Advanced Complexity Exam 2020

All written documents allowed. No Internet access, no cell phone.
The different sections are *not* independent.

1 CNF transforms

A propositional formula F is in *clausal form* if and only if it is a conjunction ($\wedge$) of clauses, where each clause is a disjunction ($\vee$) of literals, and literals are either propositional variables x or their negations $\neg x$.

SAT is the problem, given a formula in clausal form F, to decide whether F is satisfiable, and is a well-known **NP**-complete problem.

The usual translation from a formula F to a logically equivalent clausal form is exponential in time and space, in general. That translation is an algorithm which we call CNF: it takes a propositional formula F as input, pushes negations inwards, and distributes $\wedge$ over $\vee$ until a clausal form is obtained.

The purpose of this section is to explore a more clever translation, due to Tseitin (1957), and which preserves satisfiability, not logical equivalence.

Let F be a propositional formula, built from variables, negation $\neg$, truth $\top$, falsity $\bot$, binary conjunctions and disjunctions, and also binary exclusive or ($\oplus$) and $\Leftrightarrow$. Tseitin's algorithm works as follows. For each non-variable subformula G of F, we create a fresh variable y_G; for each variable x occurring in F, we consider that the notation y_x denotes x itself; and we create the following clauses:

- for each non-variable subformula G of F, say $G = G_1 \mathbin{op} G_2$ (where $op \in \{\wedge, \vee, \oplus, \Leftrightarrow\}$), we create CNF($y_G = y_{G_1} \mathbin{op} y_{G_2}$);
- we do the same for the unary operator $\neg$ (if $G = \neg G_1$, then we generate CNF($y_G = \neg y_{G_1}$)) and for the nullary operators (if $G = \top$, then we generate CNF($y_G = \top$), and similarly for $\bot$);
- finally, the unit clause y_F.

Let us call TSEITIN(F) the conjunction of all the clauses thus produced on the input formula F.

Let $x_1, \ldots, x_m$ be an enumeration of the variables that occur in F. If ρ is an assignment that satisfies F, then the assignment ρ' that extends ρ and maps each of the fresh variables y_G to the value of G under ρ satisfies TSEITIN(F). Conversely, if ρ' satisfies TSEITIN(F), then one can show by induction on the subformula G of F that the value of G under ρ' is equal to $\rho'(y_G)$; in particular, since ρ' satisfies the last clause y_F, ρ' satisfies F. Hence TSEITIN preserves satisfiability.

Question 1 Why does TSEITIN work in polynomial time? You will concentrate on the complexity of the various calls to CNF.

Very easy. 2 lines.

> *The following solution is accepted: Each call to* CNF *is on a formula of constant size (at most* 5 *in the case of binary operators). An exponential of a constant is constant...*
>
> *One should really be slightly more cautious, since the size of variables should be taken into account. If n is the size of F, we generate $O(n)$ fresh variables, so we can assume each one has size $O(\log n)$. It is easy to produce* $\mathrm{CNF}(x = y \wedge z)$, *for example: we write the clauses $\neg x \vee y$, $\neg x \vee z$, and $\neg y \vee \neg z \vee x$, in time and space $O(\log n)$. Similarly for the other operators.*

Question 2 A propositional formula F is *uniquely satisfiable* if and only if there is exactly one assignment ρ of truth values for each of the variables $x_1, \ldots, x_m$ that occur in F, such that ρ satisfies F. Show that F is uniquely satisfiable if and only if TSEITIN(F) is uniquely satisfiable.

Easy. I used 9 lines, but I am sure you don't need that much.

