauses}}\ \overbrace{\text{SAT}}^{\text{Type}}{}_{\underbrace{\text{c}}_{\text{T and F Constants}}}\ \overbrace{\text{-E4}}^{\text{Occurrences of Variables}}\ \underbrace{\text{N3P}}_{\text{Constraint on Variables}}$$

Figure 2-1: SAT notation example.

We define our SAT problems names in the following order: **Satisfiability constraint**; **Satisfiability conditions**; **modifiers to the graph constraints**; **graph constraints**; **constraints on types of clause/formulas**; **type of clause**; **constraint on number and uniqueness of variables in the clause**; **"type" of SAT**; **allowed constants (in subscript)**; **constraint on occurrences of variables**; **other constraints on the variables**. *Type of SAT* and *Satisfiability conditions* indicate the same type of modifications; however they are kept in different locations for historical reasons. To distinguish the name of the problem from the remainder of the text, we use Small Caps, as standard in literature. When there is no good short description for a constraint, we define the extra constraint after the name of the SAT problem.

The example from Figure 2-1, $k$ 1s Max Var-Linked Planar Monotone NAE EU3SAT$_c$-E4 n3p, reads as "$k$ 1s Max Var-Linked Planar Monotone Not All Equal Exactly Unique 3 SAT$_c$-Exactly 4, no 3 positive". Given a value $k$ and a value $n$, a conjunction of NAE clauses, and a planar embedding of the bipartite graph of clauses and variables together with a Hamiltonian cycle that goes through the variables, it is the problem of deciding whether it is possible to satisfy at least $n$ clauses by setting $k$ variables to True. Furthermore, we have the following restrictions on the instances: each clause has exactly 3 distinct variables; each clause has the variables all negated or all positive; 2 variables, T and F, have their values forcibly set respectively to True and False; all variables appear exactly 4 times (this includes the T and F variables); and no variable appears positively 3 or more times.

Here we define possible values for the multiple fields that define a SAT problem name.

- **Satisfiability conditions and constraints**. Conditions on how to satisfy an instance, or what a satisfiable instance of the SAT problem is.
  - Standard / not specified: the problem of deciding whether there is an assignment to the variables that satisfies the given formula.
  - **Sharp(#)**: The problem of counting how many satisfying solutions there are. When we say that a satisfiability problem is in #P or it is #P complete, we are implicitly converting the SAT problem to a #SAT problem.
  - **Circuit (C)**: The problem of deciding whether there exists an assignments of inputs to Boolean circuit constructed with gates that implement specific formulas, such that the circuit output is True (1).
  - **Quantified (Q)**: The problem of deciding whether a fully quantified Boolean formula evaluates to True.
  - ***k*-Quantified** [Pap94, p. 428]: The problem of deciding whether a quantified Boolean formula with $k-1$ alternations evaluates to True. Formally, the variables are partitioned into $k$ sets, $X_1, X_2, \dots, X_k$, and it is the problem of deciding whether $\exists X_1 \forall X_2 \exists X_3 \dots Q X_k f$ is true, where $Q$ is $\exists$ if $k$ is odd, and $\forall$ if k is even. If specified, the order of $\forall$ and $\exists$ is switched, and the first quantifier is $\forall$ instead of $\exists$.
  - **Quantifiers**: If we are given explicit quantifiers, we assume it to be the equivalent $k$-Quantified SAT instance. Example: $\forall\exists$ 3SAT is equivalent to a switched 2-Quantified 3SAT.
  - **Max**: Given $k$ and a conjunction of clauses, the problem of deciding whether there is an assignment to the variables which satisfies $k$ or more clauses.
  - **Min**: Given $k$ and a conjunction of clauses, the problem of deciding whether there is an assignment to the variables which satisfies $k$ or fewer clauses.
  - **Minmax**: Given $k$ and a conjunction of clauses $\phi(x_1, x_2, \dots, x_i, y_1, y_2, \dots, y_j)$, the problem of deciding whether $\forall \boldsymbol{x}\ \exists \boldsymbol{y}$ such that at least $k$ clauses are satisfied. It is an optimization version of $\forall\exists SAT$.
  - **Reconfiguration** [Gop+06]: Given a SAT formula and 2 solutions, $\sigma_1, \sigma_2$, the problem of deciding whether there is a sequence of solutions starting at $\sigma_1$ and ending at $\sigma_2$ where successive truth assignments differ in the value of exactly one variable. If such sequence exists, we say that there exists a **path** between $\sigma_1$ and $\sigma_2$.

- **Connectivity** [Gop+06]: Given a SAT formula, the problem of deciding whether there exists a path between every pair of solutions – whether the space of solutions is connected.
- **Exists Exists Unique ($\exists\exists!$ or $\exists \boldsymbol{x} \exists! \boldsymbol{y}$ )** [Mar08]: Given a formula $\phi(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j)$ the problem of deciding whether there exists an assignment of $x_1, x_2, \ldots, x_i$ such that there is a unique assignment for $y_1, y_2, \ldots, y_j$ that satisfies the formula. If the answer is yes, we say that $\{x_1, x_2, \ldots, x_i\}$ is a **defining set**: a set of variables such that for an assignment of truth values of this subset, there exists a unique assignment of variables that satisfy the formula while agreeing with the truth assignment.
- **Exists Exists Unique given Proper Partial Assignment ($\exists \boldsymbol{x} \exists!_t \boldsymbol{y}$)** [HM05; SU02]: Given a formula $\phi(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j)$ and a proper partial assignment over $\boldsymbol{y}$, the problem of deciding whether $x$ is a defining set. A **proper partial assignment** over $\boldsymbol{y}$ is an assignment of the variables in $\boldsymbol{y}$, such that the formula is satisfied independent of (that is, for all choices of) the assignment of variables in $\boldsymbol{x}$. For example, let

  $$\phi = (x_1 \vee y_1 \vee y_2) \wedge (\neg y_1 \neg y_2) \wedge (y_1 \neg y_2),$$

  then $y_1 = 1$, $y_2 = 0$ is a proper partial assignment, and $\boldsymbol{x}$ is a defining set, because if $x_1 = 0$, then the $\boldsymbol{y}$ assignment given is the only solution.
- **Minimum Defining Set** [HM05]: Given a formula $\phi(x_1, x_2, \ldots, x_i)$, and a value $k$, the problem of deciding whether there exists a defining set of size $k$.
- **Minimum Defining Set of Solution** [HM05]: Given a formula $\phi(x_1, x_2, \ldots, x_i)$, a solution $\boldsymbol{x}$, and a value $k$, the problem of deciding whether the solution contains a defining set of size at most $k$, i.e., there are $k$ variables such that by setting their values to the values in $\boldsymbol{x}$, there exists a unique assignment of the remaining variables that satisfies the clause, $\boldsymbol{x}$.
- **Unique given Solution($\exists_t!$)**: Given a formula $\phi(x_1, x_2, \ldots, x_i)$, and a solution $\boldsymbol{x}$, the problem of deciding whether $\boldsymbol{x}$ is the only solution.
- **Unique ($\exists!$)** [Pap94]: Given a formula $\phi$, the problem of deciding whether it has exactly one solution.
- **Critical** [Pap94]: Given a formula $\phi$ that is a conjunction of clauses, the problem of deciding whether is it true that $\phi$ is not satisfiable, but deleting any clause makes it satisfiable.
- **All-Complements**: Given a SAT instance, the problem of deciding whether the instance is satisfiable under arbitrary replacement of literals by their complements [Sze04]. In other words, given a SAT instance with $n$ variables, the problem of deciding whether all the $2^n$ SAT instances that can be constructed by complementing its variables are satisfiable.
- **Inverse**: Given a list of solutions, the problem of whether is it possible to construct a SAT instance whose solutions match the ones given, given the allowed clauses or constrains. As an example, INVERSE 3SAT is coNP-complete. The solutions to 1-IN-$(a, b, c, d)$ is a list of solutions for which it is not possible to construct a 3SAT instance.
- **Constraints**.
  * ***k* 1s**: We are limited to setting at most $k$ variables to 1 (TRUE), where $k$ is a parameter of the SAT instance. Note that this $k$ is always the letter $k$, and never substituted for a number.

    * ***k* 0s**: We are limited to setting at most $k$ variables to 0 (False), where $k$ is a parameter of the SAT instance. Note that this $k$ is always the letter $k$, and never substituted for a number.

- **Graph constraints and modifiers**. For every one of these constraints, we are given the "constraint" together with the formula in the SAT instances. For example, when solving a Planar SAT instance, we are also given its planar embedding. When solving Linked Planar SAT, we are also given the Hamiltonian cycle that goes through all the variables and clauses and its embedding.
  - **Planar**: The bipartite graph with a vertex for each clauses, a vertex for each variable, and an edge for each pair of a clause and a variable in that clause is planar. If a different graph is specified, then the specified graph is planar, as in Separable Planar SAT. When we need to distinguish the edges that connect a clause to a variable that appears in it as a positive literal from the ones that appear as a negative literal, we define the former to be edges directed from the clause to the variable and the latter to be edges directed from the variable to the clause. For convenience, when drawing such graph, we follow the convention from [Lar93] and draw the clauses as rectangles and the variables as circles, unless stated otherwise. The graph embedding is given as one of the inputs to the instance.
  - ***k*-Planar**: In the bipartite graph defined as above, each edge crosses at most $k$ other edges.
  - **Crossing Number *k***: In the bipartite graph defined as above, there are at most $k$ edge crossings.
  - **Separable** [Tip16]: The vertex set consists of all literals and all clauses. We connect the negative and positive literals of each variable with an edge. We connected each clause to the literals that it contains with edges. Figure 2-3 shows an example of a Separable Planar 3SAT graph.
  - **Separate**: Similarly to Separable, the vertex set of the incident graph consists of all literals and all clauses and we connected a clause to the literals that it contains with edges. However, we don't connect the positive and negative literals of each variable.
  - **Var-Linked Planar** [Fel+95]: In addition to the constraints in Planar, if we add a Hamiltonian cycle that goes through all the variable nodes, the graph is still planar. The cycle is given as one of the inputs to the instance. Figure 2-2 shows an example of a Var-Linked Planar 3SAT graph.
  - **Clause-Linked Planar** [Fel+95]: In addition to the constraints in Planar, if we add a Hamiltonian cycle that goes through all the clause nodes, the graph is still planar. The cycle is given as one of the inputs to the instance. Figure 2-3 shows an example of a Clause-Linked Planar 3SAT graph.
  - **Linked Planar** [Pil17]: In addition to the constraints in Planar, if we add a Hamiltonian cycle that goes through all the variable nodes, followed by going through all the clause nodes, the graph is still planar. The cycle is given as one of the inputs to the instance. Figure 2-4 shows an example of a Linked Planar 3SAT graph embedding.
  - **Rectilinear Planar**: In addition to the constraints in planar, all edges can be drawn as combinations of vertical and horizontal line segments.
  - **Tri-Legged Planar**: In addition to the constraints in Var-Linked Planar and Rectilinear Planar, all variable nodes can be drawn as straight line segments in the $x$ axis and all clauses can be drawn as horizontal lines, connected to the nodes by at most 3 vertical segments. This was

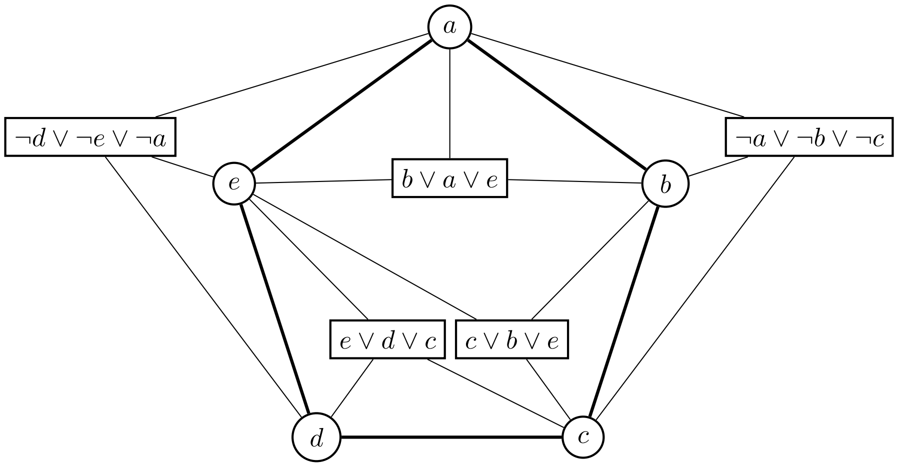


Figure 2-2: Ordered Sided Var-Linked Planar EU3SAT example.

called "tri-legged construction" by Pilz [Pil17]. Figure 2-5 shows an example of this construction. Tippenhauer [Tip16] has shown that Var-Linked Planar is equivalent to Tri-Legged Planar when all clauses have at most 3 variables.

- **Ordered Planar** [KKR16]: In addition to the constraints in Planar, when going anticlockwise around a clause node, the variables are in the same order as they appear in the clauses. This distinction from planar is only relevant when not all permutations of a relation are present in our allowed relations. For example, if $(\neg x \vee y \vee z)$ is a valid relation, but $(x \vee \neg y \vee z)$ is not. If all permutations are present, as is the case with NAE SAT, $S$-in-$k$SAT or CNF SAT, then Ordered Planar is equivalent to Planar, since we can always permute the variables in the clauses and make use of another valid relation. Figure 2-2 shows an example of a Ordered Planar 3SAT graph.
- **Sided**: Var-Linked, Clause-Linked or Linked Planar: In addition to the constraints in Var-Linked, Clause-Linked or Linked Planar, we also enforce that all edges connecting to positive literals are inside the given Hamiltonian cycle and all edges connecting to negative literals are outside it. Note that Sided Var-Linked Planar SAT implies that all clauses are monotone, and that Sided Clause-Linked Planar SAT implies that all variables are monotone – each variable appears only positively or only negatively. Figure 2-2 shows an example of a Sided Var-Linked Planar 3SAT graph.
- ***k*-Connected**: The graph of variables and clauses is $k$-connected. In other words, it is a graph with at least $k + 1$ vertices that remains connected whenever fewer than $k$ vertices are removed.

- **Constraints on types of clauses/formulas**
  - **Positive**: Each clause contains only positive literals.
  - **Monotone**: Each clause contains only positive or only negative literals.
  - **Intersecting Monotone** [EG02]: Besides being Monotone, each positive clause contains at least one variable in common with each negative clause.
  - **Intersecting** , but not Monotone: Each pair of clauses contains at least one variable in common.

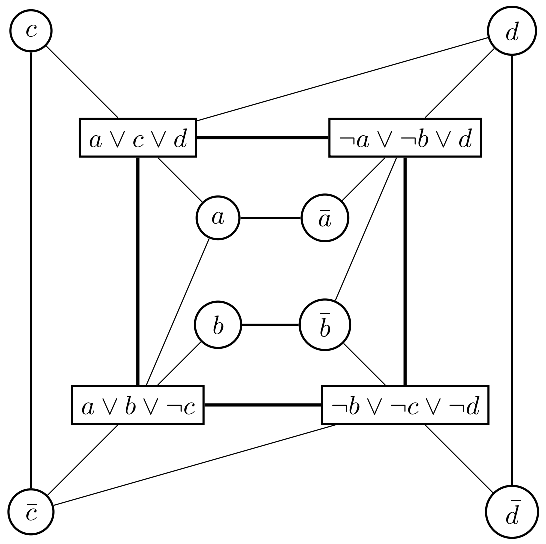


Figure 2-3: SEPARABLE CLAUSE-LINKED PLANAR EU3SAT example.

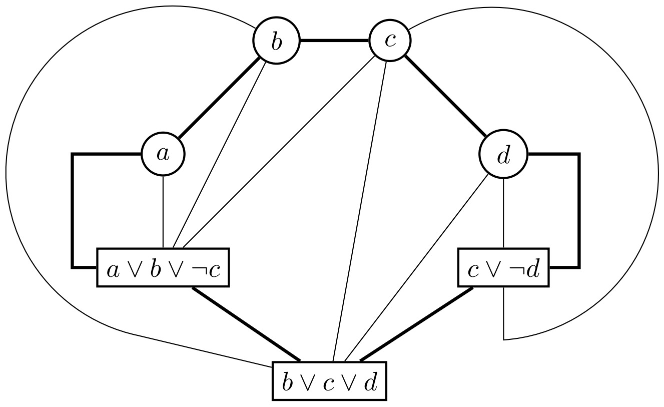


Figure 2-4: LINKED PLANAR 3SAT example.

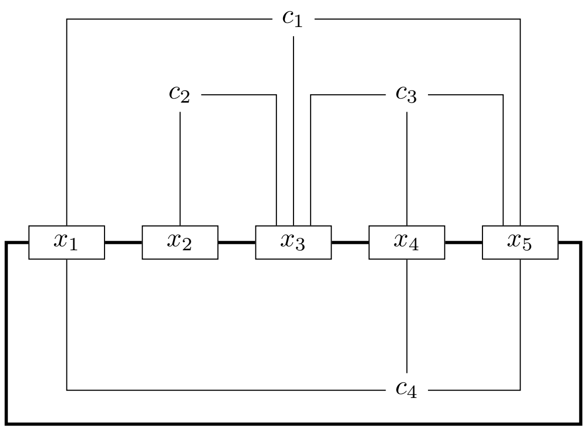


Figure 2-5: Tri-legged construction example for Rectilinear Var-Linked Planar SAT. The variables are line segments in the $x$ axis and the clauses are points.

- **n*k*p**: No clause contains $k$ or more positive literals. **n*k*n** enforces the same for negative literals.
- **e*k*p**: All clauses contain exactly $k$ positive literals. **e*k*n** enforces the same for negative literals.
- **ne*k*p**: No clause contains exactly $k$ positive literals. **ne*k*n** enforces the same for negative literals.
- **$\geq$ *k*p**: All clauses contain $k$ or more positive literals. **$\geq$ *k*n** enforces the same for negative literals.
- **Combining constraints**: If all the constraints need to be satisfied, they are listed one after the other. If only a subset of the constraints need to be satisfied at a time, we state the relation between the constraints. For example PLANAR ($\geq$ 3P OR $\geq$ 3N) SAT stands for an instance of PLANAR SAT where every clause contains at least 3 positive literals, or at least 3 negative literals.

- **Types of Clauses**
  - Standard / **not specified**: If there is a limit on the size of the clause, the allowed formulas are in **Conjunctive Normal Form (CNF)**: a **conjunction** of clauses (clauses connected by $\wedge$), each of which is a **disjunction** of literals (literals connected by $\vee$). If nothing is specified about the clause types or size, then it is assumed that all Boolean formulas are allowed.
  - **CNF**: the formula is in CNF.
  - **DNF**: The formula is in **Disjunctive Normal Form**: a disjunction of clauses, each of which is a conjunction of literals. If combined with another type of clause it will be a disjunction of clauses of that type. For example, NAE DNF SAT is a disjunction of NAE clauses.
  - **XNF**: The formula is in **XOR Normal Form**, clauses that are disjunction of literals connected by XORs. If combined with another type of clause it will be a series of those clauses connected by XORs. For example, NAE XNF SAT is a series of NAE clauses connected by XORs.
  - **XnorNF**: The formula is in **XNOR Normal Form**. Same as XNF, but with XNORs replacing the XORs.
  - **NAE**: The formula is a conjunction of **Not All Equal (NAE)** clauses of literals. A NAE clause is a clause that evaluates to TRUE iff not all of its literals are equal.

