Deciding a Multinomial over $\mathbb{Z}/2\mathbb{Z}$ is not 0 is NP-Complete

Phillip Feldman

May 2024

1 Introduction

I will show the following decision problem is NP-Complete:
Let

$$f : (\mathbb{Z}/2\mathbb{Z})^n \to \mathbb{Z}/2\mathbb{Z}$$

Where $1 \leq i \leq n$ and the following symbols may appear in f:

$$(,), \times, +, 1, 0, x_i$$

and have their common meaning from arithmetic. Decide YES if $\exists \vec{x} \in (\mathbb{Z}/2\mathbb{Z})^n$ such that $f(\vec{x}) = 1$ and NO otherwise.

2 Proof

2.1 The problem is in NP

Given some $\vec{x}$, it can be checked that $f(\vec{x}) = 1$ by performing modular arithmetic. This can be done in polynomial time.

2.2 The problem is NP-Hard

I will show this by proving a many-to-one reduction from 3-SAT.

2.2.1 Lemma: Field Isomorphism

I will use exclusive-OR (XOR) from computer science for convenience. I will also shorten TRUE to T and FALSE to F. Inspect the following tables to verify the following field isomorphism:
((T, F), XOR, AND) $\to ((1, 0), +_{\mathbb{Z}_2}, \times_{\mathbb{Z}_2})$

2.2.2 XOR is associative: (A XOR B) XOR C = A XOR (B XOR C)

A	B	C	A XOR B	B XOR C	(A XOR B) XOR C	A XOR (B XOR C)
T	T	T	F	F	T	T
T	T	F	F	T	F	F
T	F	T	T	T	F	F
T	F	F	T	F	T	T
F	T	T	T	F	F	F
F	T	F	T	T	T	T
F	F	T	F	T	T	T
F	F	F	F	F	F	F

2.2.3 AND distributes over XOR: A AND (B XOR C) = (A AND B) XOR (A AND C)

A	B	C	A AND B	A AND C	B XOR C	A AND (B XOR C)	(A AND B) XOR (A AND C)
T	T	T	T	T	F	F	F
T	T	F	T	F	T	T	T
T	F	T	F	T	T	T	T
T	F	F	F	F	F	F	F
F	T	T	F	F	F	F	F
F	T	F	F	F	T	F	F
F	F	T	F	F	T	F	F
F	F	F	F	F	F	F	F

Also note that XOR is commutative, AND is associative, and AND is commutative. You may inspect table 2.2.3 to verify that XOR behaves like + and AND behaves like $\times$ in $\mathbb{F}_2$. This proves the field isomorphism.

2.2.4 Lemma

The following tables will be useful:

2.2.5 A OR B = A XOR B XOR (A AND B)

A	B	A OR B	A AND B	A XOR B	A XOR B XOR (A AND B)
T	T	T	T	F	T
T	F	T	F	T	T
F	T	T	F	T	T
F	F	F	F	F	F

2.2.6 NOT A = A XOR T

A	NOT A	A XOR 1
T	F	F
F	T	T

2.2.7 Lemma: Eliminating repeated terms in a 3-SAT instance

Given some instance of 3-SAT, it is convenient that each grouping of disjunctions only appears once. To eliminate repeated terms: sort each grouping's literals, then sort all the groupings by the literals in first, second, then third position. Finally, perform a linear scan on the instance and eliminate any repeated terms. Since A AND A = A, the instance remains unchanged. Since the above operations can be performed in polynomial time and are done only once, the complexity class of the instance remains unchanged. This has the added benefit of framing the complexity class of 3-SAT solely in terms of n, where n is the number of variables in the instance. This is because there is a maximal worst case amount of groupings that can appear in any instance, for any n.

2.2.8 Proof of 2.2

Given the above lemmas, we can transform any instance of 3-SAT into an instance of the decision problem by rewriting the problem in $\mathbb{F}_2$. There are finitely many forms a grouping can take, and each grouping will be rewritten as a grouping that requires at most 23 symbols. I will write some of them here. I will write $A \times B$ as AB for convenience.

$$\neg x_i \to (x_i + 1)$$

$$A \wedge B \to AB$$

$$A \vee B \to A + B + AB$$

$$A \vee B \vee C \to A + B + C + AB + AC + BC + ABC$$

The transformations can be done in polynomial time. performing modular arithmetic can be done by lookup table. For each transformed grouping, the maximum calls to the lookup table required is 12. So this entire transformation is done in Polynomial Time, and a YES instance of 3-SAT transforms to a YES instance of the decision problem.