> *If ρ is the unique assignment (on $x_1, \ldots, x_m$) satisfying F, then the ρ' constructed in the main text satisfies* TSEITIN(F). *It is also the only one: as we said, any such ρ' satisfying* TSEITIN(F) *must map each variable y_G to the value of G under ρ', and then the remaining variables $x_1, \ldots, x_m$ must be mapped to their value under ρ.*
>
> *If ρ' is the unique assignment (on $x_1, \ldots, x_m$ and the various variables y_G), then the restriction ρ of ρ' to $\{x_1, \ldots, x_m\}$ satisfies F. It is also the only one, since any other assigment ρ_1 satisfying F would extend to an assignment ρ'_1 satisfying* TSEITIN(F) *that would coincide with ρ_1 on $x_1, \ldots, x_m$, hence would differ from ρ'.*
>
> *This argument can be extended to show that the map $\rho \mapsto \rho'$ is a bijection between the assignments on the variables of F that satisfy F and the assignments on the variables of* TSEITIN(F) *that satisfy* TSEITIN(F). *Hence the cardinality of the two sets of assignments is the same.*

2 The class MA

Recall that **MA** is the class of languages L such that, for every $\ell \geq 0$, there is a language $D \in \mathbf{P}$ such that, for every input x, of size n:

- if $x \in L$ then there is a y of size $p(n)$ such that $Pr_r[(x, y, r) \in D] \geq 1 - 1/2^{n^\ell}$;
- if $x \notin L$ then for every y of size $p(n)$, $Pr_r[(x, y, r) \in D] \leq 1/2^{n^\ell}$;

where the probabilities are taken over all random tapes r of size $q(n)$, and $p(n)$ and $q(n)$ are two polynomials (which may depend on ℓ).

Question 3 Show that we obtain the same class by requiring no error in the $x \in L$ case. In other words, let $\mathbf{MA}_0$ be the class defined as above, except for the clause:

- if $x \in L$ then there is a y of size $p(n)$ such that, for every r, $(x, y, r) \in D$.

You must show that $\mathbf{MA} = \mathbf{MA}_0$. As a hint, you may imitate the proof of the Sipser-Gács-Lautemann Theorem (Proposition 1.24 in the second set of lecture notes, `pcp.pdf`).

Application of a proof technique seen in class. 20 lines.

(Sipser's coding lemma will not do, since its use would require Arthur to play first and draw some hash functions at random.)

The inclusion $\mathbf{MA}_0 \subseteq \mathbf{MA}$ *is obvious.*

In the converse direction, we fix $L \in \mathbf{MA}$, *and* $\ell \in \mathbb{N}$. *For technical reasons, we shall use an* $\mathbf{MA}$ *protocol with error* $1/2^{n^{\ell+1}}$, *not* $1/2^{n^\ell}$.

Fix x. *For every* y *of size* $p(n)$, *let* R_y *be the set of random tapes* r *such that* $(x, y, r) \in D$. *If* $x \in L$, *then* R_y *is huge (covers a proportion at least* $1 - 1/2^{n^{\ell+1}}$ *of the whole space* $\Sigma^{q(n)}$ *of all random tapes). As in the proof of the Sipser-Gács-Lautemann Theorem,* $\bigcup_{i=0}^{\lceil q(n)/n^{\ell+1} \rceil} (t_i \oplus R_y) = \Sigma^{q(n)}$ *for some tuple of bitstrings* t_0, t_1, $\ldots$, $t_{\lceil q(n)/n^{\ell+1} \rceil}$, *each of size* $q(n)$. *In fact, this holds for at least half of all such tuples: the probability over* t_0, t_1, $\ldots$, $t_{\lceil q(n)/n^{\ell+1} \rceil}$ *of the complementary event, that there is an* $r \in \Sigma^{q(n)}$ *that is outside every* $t_i \oplus R_y$, *is at most* $2^{q(n)}$ *times* $(1/2^{n^{\ell+1}})(1 + \lceil q(n)/n^{\ell+1} \rceil)$, *hence at most* $1/2$.

In other words, we define the new one-round $\mathbf{MA}$ *protocol as follows: on input* x, *Merlin produces* y, *Arthur draws* r *and checks that* $(x, y, r \oplus t_i) \in D$ *for some* i, $0 \leq i \leq \lceil q(n)/n^{\ell+1} \rceil$.