- **1-in-**: The formula is a conjunction of **Exactly 1 in (1-in-)** clauses of literals. A 1-in- clause is a clause that evaluates to TRUE iff exactly one of its literals is TRUE.
- ***S*-in-**: An $S$-IN-* clause is a clause that evaluates to TRUE iff the number of TRUE literals in the clause is $\in S$. For example, NAE EU3SAT is the same as $\{1,2\}$-IN-EU3SAT.
- **Not-1-in-**: The formula is a conjunction of **Not Exactly 1 in (Not-1-in-)** clauses of literals. A Not-1-in- clause is a clause that evaluates to TRUE iff not exactly one of its literals is TRUE.
- **Horn**: The formula is a CNF formula where each clause contains at most one positive literal. An equivalent name using our notation would be: N1P CNF.
- **Dual Horn**: The formula is a CNF formula where each clause contains at most one negative literal. An equivalent name using our notation would be: N1N CNF.
- **Xor**: The formula is a conjunction of clauses, where each clause is a XORs of literals, for example $(a \oplus b) \wedge (\neg a \oplus b \oplus c) \wedge (\neg b)$.
- **S**: Some other condition or formula. Unless otherwise specified, $S$ is assumed to be a set of relations and the formula is a conjunction of clauses. Each clause evaluates to TRUE if the given relation is satisfied. For example, 2-in- would imply that each clause is TRUE iff there exist exactly 2 TRUE literals in it.
- **{connectives}**: The only connectives allowed in the formula are the ones defined in the set. The formula, in this case, is not necessarily conjunction of clauses. For example $(a \implies b \implies c) \implies (a \implies b)$ would be a valid instance of $(\{\implies\}SAT$.
- **Multiple clause types**: any of those clause types can be used in the SAT formula. Example: (NAE OR 1-IN-)3SAT, would be a SAT problem where each clause is either a NAE clause, or a 1-in- clause, with at most 3 variables each. An example of such a problem is: $\text{NAE}(x,y,z) \wedge$ 1-IN-$(y,z) \wedge \text{NAE}(\neg y, z)$

- **Constraint on number and uniqueness of literals in the clause**
  - ***k***: Clauses contain $k$ or fewer literals.
  - **E*k***: Clauses contain exactly $k$ literals.
  - **$\geq$ *k***: Clauses contain $k$ or more literals.
  - **U**: All the variables in each clause are **unique**: no variable appears more than once in the same clause.
  - **U*k***: All the variables in each clause are unique, and no clause contains more than $k$ literals.
  - **EU*k***: Clauses contain exactly $k$ literals and they are unique.
  - **$\geq$ U*k***: All the variables in each clause are unique, and all clauses contain $k$ or more literals.
  - **Multiple constraints**: The constraints are combined by Boolean connectives, for example ($\geq$ 3P OR $\geq$ 2N) means that each clause contains 3 or more positive literals or 2 or more negative literals. If no connective is used, we assume there to be an AND, for example: ($\geq$ 2P)EU4, means that each clause contains exactly 4 unique variables, with at least 2 positive ones.

- **Type of SAT** - They are generally incompatible with the SAT conditions field since they perform a similar role. They are placed here, however, for historical reasons, or when it makes the problem name easier to read.

  - **SAT**: The standard Boolean Satisfiability problem. Given a Boolean formula, the problem of deciding whether there exists an assignment of the variables that makes the formula evaluate to TRUE; unless we have set a different SAT condition
  - **QSAT**: See QUANTIFIED under Satisfiability conditions.
  - **QSAT*k***: See *k*-QUANTIFIED under Satisfiability conditions.
  - **Circuit SAT (CSAT)**: See CIRCUIT under Satisfiability conditions.
  - **UNSAT**: The problem of deciding whether a Boolean formula is not satisfiable, namely, deciding whether it evaluates to FALSE for every variable assignment.
  - **SAT-UNSAT** [Pap94]: Given 2 Boolean formulas, $\phi$ and $\phi'$, the problem of deciding whether it is true that $\phi$ is satisfiable and $\phi'$ is not.
  - **TAUTOLOGY**: The problem of deciding whether a Boolean formula is always satisfiable, namely. deciding whether it is TRUE for every variable assignment.
  - **NOT TAUTOLOGY**: The problem of deciding whether there exists an assignment of the variables that makes the formula false.
  - **#SAT**: See SHARP under Satisfiability conditions.

- **Allowed constants (in subscript)** - The distinction between a single variable set to TRUE or FALSE, and an unlimited number is relevant when discussing SAT problems where clauses can only contain unique variables, or when there is a limit on how many times a variable can be used.
  - $_{\mathbf{c}}$: A single variable can be forcibly set to TRUE (1), and a single to FALSE(0). Example, SAT$_{\mathrm{C}}$.
  - $_{\mathbf{cs}}$: An unlimited number of variables can be forcibly set to TRUE or FALSE.
  - $_{\mathbf{t}}$: A single variable can be forcibly set to TRUE. $_{\mathbf{f}}$ is the equivalent for FALSE. These are generally only relevant when using MONOTONE or POSITIVE constraints; otherwise one should use $_{\mathrm{C}}$.
  - $_{\mathbf{ts}}$: An unlimited number of variables can be forcibly set to TRUE. $_{\mathbf{fs}}$ is the equivalent for FALSE.

- **Constraint on occurrences of variables**.
  - **-*k***: Each variable **occurs** at most *k* times. We define the number of occurrences of a variable to be the number of times its literals appear in the formulas. For example, in the following CNF formula, the variable $x$ occurs 3 times and the variable $y$ occurs twice: $(x \vee x \vee y) \wedge (x \vee y)$. For another example, SAT-2 is CNF SAT where each variable appears at most twice.
  - **-E*k***: Each variable occurs exactly *k* times. For example, E3SAT-E3, where each variable occurs exactly 3 times and each clause contains exactly 3 variables.

- **Other constraints on the variables** - It is very rare at the moment to constrain the individual variables. One example was N3P 3SAT-3 N2N as defined by Ding [DFZ11], where they defined that every variable occurred as a negative literal at most 1 time. Even though this is a direct consequence of being a CNF SAT-3, it is still worth mentioning. For those and others constraints we use the same notation as for the clauses.
  - **n*k*p**, **n*k*n**, **e*k*p**, **e*k*n**, **$\geq$ *k*p**, **$\geq$ *k*n**, **ne*k*p**, **ne*k*n**: All enforce the same constraint as the respective clause constraint, except that instead of referring to the literals of a clause, this constraint refers to the occurrences of a variable.

  - **Monotone**: Each variable appears only positively or only negatively.

We also define some other terms that will be useful in our reductions, proofs, and Table 2.1.

- **Hypergraphs**. Terms related to hypergraphs.
  - **Hypergraph**: A hypergraph is a set of hyperedges. Each hyperedge is a set of 1 or more nodes.
  - **2-colorable Hypergraph**: A hypergraph where it is possible to color the nodes using 2 colors such that every hyperedge contains nodes of both colors.
  - ***k*-uniform Hypergraph**: A hypergraph where every hyperedge contains $k$ nodes.
  - ***k*-regular Hypergraph**: A hypergraph where every node appears in $k$ hyperedges.
  - **Condenser**: A hypergraph that is minimally non-2-colorable. By removing a single hyperedge, the hypergraph becomes 2-colorable.
  - **Square Hypergraph**: A hypergraph with the same number of nodes and hyperedges.
  - **Transversal Hypergraph problem (TRANS-HYP)** [EG02]. A **transversal** of a hypergraph $\mathcal{H}$ is a subset of the vertices that intersects every hyperedge. A transversal is **minimal** if removing any vertex from it makes it stop being a transversal. The **transversal hypergraph** of $\mathcal{H}$ is a hypegraph $\mathcal{G}$ whose hyperedges are all the minimal transversals of $\mathcal{H}$. The **Transversal Hypergraph problem** is the problem of given 2 hypergraphs $\mathcal{H}$ and $\mathcal{G}$, deciding whether $\mathcal{G}$ is the transversal of $\mathcal{H}$. This problem is relevant because some instances of SAT are co-TRANS-HYP complete. It has been shown that the problem is in co-$NP$, but no co$NP$-completeness proof has been found. It is believed that TRANS-HYP is co$NP$-intermediate, in a complexity class between $P$ and co$NP$ [EG95]. It has been shown that TRANS-HYP can be recognized in $O\left(n^{\log n}\right)$ [FK96].

- **Relations**. Terms related to relations and their classifications. Useful when describing dichotomoies.
  - **Relation**: A relation is a set of tuples specifying valid assignments of literals.
    For example, the only relation in Positive NAE EU3SAT is $\{(1,0,0),(0,1,0),(0,0,1),(1,1,0),(1,0,1),(0,1,1)\}$, and the only Positive 1-in-EU3 relation is 1-in-$(x,y,z)$ $\{(1,0,0),(0,1,0),(0,0,1)\}$. The Positive NAE u3SAT problem contains 2 relations, NAE$(x,y,z)$ $\{(1,0,0),(0,1,0),(0,0,1),(1,1,0),(1,0,1),(0,1,1)\}$ and NAE$(x,y)$ $\{(1,0),(0,1)\}$, depending on the clause size. 1-in-U3 SAT contains 3 relations, $\{(1,0,0),(0,1,0),(0,0,1)\}$, $\{(1,0),(0,1)\}$ and $\{(1)\}$.
    NAE 3SAT Contains 6 relations: NAE$(x,y,z)$, NAE$(\neg x,y,z)$, NAE$(x,\neg y,z)$, NAE$(x,y,\neg z)$, NAE$(x,y)$, NAE$(\neg x,y)$. The other NAE clauses with other variables negated, or with variables repeated are equivalent to one of those 6 cases.
  - **Bijunctive** [Gop+06]: A relation is bijunctive if it is expressible as a 2-CNF. This implies that, given 3 valid assignments of variables $a,b,c$, $\textsc{maj}(a,b,c)$ is also a valid assignment, where $\textsc{maj}$ is the majority operation. $\textsc{maj}(a,b,c)$ is the vector whose $i$th element is the majority of $a_i,b_i,c_i$.
  - **Affine**: A relation is affine if, given 3 valid assignments of variables $a,b,c$, $(a\oplus b\oplus c)$ is also a valid assignment. $(a\oplus b\oplus c)$ is the vector whose $i$th element is $a_i\oplus b_i\oplus c_i$, the XOR of the $i$th variable assignments.
  - **Horn**: A relation is Horn if, given 2 valid assignments of variables $a$ and $b$, $(a\vee b)$ is also a valid assignment. $(a\vee b)$ is the vector whose $i$th element is $a_i\vee b_i$.

- **Dual Horn**: A relation is Dual Horn if, given 2 valid assignments of variables $a$ and $b$, $(a \wedge b)$ is also a valid assignment. $(a \wedge b)$ is the vector whose $i$th element is $a_i \wedge b_i$.
- **OR-Free** [Gop+06]: A relation in $k$ variables is OR-free if the relation $\{(0,1),(1,0),(1,1)\}$ cannot be obtained by setting $k-2$ of the coordinates to constants.
- **NAND-Free** [Gop+06]: A relation in $k$ variables is NAND-free if the relation $\{(0,0),(0,1),(1,0)\}$ cannot be obtained by setting $k-2$ of the coordinates to constants. All relations in Positive 3SAT are NAND-free. In 3SAT; however, some of them are not NAND-free, such as $(\neg x \vee \neg y)$.
- **Connected Component** [Gop+06]: Two tuples $t_1$ and $t_2$ of a relation are part of the same connected component if there is a sequence of tuples starting at $t_1$ and ending at $t_2$ where successive tuples differ in the value of exactly one variable.
- **Tight** [Gop+06]. Tight is an important property for Reconfiguration SAT and Connectivity SAT. A set $S$ of relations is tight if one of the following is true:
  1. Every connected component of every relation in $S$ is bijunctive.
  2. Every relation in S is OR-free.
  3. Every relation in S is NAND-free.
- **Self-Complementary relation**: A relation is self-complementary iff, given an assignment of variables that satisfies the relation, negating the assignment also satisfies the relation. The Not All Equal relations are self-complementary. The 1-in-$k$ relations are not, unless $k = 2$
- ***d* Relation** [KKR16]: Let $R$ be a relation. $dR = \{dT : T \in R\}$. Let $n$ be the length of the tuple T. Then: $dT = \{t_i + t_{i+1} \mod 2 : \ i = 1, 2, \ldots, n\}$, with $t_{n+1} = t_1$. For example, let $R$ be the Positive 1-in-EU3 relation. Then $R = \{(1,0,0),(0,1,0),(0,0,1)\}$, and $dR = \{(1,0,1),(1,1,0),(0,1,1)\}$. If we have $R$ be the Positive All Equal relation in 3 variables, then $R = \{(1,1,1),(0,0,0)\}$ and $dR = \{(0,0,0)\}$.

- **Delta Matroids**. Terms related to Delta Matroids and the Planar Dichotomy [KKR16].
  - **Delta Matroid (Δ-matroid)** [KKR16]: A non-empty relation is a Δ-Matroid if whenever there are 2 assignments of variables, $f$ and $g$, that satisfy that relation, the following is true: Let $x$ be a variable whose assignments in $f$ and $g$ differ. Then there exists a variable $y$ (which could just be $x$) whose assignments in $f$ and $g$ differs such that if we negate both $x$ and $y$ in the assignment $f$, we obtain another valid assignment.
    In other words, S is a Δ-matroid if

    $$\forall T_1 \in S, \forall T_2 \in S. T_1 \neq T_2. \, \forall x \in T_1 \Delta T_2. (((T_1 \oplus x) \in S) \text{OR } (\exists y \in T_1 \Delta T_2, x \neq y \text{ s.t. } ((T_1 \oplus x \oplus y) \in S))),$$

    where $\boldsymbol{T_1 \Delta T_2}$ is the set of variables with different values in $T_1$ and $T_2$, and $\oplus x$ means to negate the variable $x$, i.e. replace 0s by 1s, and vice-versa in the variable location in the tuple.
    An example of Δ-Matroid would be the Positive 1-in-$k$ relation, since we can always convert a valid assignment to another by negating the 2 literals that differ between them. Other example would be $\{(1,1,0),(1,0,0)\}$, which evaluates to True if the first 2 variables is True, or if only the first variable is True. An all equal relation in 3 variables, $\{(1,1,1),(0,0,0)\}$ is not a Δ-matroid, and $\{(1,1,0),(0,0,1)\}$ is also not a Δ-matroid. By this definition a relation that contains only one tuple is a Δ-matroid.

- **Even Delta Matroid (even Δ-matroid)** [KKR16]: If in addition of the relation being a delta matroid, all valid assignments for the relation contain the same parity of literals set to True. Positive 1-in-EU$k$ relation is an even Δ-matroid, but $\{(1,1,0),(1,0,0)\}$ is not. The NAE 3SAT relation is also not an even Δ-matroid, since the relation can be satisfied with either 1 or 2 variables set to True. By this definition a relation that contains only one tuple is an even Δ-matroid.

- **Sets**. Terms related to $S$-in-E$kSAT$ proofs.
  - **Self-dual**: The set $S$ in an $S$-in-E$kSAT$ relation is self dual iff $x \in S$ implies $(k-x) \in S$.

## 2.3 Notation incompatibility with previous papers.

The notation we propose is incompatible with the notation used in other papers, so here we display some of those differences.

- Laroche [Lar93] calls Planar Positive 1-in-EU3 SAT by Planar 1-in-3 SAT.
- Moore et al. [MR01] call "Planar Positive 1-in-EU3SAT-E3" by "Monotone 1-in-3 Satisfiability for planar cubic graphs".
- Schaefer [Sch78] calls "Positive Not-1-in-u3 SAT$_T$" by Not-1-in-3 Satisfiability.
- Dehghan [Deh16] calls "Planar Positive NAE 3SAT$_{\mathrm{Ts}}$" by "Restricted Planar Monotone Not-All-Equal 3SAT".
- Eiter et al. [EG02] call Positive NAE SAT by Symmetric Monotone SAT.
- de Berg et al. [BK10] call Sided Var-Linked Rectilinear Planar Monotone 3SAT by Planar Monotone 3-sat.
- Chaplick et al. [Cha+16] call Clause-Linked Planar Positive 1-in-EU3SAT by Positive Planar Cycle 1-in-3-Sat.
- Johnson et al. [GJS76] calls MAX 2SAT by MAX SAT2.
- Hunt et al. [III+98] calls Planar Positive 1-in-EU3SAT by Pl-1-Ex3MonoSat.
- In [Deh16], Dehghan uses a "NAE assignment to a CNF formula" to mean that each conjunct is a NAE clause of literals.
- Lichtenstein [Lic82] calls Var-Linked Planar 3SAT by Planar 3SAT, and calls Var-Linked Planar 3QSAT by Planar TF. Mansfield [Man83], wrongly cites the problem solved by Lichtenstein by omitting the Var-Linked restriction in its description.
- Feder [Fed01] and Kazda et al. [KKR16] call U SAT E2 by Boolean edge CSP. Kazda et al. also call Planar SAT by Planar Boolean CSP.
- Szeider [Sze04] calls All-Complements SAT by Graph SAT.

# Chapter 3

# NAE SAT variants

In this chapter we present our results in NAE SAT variants. We first present some problems with no graph restrictions that are in P, followed by NP-complete problems. Then we present almost planar NAE SAT variants, and at last, a result in $\Sigma_2^P = \mathrm{NP}^{\mathrm{NP}}$.

## 3.1 NAE SAT-2

**Theorem 3.1.1.** NAE SAT-2 *is in P.*

*Proof.* If there exists a clause with a single variable, then it is not satisfiable. If not, then every clause contains at least 2 variables.

First we convert it to a Positive NAE SAT-2 instance. If a variable ($x$) occurs as both as a positive and a negative literal, we create a new variable and a new clause $NAE(x, x')$, and we replace $\neg x$ by $x'$. If it occurs only as a negative literal, we can replace the negative literal with the positive literals, without affecting the satisfiability of the instance. If one represents the clauses as hyperedges, and the variables as nodes in a hypergraph, then the NAE SAT instance is satisfiable if and only if the hypergraph can be 2-colored. Supposing the hypergraph is not 2-colorable, this means it contains a subset of hyperedges that are minimally non-2-colorable, also known as a condenser.

It is known that any condenser contains at least as many edges as nodes [Sey74].

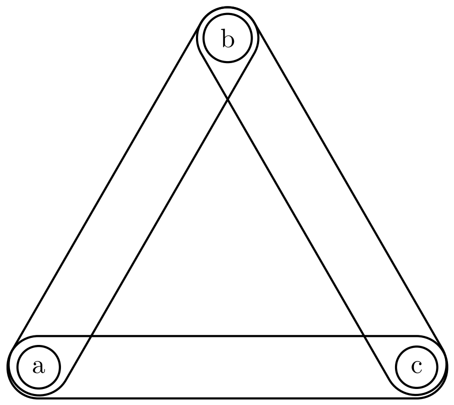
Since each node shows up in at most 2 clauses, the degree of each node is at most 2. The number of edges in any subset $E$ is

$$|E| = \sum_{v \in V[E]} \frac{\text{degree of node } v}{\text{average degree of the hyperedges in } E} \leq \sum_{v \in V[E]} \frac{2}{2} \leq |V[E]|,$$

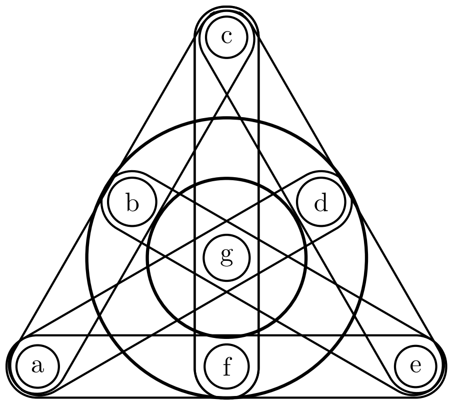
with equality iff each hyperedge and each node in $E$ has degree 2.

Therefore if a hypergraph is not 2-colorable, it contains a subset $E$ such that each hyperedge and each node of the subset has degree 2, and this subset is not 2-colorable.

Therefore we can remove all hyperedges with more than 2 variables, converting the hypergraph to a graph to test the resulting graph for 2-colorability. Since 2-coloring a graph is in P, the original problem is also in P. □



(a) A 2-regular 2-uniform square condenser with 3 hyperedges



(b) The Fano plane, a 3-regular 3-uniform square condenser, with 7 hyperedges.

Figure 3-1: Examples of square condensers.

## 3.2 Positive NAE EU$k$SAT-E$k$

The NAE EU$k$ SAT problem is equivalent to the 2-coloring a $k$-uniform hypergraph, by converting the variables to nodes, and having each clause represent a hyperedge, since each clause contains exactly $k$ different nodes, all positive. Similarly, NAE U SAT-$Ek$ is equivalent to the 2-coloring a $k$-regular hypergraph, since each variable is part of exactly $k$ clauses, and they are all positive. Therefore we can use known results on hypergraph 2 coloring to prove results on Positive NAE SAT.

**Theorem 3.2.1.** Positive NAE EU$k$SAT-E$k$ *is always satisfiable for* $k \geq 4$

*Proof.* We can represent our Positive NAE EU$k$SAT-E$k$ instance as a hypergraph 2-coloring problem, where the clauses are the hyperedges and the variables are the nodes. Henning at al [HY13] showed that all $k$-uniform-$k$-regular hypergraphs, for $k \geq 4$ are 2 colorable. Therefore every Positive NAE EU$k$SAT-E$k$ instance is 2-colorable for $k \geq 4$. □

**Theorem 3.2.2.** Positive NAE EU$kSAT\frac{2^{k-3}}{k}$ *is always satisfiable.*

*Proof.* Paul Erdős [EL73] showed that a $(r+1)$-chromatic $k$-uniform hypergraph contains at least one vertex of degree $> \frac{r^{k-1}}{4k}$; therefore, if a $k$-uniform hypergraph is not 2 colorable, i.e. it is at least 3 chromatic, it contains at least one vertex of degree $> \frac{2^{k-1}}{4k} = \frac{2^{k-3}}{k}$.

Therefore all $k$-uniform hypergraphs with all vertices with degree $\leq \frac{2^{k-3}}{k}$ are 2-colorable. As a consequence, all instances of Positive NAE EU$k$SAT$\frac{2^{k-3}}{k}$ are satisfiable. □

## 3.3 NAE E$k$SAT-$k$

**Theorem 3.3.1.** NAE E$k$SAT-$k$ *is in P.*

It was shown by Robertson et al. [RST99] that the problem of detecting if a hypergraph is a **square condenser**, i.e. has $k$ nodes and $k$ hyperedges and is *minimally* non-2-colorable, is in $P$. Examples of square condensers can be seen in Figure 3-1 . We will use it to show that NAE E$k$SAT-$k$ is also in $P$, even when a clause can repeat the same variable, as long as the occurrences of each variable is bounded by $k$.

First let's prove a positive version of Theorem 3.3.1.

**Theorem 3.3.2.** Positive E$k$NAE SAT-$k$ *is in P.*

*Proof.* Let $H$ be the hypergraph representation of the Positive NAE SAT instance. All the clauses are converted to hyperedges that contain $k$ nodes, and all the variables are converted to nodes. We allow hyperedges to contain multiple instances of the same node. This is equivalent to containing a single instance of each node while lowering the degree of the hyperedge to the number of unique nodes in it.

Our original instance is satisfiable if and only if this hypergraph is 2-colorable, i.e. if it is possible to color the nodes in such a way that no hyperedge is monochromatic.

Suppose the hypergraph is not 2-colorable, but it is not a condenser. This means that a strict subset of the hypergraph, $E \subsetneq H$ is a condenser. Since $E$ is a condenser, we have that $|V[E]| \leq |E|$ [Sey74], where $|V[E]|$ are the vertices of $E$. Since each hyperedge contains $k$ nodes (possibly with repetitions), and each node appears at most $k$ times; we know that among the $|E|$ hyperedges there exists at least $|E|$ nodes, so $|V[E]| \geq |E|$. Therefore, $|V[E]| = |E|$, so $E$ is a square condenser. The above also implies that each node appears exactly $k$ times in $|E|$.

Now, suppose one of those node is also part of a hyperedge in $H - E$. Then in $H$ the node will appear in at least $k+1$ locations, which is a contradiction. Therefore any condenser in $H$ is a connected component which is not connected to the rest of $H$. Therefore the hypergraph is not 2-colorable if and only it is a square condenser or if one of its connected components is a square condenser.

Since we have a polynomial time algorithm to check if a hypergraph is a square condenser, and the largest number of connected components in a hypergraph $H$ is $O(|H|)$, we can check if $H$ is 2-colorable in polynomial time. Therefore checking if a Positive E$k$NAE SAT-$k$ instance is satisfiable is in P. □

**Corollary 3.3.3.** Positive EU$k$ NAE SAT-E-$k$ *is in P.*

*Proof.* This follows immediately from Theorem 3.3.2. □

**Corollary 3.3.4.** *2-coloring of $k$-regular-$k$-uniform hypergraphs is in P.*

*Proof.* We convert readily convert an instance of a 2-coloring of $k$-regular-$k$-uniform hypergraph problem into an instance of Positive EU$k$ NAE SAT-E-$k$. Then, this follows from Theorem 3.3.2. □

Now we are ready to solve the problem without the positive restriction.

**Theorem 3.3.1.** NAE E$k$SAT-$k$ *is in P.*

*Proof.* Suppose a variable $x$ appears in $n$ clauses as a positive literal and in $q \leq k - n$ clauses as a negative literal.

We create 2 new variables: $x_1, x_2$, and we add a new clause $(\underbrace{x_1, x_1, \ldots, x_1}_{k-n\text{-times}}, \underbrace{x_2, x_2, \ldots, x_2}_{n\text{-times}})$. This clause enforces $x_1 \neq x_2$. We replace every positive occurrence of $x$ with $x_1$ and every negative occurrence with $x_2$. $x_1$ will occur in $k$ locations and $x_2$ will occur in $n + q \leq k$ locations.

After this we will have converted all negative literals into positive literals, at the cost of increasing the number of variables by at most 2. The original problem is satisfiable if and only if this problem is satisfiable. Since we have proven that Positive E$k$NAE SAT-$k$ with variable repetitions is in P, then E$k$NAE SAT-$k$ is in P. □

## 3.4 Positive NAE 3SAT-3

**Theorem 3.4.1.** Positive NAE 3SAT-3 *is NP-complete.*

*Proof.* We reduce from NAE 3SAT. If a variable $x$ appears in $\leq k$ times positively and $\leq k$ times negatively, replace it with variables $x_1, x_1', x_2, \ldots, x_k, x_k'$, and force all $x_i$s equal and opposite from all $x_i'$s by adding a cycle of NAE clauses:

$$\text{NAE}(x_i, x_i') \quad (\text{i.e., } x_i \neq x_i') \quad \text{and} \tag{3.1}$$

$$\text{NAE}(x_i', x_{i+1}) \quad (\text{i.e., } x_i' \neq x_{i+1}) \quad \text{for } i \in \{1, 2, \ldots, k\} \tag{3.2}$$

(indices modulo $k$).

Then we replace each occurrence of $x$ with an $x_i$, and each occurrence of $\neg x$ with and $x_i'$ . Each $x_i$ and $x_i'$ will occur at most 3 times. □

## 3.5 Positive NAE E3SAT-4

**Theorem 3.5.1.** Positive NAE E3SAT-4 *is NP-complete.*

*Proof.* We perform a reduction from Positive NAE 3SAT-3, which we showed to be NP-complete on Theorem 3.4.1.

Given an instance of Positive NAE 3SAT-3, suppose a clause contains only two variables NAE$(x, y)$, we create the following clauses: NAE$(x, x_1, x_1)$, NAE$(x_1, x_1, x_2)$ and NAE$(x_2, x_2, y)$. This forces $x = x_2$, and is equivalent to NAE$(x_2, y)$. Each newly created variable appears 4 or fewer times, and each of the original variables appear at most 3 times, since they were not duplicated in the reduction. □

## 3.6 1-Planar Positive NAE 3SAT

**Theorem 3.6.1.** 1-Planar Positive NAE 3SAT *is NP-complete.*

*Proof.* We reduce from Positive NAE 3SAT. If an edge coming from a variable $x_i$ crosses $k$ other edges, we create $2k$ copies of the variable $x$, i.e $x_1, x_2, \ldots, x_{2k}$, together with $2k$ clauses, 2 for each crossing. The clauses will force all the $x$s with even index to be equal. NAE$(x, x_1)$, NAE$(x_1, x_2)$, …, NAE$(x_{2k-1}, x_{2k})$. At last we substitute $x$ for $x_{2k}$ in the original clause. We show the results of this procedure for a single crossing in Figure 3-2.

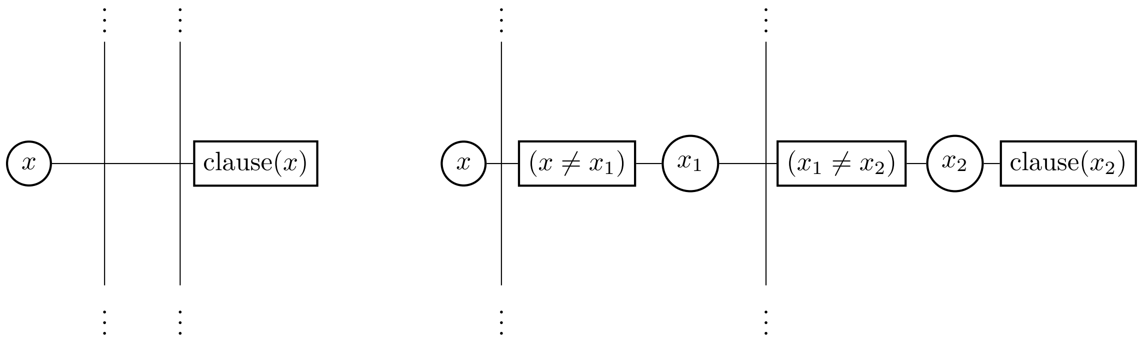
The original problem will have a solution iff this one has a solution.

Each of the newly created edges are part of at most one crossing.

If we repeat this in every edge of the graph, we will obtain a graph that is 1-planar. We can do this in polynomial time, since there are at most $|E|^2$ crossings. Therefore 1-Planar Positive NAE 3SAT is NP-complete. □

## 3.7 Crossing number $n^\epsilon$ NAE 3SAT

We define "Crossing number $n^\epsilon$ NAE 3SAT" to be an instance of NAE 3SAT, given with a graph representation that contains at most $n^\epsilon$ edge crossings.



(a) Edge with multiple crossings (b) Edges with at most one crossing

Figure 3-2: Converting Positive NAE SAT to 1-Planar Positive NAE SAT

**Theorem 3.7.1.** Crossing number $n^{\epsilon}$ NAE 3SAT *is NP-complete.*

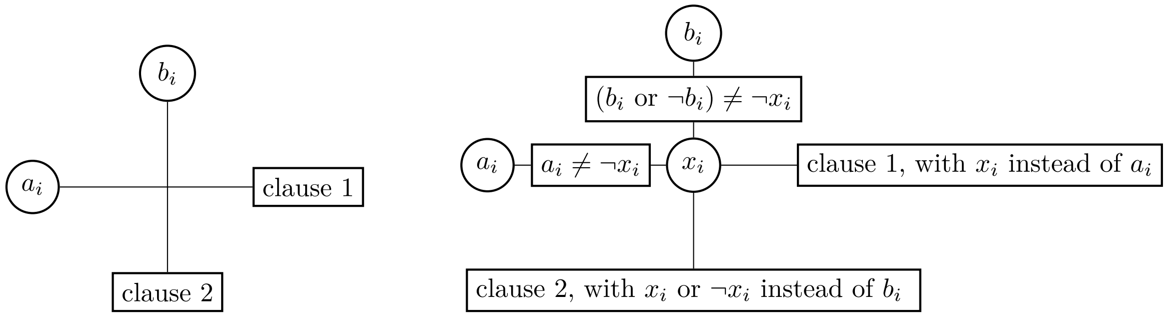
*Proof.* By Theorem 3.6.1, Crossing number $n^1$ NAE 3SAT is NP-complete. We can convert any instance of Crossing number $n^1$ NAE 3SAT to a Crossing number $n^{\epsilon}$ NAE 3SAT by adding a trivially solvable $n^{\frac{1}{\epsilon}}$ sized planar 3SAT instance to the graph. □

## 3.8 Crossing number $O(\log n)$ NAE 3SAT

**Theorem 3.8.1.** Crossing number $O(\log n)$ NAE 3SAT *is in P.*

*Proof.* Let $c$ be the crossing number. We can reduce our problem to $2^c = n^{O(1)}$ instances of Planar NAE 3SAT which is in P. Our problem is satisfiable iff at least one of these instances is satisfiable. The reduction is as follows:

Suppose 2 edges cross each other, as seen in Figure 3-3 (a), for example the one connecting variable $a_i$ to clause $c_1$ and the one connecting $b_i$ with $c_2$. We will duplicate the number of instances, at the intersection point add 2 clauses and 1 new variable: $\text{NAE}(a_i, \neg y), \text{NAE}(b_i, y'), y$, where $y'$ is $\neg y$ in the first copy of the instances, and $y$ in the second copy. We replace $a_i$ and $b_i$ by $y$ in clauses $c_1$ and $c_2$ in the first copy, and by $y$ and $\neg y$ in the second copy, respectively. This removes the crossing from the graph, as seen in Figure 3-3 (b).



$b_i$
$a_i$
clause 1
clause 2
$b_i$
$(b_i$ or $\neg b_i) \neq \neg x_i$
$a_i$
$a_i \neq \neg x_i$
$x_i$
clause 1, with $x_i$ instead of $a_i$
clause 2, with $x_i$ or $\neg x_i$ instead of $b_i$

(a) Crossing of 2 edges (b) Doubling the instances

Figure 3-3: Removing a crossing of 2 variables by doubling the instances

By doing this we will be solving 2 problems: is the original instance satisfiable if $a_i = b_i$, and is the original instance satisfiable if $a_i \neq b_i$.

Since there are $O(\log(n))$ crossings, and for each crossing we are duplicating the number of instances, we will solve a polynomial $n^{O(1)}$ number of problems in P, and therefore our original problem is also in P. $\square$

## 3.9 Bounded-genus Orientable Surface NAE 3SAT

**Theorem 3.9.1.** BOUNDED-GENUS ORIENTABLE SURFACE NAE 3SAT *is in P.*

*Proof.* We define the problem as follows: given a NAE 3SAT formula, together with an embedding of the formula onto an orientable surface of genus $k$, where $k$ is a constant, is the formula satisfiable?

Moret [Mor88] showed a reduction from NAE 3SAT to MAX CUT which preserves planarity. Galluccio et al. [GLV01] showed that MAX CUT is in P for any fixed orientable surface of genus bounded by a constant. Therefore, if given an instance of NAE 3SAT and an embedding of its variable clause graph onto an orientable surface of bounded genus, we can use the planarity preserving reduction to reduce it to MAX CUT in an orientable surface of bounded genus. Therefore BOUNDED-GENUS ORIENTABLE SURFACE NAE 3SAT is in P. $\square$

## 3.10 Exists Exists-Unique NAE 3SAT

**Definition** (Exists Exists-Unique NAE 3SAT)**.** Given a 3CNF formula $\phi(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j)$, **Exists Exists-Unique NAE 3SAT ($\exists\exists!$ NAE 3SAT)** is the problem of deciding whether there exists an assignment of $x_1, x_2, \ldots, x_i$ such that there is a unique assignment for $y_1, y_2, \ldots, y_j$ such that every clause in $\phi$ have both TRUE and FALSE literals. **Exists Exists-Unique NAE 3SAT$_c$ ($\exists\exists!$ NAE 3SAT$_c$)** is the problem of deciding whether there exists such assignment, if the formula may also contain 1 variables whose value is set to $T$ and one whose value is set to $F$.

**Theorem 3.10.1.** EXISTS EXISTS-UNIQUE NAE 3SAT$_c$ *is $\Sigma_2^P$-hard.*

*Proof.* We reduce from EXISTS EXISTS-UNIQUE 3SAT, by using a similar reduction as used on [BLS12].

We convert a 3CNF formula to one to be used by the NAE-3SAT$_c$. We create $k$ new variables, one for each clause, $z_1, z_2, \ldots, z_k$. We replace $(a_k \vee b_k \vee c_k)$ with the following 4 clauses: $\text{NAE}(a_k, b_k, z_k) \wedge \text{NAE}(\neg z_k, c_k, F) \wedge \text{NAE}(a_k, z_k, T) \wedge \text{NAE}(b_k, z_k, T)$.

There is exactly one possible value for $z_k$ i.e. $a_k \vee b_k$, and the above 4 formulas enforce $(a_k \vee b_k \vee c_k)$. Therefore this is a reduction from 3SAT to NAE 3SAT$_C$ that preserves the number of solutions.

Therefore we can convert an instance of EXISTS EXISTS-UNIQUE 3SAT, if we convert its 3CNF formula $\phi(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j)$, as stated above, and we insert all the newly created variables in the second set of variables, i.e. on the list of $y_i$s, since there is exactly one valid value that the new variables can take. This can clearly be done in polynomial time, and at the end we have an instance of $\phi(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j, z_1, z_2, \ldots, z_k)$ EXISTS EXISTS-UNIQUE NAE 3SAT. This instance will be satisfiable iff the original problem was satisfiable. $\square$

**Theorem 3.10.2.** EXISTS EXISTS-UNIQUE NAE 3SAT *is $\Sigma_2^P$-hard.*

*Proof.* We reduce from EXISTS EXISTS-UNIQUE NAE 3SAT$_{\text{C}}$. Given an instance of $\phi(x_1, x_2, \ldots, x_i, y_1, y_2, \ldots, y_j)$ EXISTS EXISTS-UNIQUE NAE$_{\text{C}}$ 3SAT}, we create a new clause $(T, T, F)$, which forces $T$ and $F$ to be different. We then add $T$ and $F$ to the set of $x_i$ variables.

Suppose the original problem was satisfiable. Then, setting the variables $x_i$ to the solution to the original instance and $T$ to TRUE, will generate enforce that there exists a unique solution to the remaining of the NAE 3SAT formula.

Suppose that this problem is satisfiable. If in the solution we set $T$ to TRUE, then the exact same assignment is a solution to the original instance. Suppose we set $T$ to FALSE. Then since the NAE relation is dual, if we negate all variables, we have another valid solution with $T$ set to TRUE, and therefore the original problem is also satisfiable. □

# Chapter 4

# $S$-in-$k$SAT variants

In this chapter we present results on 1-IN-$k$SAT and $S$-IN-$k$SAT variants. We start with a problem that is in P, followed by the characterization of $S$-in-E$k$SAT and results on $S$-IN-E4SAT-$O(1)$. At last we show that $\{0, 1, k-1, k\}$-IN-EU$k$SAT is NP-complete, which is an interesting result since the same problem with a planarity restriction is in P, as seen in Theorem 6.7.9.

## 4.1 1-in-$k$SAT-2

**Theorem 4.1.1.** 1-IN-$k$SAT-2 *is in P.*

*Proof.* We will reduce the problem to MAXIMUM WEIGHT MATCHING, which was shown by Edmonds [Edm65] to be in P.

1. The monotone variables that appear twice can be regarded as edges of weight 2 connecting 2 clauses. If the variable appeared as positive literals, the edge being in the matching means that the variable was set to TRUE. If negative, then to FALSE.
2. The variables that appear as both positive and negative literals can be regarded as 2 edges of weight 1 that share one dummy node, and connect 2 clauses. Which edge is matched indicates if the variable was set to TRUE or FALSE.
3. The variables that appear only once can be regarded as an edge of weight 1 connecting a clause to a dummy node. It indicates being set to TRUE or FALSE similarly to the monotone variables in item 1.
4. The clauses can be regarded as nodes of degree $k$, whose edges are the nodes in it connecting it. We will call those nodes **clause-nodes**.

The edges connecting 2 clause-nodes have weight 2. The edges that are part of a single clause-node have weight one. There are no other types of edges.

If there is a solution $E$ of weight $n$, this means that $n$ clause-nodes are connected to at least one edge in $E$. Since it is a match, this means that $n$ clause-node are connected to exactly one edge in $E$, and therefore each of those clause contain exactly 1 TRUE literal. If $n$ is the number of clause nodes, it is not possible to have a total weight greater than $n$.

If there was a solution to the 1-IN-$k$SAT-2 that contained $n$ clauses, then the edges associated with the variable choices form a matching $E$ with weight $n$. □

**Corollary 4.1.2.** Max 1-in-$k$SAT-2 *is in P.*

*Proof.* The maximum weight matching from the proof above has weight equal to the maximum number of clauses that can be simultaneously satisfiable. □

## 4.2 $S$-in-E$k$SAT characterization

For each subset $S$ of $\{0, 1, 2, \dots k\}$, define $S$-in-E$k$SAT to be the problem of deciding whether $n$ Boolean variables $x_1, \dots, x_n$ can be set to satisfy all given clauses, where each clause involves $k$ literals (each of the form $x_i$ of $\neg x_i$) and requires that, among these $k$ literals, the number that evaluate to True is contained in $S$. For example, $\{1, 2, 3\}$-E3SAT is the standard E3SAT problem: each clause is satisfied if 1, 2, or 3 literals are True.