We have just seen that, if $x \in L$, *then Merlin can find a* y *such that Arthur will accept, whatever value of* r *is drawn. If* $x \notin L$, *then for any answer* y *that Merlin may give, the probability that Arthur will accept is at most* $q(n) + 1$ *times (one for each* i*) the probability that* $(x, y, r \oplus t_i)$ *is in* D *(at most* $1/2^{n^{\ell+1}}$*). Asymptotically, this is at most* $1/2^{n^\ell}$, *and we conclude.*

Question 4 Deduce that $\mathbf{MA} \subseteq \Sigma_2^p$.

Easy. 5 lines.

If $L \in \mathbf{MA} = \mathbf{MA}_0$, then $x \in L$ if and only if there is a y of size $p(n)$ such that for every r of size $q(n)$, $(x, y, r) \in D$. This is clear when $x \in L$. When $x \notin L$, for every y a huge proportion of random tapes r (at least $1 - 1/2^{n^\ell}$) will falsify the claim that $(x, y, r) \in D$, hence at least one.

Now one can decide L by guessing y existentially, then r universally, then checking $(x, y, r) \in D$ in polynomial time.

3 The Zachos Lemma

Let us recall the $\mathbf{BP}\cdot$ operator from the lectures: for any complexity class $\mathcal{C}$, $\mathbf{BP} \cdot \mathcal{C}$ is the class of languages L such that there is a randomized polynomial time Turing machine $\mathcal{A}'$ and a language $D' \in \mathcal{C}$ such that, on input x (of size n):

- If $x \in L$, then $Pr_r[\mathcal{A}'(x, r) \in D'] \geq 2/3$;
- If $x \notin L$, then $Pr_r[\mathcal{A}'(x, r) \in D'] \leq 1/3$.

where probabilities are taken on random strings r of size $q(n)$, for some polynomial q in n.

Recall that an oracle machine is a multi-tape machine with a specific query tape, three extra control states Q, YES and NO. Let A be a language. The semantics of the machine with oracle A is as usual, except that when the machine reaches state Q, it then proceeds to state YES if the contents of the query tape is in A, and to NO otherwise. The *relativized* classes $\mathbf{P}^A$, $\mathbf{NP}^A$, $\mathbf{BPP}^A$, etc., are obtained from their classical counterpart by changing the underlying Turing machine model to the corresponding oracle machine, with oracle A.

For a complexity *class* $\mathcal{C}$, we write $\mathbf{NP}^{\mathcal{C}}$ for the union of the classes $\mathbf{NP}^{L'}$, $L' \in \mathcal{C}$.

It is clear that $\mathcal{C} \subseteq \mathcal{C}'$ implies $\mathbf{NP}^{\mathcal{C}} \subseteq \mathbf{NP}^{\mathcal{C}'}$.

Question 5 Show that $\mathbf{NP}^{\mathbf{BPP}} \subseteq \mathbf{MA}$.

Requires a bit of work, but no crazy new idea. 29 lines.

Let $L \in \mathbf{NP}^{\mathbf{BPP}}$. So there is language L' in $\mathbf{BPP}$ such that $L \in \mathbf{NP}^{L'}$. Let $\mathcal{M}$ be a non-deterministic Turing machine with oracle L' deciding L. We can assume all calls to the oracle to work with queries of a fixed polynomial size. Since $L' \in \mathbf{BPP}$, there is a polynomial-time language D such that if $x' \in L'$ then $Pr_r[(x', r) \in D] \geq 1 - 1/2^{p(n)}$ (for any fixed polynomial p, and n is the size of the input—not x'), and if $x' \notin L'$, then $Pr_r[(x', r) \in D] \leq 1/2^{p(n)}$.

We will fix $p(n)$ later on. We aim at providing an error bound $\leq 1/2^{n^\ell}$. Let m be the number of calls to the oracle; that is polynomial in n. We will see that we need to set $p(n) \geq \log_2 m + n^\ell$, hence $n^{\ell+1}$ fits asymptotically, for example.