We give a characterization of which $S$-in-E$k$SAT, for $k \geq 3$, are NP-complete and which are in P We show that (assuming P $\neq$ NP) $S$-in-E$k$SAT, is in P if and only if $S$ is one of the following:

(a) $\{0\}$

(b) $\{k\}$

(c) $\{0, k\}$

(d) $\{0, 1, 2, \dots, k\}$

(e) $\emptyset$

(f) $\{\text{even}\}$ and $\{\text{odd}\}$

(g) $\{k - 1, k\}$

(h) $\{0, 1\}$

**Lemma 4.2.1.** *If $S$ is one of the above, then $S$-in-E$k$SAT is in P.*

*Proof.* The clauses of the $\{0\}$ case can be written as a CNF formula where every conjunct contains at most one literal, since every single literal has to be set to False. By Schaefer's dichotomy, this is in $P$. By symmetry, so is $\{k\}$.

The clauses of the $\{0, k\}$ case are satisfied iff all literals are equal. Therefore, each clause, $\{0, k\}(x_1, x_2, \dots, x_k)$, can be rewritten as a cycle of clauses with 2 variables, $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3) \dots (x_{k-1} \vee \neg x_k) \wedge (x_k \vee \neg x_1)$. Since 2SAT is in P so is $\{0, k\}$-in-E$k$SAT

For $\{0, 1, 2, \dots, k\}$, any assignment will satisfy. For $\emptyset$, no assignment will satisfy. Therefore both are in P.

$\{\text{even}\}$ and $\{\text{odd}\}$ can be represented by linear equations modulo 2, which is in P.

$\{k - 1, k\}$ reduce to 2SAT with $x_i \vee x_j$ for all pairs $i, j$ in the clause $\{0, 1\}$ is easy by symmetry □

Now we can prove our main theorem by induction.

**Theorem 4.2.2.** *Assuming $P \neq NP$, iff*

$$S \in \{\emptyset, \{0\}, \{k\}, \{0, k\}, \{k - 1, k\}, \{0, 1\}, \{even\}, \{odd\}, \{0, 1, 2, \dots, k\}\}$$

*and $k \geq 3$, then $S$-in-E$k$SAT is in $P$. Otherwise, it is NP-complete.*

We start by proving the base case for $k = 3$.

**Lemma 4.2.3.** *For $k = 3$, iff $S$ is not one of the sets from the list above, then $S$-IN-E3SAT is NP-complete.*

*Proof.* We have already proved above that the following 9 cases $\{0, 1, 2, 3\}$, $\{\}$, $\{0\}$, $\{3\}$, $\{2, 3\}$, $\{0, 1\}$, $\{0, 2\}$, $\{1, 3\}$ and $\{0, 3\}$ are in $P$.

The other 7 cases are problems known to be NP-complete.

- $\{1\}$ and $\{2\}$ are 1-IN-E3SAT. Exactly 1 literal is TRUE, or alternatively exactly 1 is FALSE.
- $\{1, 2, 3\}$ and $\{0, 1, 2\}$ are E3SAT. At least 1 literal is TRUE/FALSE.
- $\{1, 2\}$ is NAE-E3SAT, as we forbid all literals from being TRUE or all from being FALSE.
- $\{0, 2, 3\}$ and $\{0, 1, 3\}$ are NOT EXACTLY-1-IN-E3SAT (Theorem 6.1.2), not exactly one literal is TRUE/FALSE.

□

For the induction step, lets first prove the following Lemmas:

**Lemma 4.2.4.** *If $S$ is not self-dual, we can force a variable to be* TRUE.

*Proof.* If $S$ is not self-dual, then there is an $i \in Y$ such that $k - i \notin S$. Then the clause

$$\underbrace{T + \cdots + T}_{i} + \underbrace{\neg T + \cdots + \neg T}_{k-i} \in S$$

constructs a TRUE value $T$. □

**Lemma 4.2.5.** *If $S$ is self-dual, an instance of $S$-IN-E$k$SAT is satisfiable iff it is satisfiable even when a single arbitrary chosen variable is set to* TRUE.

*Proof.* Suppose an instance of $S$-IN-E$k$SAT is satisfiable. Consider one possible assignment of the variables that satisfies that instance. Choose one of the variables and call it $T$. If this variable is not set to TRUE, then, since $S$ is dual, negating all the variables will result in another solution to the problem where $T$ is TRUE. The reverse direction is trivial. □

With Lemmas 4.2.4 and 4.2.5, we show that in $S$-IN-E$k$SAT instance we can always construct or define a variable $T$ such that the instance will be satisfiable iff it is satisfiable when $T$ is TRUE, and by consequence, when $\neg T$ is FALSE. We will call $\neg T$ as $F$.

Now we will prove the rest of the induction to prove Theorem 4.2.2.

**Theorem 4.2.2.** *Assuming $P \neq NP$, iff*

$$S \in \big\{\emptyset, \{0\}, \{k\}, \{0, k\}, \{k-1, k\}, \{0, 1\}, \{even\}, \{odd\}, \{0, 1, 2, \ldots, k\}\big\}$$

*and $k \geq 3$, then $S$-IN-E$k$SAT is in $P$. Otherwise, it is NP-complete.*

*Proof.* If $S$-IN-E$k$SAT is hard, then so are

1. $S$-IN-E$(k + 1)$SAT: add $F$ to each one of the clauses.

2. $(S \cup \{k+1\})$-IN-E$(k+1)$SAT: same reduction as above. Because $F = false$, obtaining $k+1$ TRUE variables in a clause is impossible, since at least one variable will be FALSE.

3. $\{s+1 \mid s \in S\}$-IN-E$(k+1)$SAT: add $T$ to each of the clauses, where $T$ is constructed as above.

4. $\{0\} \cup \{s+1 \mid s \in S\}$-IN-E$(k+1)$SAT: same reduction as above. Because $T =$TRUE, obtaining 0 TRUE variables in a clause is impossible.

Now we prove the other cases to be NP-hard:

**Case 1:** $k \notin S$. Then because of 1, we we only need to consider $S$'s that are easy for $k-1$ but not for $k$. These are:

- $\{k-1\}$: Hard by applying 3 to $\{k-2\}$-IN-E$(k-1)$.
- $\{0, k-1\}$: Hard by applying 4 to $\{k-2\}$-IN-E$(k-1)$ and adding 0.
- $\{k-2, k-1\}$: Hard by applying 3 to $\{k-3, k-2\}$-IN-E$(k-1)$

**Case 2:** $k \in S$. Then because of 2, we only need to consider cases where $S - \{k\}$ is easy for $k-1$ but $S$ is not easy for $k$. These are:

- $\{k-2, k-1, k\}$: Hard by applying 3 to $\{k-3, k-2, k-1\}$-IN-E$(k-1)$SAT
- $\{0, 1, k\}$: Hard by reduction from $\{0, 1, k-1\})$-IN-E$(k-1)$SAT by converting each clause $a_1 + \cdots + a_{k-1} \in \{0, 1, k-1\}$ into:
  - Variable $X$ local to this clause
  - $a_1 + \cdots + a_{k-1} + X \in \{0, 1, k\}$
  - $a_i + X + \cdots + X \in \{0, 1, k\}$ for each $i$
  - Iff $X$ is TRUE, then all $a_i$s must be TRUE (by second set of constraints).
  - Iff $X$ is FALSE, then at most one $a_i$ must be TRUE (by first constraint).

This finishes the proof for the cases where $S$-IN-E$k$SAT is NP-complete, and we showed in Lemma 4.2.1 the cases where $S$-IN-E$k$SAT are in $P$. □

## 4.3 $S$-in-E4SAT-$O(1)$

The following are known:

- $\{1, 2, 3\}$-IN-E3SAT-4 is NP-complete, because of E3SAT-4 [Tov84].
- $1, 2$-IN-E3SAT-4 is NP-complete, because NAE E3SAT-4 is NP-hard, Theorem 3.5.1.
- 1-IN-E3SAT-3 is NP-complete [Lar93].
- 2-IN-E3SAT-3 is NP-complete, by symmetry from 1-IN-E3SAT-3.
- $0, 2, 3$-IN-E3SAT-3 is NP-complete, because NOT 1-IN-EU3SAT-3 is NP-hard, Theorem 6.1.2.
- $0, 1, 3$-IN-E3SAT-3 is NP-complete, by replacing 1s and 0s, we are back to the previous case.

**Theorem 4.3.1** ($\{S\}$-IN-E4SAT-$O(1)$ hardness.). *The following are NP-complete*

- $\{1\}$-IN-E4SAT-4
- $\{2\}$-IN-E4SAT-5
- $\{3\}$-IN-E4SAT-4
- $\{0,2\}$-IN-E4SAT-3
- $\{1,2\}$-IN-E4SAT-4
- $\{2,3\}$-IN-E4SAT-4
- $\{0,1,3\}$-IN-E4SAT-4
- $\{0,1,4\}$-IN-E4SAT-6
- $\{0,2,3\}$-IN-E4SAT-3
- $\{0,3,4\}$-IN-E4SAT-6
- $\{1,2,3\}$-IN-E4SAT-5
- $\{0,1,3,4\}$-IN-E4SAT-5
- $\{0,2,3,4\}$-IN-E4SAT-5
- $\{1,2,3,4\}$-IN-E4SAT-5

*Proof.* We will consider 3 cases, when $S$ is not dual and $\{0,4\} \cap S = \emptyset$, when $S$ is not dual and $\{0,4\} \cap S \neq \emptyset$, and at last when $S$ is dual.

*Case* 1 ($S$ is not dual and $\{0,4\}\cap$S$=\emptyset$). We can modify Lemma 4.2.4 slightly to create a TRUE and a FALSE variable.

*Lemma* 4.3.2. *If $S$ is not self-dual, and $0 \notin S$ and $k \notin S$, we can force a variable to be* TRUE *and a variable to be* FALSE *in $S$-IN-$k$SAT.*

*Proof.* If $S$ is not self-dual, then there is an $i \in Y, i \neq 0, i \neq k$ such that $k - i \notin S$. Then the clause

$$\underbrace{T + \cdots + T}_{i} + \underbrace{F + \cdots + F}_{k-i} \in S$$

constructs a TRUE value $T$ and a FALSE value F, because since $k \notin S$, both variables cannot be TRUE, and since $0 \notin S$, both variables cannot be FALSE. □

By Lemma 4.3.2, we can construct a TRUE variable and a FALSE variable. the clause that forces them to be TRUE and FALSE will use exactly 4 of their occurrences, and, without loss of generality, at most 2 instances of the variable that appears fewer times. If this variable was the one set to F, we can simply negate it in the clause in order to construct another TRUE variable.

Therefore, we can construct a TRUE variable while using at most 2 occurrences of the variable, and, by symmetry, construct F while using at most 2 occurrences of the variable. For each time we want to use a

TRUE or FALSE variable we can just repeat this construction, and this guarantees that none of our variables forcibly set to TRUE or FALSE will occur more than 3 times.

Therefore, by using the constructions from the proof of Theorem 4.2.2:

- $\{1\}$-IN-E4SAT-4 is NP-complete (from adding F to $\{1\}$-IN-E3SAT-4)
- $\{3\}$-IN-E4SAT-4, is NP-complete (from adding T to $\{2\}$-IN-E3SAT-4)
- $\{1, 2\}$-IN-E4SAT-4 is NP-complete (from adding F to $\{1, 2\}$-IN-E3SAT-4)
- $\{2, 3\}$-IN-E4SAT-4 is NP-complete (from adding T to $\{1, 2\}$-IN-E3SAT-4)

*Case* 2. In this case we can force a variable to be TRUE (or FALSE) by using a specific clause for each case. For each clause we will construct a different TRUE (or false) value. We list the clause we use to construct the TRUE (or FALSE) variable next to the particular cases. We use '_' to denote a (repeated) dummy variable unique to that clause.

- $\{1, 2, 3, 4\}$-IN-4SAT-5 is NP-complete (from adding F to $\{1, 2, 3\}$-IN-E3SAT-4,$(\neg F, \neg F, \neg F, \neg F)$)
- $\{0, 2, 3, 4\}$-IN-4SAT-5 is NP-complete (from adding T to $\{1, 2, 3\}$-IN-E3SAT-4 $(T, T, T \neg T)$)
- $\{0, 3\}$-IN-E4SAT-3 is NP-complete (from adding T to $\{2\}$-IN-E3SAT-3, $(\neg T, \neg T, \_, \_)$)
- $\{0, 2\}$-IN-E4SAT-3 is NP-complete (from adding T to $\{1\}$-IN-E3SAT-3, $(\neg T, F, F, F)$)
- $\{0, 1, 3\}$-IN-E4SAT-4 is NP-complete (from adding F to $\{0, 1, 3\}$-IN-EU3SAT-3, $(\neg T, \neg T, F, F)$)
- $\{0, 2, 3\}$-IN-E4SAT-3 is NP-complete (from adding F to $\{0, 2, 3\}$-IN-EU3SAT-3, $(F, \_, \_, \_)$)

The $\{0, 1, 4\}$ case is harder. First notice that we can copy a variable by adding the following clause: $(x, x, x', x')$. This forces $x$ and $x'$ to be the same. Therefore, we can reduce from $\{1\}$-IN-E3SAT-3, by converting a $(a, b, c)$ clause into the following ones: $(a', a', \neg b', \neg c')$, $(b', b', \neg a', \neg c')$, $(c', c', \neg a', \neg b')$, $(a, a, a', a')$, $(b, b, b', b')$, $(c, c, c', c')$. For each clause, we create 3 new variables, each of which occurs 6 times, and we duplicate the number of occurrences of the original variable, so each variable appears at most 6 times. Therefore:

- $\{0, 1, 4\}$-IN-E4SAT-6 is NP-complete.
- $\{0, 3, 4\}$-IN-E4SAT-6 is NP-complete, by symmetry.

*Case* 3 ($S$ is dual). If $S$ is self-dual, then it is one of $\{1, 2, 3\}$, $\{2\}$, $\{0, 1, 3, 4\}$. In the first 2 cases, we can force 2 variables to be the same by adding a clause $(x, x, \neg y, \neg y)$. Therefore, if we create a series of clauses $(x_0, x_0, \neg x_1, \neg x_1), (x_1, x_1, \neg x_2, \neg x_2) \ldots$, we can create $n$ variables, all with the same value, that appear exact 4 times each.

If we define them to be $T$ (or $F$), as done in the proof of Theorem 4.2.2, and add one of them to each clause, we then show that

- $\{1, 2, 3\}$-IN-E4SAT-5 is NP-complete (from adding F to $\{1, 2, 3\}$-IN-E3SAT-4)
- $\{2\}$-IN-E4SAT-5 is NP-complete (from adding F to $\{2\}$-IN-E3SAT-3)

For the last case, $\{0, 1, 3, 4\}$, we do exactly the same, but we instead use a series of clauses of the form $(x_0, x_0, x_1, x_1)$, $(x_1, x_1, x_2, x_2)$, $\ldots$, instead of alternating positive and negative literals. Therefore

- $\{0,1,3,4\}$-IN-E4SAT-5 is NP-complete (from adding T to $\{0,2,3\}$-IN-E3SAT-4)

□

## 4.4 $\{0,1,k-1,k\}$-in-EU$k$SAT

**Theorem 4.4.1.** $\{0,1,k-1,k\}$-IN-EU$k$SAT *is NP-complete.*

*Proof.* We reduce from $\{0,1,k-1,k\}$-IN-E$k$SAT, which is NP-complete (Theorem 4.2.2).

Each variable can appear at most $k$ times in an instance of $\{0,1,k-1,k\}$-IN-EU$k$SAT. If all $k$ appearances are in a single clause and 0, 1, $k-1$ or $k$ of them are positive literals, we can delete that clause because it is always satisfiable and delete that variable because it does not appear in any other clause. If all $k$ appearances are in a single clause with some other number of positive literals, our instance is unsatisfiable so our reduction produces an arbitrary unsatisfiable instance of $\{0,1,k-1,k\}$-IN-E$k$SAT.

For each remaining variable $x$ (appearing in a single clause at most $k-1$ times), we create the following 2 $\{0,1,k-1,k\}$-IN-EU$k$ clauses:

$$(x, x_1, x_2, x_3, \ldots, x_{k-1})$$
$$(\neg x, x_1, x_2, x_3, \ldots, x_{k-1})$$

These clauses force variables $x_1$ through $x_{k-1}$ to have the same value, regardless of the value of $x$. Then, we replace every occurrence of $x$ from the $\{0,1,k-1,k\}$-IN-E$k$SAT clauses by one of the $x_i$s, such that no clause contains 2 or more of the same $x_i$. This concludes the reduction from $\{0,1,k-1,k\}$-IN-E$k$SAT to $\{0,1,k-1,k\}$-IN-EU$k$SAT. □

# Chapter 5

# Max SAT variants

In this chapter we present results on Max SAT variants. Most of the results are in Planar Max SAT variants, and so this Chapter makes extensive use of figures to show that the reductions preserve planarity.

## 5.1 Max All Equal EU3SAT

**Theorem 5.1.1.** Max All Equal EU3SAT *is NP-complete.*

*Proof.* We reduce from Min EU2SAT [KKM94].

Given an instance $\phi$, $k$ of Min EU2SAT, we add a variable $z$ to every clause and we set $k' = m - k$ to obtain an instance $\phi'$, $k'$ of Max All Equal EU3SAT. So every clause in $\phi'$ will be of the form ALL EQUAL$(x_i, x_j, z)$.

Suppose that there is a solution that satisfies $k'$ of the clauses; this happens iff it is possible to set the variables of $k'$ of the original instance clauses to all be equal to $z$. If we define $z$ to be 0 (which we can do since negating all variables doesn't change the number of satisfied clauses in Max All Equal SAT), $k'$ or more clauses will be satisfied in the Max All Equal EU 2SAT instance iff there exists an assignment that satisfies $k$ or fewer clauses in the original instance. □

## 5.2 Max Planar 2SAT

Garey, Johnson and Stockmeyer [GJS76] gave a reduction from EU3SAT to Max 2SAT. Here we propose a small change to the reduction to make it parsimonious, and we show that this new reduction preserves planarity.

**Theorem 5.2.1.** Max 2SAT *is NP/ASP/#P-complete.*

*Proof.* Garey, Johnson and Stockmeyer [GJS76] reduction from EU3SAT to Max 2SAT creates a new variable for each clause, $(a_i, b_i, c_i)$, and replaces it with 10 new clauses: $(a_i)$, $(b_i)$, $(c_i)$, $(d_i)$, $(\neg a_i, \neg b_i)$, $(\neg a_i, \neg c_i)$, $(\neg b_i, \neg c_i)$, $(a_i, \neg d_i)$, $(b_i, \neg d_i)$, $(c_i, \neg d_i)$. And the result is an instance of Max 2SAT, where $7m$ clauses can simultaneously satisfied, iff the original $m$ clauses could be satisfied.

This reduction was not parsimonious, because if in a certain clause $a_i = b_i = c_i = 1$, 7 of the new clauses would be satisfied independent of the value of $d_i$.

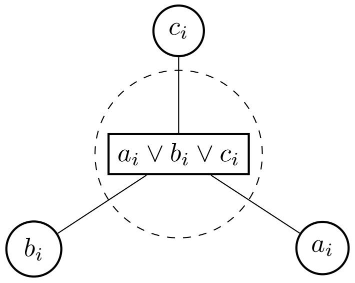


Figure 5-1: A 3SAT clause.

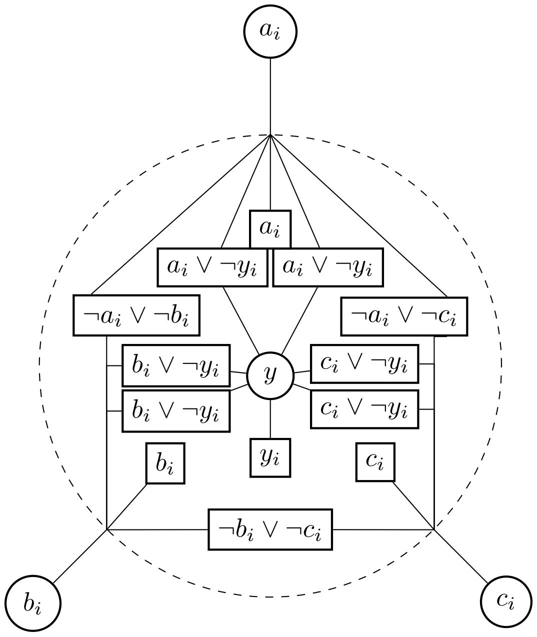


Figure 5-2: Converting the simple clause from PLANAR 3SAT to clauses of the MAX 2SAT instance

If we add 3 extra clauses, for a total of 13 clauses, $(a_i, \neg d_i)$, $(b_i, \neg d_i)$, $(c_i, \neg d_i)$, i.e. we repeat 3 of the above clause, then we have a parsimonious reduction. For each clause, we get that 10 of the newly created clauses can be satisfied, iff the original clause was satisfied. Furthermore, there is only one assignment of the variables that results in 10 being satisfied, and it is not possible to satisfy more than 10 clauses. This can be easily verified, and is left as an exercise for the reader.

Therefore $10m$ clauses can simultaneously satisfied, iff the original $m$ clauses could be satisfied; and the reduction is parsimonious. Therefore MAX 2SAT is NP/ASP/#P-complete. □

**Lemma 5.2.2.** *The above reduction preserves planarity.*

*Proof.* This can be seen on Figures 5-1 and 5-2. Each clause is substituted by a cluster, and the cluster does not affect other clauses. □

**Theorem 5.2.3.** MAX PLANAR 2SAT *is NP/ASP/#P-complete.*

*Proof.* Since PLANAR EU3SAT [III+98] is NP/ASP/#P-complete, and the above reduction preserves planarity, PLANAR MAX 2SAT is NP/ASP/#P-complete. □

**Theorem 5.2.4.** Max Planar EU2SAT *is NP-complete.*

*Proof.* We use the same reduction from Max Planar 2SAT to Max Planar EU2SAT used in the original paper [GJS76]. Given an instance $\Phi(\boldsymbol{X}), k$ of Max Planar 2SAT, suppose that there exists $q$ clauses with exactly 1 variable (either a clause with the same literal twice, or a clause with a single literal, since clauses with a positive and negative literal can simply be removed since they are always satisfiable, as long as we decrease the value of $k$).

We replace each clause $(a_i, a_i)$ or $(a_i)$ with:

$$(a_i, y_i), \quad (a_i, \neg y_i)$$

, where $y_i$ is a variable unique to that clause.

At last we increase the value $k$ to $k + q$. This reduction clearly preserves planarity; however, it is not parsimonious. □

**Theorem 5.2.5.** Min Planar EU2SAT *is NP-complete.*

*Proof.* Kohli et al. [KKM94] reduction of Max EU2SAT to Min EU2SAT preserved planarity. It consisted of splitting each clause $(a_i, b_i)$ into 2 clauses $(\neg a_i, y_i)$, $(\neg b_i, \neg y_i)$, where $y_i$ is a variable unique to that clause. Therefore, since Max Planar EU2SAT is NP-complete, so is Min Planar EU2SAT. □

## 5.3 Max Planar Xor EU2SAT

**Theorem 5.3.1.** Max Planar Xor EU2SAT *is in P.*

We reduce the problem to Planar Weighted Max-Cut, which is known to be in P [SWK90].

**Definition (Planar Weighted Max-Cut).**
*Instance*: An undirected connected planar graph $G = (V, E)$,and a weight $w(e)$ for each edge $e \in E$.
*Question*: What is the value of the weighted maximum cut? The value of a cut is, given a partition of a graph into 2 sets, the sum of the weights of the edges connecting the two sets. The value of the weighted max-cut is the max weight of a cut over all partitions into 2 non-empty subsets.

*Proof.* Let $x_i$ be variables. Construct the following graph, whose nodes are the variables:

- For each clause $(x_i \oplus x_j)$ or $(\neg x_i \oplus \neg x_j)$, add an edge $(x_i, x_j)$ with weight $-1$;
- For each clause $(\neg x_i \oplus x_j)$ or $(x_i \oplus \neg x_j)$, add an edge $(x_i, x_j)$ with weight 1.

Then, this instance of Weighted Max Cut is equivalent to our instance of Max Planar Xor EU2SAT: if there are $k$ edges with weight $-1$ and the weighted max cut has value $C$, then the maximum number of clauses satisfiable is $k + C$. Since this problem is in P [SWK90], so is Max Planar Xor EU2SAT. □

**Corollary 5.3.2.** Max Planar Xnor EU2SAT *is in P.*

*Proof.* By negating one literal in each clause, we reduce an instance of Xor SAT to Xnor SAT. □

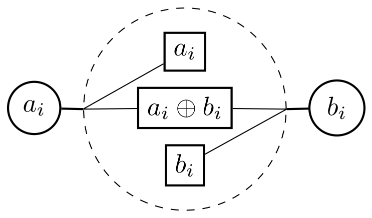


Figure 5-3: Converting a 2CNF clause to 2 XOR clauses.

## 5.4 Max Planar Xor 2SAT

**Theorem 5.4.1.** MAX PLANAR XOR 2SAT *is NP/ASP/#P-complete.*

*Proof.* We can reduce from MAX PLANAR 2SAT, Theorem 5.2.3.

For each clause with a single variable, we duplicate them. For each clause $(a_i \vee b_i)$ with 2 variables, replace it with $(a_i)$, $(b_i)$, $(a_i \oplus b_i)$. This reduction preserves planarity, as seen in Figure 5.4.2.

Exactly 2 of those clauses will be satisfied iff at least one of the variables are TRUE, otherwise only one clause will be satisfied. Therefore, at least $m$ clauses will be satisfied in the original instance, if at least $2m$ clauses are satisfied in the MAX PLANAR XOR 2SAT instance. In particular, if $m$ is the maximum number of clauses that can be satisfied in the original problem, $2m$ will be the maximum in the new instance, and there will be a 1 to 1 correspondence between variable assignments in both problems. This shows that this problem is ASP-hard. Furthermore, if we combine this reduction with the reduction from PLANAR 3SAT to to MAX PLANAR 2SAT, it shows that this problem is #P-hard, since the number of solutions will be the same as in the original PLANAR 3SAT instance. □

**Corollary 5.4.2.** MAX PLANAR XNOR 2SAT *is NP/ASP/#P-complete.*

*Proof.* We replace each clause with a single variable with a clause with a single variable, but negated. Then for each clause $(x_i \vee x_j)$, we replace it with $(\neg x_i)$, $(\neg x_j)$, $(x_i \not\oplus x_j)$. The remaining of the proof is the same. □

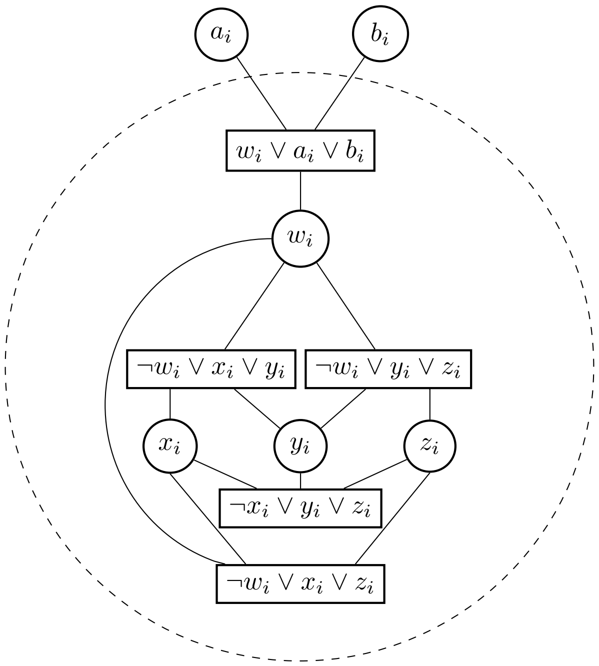
## 5.5 Max Planar Xor EU3SAT

**Theorem 5.5.1.** MAX PLANAR XOR EU3SAT *is NP/ASP/#P-complete.*

*Proof.* We will reduce from MAX PLANAR XOR 2SAT. First notice that XOR EU3SAT is the same as {\ODD\-in-EU}3SAT. We will convert each 2 clause to a series of 3 clauses, such that the only way to maximize the number of satisfied clauses is if the 2 clause summed to 1.

Given an instance of MAX XOR 2SAT with $m$ clauses, and its $k$, we replace each of its clauses $(a_i \oplus b_i)$, with 5 clauses : $(a_i \oplus b_i \oplus w_i)$, $(\neg w_i \oplus x_i \oplus y_i)$, $(\neg w_i \oplus y_i \oplus z_i)$, $(\neg w_i \oplus x_i \oplus z_i)$, $(\neg x_i \oplus y_i \oplus z_i)$ where $x_i.y_i, z_i, w_i$ are local to those clauses. The only way to satisfy the last clause, is if $(x_i + y_i, +z_i) = \textit{even}$, and, unless $x_i = y_i = z_i$, at least one of the other clauses will not be satisfied. This forces $x_i = y_i = z_i = w_i = 0$, and that $a_i \neq b_i$. This can be seen in Figure 5-4.

Similarly, we replace each clause $(a_i)$ with $(a_i \oplus x_i \oplus y_i)$, $(a_i \oplus x_i \oplus z_i)$, $(a_i \oplus y_i \oplus z_i)$, $(a_i \oplus y_i \oplus z_i)$ $(\neg x_i \oplus y_i \oplus z_i)$. The only way to satisfy the last clause, is if $(x_i + y_i, +z_i) = \textit{even}$. And, unless $x_i = y_i = z_i$, at least one of the first 3 clauses will not be satisfied. This forces $x_i = y_i = z_i = w_i = 0$, and that $a_i = 1$. We also add a clause that can always be trivially satisfied $(a_i \oplus w_i \oplus z_i)$ This can be seen in Figure 5-5.



$a_i$
$b_i$
$w_i \vee a_i \vee b_i$
$w_i$
$\neg w_i \vee x_i \vee y_i$
$\neg w_i \vee y_i \vee z_i$
$x_i$
$y_i$
$z_i$
$\neg x_i \vee y_i \vee z_i$
$\neg w_i \vee x_i \vee z_i$

Figure 5-4: Converting a 2 variable XOR clause to 3 variable clauses while preserving planarity.

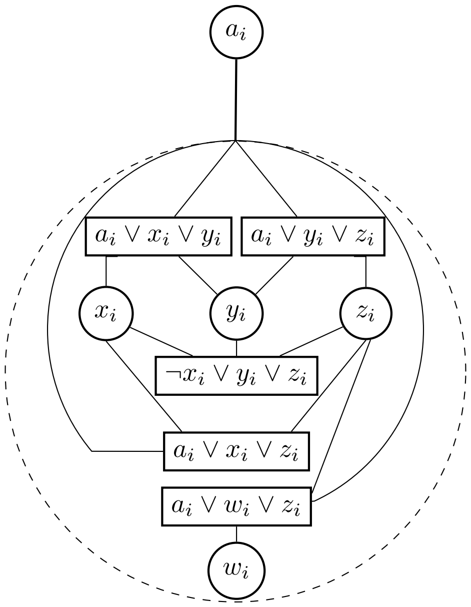


Figure 5-5: Converting a 1 variable XOR clause to 3 variable clauses while preserving planarity.

Therefore, we can satisfy $k$ clauses in the original instance iff we can satisfy $5k$ clauses in the MAX PLANAR XOR EU3SAT. If $k$ is the maximum number of clauses that can be satisfied, this reduction will preserve the number of solutions. Therefore, similar to the MAX PLANAR XOR 2SAT proof (Theorem 5.4.1), this shows that this problem is #P and ASP-hard. $\square$

**Corollary 5.5.2.** MAX PLANAR XNOR EU3SAT *is NP/ASP/#P-complete.*

*Proof.* In each of the above clauses in the above reduction for MAX PLANAR XOR EU3SAT, we choose out of the positive $x_i$, $y_i$ or $z_i$ literals an negate it. The rest of the reduction follows. $\square$

## 5.6 Max Planar Horn 2SAT

**Theorem 5.6.1.** MAX PLANAR HORN 2SAT *is NP/ASP/#P-complete.*

*Proof.* We reduce from MAX PLANAR XOR 2SAT.

Given an instance of MAX PLANAR XOR 2SAT, $\Phi(\boldsymbol{X}), k$ we replace each $(a_i \oplus \neg b_i)$ or $(\neg a_i \oplus b_i)$ clause with the 3 clauses $(a_i \vee \neg b_i)$, $(a_i, \neg b_i)$, $(\neg a_i \vee b_i)$. Therefore, iff $a \vee b$, the original clause will be satisfied and 3 of the new clauses will be satisfied. Otherwise, the original clause will not be satisfied, and only 1 or 2 of the new clauses will be satisfied. This reduction preserves planarity, as seen in Figure 5-6.

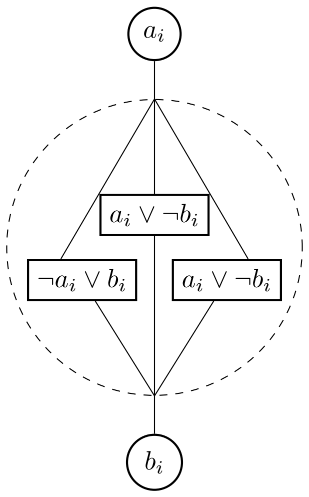


Figure 5-6: Conversion of $(a_i \oplus \neg b_i)$ clause from MAX PLANAR XOR 2SAT to MAX PLANAR HORN 2SAT.

We replace each $(a_i \oplus b_i)$ or $(\neg a_i \oplus \neg b_i)$ clause with the 4 clauses $(\neg a_i \vee \neg b_i)$, $(\neg a_i \vee \neg b_i)$, $(a_i)$, $(b_i)$. Therefore, iff $a \neq b$, the original clause will be satisfied and 3 of the new clauses will be satisfied. Otherwise, the original clause will not be satisfied and only 2 of the new clauses will be satisfied. This reduction preserves planarity, as seen in Figure 5-7.

At last, we triplicate each clause that contains only one variable. We obtain a $\Phi(\boldsymbol{X'}), 3k$ instance of MAX PLANAR HORN 2SAT.

$3k$ clauses are simultaneously satisfiable in the new instance iff at least $k$ of the original instance clauses are simultaneously satisfiable. If $k$ is the maximum number of clauses that can be satisfied in the original problem, this reduction will preserve the number of solutions. Therefore, similar to the MAX XOR 2SAT proof (Theorem 5.4.1), this shows that this problem is #P and ASP-hard. $\square$

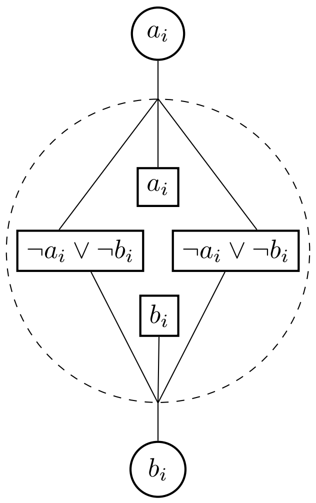


Figure 5-7: Conversion of $(a_i \oplus b_i)$ clause from Max Planar Xor 2SAT to Max Planar Horn 2SAT.

**Corollary 5.6.2.** Max Planar Dual Horn 2SAT *is NP/ASP/#P-complete.*

*Proof.* In all the new clauses from the Max Planar Horn 2 SAT above, we negate all the literals (negative literals become positive, and vice versa). The same results follow. □

# Chapter 6

# Other SAT variants

In this chapter we present results on SAT variants or problems related to SAT that did not fit the other chapters. We show results NP-complete results for bounded 3SAT variants; extend the NP-completeness result of Linked Planar 3SAT; to show that Quantified Linked Planar 3SAT is PSPACE-complete, analyze a problem about deciding Schaefer's Dichotomy, and a variant of Circuit SAT; we do a partial characterization of XNF-SAT problems; we apply the ordered planar SAT dichotomy to show that 2 planar SAT problems results are in P; and at last show that the equivalence of Tri-legged Planar 3SAT and Var-Linked Planar 3SAT imply the equivalence of Sided Tri-legged Planar 3SAT and Sided Var-Linked Planar 3SAT.

## 6.1 Not 1-in-E3SAT-3

**Lemma 6.1.1.** Not-1-in-E3SAT-6 *is NP-complete*

*Proof.* We reduce from 3SAT-3.

For each clause, replace the clause,$(a \vee b \vee c)$, with 2 clauses $(a + b + x \neq 1)$ and $(\neg x + c + c \neq 1)$. The latter clause is equivalent to $(\neg x \Rightarrow c)$, which is always satisfied when $x$ is True. The only case when the former clause forces $x$ to be False is when both $a$ and $b$ are False. Thus $(\neg a \wedge \neg b) \Rightarrow \neg x \Rightarrow c$, which enforces 3SAT, and those are the only constraints on $a, b, c$. (This reduction is not parsimonious when $a, b, c$ are all True; then $x$ is free.).

We also replace clauses $(a \vee b)$ with $(a+b+x \neq 1)$ and $(\neg x+x+x \neq 1)$. This forces $x = 1$ and $a+b \neq 0$.

The number of instances where a variable appears will be, in the worst case, doubled. □

**Theorem 6.1.2.** Not-1-in-EU3SAT-3 *is NP-complete.*

*Proof.* We reduce from Not-1-in-E3SAT-6, from Lemma 6.1.1.

First, notice that the following 2 clauses $(a+b+c \neq 1)$, $(a+\neg b+c \neq 1)$ forces $a$ and $c$ to both be True, also notice that $(a + F + c \neq 1)$, where $F = 0$ is equivalent to $(a = c)$. Therefore we can create copies of variables.

For each variable $x_i$, we create 5 new $F$ variables by using the above construction and 5 new clauses: $(x_i + F + x'_{i1} \neq 1)$, $(x_{i1} + F + x'_{i2} \neq 1)$, $(x_{i2} + F + x'_{i3} \neq 1)$, $(x_{i3} + F + x'_{i4} \neq 1)$, $(x_{i4} + F + x'_{i5} \neq 1)$. This forces variable $x_i$ to be equal to variables $x'_{i1}$, $x'_{i2}$, $x'_{i3}$, $x'_{i4}$, $x'_{i5}$.

Then, we can replace $x$ in its original clauses by at most 1 instances of each of $x'_{i1}$, $x'_{i2}$, $x'_{i3}$, $x'_{i4}$, $x'_{i5}$. This way, $x_i$ will occur at most twice (once in an original clause, once in the new clauses), each $x'_i$, will occur at most three times (twice in the new clauses, once in one of the original clauses where $F$ appeared). Each $F$ variable appears exactly 3 times. None of the variables appears twice in the same clause.

Therefore NOT-1-IN-EU3SAT-3 is NP-complete. □

## 6.2 Monotone 3SAT-3

**Theorem 6.2.1.** MONOTONE 3SAT-3 *is NP-complete*

*Proof.* This is a consequence of PLANAR MONOTONE (EU2 OR EU3)SAT-3E [DDD16] being NP-complete, but we provide a simpler proof here.

We reduce from MONOTONE 3SAT. If a variable $x$ appears $k$ times, replace it with variables $x_1, x'_1, x_2, \ldots, x_k, x'_k$, and force all $x_i$s equal and opposite from all $x'_i$s by adding a cycle of implications that are MONOTONE 2SAT clauses:

$$\begin{aligned} &x_i \vee x'_i \quad (\text{i.e., } \neg x_i \Rightarrow x'_i) \quad \text{and} \\ &\neg x'_i \vee \neg x_{i+1} \quad (\text{i.e., } x'_i \Rightarrow \neg x_{i+1}) \quad \text{for } i \in \{1, 2, \ldots, k\} \end{aligned}$$

(indices modulo $k$).