We build the following **MA** *game. Merlin guesses a trace of* $\mathcal{M}$*, including the answers of the oracle. Then Arthur checks all the guessed answers to the oracle using his* **BPP** *algorithm, and also checks that the trace given by Merlin is a valid trace and one that accepts.*

If $x \in L$*, then Merlin can provide all the right answers to the oracle, and Arthur will confirm that each of those answers is right with probability at least* $1-1/2^{p(n)}$*. Hence Arthur can only reject provided he made a mistake at least once while checking Merlin's guess. That is, the probability that Arthur will reject is at most* $m/2^{p(n)}$*, where* m *is the number of calls to the oracle. Taking* $p(n) \geq \log_2 m + n^\ell$*, this is at most* $1/2^{n^\ell}$*, as desired.*

If $x \notin L$*, and assuming Merlin's trace is valid and accepts, Merlin must have cheated and given a wrong answer to at least one of the oracle calls. The probability that he goes undetected by Arthur is then at most* $m/2^{p(n)} \leq 1/2^{n^\ell}$*.*

Therefore $L \in$ **MA***.*

An alternative argument works by letting Merlin guess, not a trace of $\mathcal{M}$*, but a sequence of* $q(n)$ *bits, where* $q(n)$ *is a polynomial exceeding the time complexity of* $\mathcal{M}$*. Then Arthur simulates* $\mathcal{M}$*, resolving the* i*th call to the oracle by reading Merlin's* i*th bit. The error bounds are as above.*

Question 6 Show the *Zachos Lemma*: if **NP** $\subseteq$ **BPP**, then **PH** $\subseteq$ **BPP**. Here is the proof, your task is to replace the "why?" questions by appropriate justifications. If **NP** $\subseteq$ **BPP**, then:

Easy if you know your lessons. 15 lines, concentrated on one of the subquestions.

$$
\begin{aligned}
\mathbf{PH} &= \Sigma_2^p & &\text{why? (1)}\\
&= \mathbf{NP}^{\mathbf{NP}} & &\text{final comments in the } \texttt{nl.pdf} \text{ lecture notes}\\
&\subseteq \mathbf{NP}^{\mathbf{BPP}} & &\mathcal{C} \subseteq \mathcal{C}' \text{ implies } \mathbf{NP}^{\mathcal{C}} \subseteq \mathbf{NP}^{\mathcal{C}'}\\
&\subseteq \mathbf{MA} & &\textbf{Question 5}\\
&\subseteq \mathbf{AM} & &\text{why? (2)}\\
&= \mathbf{BP}\cdot\mathbf{NP} & &\text{why? (3)}\\
&\subseteq \mathbf{BP}\cdot\mathbf{BPP} & &\text{why? (4)}\\
&\subseteq \mathbf{BPP} & &\text{why? (5).}
\end{aligned}
$$

(1) Since **NP** $\subseteq$ **BPP***, this is Corllary 1.23 in the lecture notes (consequence of Adleman and Karp-Lipton).*

(2) Babai's theorem (Theorem 3.12).

(3) **AM** $=$ **BP** $\cdot$ **NP***: Proposition 3.5 of the lecture notes.*

(4) **BP**$\cdot$ *is a monotonic operator (Lemma 3.7).*

(5) Let $L \in \mathbf{BP} \cdot \mathbf{BPP}$. There is a language L' in $\mathbf{BPP}$ such that $x \in L$ implies $Pr_r[(x, r) \notin L'] \leq 1/6$, and $x \notin L$ implies $Pr_r[(x, r) \in L'] \leq 1/6$, in other words we can decide with error $\leq 1/6$. Since L' is in $\mathbf{BPP}$, we can decide $(x, r) \in L'$) with error at most $1/6$: there is a polynomial-time Turing machine $\mathcal{M}$ such that if $(x, r) \in L'$ then $Pr_{r'}[\mathcal{M}(x, r, r') = \bot] \leq 1/6$ and otherwise $Pr_{r'}[\mathcal{M}(x, r, r') = \top] \leq 1/6$.