Then replace each occurrence of $x$ by an occurrence of an $x_i$. Each $x_i$ will appear in 3 clauses and each $x'_i$ will appear in 2 clauses. □

## 6.3 Quantified Linked Planar 3SAT

**Definition. Quantified Linked planar 3SAT** is a fully quantified Linked Planar 3SAT instance, where the cycle that goes through the variables and clauses go through all the variables in the same order that they are quantified in the Quantified Boolean Formula. The motivation is that this can possibly help simulate games where where 2 players take turns controlling a shared agent.

**Theorem 6.3.1.** QUANTIFIED LINKED PLANAR 3SAT *is PSPACE-complete.*

*Proof.* QUANTIFIED LINKED PLANAR 3SAT is in PSPACE by trivial reduction to QUANTIFIED3SAT. Now, we will show that it is in PSPACE-complete by a reduction from QUANTIFIED 3SAT. We will combine the reduction from QUANTIFIED3SAT to QUANTIFIED PLANAR 3SAT from Lichtenstein [Lic82] with the reduction from PLANAR 3SAT to LINKED PLANAR 3SAT by Pilz [Pil17]. We will omit some details from the reductions, which can be found in the original papers, so consult them for more information.

Lichtenstein [Lic82] showed a reduction from QUANTIFIED 3SAT to PLANAR QUANTIFIED 3SAT that allows us to choose the location of each of the original clauses and variables of the QUANTIFIED 3SAT instance to coincide with the positive $x$ and positive $y$ axis (if variables are in the $y$ axis, then clauses are in the $x$ axis, and vice-versa), in any order, and keep the new variables introduced variables strictly in the positive quadrant. This can be seen in Figures 6-1 and 6-2, where we show the location of the original variable and clauses and the location of the added crossover gadgets, which adds new clauses and variables to the instance.

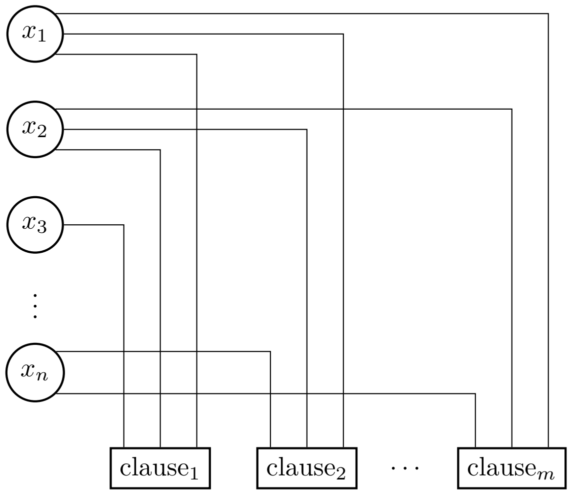


Figure 6-1: Start of Lichtenstein's reduction, by setting variables on the $y$ axis and clauses on the $x$ axis.

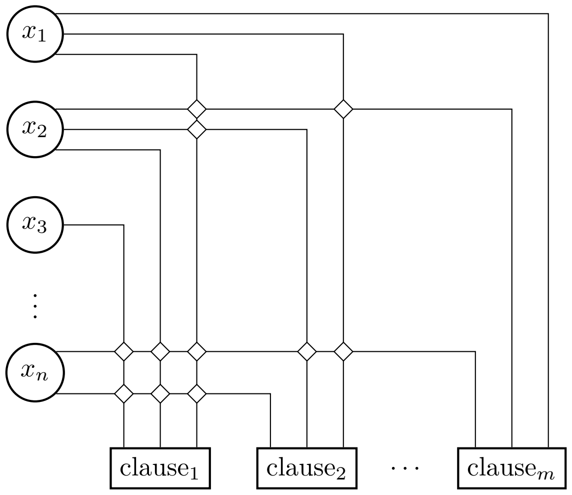


Figure 6-2: Putting crossover gadgets on the intersections, to make the graph planar.

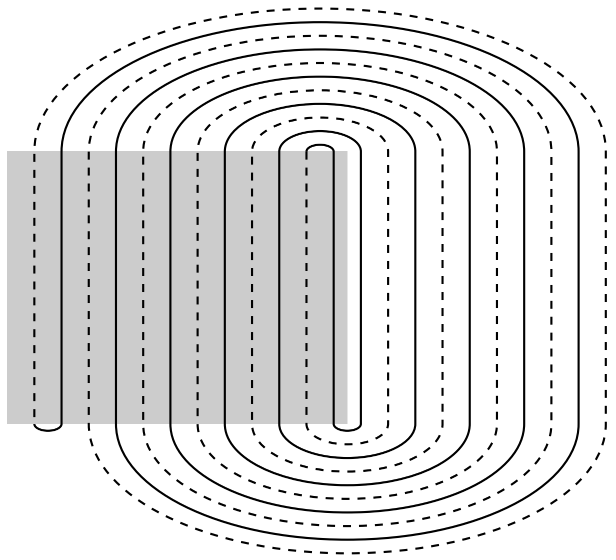


Figure 6-3: Pilz's variable and clause cycle. The shaded area is where the original clauses and variable nodes are located. The dotted line contains the clauses and the solid line contains the variables. Every time an edge crosses the cycle, a crossover gadgets is added, which are the only newly added variables and clauses. Figure from: [mar19], based on figure from [Pil17].

Pilz [Pil17] showed a reduction from Planar 3SAT to Linked Planar 3SAT, as long as no clause has the same $x$ coordinate as a variable in the original instance planar embedding. This reduction first visits all the variables, followed by all the clauses, first visiting the ones with smaller $x$ coordinates. In the case multiple variables have the same $x$ coordinate, we break the ties based on the $y$ coordinate, sometimes in ascending order, others in descending order, but always consistent in the same $x$ coordinate. Furthermore, all the new variables created by this reduction are necessarily to the right of the leftmost variable of clause of the original problem. The construction of the Variable-Clause cycle can be seen on Figure 6-3.

Therefore if we make all the newly created variables to use the Existential Quantifier, and to be the last ones in the quantifier list, we can use Lichtenstein [Lic82] reduction to convert Quantified 3SAT to a Quantified Planar 3SAT instance with all the original variables in the $y$ axis in the correct order. Then, we can move the clause nodes slightly to make sure that none of them have the same $x$ coordinate as a clause node. And, at last, we can use Pilz [Pil17] reduction to convert this to a Quantified Linked Planar 3SAT, which will first visit all the original variables in the correct order before visiting the new variables. At the end, we are not guaranteed to visit the newly created variables in the correct order; however, since the existential quantifiers commute with each other, we can just change the order of those variables obtaining an equivalent Quantified 3SAT instance. □

## 6.4 Not Implies Circuit SAT

Not Implies Circuit SAT is the problem of deciding whether the inputs of a particular circuit with Boolean inputs, where every gate is a "not implies" gate, can be set such that the output of the circuit is 1. A not implies gate is a gate that takes 2 inputs, $x$, $y$, and outputs $(x \wedge \neg y)$.

**Theorem 6.4.1.** Circuit Not Implies SAT *is NP-complete.*

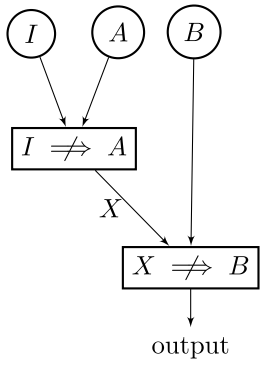


Figure 6-4: Circuit whose output is 0 if I is 0, or NOR(A,B) if I is 1.

*Proof.* Using not implies gates we can construct a 3 input gate whose output is 0 if the first input is 0, or Nor$(A, B)$ if it is 1, where A and B are respectively the second and third inputs, as seen in Figure 6-4. We will call this circuit a Conditional Nor.

If we construct a circuit using only copies of the Conditional Nor and we make the first input of all the copies be shared, the only way for the final output to be different from 0, is if this first input is 1. Therefore the only way for the output to not be zero is if $I \neq 0$, and so all the Conditional Nors behave like standard Nor gates.

Since Nor can simulate any circuit, the above construction can simulate any circuit (as long $I \neq 0$, which will happen if there exists an input assignment which makes the output be 1), and therefore the problem is NP-complete. $\square$

## 6.5 Deciding CNF Schaefer's Dichotomy

**Definition. Deciding CNF Schaefer's Dichotomy** is the problem of, given $r$ CNF Boolean formulas $\phi_1(x_1, \ldots, x_{k_1}), \phi_2(x_1, \ldots, x_{k_2}), \ldots, \phi_r(x_1, \ldots, x_{k_r})$ each defining a relation on an arbitrary subset of variables, decide whether satisfiability of $m$ clauses defined by these formulas (each applied to an arbitrary set of variables, and then all clauses get andded together) is **polynomial** or **NP-complete**. This differs from how the formulas are given in the original paper by Schaefer [Sch78]. In the original paper the Boolean formulas were given in DNF.

**Theorem 6.5.1.** Deciding CNF Schaefer's Dichotomy *is NP-complete.*

*Proof.* We reduce from CNF SAT. Given a CNF formula $\varphi(x_1, \ldots, x_n)$, construct the Boolean formula:

$$\varphi' = \textsc{Exactly-1-in-}(x_{n+1}, x_{n+2}, x_{n+3}) \land \varphi(x_1, \ldots, x_n),$$

where Exactly-1-in- is a 1-in-3 clause:

$$\textsc{Exactly-1-in-}(a, b, c) = (a \lor b \lor c) \land (\neg a \lor \neg b) \land (\neg b \lor \neg c) \land (\neg c \lor \neg a).$$

If $\varphi$ is unsatisfiable, then so is $\varphi'$. Therefore deciding satisfiability of $\varphi'$ clauses is polynomial, since the answer is always no.

If $\varphi$ is satisfiable, then deciding satisfiability of $\varphi'$ clauses is NP-complete, because we can represent an instance of POSITIVE 1-IN-3SAT as follows: we create $n$ new dummy variables, $x_1, x_2, \dots$ and we replace each clause $(a_i + b_i + c_i) = 1$ with EXACTLY-1-IN$(a_i, b_i, c_i) \wedge \varphi(x_1, \dots, x_n)$. Because $\varphi$ has a satisfying assignment, we can set $x_1, \dots, x_n$ accordingly. The remaining variables will have a satisfying assignment if and only if the 1-IN-3SAT instance did. Therefore the problem is NP-hard. □

## 6.6 XNF-SAT

We will prove that multiple XNF-SAT problems are in RP (defined below) by reducing it to the complement of a restricted version of the Evaluates Zero Everywhere problem which we will show to be equivalent to the Polynomial Identity Testing problem.

First we define some terms:

- $\boldsymbol{F_2}$: The finite field of order 2. $F_2 = \mathbb{Z}/2\mathbb{Z} = \mathbb{Z}_2$. $1 + 1 = 0$ in $F_2$.
- **RP**: A problem is in RP if a probabilistic algorithm that runs in polynomial time
  - always returns FALSE if the correct answer is FALSE
  - returns TRUE with probability $> \frac{1}{2}$ if the correct answer is TRUE (otherwise returns FALSE).
- **coRP**: Similar to RP, but with TRUE replace with FALSE and FALSE replaced with TRUE. A problem is in RP if a probabilistic algorithm that runs in polynomial time
  - always returns TRUE if the correct answer is TRUE
  - returns FALSE with probability $> \frac{1}{2}$ if the correct answer is FALSE (otherwise returns TRUE).
- **Evaluates Zero Everywhere (EZE)**: The problem of deciding whether a polynomial evaluates to 0 everywhere. The polynomial $x^2+x$ in $F_2$ evaluates to 0 for every possible value of $x$ (1 or 0). Therefore for the input $x^2 + x$ in $F_2$, EZE returns TRUE. EZE in $F_2$ is coNP-hard [Har11].
- **co-EZE**: The complement of EZE. The problem of deciding whether a polynomial does not evaluate to 0 everywhere. This problem is NP-hard, since EZE is coNP-hard.
- **Polynomial Identity Testing (PIT)**: The problem of deciding whether a polynomial is identical to 0. The polynomial $x^2 + x$ in $F_2$ is not identical to 0, so PIT would return FALSE for this input. The polynomial $(x + 1)^2 + x^2 + 1$ in $F_2$ is identical to 0, since after expanding and combining the terms, all of them cancel one another, so PIT would return TRUE for this input. PIT is in coRP [KI04] if the polynomial can be evaluated in polynomial time.
- **co-PIT**: The complement of PIT. The problem of deciding whether a polynomial is not identical to 0. It is in RP since PIT is in coRP.

Now we are ready to prove our Lemma showing the equivalence of a restricted EZE problem to the PIT problem.

**Lemma 6.6.1.** *If every variable has degree at most 1, the EZE problem is equivalent to the PIT problem. And therefore it is in coRP.*

*Proof.* Let $Q$ be a polynomial where every variable has degree at most 1. Fully expand the polynomial. Suppose the polynomial is not identical to zero. Consider a term $T$ with lowest degree, that is, a term with the smallest number of variables. Any other term has at least one variable not in this term. If we set the variables in $T$ to 1, and all the other variables to 0, then the polynomial will evaluate to 1.

Therefore the polynomial will only evaluate to 0 everywhere if it is identically to zero. Therefore, for polynomials with every variable with degree at most 1, EZE is equivalent to PIT. And PIT has been proved to be in coRP [KI04].

Now we can prove our main theorem.

**Theorem 6.6.2.** AND XNF $\text{SAT}_{\text{C}}$ *is in RP.*

The main idea of the proof of Theorem 6.6.2 is to convert the problem to a co-EZE problem with a polynomial in $F_2$ as its input. In this polynomial, if it was expanded, every variable would have degree at most 1. Therefore we don't need to expand in order to solve the co-EZE problem, since EZE is equivalent to PIT when every variable has degree at most 1.

*Proof.* We reduce the problem to the complement of Polynomial Identity Testing in $F_2$, which is in RP.

For each clause, if a literal appears more than once, we can delete all but one occurrences, for example $(x \wedge x \wedge x \wedge y) = (x \wedge y)$. We can also remove every clause that has a variable that appears both positively and negatively, because it always evaluates to zero.

Then we can rewrite the problem as a polynomial in $F2$. We replace the XORs by $\oplus$, ANDs by products, and negative literals by "1$\oplus$literal". For example $(\neg x \wedge y) \oplus (z \wedge y) \rightarrow ((1+x)y) \oplus (zy)$. Our problem then becomes equivalent to co-EZE.

If we were to expand the polynomial, every single variable would appear with degree at most 1 in each of the terms. By Lemma 6.6.1, this means that our problem is equivalent to co-PIT, which is in RP. Note that at no moment we expand the polynomial, since doing so could take an exponential amount of time; the only condition for co-PIT to be in RP is that the polynomial can be evaluated in polynomial time, which we can do without expanding it. ☐

**Corollary 6.6.3.** AND XnorNF $\text{SAT}_{\text{C}}$ *is in RP.*

*Proof.* Replace each XOR operation with an XNOR operation, and adjust the equations accordingly, on the proof of Theorem 6.6.2, and the proof is still valid. ☐

**Theorem 6.6.4.** *$S$* XNF $\text{SAT}_{\text{C}}$ *is in RP if each relation in $S$ can be written as "the (possibly complemented) disjunction of mutually exclusive AND clauses. A set of clauses is* ***mutually exclusive*** *if no 2 clauses can evaluate to* True *simultaneously.*

*Proof.* If a relation can be written as a disjunction of a polynomial number of mutually exclusive AND clauses, we can replace each instance of the relation by the XOR of those AND clauses. Since the clauses are mutually exclusive, at most one will evaluate to True. Therefore the XOR of these clauses evaluates to true if and only if the original relation evaluated to True.

If a relation can be written as the disjunction of the complement of a polynomial number of mutually exclusive AND clauses, we can replace each instance of the relation by the XOR of those AND clauses, and XOR it with (1). Since the clauses are mutually exclusive, at most one will evaluate to True at a time. Therefore, iff the original relation evaluated to True, all these AND clauses will evaluate to False, and therefore the XOR of these clauses XORed with (1) will evaluate to True.

Therefore, we can reduce our (S) XNF SAT instance into an instance of XNF SAT, with a polynomial increase in the number of clauses, which concludes our proof. ☐

**Corollary 6.6.5.** *$S$ XNORNF SAT$_C$ is in RP if each relation in $S$ can be written as the disjunction of a polynomial number of mutually exclusive AND clauses, or as the complement of the disjunction of a polynomial number of mutually exclusive AND clauses.*

*Proof.* We follow the steps of the reduction of Theorem 6.6.4; however we substitute each XOR with XNOR (1) XNOR. The result follows. ☐

**Corollary 6.6.6.** NAE XNF SAT$_c$*,* OR XNF SAT$_c$*,* NOR XNF SAT$_c$*,* $c$-IN-$k$ XNF SAT$_c$ *are in RP.*

*Proof.* These all follow from Theorem 6.6.4 by analyzing or rewriting the formula.

- A NAE relation is the complement of the disjunction of the conjunction of the literals in the clause and the conjunction of the negated literals in the clause (the complement of the OR of [the ANDs of the literals and the AND of the negated literals]).
- A OR relation is the complement of the conjunction of the negated literals in the clause (the complement of the AND of the negated literals).
- A NOR relation is the conjunction of the negated literals in the clause (the AND of the negated literals).
- A $c$-IN-$k$ XNF SAT relation can be written as the disjunction of at most $\binom{c}{k}$ mutually exclusive AND clauses, one for each combination that evaluates to TRUE. If any variable repeats more than once, we remove the repetitions of the equivalent AND clauses, so we are left with the exclusive ones.

☐

**Theorem 6.6.7.** *$S$ XNF U SAT$_C$ is in RP if each relation in $S$ is multilinear. A relation is* ***multilinear*** *if it can be rewritten as a multilinear polynomial in $F_2$.*

*Proof.* If each relation in $S$ is multilinear, and every variable appears at most once in each clause, then the instance can be converted into a polynomial in $F_2$ in which the maximum degree of each variable is 1. By Theorem 6.6.1, co-EZE for this problem is in RP. Therefore the problem is in RP. ☐

## 6.7 Planar Dichotomy

We will be showing that PLANAR (NAE OR ALL EQUAL) SAT, i.e, a Planar SAT problem whose formula contains NAE clauses well as All Equal clauses, is in P. We will also be showing that PLANAR $\{0, 1, k-1, k\}$-IN-EU$k$SAT is in P.

In order to show this, first we will prove 2 interesting Lemmas related to the Ordered Planar Dichotomy. Lemma 6.7.4 describes some situations where we can compose clause types of PLANAR SAT problems in P, resulting in a new problem in P. Lemma 6.7.6 shows a way to generalize PLANAR POSITIVE SAT hardness results to PLANAR SAT hardness results.

Here is the Ordered Planar Dichotomy [KKR16] we will be referring to during the rest of this section, as defined on Table 2.1.

Ordered Planar ($S$) SAT is in P if $S$ SAT is in P, or if $S$ only contains self-complementary relations such that $dR$ is an even $\Delta$-matroid. Otherwise it is NP-complete

**Lemma 6.7.1.** *Let $R$ be a relation. Then all tuples in $dR$ have even parity.*

*Proof.* Let $T$ be a tuple in $R$. Then $dT = \{t_i + t_{i+1} \mod 2 : \; i = 1, 2, \dots, n\}$. The parity of $dT$ is

$$\begin{aligned}\left(\sum_{i=1}^{n} dt_i\right) \mod 2 &= \left(\sum_{i=1}^{n}(t_i + t_{i+1})\right) \mod 2\\ &= \left(\sum_{i=1}^{n} 2t_i\right) \mod 2\\ &= 0\end{aligned}$$

Therefore all the tuples in $dR$ have even parity. □

Now we define the **entry-wise XOR** of two tuples. Let $T_1 = (t_{1_1}, t_{1_2}, \dots, t_{1_i})$, and let $T_2 = (t_{2_1}, t_{2_2}, \dots, t_{2_i})$. Then

$$T_1 \oplus T_2 = (t_{1_1} \oplus t_{2_1}, t_{1_2} \oplus t_{2_2}, \dots t_{1_i} \oplus t_{2_i})$$

And now we define the **pairwise entry-wise XOR** of a relation. Let $R$ be a relation, $R = \{T_0, T_1, ..., T_i\}$, where each $T_i$ is a tuple. The pairwise entry-wise XOR, is the set of sets of the the entry-wise XORs of each tuple with all other tuples,

$$\text{pairwise entry-wise XOR of } R = \{\{T_i \oplus T_j \mid \forall T_j \neq T_i \in R\}, \forall T_i \in R\}.$$

And we define the $\boldsymbol{\Delta}$ operator on 2 tuples:

$$T_1 \Delta T_2 = \text{the set of variables that differ between tuples } T_1 \text{ and } T_2$$

**Lemma 6.7.2.** *If $R$ and $S$ are relations that have the same pairwise entry-wise XOR and $R$ is a $\Delta$-matroid then $S$ is a $\Delta$-matroid.*

*Proof.* First, notice that if 2 pairs of tuples $T_1,\ T_2$ and $T_1',\ T_2'$ have the same entry-wise XOR, then $T_1 \Delta T_2 = T_1' \Delta T_2'$, i.e. the set of variables that differ between tuples $T_1,\ T_2$ is the same as the set of variables that differ between $T_1',\ T_2'$.