Then, if $x \in L$, at most $1/6$ of the possible values of r are such that $(x, r) \notin L'$. For the remaining ones, at most $1/6$ of the possible values of r' are such that $\mathcal{M}(x, r, r') = \bot$. Hence at most $1/6 + 1/6.5/6 \leq 1/6 + 1/6 = 1/3$ of the possible pairs (r, r') are such that $\mathcal{M}(x, r, r') = \bot$. Therefore $Pr_{r,r'}[\mathcal{M}(x, r, r') = \bot] \leq 1/3$.

When $x \notin L$, a similar argument shows that $Pr_{r,r'}[\mathcal{M}(x, r, r') = \top] \leq 1/3$.

4 The Valiant-Vazirani theorem

Let $\Sigma = \mathbb{Z}/2\mathbb{Z}$ in this Section. Recall that a linear hash function $h \colon \Sigma^m \to \Sigma^{m'}$ is a linear map from $\mathbb{Z}/2\mathbb{Z}^m$ to $\mathbb{Z}/2\mathbb{Z}^{m'}$.

Question 7 Let F be a propositional formula in clausal form, built on propositional variables $x_1, \ldots, x_m$, say. Let X be the set of environments (mappings from the propositional variables $x_1, \ldots, x_m$ to truth-values) ρ that satisfy F (in notation, $\rho \models F$). Let $m' \geq 2$ be a number such that $2^{m'-2} \leq |X| \leq 2^{m'-1}$, where $|X|$ is the cardinality of X. Identify each environment ρ with the obvious vector in Σ^m. Show that:

$$Pr_{h,b}[\exists! \rho \in \Sigma^m \cdot \rho \models F \text{ and } h(\rho) = b] \geq \frac{1}{8}$$

where h is drawn at random uniformly among all linear hash functions from Σ^m to $\Sigma^{m'}$, and b is drawn at random uniformly, and independently, in $\Sigma^{m'}$. We write $\exists!$ for "there exists a unique". (Hint: given a fixed ρ, find a lower bound for the probability of the event $C_\rho(h, b)$, defined as holding whenever $h(\rho) = b$ but $h(\rho') \neq b$ for every $\rho' \in X$ such that $\rho' \neq \rho$.) As in class. 21 lines.

Given a fixed $\rho \in X$:

$$\begin{aligned} Pr_{h,b}[C_\rho(h, b)] &= Pr_{h,b}[h(\rho) = b \textit{ and } \rho \textit{ is not a collision for } h \textit{ in } X] \\ &= Pr_{h,b}[h(\rho) = b \mid \rho \textit{ is not a collision for } h \textit{ in } X] \\ &\quad .Pr_{h,b}[\rho \textit{ is not a collision for } h \textit{ in } X] \end{aligned}$$

by Bayes' law for example. The probability that $h(\rho) = b$ knowing that ρ is not a collision for h in X (assuming such h and b exist) is exactly *the mean of all probabilities over b that $b = b_0$, for various values of b_0 (which do not matter, those are the values of the form $h(\rho)$, $h \in X$), i.e., $1/2^{m'}$.*

We now use the technique we used to show Sipser's first coding lemma (Lemma 3.13 in the notes) with $\ell = 1$. The probability over h and b (hence in fact over h alone) that ρ is a collision for h in X is:

$$\begin{aligned} Pr_h[\exists \rho' \in X \cdot \rho' \neq \rho \text{ and } h(\rho') = h(\rho)] &\leq \sum_{\rho' \neq \rho} \frac{1}{2^{m'}} \\ &\leq \frac{2^{m'-1} - 1}{2^{m'}} \leq \frac{1}{2} \end{aligned}$$

since $|X| \leq 2^{m'-1}$. So the probability that ρ is not a collision for h in X is at least $1/2$.

We therefore obtain:

$$Pr_{h,b}[C_\rho(h, b)] \geq \frac{1}{2^{m'+1}}$$

Knowing this,

$$\begin{aligned} Pr_{h,b}[\exists! \rho \in \Sigma^m \cdot \rho \models F \text{ and } h(\rho) = b] &= Pr_{h,b}[\exists \rho \in X \cdot C_\rho(h, b)] \\ &= \sum_{\rho \in X} Pr_{h,b}[C_\rho(h, b)] \end{aligned}$$

since the events $C_\rho(h, b)$ are disjoint as ρ varies. Since $2^{m'-2} \leq |X|$ and $Pr_{h,b}[C_\rho(h, b)] \geq \frac{1}{2^{m'+1}}$, this is greater than or equal to $2^{m'-2} \cdot \frac{1}{2^{m'+1}} = \frac{1}{8}$.

Question 8 We take F and m as above, but we no longer assume that m' is known. Show that, if we draw m' at random uniformly among $\{2, 3, \ldots, m + 1\}$, and a linear hash function $h\colon \Sigma^m \to \Sigma^{m'}$ and a vector b in $\Sigma^{m'}$ at random as before, then: Very easy, 3 lines.

- if F is satisfiable, then $Pr_{m',h,b}[\exists! \rho \in \Sigma^m \cdot \rho \models F \text{ and } h(\rho) = b] \geq 1/(8m)$.

If F is satisfiable, then the set X has cardinality between $2^{m'-2}$ and $2^{m'-1}$ for some unique $m' \in \{2, 3, \ldots, m + 1\}$. We find the right m', say m'_0, with probability $1/m$, and then $Pr_{h,b}[\exists! \rho \in \Sigma^m \cdot \rho \models F \text{ and } h(\rho) = b] \geq$

$1/8$ for $m' = m'_0$. Formally:

$$
\begin{aligned}
& Pr_{m',h,b}[\exists! \rho \in \Sigma^m \cdot \rho \models F \textit{ and } h(\rho) = b] \\
& = \frac{1}{m} \sum_{m'=2}^{m+1} Pr_{h\colon \Sigma^m \to \Sigma^{m'}, b \in \Sigma^{m'}}[\exists! \rho \in \Sigma^m \cdot \rho \models F \textit{ and } h(\rho) = b] \\
& \quad \textit{reflecting that } m' \textit{ is chosen uniformly among the } m \textit{ values } 2, \ldots, m+1 \\
& \geq \frac{1}{m} Pr_{h\colon \Sigma^m \to \Sigma^{m'_0}, b \in \Sigma^{m'_0}}[\exists! \rho \in \Sigma^m \cdot \rho \models F \textit{ and } h(\rho) = b] \\
& \quad \textit{since the sum is larger than its summand for } m' = m'_0 \\
& \geq \frac{1}{m} . \frac{1}{8},
\end{aligned}
$$

by the previous question.

Question 9 Define a randomized polynomial time algorithm $\mathcal{W}$ that takes a propositional formula F in clausal form as input (on m variables $x_1, \ldots, x_m$ as above) and returns a propositional formula F' in clausal form such that: (a) if F is satisfiable, then F' is uniquely satisfiable with probability at least $1/(8m)$, and (b) if F is unsatisfiable, then F' is unsatisfiable.

You need to combine a few things you know here. 11 lines.

The formula F' is a conjunction of F with some set S of clauses that represent the condition $h(\rho) = b$, where (m' and) h and b are found at random, as before.

The difficulty is that, although $h(\rho) = b$ is a propositional formula, converting it to clausal form may take time exponential in m, hence in the size of the input. This is where we will use the Tseitin transform.