Now suppose $R$ is a $\Delta$-matroid, but $S$ is not. Then there exists tuples $S_i, S_j$ such that for one of the variables that differs between them, negating that variable and any other variable that differ among them from $S_i$ will not result in a new tuple in $S$. Formally:

$$\exists x \in S_i \Delta S_j \; \forall y \in S_i \Delta S_j, x \neq y \; ((S_i \oplus x) \notin S) \wedge ((S_i \oplus x \oplus y) \notin S). \tag{6.1}$$

However, since $R$ and $S$ have the same pairwise entry-wise XOR, then there exists a tuple $R_i \in R$ that has the same entry-wise XOR with all the other tuples in $R$, as $S_i$ has with the tuples in $S$.

$$\{R_i \oplus R_k \mid \forall R_k \neq R_i \in R\} = \{S_i \oplus S_k \mid \forall S_k \neq S_i \in S\}.$$

In particular, there also exists a $R_j \in R$ such that

$$R_i \oplus R_j = S_i \oplus S_j.$$

Which means that $x \in R_i \Delta R_j$.

Since $R$ is a $\Delta$-matroid, either negating $x$ in $R_i$ will result in another tuple in $R$, or there is a $y$ that differs between $R_i$ and $R_j$ such that negating both $x$ and $y$ in $R_i$ will result in another tuple in $R$. Formally:

$$((R_i \oplus x) \in R)\text{OR } (\exists y \in R_i \Delta R_j, x \neq y \text{ s.t. } ((R_i \oplus x \oplus y) \in R)),$$

where $x$ is the same $x$ from Equation 6.1.

Now we have 2 cases:

*Case* 1 $((R_i \oplus x) \in R)$. Negating $x$ in $R_i$ results in another tuple in $R$.

We show that negating $x$ in $S_i$ results in another tuple in $S$.

Let $R_l = (R_i \oplus x)$.

Since

$$\{R_i \oplus R_k \mid \forall R_k \neq R_i \in R\} = \{S_i \oplus S_k \mid \forall S_k \neq S_i \in S\},$$

then

$$\begin{aligned}
&(R_i \oplus R_l) \in \{S_i \oplus S_k \mid \forall S_k \neq S_i \in S\} \\
&\implies x \in \{S_i \oplus S_k \mid \forall S_k \neq S_i \in S\} \\
&\implies \exists S_l . S_i \oplus S_l \ = x \\
&\implies (S_i \oplus x) \in S
\end{aligned}$$

Which contradicts Equation 6.1.

*Case* 2 ( $\exists y \in R_i \Delta R_j, x \neq y$ s.t. $((R_i \oplus x \oplus y) \in R)$. Negating $x$ and $y$ in $R_i$ results in another tuple in $R$. We show that negating $x$ and $y$ in $S_i$ results in another tuple in $S$.

Let $R_l = (R_i \oplus x \oplus y)$. The, by repeating the proof from the previous case:

Since

$$\{R_i \oplus R_k \mid \forall R_k \neq R_i \in R\} = \{S_i \oplus S_k \mid \forall S_k \neq S_i \in S\},$$

then

$$\begin{aligned}
&(R_i \oplus R_l) \in \{S_i \oplus S_k \mid \forall T_k \neq T_i \in R\} \\
&\implies \exists S_l . S_i \Delta S_l \ = \{x, y\} \\
&\implies (S_i \oplus x \oplus y) \in S
\end{aligned}$$

Which contradicts Equation 6.1.

Therefore, by contradiction, $S$ is a $\Delta$-matroid. □

**Corollary 6.7.3.** *If $R$ and $S$ have the same pairwise entry-wise XOR and $dR$ is a $\Delta$-matroid, then $dS$ is a $\Delta$-matroid.*

*Proof.* Since, $R_i \oplus R_j = S_i \oplus S_j$ implies $dR_i \oplus dR_j = dS_i \oplus dS_j$, the proof is almost identical to the proof of Lemma 6.7.2. □

**Lemma 6.7.4** (Planar P Composition)**.** *If $S$ and $Q$ only contain self-complementary relations, $S_i$ and $Q_i$, such that $dS_i$ and $dQ_i$ are even $\Delta$-matroid, then* Planar *(S OR Q) SAT is in P.*

*Proof.* We have defined ($S$ OR $Q$) SAT to mean a SAT problem that contains both $S$ clauses or $Q$ clauses. Therefore all the relations in ($S$ OR $Q$) SAT are even $\Delta$-matroid, and by the Ordered Planar Dichotomy, Planar ($S$ OR $Q$) SAT is in P. □

**Corollary 6.7.5.** *If $S$ SAT and $Q$ SAT are NP-complete, where $S$ and $Q$ are a set of relations or of allowed clause types, and* Planar *$S$ SAT and* Planar *$Q$ SAT are in P, then* Planar *(S OR Q) SAT is in P.*

*Proof.* By the Ordered Planar Dichotomy, this implies that $S$ and $Q$ only contain self-complementary relations, $S_i$ and $Q_i$, such that $dS_i$ and $dQ_i$ are even $\Delta$-matroid. Therefore, by Lemma 6.7.4, Planar ($S$ OR $Q$) SAT is in P. □

**Lemma 6.7.6** (Planar Positive P Generalization)**.** *Let $S$ be a type of clause and if for every relation $R'$ from* Positive *$S$, $dR'$ is an even $\Delta$-matroid, then for every relation $R$ from $S$, $dR$ is an even $\Delta$-matroid.*

First we give some example from what the above means. Let $S$ be NAE EU3SAT. The valid relations are NAE$(x, y, z)$, NAE$(\neg x, y, z)$, NAE$(x, \neg y, z)$, NAE$(x, y, \neg z)$. Other relations are equivalent to those. Let $R =$

$$\text{NAE}(x, y, z) = \{(1,0,0), (0,1,0), (0,0,1), (1,1,0), (1,0,1), (0,1,1)\}.$$

If we show that $dR$ is an even $\Delta$-matroid, then $dQ$ for $Q$ in

$$\begin{aligned}\{&\text{NAE}(\neg x, y, z) = \{(0,0,0), (1,1,0), (1,0,1), (0,1,0), (0,0,1), (1,1,1)\},\\ &NAE(x, \neg y, z) = \{(1,1,0), (0,0,0), (0,1,1), (1,0,0), (1,1,1), (0,0,1)\},\\ &NAE(x, y, \neg z) = \{(1,0,1), (0,1,1), (0,0,0), (1,1,1), (1,0,0), (0,1,0)\}\}\end{aligned}$$

are all even $\Delta$-matroids.

*Proof.* Let $R$ be a relation from Positive $S$, and $R'$ be an equivalent relation from $S$ with some of the variables negated, say the set of variables $X$. For example, $R$ could be NAE$(x, y, z)$ and $R'$ be, NAE$(x, \neg y, z)$. Let $T_i$ be a tuple from $R$, and $T'_i$ be the equivalent tuple from $R'$. The only difference between between $T_i$ and $T'_i$ are the variables in $X$. Therefore, $T_i \oplus T_j = T'_i \oplus T'_j$. Therefore, those 2 relations have the same pairwise entry-wise XOR.

By Corollary 6.7.3, for every relation $R$ from $S$, $dR$ is a $\Delta$-matroid, since for every relation $R'$ from Positive $S$, $dR'$ is a $\Delta$-matroid.

By Lemma 6.7.1, if $dR$ or $dR'$ are $\Delta$-matroids, they are even $\Delta$-matroids.

Therefore, for every relation $R$ from $S$, $dR$ is an even $\Delta$-matroid □

**Corollary 6.7.7.** *If* Positive *$S$ SAT is NP-complete, where $S$ is a set of relations or of allowed clause types, and* Planar Positive *$S$ SAT is in P, then* Planar *$S$ SAT is in P.*

*Proof.* By the Ordered Planar Dichotomy, this implies that Positive $S$ only contain self-complementary relations, $S_i$, such that $dS_i$ is an even $\Delta$-matroid. By Lemma 6.7.6, we can drop the positive restriction. $S$ only contain self-complementary relations, $S_i$, such that $dS_i$ is an even $\Delta$-matroid. By the Ordered Planar Dichotomy, Planar $S$ SAT is in P. □

**Theorem 6.7.8.** Planar (NAE or All Equal) SAT *is in P*

*Proof.* First notice that Planar (NAE or All Equal) SAT and Planar (NAE or All Equal) U SAT are equivalent problems, since repeating a variable in a clause is equivalent to decreasing the size of the clause. Therefore we will be considering the relations in Planar (NAE or All Equal) U SAT.

Let $R_i$ be the Positive NAE relation for $i$ variables, and let $Q_i$ be the Positive All Equal relation for $i$ variables. By the Ordered Planar Dichotomy since every $R_i$ and $Q_i$ are self complementary, it is sufficient to show that all $dR_i$ and $dQ_i$ are even $\Delta$-matroids to show that Planar Positive (NAE or All Equal) SAT is in P.

Since Planar NAE SAT is in P [Alo+09], we know that all $dR_i$ are even $\Delta$-matroid. All $Q_i$s contain exactly 2 tupples that are self dual, $Q_i = \{(1,1,\dots,1),(0,0,\dots,0)\}$. We know, therefore, that every $dQ_i$ contain exactly one tuple, $dQ_i = \{(0,0,\dots,0)\}$. Therefore, every $dQ_i$ is an even $\Delta$-matroids. By Lemma 6.7.4, this shows that Planar Positive (NAE or All Equal) SAT is in P.

By Lemma 6.7.6, Planar (NAE or All Equal) SAT is also in P. □

**Theorem 6.7.9.** Planar $\{0,1,k-1,k\}$-in-EU$k$SAT *is in P.*

Even though $\{0,1,k-1,k\}$-in-EU$k$SAT is NP-complete (Theorem 4.4.1), the planar version is not. We will use the ordered planar dichotomy to show this.

**Lemma 6.7.10.** *Let $R$ be the* Positive $\{0,1,k-1,k\}$-in-EU$k$ *relation. Then $dR$ is an even $\Delta$-matroid.*

*Proof.* $R$ is the set of $k$-tuples that contain exactly one 0, exactly one 1, no 0s, or no 1s. Therefore:

$$\begin{aligned} R =&\{(0,0,\dots,0),(1,0,0,\dots,0),(0,1,0,0,\dots,0),\dots,(0,0,\dots,0,1),\\ &(1,1,1,\dots,1),(0,1,1,\dots,1),(1,0,1,\dots,1),\dots,(1,1,\dots,1,0)\}\\ dR =&\{(0,0,\dots 0,0),(1,1,0,0,0\dots,0),(0,1,1,0,0,0,\dots,0),\dots,(0,0,\dots,0,1,1),(1,0,0,0,\dots,0,1)\} \end{aligned}$$

And we can see that $dR$ is a $\Delta$-matroid. If 2 tuples differ by exactly 2 variables, one can convert one into the other by negating those 2 variables. If they differ by 4 variables, we can convert one of them into the $\mathbf{0}$ tuple by negating 2 variables that differ.

By Lemma 6.7.1, we have shown that all the tuples in $dR$ have the same parity. Therefore, $dR$ is an even $\Delta$-matroid. □

Now we are ready to prove Theorem 6.7.9.

*Proof.* We know that all relations in Planar $\{0,1,k-1,k\}$-in-EU$k$SAT are self complementary. By Lemmas 6.7.10 and 6.7.6, we know that for every relation $R$ in Planar $\{0,1,k-1,k\}$-in-EU$k$SAT, $dR$ is an even $\Delta$-matroid. Therefore, by the Planar Dichotomy, it is in $P$. □

## 6.8 Sided Var-Linked

We note that a Theorem proved by Tippenhauer [Tip16] showing the equivalence of Var-Linked and Tri-Legged 3SAT instances also implies the equivalence of their respective Sided instances.

**Theorem 6.8.1.** *Let $\Phi$ be a 3SAT formula with* Var-Linked *planar embedding. Then $\Phi$ has a* Tri-Legged *embedding [Tip16]*

**Theorem 6.8.2.** *Let $\Phi$ be a 3SAT formula with* Sided Var-Linked *planar embedding. Then $\Phi$ has a* Sided Tri-Legged *embedding.*

*Proof.* In the proof of Theorem 6.8.1 by Tippenhauer, all the edges that were inside the variable loop stayed inside the loop, and all the edges that were outside the loop stayed outside the loop. Therefore the proof preserves the Sided property of the graph. $\square$

# Chapter 7

# Open Problems

There are still many open problems in Boolean Satisfiability. Here we report some problems that have been proposed, and propose some new ones.

- If $s \leq 2^{r-1} - 1$, is every instance of EU$r$SAT-$s$ always satisfiable [Tov84]?
- Is the complexity of Var-Linked Planar $S$ 3SAT always the same as the complexity of Planar $S$ SAT [Tip16] ?
- Are the complexities of Clause-Linked Planar $S$ 3SAT, Var-Linked Planar $S$ 3SAT, Linked Planar $S$ 3SAT and Planar $S$ 3SAT always the same?
- For what values of S is Positive $S$-in-EU$(2k+1)$-SAT-E2 always satisfiable? [Kol+19] . In particular, is Positive $\{1,4\}$-in-EU5$(2k+1)$-SAT-E2 always satisfiable?
- What is the complexity of Max $S$ SAT-E2 if every relation is a even $\Delta$-matroid [KKR16]?
- Ordered Planar $S$ U SAT being NP-complete implies that Planar $S$ u SAT is also NP-complete. Is the reverse true? It is probably not true, and maybe it can be shown with a non-symmetric relation.
- Is there a Quantified Planar Dichotomy? We have an Ordered Planar Dichotomy for SAT, but not for Quantified Sat.
- Is Quantified Clause-Linked Planar SAT PSPACE-complete? It should be since Quantified Var-Linked Planar SAT is PSPACE-complete, but the proof is missing.
- Is $k$ 1s Planar NAE SAT NP-complete? More generally, most $k$ 1s and $k$ 0s SAT instances have not been investigated. Maybe many of the polynomial problems might become NP-complete.
- Is there a characterization of Planar Reconfiguration SAT and Planar Connectivity SAT?
- Can we extend results of SAT with a bounded number of variables to $k$-Quantified SAT? This seems to be hard. For example $\forall\exists$ EU3SAT-3 is $\Pi_2^P$ hard, even though EU3SAT-3 is in P.

# Bibliography

[All+09] Eric Allender et al. “The Complexity of Satisfiability Problems: Refining Schaefer’s Theorem”. In: *Journal of Computer and System Sciences* 75.4 (2009), pp. 245–254. DOI: 10.1016/j.jcss.2008.11.001. URL: https://doi.org/10.1016/j.jcss.2008.11.001.

[Alo+09] Noga Alon et al. “Polychromatic Colorings of Plane Graphs”. In: *Discrete & Computational Geometry* 42.3 (June 2009), pp. 421–442. ISSN: 1432-0444. DOI: 10.1007/s00454-009-9171-5. URL: http://dx.doi.org/10.1007/s00454-009-9171-5.

[Alo+14] Greg Aloupis et al. “Classic Nintendo Games Are (Computationally) Hard”. In: *Fun with Algorithms* (2014), pp. 40–51. ISSN: 1611-3349. DOI: 10.1007/978-3-319-07890-8_4. URL: http://dx.doi.org/10.1007/978-3-319-07890-8_4.

[BG82] Andreas Blass and Yuri Gurevich. “On the unique satisfiability problem”. In: *Information and Control* 55.1-3 (Oct. 1982), pp. 80–88. ISSN: 0019-9958. DOI: 10.1016/s0019-9958(82)90439-9. URL: http://dx.doi.org/10.1016/S0019-9958(82)90439-9.

[BK10] Mark de Berg and Amirali Khosravi. “Optimal Binary Space Partitions in the Plane”. In: *Computing and Combinatorics* (2010), pp. 216–225. ISSN: 1611-3349. DOI: 10.1007/978-3-642-14031-0_25. URL: http://dx.doi.org/10.1007/978-3-642-14031-0_25.

[BLS12] M. L. Bonet, S. Linz, and K. St. John. “The Complexity of Finding Multiple Solutions to Betweenness and Quartet Compatibility”. In: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 9.1 (Jan. 2012), pp. 273–285. ISSN: 1545-5963. DOI: 10.1109/TCBB.2011.108.

[Cha+16] Steven Chaplick et al. “The Complexity of Drawing Graphs on Few Lines and Few Planes”. In: *CoRR* abs/1607.06444 (2016). arXiv: 1607.06444. URL: http://arxiv.org/abs/1607.06444.

[Cha+90] Vijaya Chandru et al. “On renamable Horn and generalized Horn functions”. In: *Annals of Mathematics and Artificial Intelligence* 1.1-4 (Sept. 1990), pp. 33–47. ISSN: 1573-7470. DOI: 10.1007/bf01531069. URL: http://dx.doi.org/10.1007/BF01531069.

[Che09] Hubie Chen. “A rendezvous of logic, complexity, and algebra”. In: *ACM Computing Surveys* 42.1 (Dec. 2009), pp. 1–32. ISSN: 0360-0300. DOI: 10.1145/1592451.1592453. URL: http://dx.doi.org/10.1145/1592451.1592453.

[CKS01] Nadia Creignou, Sanjeev Khanna, and Madhu Sudan. “Complexity Classifications of Boolean Constraint Satisfaction Problems”. In: (Jan. 2001). DOI: 10.1137/1.9780898718546. URL: http://dx.doi.org/10.1137/1.9780898718546.

[CLX09] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. “Holant problems and counting CSP”. In: *Proceedings of the 41st annual ACM symposium on Symposium on theory of computing - STOC ’09* (2009). DOI: 10.1145/1536414.1536511. URL: http://dx.doi.org/10.1145/1536414.1536511.

[CLX10] Jin-Yi Cai, Pinyan Lu, and Mingji Xia. “Holographic Algorithms with Matchgates Capture Precisely Tractable Planar #CSP”. In: *2010 IEEE 51st Annual Symposium on Foundations of Computer Science* (Oct. 2010). DOI: 10.1109/focs.2010.48. URL: http://dx.doi.org/10.1109/FOCS.2010.48.

[Con+97] Anne Condon et al. “Probabilistically Checkable Debate Systems and Nonapproximability of PSPACE-Hard Functions”. In: (Sept. 1997).

[Coo71] Stephen A. Cook. “The complexity of theorem-proving procedures”. In: *Proceedings of the third annual ACM symposium on Theory of computing - STOC ’71*. - 1971, nil. DOI: 10.1145/800157.805047. URL: https://doi.org/10.1145/800157.805047.

[Cre95] N. Creignou. “A Dichotomy Theorem for Maximum Generalized Satisfiability Problems”. In: *Journal of Computer and System Sciences* 51.3 (Dec. 1995), pp. 511–522. ISSN: 0022-0000. DOI: 10.1006/jcss.1995.1087. URL: http://dx.doi.org/10.1006/jcss.1995.1087.

[DD16] Andreas Darmann and Janosch Döcker. “Monotone 3-Sat-4 is NP-complete”. In: *CoRR* abs/1603.07881 (2016). arXiv: 1603.07881. URL: http://arxiv.org/abs/1603.07881.