The condition $h(\rho) = b$ is a conjunction of formulae $A_i \cdot \rho = b_i$, $1 \leq i \leq m'$, one for each row (seeing h as a matrix). Hence it suffices to be able to convert formulae of this form to clausal form, in polynomial time, by adding extra variables, and preserving unique *solutions. This is done by using **Question 2** (and is also the reason why we included the operators $\oplus$ and $=$ there: $A_i \cdot \rho = b_i$ is really the formula $x_{i_1} \oplus \cdots \oplus x_{i_k} = b_i$, where $i_1, \ldots, i_k$ are the indices of the 1 entries in A_i).*

Question 10 On input F (a clausal form again), we now build k formulae $F_1, \ldots, F_k$ in clausal form, by calling $\mathcal{W}$ k times, and where k is a parameter, depending polynomially on the size n of F. Let $\epsilon \in]0, 1[$ be an arbitrary parameter (possibly depending on the size n of F). We wish to find k such that: (a) if F is satisfiable, then at least one of $F_1, \ldots, F_k$ is uniquely satisfiable, with probability at least $1 - \epsilon$, and (b) if F is unsatisfiable, then no formula F_i is uniquely satisfiable. Show that one can achieve this, by giving an explicit

Elementary. 4 lines.

formula for k as a function of n and ϵ.

The probability that no F_i is uniquely satisfiable in case (a) is the product of the individual probabilities, by independence, hence is at most $(1 - 1/(8m))^k$. Since $m \neq n$, this is at most $(1 - 1/(8n))^k$. We wish this to be at most ϵ, hence we must have $k \geq \log \epsilon / \log(1 - 1/(8n)) \sim -8n \log \epsilon$.

Question 11 Deduce the *Valiant-Vazirani theorem*: if $USAT \in \mathbf{P}$, then $\mathbf{NP} = \mathbf{RP}$. Here USAT is the unique satisfiability problem: given a clausal form F (on variables $x_1, \ldots, x_m$), is there a unique $\rho \in \Sigma^m$ that satisfies F? Not too hard. 9 lines.

We know that $\mathbf{RP} \subseteq \mathbf{NP}$ (lemma 1.6 in `pcp.pdf`). Conversely, it suffices to show that SAT is in $\mathbf{RP}$, since $\mathbf{RP}$ is closed under polynomial time reductions and SAT is $\mathbf{NP}$-complete.

On input F (a clausal form), we run $\mathcal{W}(F)$ k times, as in the previous question, and we obtain k formulae $F_1, \ldots, F_k$. We design k so that $\epsilon = 1/2$, i.e., $k \sim 8n \log 2$.

Then we test whether some F_i is uniquely satisfiable (in polynomial time, by assumption). If F is satisfiable, then this will succeed with probability at least $1 - \epsilon = 1/2$. Otherwise, this will always fail.

Hence we have obtained an $\mathbf{RP}$ algorithm deciding SAT.

We define another operator $\oplus\cdot$ ("parity") as follows: $L \in \oplus \cdot \mathcal{C}$ iff there is a language L' in $\mathcal{C}$, and a polynomial $p(n)$, such that:

- $x \in L$ iff the number of strings y of size $p(n)$ such that $(x, y) \in L'$ is *odd*.

I.e., $\oplus \cdot \mathbf{P}$ is the class of languages decidable on a (balanced, i.e., binary branching and whose branches all have the same length) non-deterministic Turing machine by accepting iff the number of accepting branches is odd.

Question 12 Using the same ideas as before, show that $\mathbf{NP} \subseteq \mathbf{RP}^{\oplus \cdot \mathbf{P}}$. Largely doable. 9 lines.

*Instead of testing whether some F_i is uniquely satisfiable, we test whether some some F_i has an odd number of satisfying assignments. If F is satisfiable, this will succeed with probability at least $1 - \epsilon = 1/2$ for the same reason as before: if F_i is uniquely satisfiable, then it has an odd number of satisfying assignments. The new thing is that if F is unsatisfiable, then every F_i has zero (an even number) of satisfying assignments (this is stronger than condition (b) of **Question 10**).*

Now, checking whether a formula has an odd number of satisfying assignments can be done by a non-deterministic Turing machine that guesses *the values of each variable non-deterministically, and once this is done, evaluates the formula. All the branches have the same computation length, and the number of accepting branches is the number of satisfying assignments.*