[DDD16] Andreas Darmann, Janosch Döcker, and Britta Dorn. “On planar variants of the monotone satisfiability problem with bounded variable appearances”. In: *International Journal of Foundations of Computer Science* 29 (Apr. 2016). DOI: 10.1142/S0129054118500168.

[Deh15] Ali Dehghan. “On strongly planar not-all-equal 3SAT”. In: *Journal of Combinatorial Optimization* 32.3 (May 2015), pp. 721–724. ISSN: 1573-2886. DOI: 10.1007/s10878-015-9894-6. URL: http://dx.doi.org/10.1007/s10878-015-9894-6.

[Deh16] Ali Dehghan. “On strongly planar not-all-equal 3SAT”. In: *Journal of Combinatorial Optimization* 32.3 (Oct. 2016), pp. 721–724. ISSN: 1573-2886. DOI: 10.1007/s10878-015-9894-6. URL: https://doi.org/10.1007/s10878-015-9894-6.

[DF03] Victor Dalmau and Daniel K. Ford. “Generalized Satisfiability with Limited Occurrences per Variable: A Study through Delta-Matroid Parity”. In: *Lecture Notes in Computer Science* (2003), pp. 358–367. ISSN: 1611-3349. DOI: 10.1007/978-3-540-45138-9_30. URL: http://dx.doi.org/10.1007/978-3-540-45138-9_30.

[DF86] M.E Dyer and A.M Frieze. “Planar 3DM is NP-complete”. In: *Journal of Algorithms* 7.2 (June 1986), pp. 174–184. ISSN: 0196-6774. DOI: 10.1016/0196-6774(86)90002-7. URL: http://dx.doi.org/10.1016/0196-6774(86)90002-7.

[DFZ11] Liang Ding, Bin Fu, and Binhai Zhu. “Minimum Interval Cover and Its Application to Genome Sequencing”. In: *Combinatorial Optimization and Applications*. Ed. by Weifan Wang, Xuding Zhu, and Ding-Zhu Du. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 287–298. ISBN: 978-3-642-22616-8.

[Día+12] Josep Díaz et al. “On the Complexity of Metric Dimension”. In: *Lecture Notes in Computer Science* (2012), pp. 419–430. ISSN: 1611-3349. DOI: 10.1007/978-3-642-33090-2_37. URL: http://dx.doi.org/10.1007/978-3-642-33090-2_37.

[Edm65] Jack Edmonds. “Maximum matching and a polyhedron with 0, 1-vertices”. In: *Journal of research of the National Bureau of Standards B* 69.125-130 (1965), pp. 55–56.

[EG02] Thomas Eiter and Georg Gottlob. “Hypergraph Transversal Computation and Related Problems in Logic and AI”. In: *Lecture Notes in Computer Science* (2002), pp. 549–564. ISSN: 0302-9743. DOI: 10.1007/3-540-45757-7_53. URL: http://dx.doi.org/10.1007/3-540-45757-7_53.

[EG95] Thomas Eiter and Georg Gottlob. “Identifying the Minimal Transversals of a Hypergraph and Related Problems”. In: *SIAM Journal on Computing* 24.6 (Dec. 1995), pp. 1278–1304. ISSN: 1095-7111. DOI: 10.1137/s0097539793250299. URL: http://dx.doi.org/10.1137/S0097539793250299.

[EG96] Thomas Eiter and Georg Gottlob. “Note on the Complexity of Some Eigenvector Problems”. In: (Feb. 1996).

[EL73] Paul Erdős and László Lovász. “Problems and results on 3-chromatic hypergraphs and some related questions”. In: *COLLOQUIA MATHEMATICA SOCIETATIS JANOS BOLYAI 10. INFINITE AND FINITE SETS, KESZTHELY (HUNGARY)*. Citeseer. 1973.

[Fed01] Tomás Feder. “Fanout limitations on constraint systems”. In: *Theoretical Computer Science* 255.1-2 (Mar. 2001), pp. 281–293. ISSN: 0304-3975. DOI: 10.1016/s0304-3975(99)00288-1. URL: http://dx.doi.org/10.1016/S0304-3975(99)00288-1.

[Fel+95] M. R. Fellows et al. “The complexity of induced minors and related problems”. In: *Algorithmica* 13.3 (Mar. 1995), pp. 266–282. ISSN: 1432-0541. DOI: 10.1007/BF01190507. URL: https://doi.org/10.1007/BF01190507.

[FK96] Michael L. Fredman and Leonid Khachiyan. "On the Complexity of Dualization of Monotone Disjunctive Normal Forms". In: *Journal of Algorithms* 21.3 (Nov. 1996), pp. 618–628. ISSN: 0196-6774. DOI: 10.1006/jagm.1996.0062. URL: http://dx.doi.org/10.1006/jagm.1996.0062.

[GJS76] M.R. Garey, D.S. Johnson, and L. Stockmeyer. "Some Simplified Np-Complete Graph Problems". In: *Theoretical Computer Science* 1.3 (1976), pp. 237–267. DOI: 10.1016/0304-3975(76)90059-1. URL: https://doi.org/10.1016/0304-3975(76)90059-1.

[GLV01] Anna Galluccio, Martin Loebl, and Jan Vondrák. "Optimization via enumeration: a new algorithm for the max cut problem". In: *Mathematical Programming* 90.2 (2001), pp. 273–290.

[Gol77] Leslie M. Goldschlager. "The monotone and planar circuit value problems are log space complete for P". In: *ACM SIGACT News* 9.2 (July 1977), pp. 25–29. ISSN: 0163-5700. DOI: 10.1145/1008354.1008356. URL: http://dx.doi.org/10.1145/1008354.1008356.

[Gol78] E Mark Gold. "Complexity of automaton identification from given data". In: *Information and Control* 37.3 (June 1978), pp. 302–320. ISSN: 0019-9958. DOI: 10.1016/s0019-9958(78)90562-4. URL: http://dx.doi.org/10.1016/S0019-9958(78)90562-4.

[Gop+06] Parikshit Gopalan et al. "The Connectivity of Boolean Satisfiability: Computational and Structural Dichotomies". In: *Lecture Notes in Computer Science* (2006), pp. 346–357. ISSN: 1611-3349. DOI: 10.1007/11786986_31. URL: http://dx.doi.org/10.1007/11786986_31.

[Gut96] Shai Gutner. "The complexity of planar graph choosability". In: *Discrete Mathematics* 159.1-3 (Nov. 1996), pp. 119–130. ISSN: 0012-365X. DOI: 10.1016/0012-365x(95)00104-5. URL: http://dx.doi.org/10.1016/0012-365X(95)00104-5.

[Har11] Nick Harvey. *Randomized Algorithms Lecture 9*. 2011. URL: https://www.cs.ubc.ca/~nickhar/W12/Lecture9Notes.pdf.

[HM05] Hamed Hatami and Hossein Maserrat. "On the computational complexity of defining sets". In: *Discrete Applied Mathematics* 149.1-3 (Aug. 2005), pp. 101–110. ISSN: 0166-218X. DOI: 10.1016/j.dam.2005.03.004. URL: http://dx.doi.org/10.1016/j.dam.2005.03.004.

[Hor51] Alfred Horn. "On Sentences Which Are True of Direct Unions of Algebras". In: *The Journal of Symbolic Logic* 16.01 (1951), pp. 14–21. DOI: 10.2307/2268661. URL: https://doi.org/10.2307/2268661.

[HRT07] Ishay Haviv, Oded Regev, and Amnon Ta-Shma. In: *Theory of Computing* 3.1 (2007), pp. 45–60. ISSN: 1557-2862. DOI: 10.4086/toc.2007.v003a003. URL: http://dx.doi.org/10.4086/toc.2007.v003a003.

[HY13] Michael A. Henning and Anders Yeo. "2-colorings ink-regulark-uniform hypergraphs". In: *European Journal of Combinatorics* 34.7 (Oct. 2013), pp. 1192–1202. ISSN: 0195-6698. DOI: 10.1016/j.ejc.2013.04.005. URL: http://dx.doi.org/10.1016/j.ejc.2013.04.005.

[III+98] Harry B. Hunt III et al. "The Complexity of Planar Counting Problems". In: *CoRR* cs.CC/9809017 (1998). URL: http://arxiv.org/abs/cs.CC/9809017.

[JM95] Klaus Jansen and Haiko Müller. "The minimum broadcast time problem for several processor networks". In: *Theoretical Computer Science* 147.1-2 (Aug. 1995), pp. 69–85. ISSN: 0304-3975. DOI: 10.1016/0304-3975(94)00230-g. URL: http://dx.doi.org/10.1016/0304-3975(94)00230-G.

[Kar72] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations* (1972), pp. 85–103. DOI: 10.1007/978-1-4684-2001-2_9. URL: http://dx.doi.org/10.1007/978-1-4684-2001-2_9.

[KI04] Valentine Kabanets and Russell Impagliazzo. "Derandomizing Polynomial Identity Tests Means Proving Circuit Lower Bounds". In: *computational complexity* 13.1-2 (Dec. 2004), pp. 1–46. ISSN: 1420-8954. DOI: 10.1007/s00037-004-0182-6. URL: http://dx.doi.org/10.1007/s00037-004-0182-6.

[KKM94] Rajeev Kohli, Ramesh Krishnamurti, and Prakash Mirchandani. "The Minimum Satisfiability Problem". In: *SIAM J. Discret. Math.* 7.2 (May 1994), pp. 275–283. ISSN: 0895-4801. DOI: 10.1137/S0895480191220836. URL: http://dx.doi.org/10.1137/S0895480191220836.

[KKR16] Alexandr Kazda, Vladimir Kolmogorov, and Michal Rolínek. "Even Delta-Matroids and the Complexity of Planar Boolean CSPs". In: *CoRR* abs/1602.03124 (2016). arXiv: 1602.03124. URL: http://arxiv.org/abs/1602.03124.

[KLN91] Jan Kratochvíl, Anna Lubiw, and Jaroslav Nešetřil. "Noncrossing subgraphs in topological layouts". In: *SIAM Journal on Discrete Mathematics* 4.2 (1991), pp. 223–244.

[KMT10] Yasuaki Kobayashi, Yuichiro Miyamoto, and Hisao Tamaki. "k-cyclic Orientations of Graphs". In: *Lecture Notes in Computer Science* (2010), pp. 73–84. ISSN: 1611-3349. DOI: 10.1007/978-3-642-17514-5_7. URL: http://dx.doi.org/10.1007/978-3-642-17514-5_7.

[Kol+19] Sanjana Kolisetty et al. "The Complexity of Finding S-factors in Regular Graphs". In: *eccc* - (2019). arXiv: 1603.07881. URL: https://eccc.weizmann.ac.il/report/2019/040/.

[Kra03] Jan Kratochvíl. "Complexity of Hypergraph Coloring and Seidel's Switching". In: *Lecture Notes in Computer Science* (2003), pp. 297–308. ISSN: 1611-3349. DOI: 10.1007/978-3-540-39890-5_26. URL: http://dx.doi.org/10.1007/978-3-540-39890-5_26.

[Kra91] Jan Kratochvíl. "String graphs. II. recognizing string graphs is NP-hard". In: *Journal of Combinatorial Theory, Series B* 52.1 (May 1991), pp. 67–78. ISSN: 0095-8956. DOI: 10.1016/0095-8956(91)90091-w. URL: http://dx.doi.org/10.1016/0095-8956(91)90091-W.

[Kro67] M. R. Krom. "The Decision Problem for a Class of First-Order Formulas in Which all Disjunctions are Binary". In: *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik* 13.1-2 (1967), pp. 15–20. ISSN: 1521-3870. DOI: 10.1002/malq.19670130104. URL: http://dx.doi.org/10.1002/malq.19670130104.

[KS98] Dimitris Kavvadias and Martha Sideri. "The Inverse Satisfiability Problem". In: *SIAM Journal on Computing* 28.1 (Jan. 1998), pp. 152–163. ISSN: 1095-7111. DOI: 10.1137/s0097539795285114. URL: http://dx.doi.org/10.1137/S0097539795285114.

[KW13] Stefan Kratsch and Magnus Wahlström. "Two edge modification problems without polynomial kernels". In: *Discrete Optimization* 10.3 (Aug. 2013), pp. 193–199. ISSN: 1572-5286. DOI: 10.1016/j.disopt.2013.02.001. URL: http://dx.doi.org/10.1016/j.disopt.2013.02.001.

[Lad75] Richard E. Ladner. "The circuit value problem is log space complete for P". In: *ACM SIGACT News* 7.1 (Jan. 1975), pp. 18–20. ISSN: 0163-5700. DOI: 10.1145/990518.990519. URL: http://dx.doi.org/10.1145/990518.990519.

[Lar93] Philippe Laroche. "Planar 1-in-3 satisfiability is NP-complete". In: *COMPTES RENDUS DE L ACADEMIE DES SCIENCES SERIE I-MATHEMATIQUE* 316.4 (1993), pp. 389–392.

[Lev73] L. A. Levin. "Universal Sequential Search Problems". In: *Probl. Peredachi Inf.* 9.3 (1973), pp. 115–116.

[Lew79] Harry R Lewis. "Satisfiability problems for propositional calculi". In: *Mathematical Systems Theory* 13.1 (1979), pp. 45–53.

[Lic82] David Lichtenstein. "Planar Formulae and Their Uses". In: *SIAM Journal on Computing* 11.2 (1982), pp. 329–343. DOI: 10.1137/0211025. URL: https://doi.org/10.1137/0211025.

[LMM05] Van Bang Le, Raffaele Mosca, and Haiko Müller. "On Stable Cutsets in Claw-Free Graphs and Planar Graphs". In: *Lecture Notes in Computer Science* (2005), pp. 163–174. ISSN: 1611-3349. DOI: 10.1007/11604686_15. URL: http://dx.doi.org/10.1007/11604686_15.

[Man83] Anthony Mansfield. "Determining the thickness of graphs is NP-hard". In: *Mathematical Proceedings of the Cambridge Philosophical Society*. Vol. 93. 1. Cambridge University Press. 1983, pp. 9–23.

[Mar08] Dániel Marx. "Complexity of unique list colorability". In: *Theoretical Computer Science* 401.1-3 (July 2008), pp. 62–76. ISSN: 0304-3975. DOI: 10.1016/j.tcs.2008.03.018. URL: http://dx.doi.org/10.1016/j.tcs.2008.03.018.

[mar19] marmot. *How can I draw this spiral (with vertical sides) in tikz?* Tex Stack Exchange. 2019. URL: https://tex.stackexchange.com/a/490434/188729.

[MG08] JÁN MAŇUCH and DAYA RAM GAUR. "FITTING PROTEIN CHAINS TO CUBIC LATTICE IS NP-COMPLETE". In: *Journal of Bioinformatics and Computational Biology* 06.01 (Feb. 2008), pp. 93–106. ISSN: 1757-6334. DOI: 10.1142/s0219720008003308. URL: http://dx.doi.org/10.1142/S0219720008003308.

[Mor88] B. M. E. Moret. "Planar NAE3SAT is in P". In: *SIGACT News* 19.2 (June 1988), pp. 51–54. ISSN: 0163-5700. DOI: 10.1145/49097.49099. URL: http://doi.acm.org/10.1145/49097.49099.

[MR01] C. Moore and J.M. Robson. "Hard Tiling Problems with Simple Tiles". In: *Discrete & Computational Geometry* 26.4 (Jan. 2001), pp. 573–590. ISSN: 1432-0444. DOI: 10.1007/s00454-001-0047-6. URL: http://dx.doi.org/10.1007/s00454-001-0047-6.

[MR08] Wolfgang Mulzer and Günter Rote. "Minimum-Weight Triangulation Is Np-Hard". In: *Journal of the ACM* 55.2 (2008), pp. 1–29. DOI: 10.1145/1346330.1346336. URL: https://doi.org/10.1145/1346330.1346336.

[MTY07] Kazuhisa Makino, Suguru Tamaki, and Masaki Yamamoto. "A Dichotomy Theorem within Schaefer for the Boolean Connectivity Problem". In: *Electronic Colloquium on Computational Complexity (ECCC)* 14 (2007).

[Pap94] Christos H. Papadimitriou. *Computational complexity.* Addison-Wesley, 1994, pp. I–XV, 1–523. ISBN: 978-0-201-53082-7.

[Pil17] Alexander Pilz. "Planar 3-SAT with a Clause/Variable Cycle". In: *CoRR* abs/1710.07476 (2017). arXiv: 1710.07476. URL: http://arxiv.org/abs/1710.07476.

[RST99] Neil Robertson, P. D. Seymour, and Robin Thomas. "Permanents, Pfaffian orientations, and even directed circuits". In: *arXiv Mathematics e-prints*, math/9911268 (Oct. 1999), math/9911268. arXiv: math/9911268 [math.CO].

[Sch10] Tatjana Schmidt. "Computational Complexity of SAT, XSAT and NAE-SAT for linear and mixed Horn CNF formulas". PhD thesis. Universität zu Köln, 2010. URL: https://kups.ub.uni-koeln.de/3129/.

[Sch78] Thomas J. Schaefer. "The complexity of satisfiability problems". In: *Proceedings of the tenth annual ACM symposium on Theory of computing - STOC '78.* - 1978, nil. DOI: 10.1145/800133.804350. URL: https://doi.org/10.1145/800133.804350.

[Sey74] Paul D Seymour. "On the two-colouring of hypergraphs". In: *The Quarterly Journal of Mathematics* 25.1 (1974), pp. 303–311. URL: http://dx.doi.org/10.1016/S0019-9958(78)90562-4.

[SM73] L. J. Stockmeyer and A. R. Meyer. "Word problems requiring exponential time(Preliminary Report)". In: *Proceedings of the fifth annual ACM symposium on Theory of computing - STOC '73* (1973). DOI: 10.1145/800125.804029. URL: http://dx.doi.org/10.1145/800125.804029.

[Sto76] Larry J. Stockmeyer. "The polynomial-time hierarchy". In: *Theoretical Computer Science* 3.1 (Oct. 1976), pp. 1–22. ISSN: 0304-3975. DOI: 10.1016/0304-3975(76)90061-x. URL: http://dx.doi.org/10.1016/0304-3975(76)90061-X.

[SU02] Marcus Schaefer and Christopher Umans. "Completeness in the Polynomial-Time Hierarchy A Compendium". In: *Sigact News - SIGACT* 33 (Jan. 2002).

[SWK90] W.-K. Shih, S. Wu, and Y. S. Kuo. "Unifying Maximum Cut and Minimum Cut of a Planar Graph". In: *IEEE Trans. Comput.* 39.5 (May 1990), pp. 694–697. ISSN: 0018-9340. DOI: 10.1109/12.53581. URL: https://doi.org/10.1109/12.53581.

[Sze04] Stefan Szeider. "Generalizations of matched CNF formulas". In: *Annals of Mathematics and Artificial Intelligence* 43.1-4 (Dec. 2004), pp. 223–238. ISSN: 1573-7470. DOI: 10.1007/s10472-005-0432-6. URL: http://dx.doi.org/10.1007/s10472-005-0432-6.

[Tip16] Simon Tippenhauer. "On Planar 3-SAT and its Variants". MA thesis. Germany: Freie Universität Berlin, 2016. URL: https://www.mi.fu-berlin.de/inf/groups/ag-ti/theses/master_finished/tippenhauer_simon/index.html.

[Tov84] Craig A. Tovey. "A simplified NP-complete satisfiability problem". In: *Discrete Applied Mathematics* 8.1 (Apr. 1984), pp. 85–89. ISSN: 0166-218X. DOI: `10.1016/0166-218x(84)90081-7`. URL: `http://dx.doi.org/10.1016/0166-218X(84)90081-7`.

[Wra76] Celia Wrathall. "Complete sets and the polynomial-time hierarchy". In: *Theoretical Computer Science* 3.1 (Oct. 1976), pp. 23–33. ISSN: 0304-3975. DOI: `10.1016/0304-3975(76)90062-1`. URL: `http://dx.doi.org/10.1016/0304-3975(76)90062-1`.

[Wu15] Lidong Wu. "On strongly planar 3SAT". In: *Journal of Combinatorial Optimization* 32.1 (Apr. 2015), pp. 293–298. ISSN: 1573-2886. DOI: `10.1007/s10878-015-9878-6`. URL: `http://dx.doi.org/10.1007/s10878-015-9878-6`.

# Index