The Complexity of Verifying Loop-free Programs as Differentially Private

Marco Gaboardi
Boston University, USA

Kobbi Nissim
Georgetown University, USA

David Purser
University of Warwick, UK
MPI-SWS, Saarbrücken, Germany

Abstract

We study the problem of verifying differential privacy for loop-free programs with probabilistic choice. Programs in this class can be seen as randomized Boolean circuits, which we will use as a formal model to answer two different questions: first, deciding whether a program satisfies a prescribed level of privacy; second, approximating the privacy parameters a program realizes. We show that the problem of deciding whether a program satisfies ε-differential privacy is $\mathbf{coNP}^{\#\mathbf{P}}$-complete. In fact, this is the case when either the input domain or the output range of the program is large. Further, we show that deciding whether a program is (ε, δ)-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-hard, and in $\mathbf{coNP}^{\#\mathbf{P}}$ for small output domains, but always in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$. Finally, we show that the problem of approximating the level of differential privacy is both **NP**-hard and **coNP**-hard. These results complement previous results by Murtagh and Vadhan [34] showing that deciding the optimal composition of differentially private components is #**P**-complete, and that approximating the optimal composition of differentially private components is in **P**.

2012 ACM Subject Classification Security and privacy; Theory of computation → Probabilistic computation

Keywords and phrases differential privacy, program verification, probabilistic programs

Funding M. G. and K. N. were supported by NSF grant No. 1565387 TWC: Large: Collaborative: Computing Over Distributed Sensitive Data. D. P. was supported by the UK EPSRC Centre for Doctoral Training in Urban Science (EP/L016400/1).

Acknowledgements Research partially done while M.G. and K.N. participated in the "Data Privacy: Foundations and Applications" program held at the Simons Institute, UC Berkeley in spring 2019.

1 Introduction

Differential privacy [22] is currently making significant strides towards being used in large scale applications. Prominent real-world examples include the use of differentially private computations by the US Census' OnTheMap project[1], applications by companies such as Google and Apple [24, 37, 4, 18], and the US Census' plan to deploy differentially private releases in the upcoming 2020 Decennial [1].

More often than not, algorithms and their implementations are analyzed "on paper" to show that they provide differential privacy. This analysis—a proof that the outcome distribution of the algorithm is stable under the change in any single individual's information—is often intricate and may contain errors (see [31] for an illuminating discussion about several wrong versions of the sparse vector algorithm which appeared in the literature). Moreover,

[1] `https://onthemap.ces.census.gov`

even if it is actually differentially private, an algorithm may be incorrectly implemented when used in practice, e.g. due to coding errors, or because the analysis makes assumptions which do not hold in finite computers, such as the ability to sample from continuous distributions (see [33] for a discussion about privacy attacks on naive implementations of continuous distributions). Verification tools may help validate, given the code of an implementation, that it would indeed provide the privacy guarantees it is intended to provide. However, despite the many verification efforts that have targeted differential privacy, e.g. [38, 9, 41, 25, 7, 45, 6, 2, 14, 15] based on automated or interactive techniques, little is known about the complexity of some of the basic problems in this area. Our aim is to clarify the complexity of some of these problems.

In this paper, we consider the computational complexity of determining whether programs satisfy (ε, δ)-differential privacy. The problem is generally undecidable, and we hence restrict our attention to probabilistic loop-free programs, which are part of any reasonable programming language supporting random computations. To approach this question formally, we consider probabilistic circuits. The latter are Boolean circuits with input nodes corresponding both to input bits and to uniformly random bits ("coin flips") where the latter allow the circuit to behave probabilistically (see Figure 1). We consider both decision and approximation versions of the problem, where in the case of decision the input consists of a randomized circuit and parameters ε, δ and in the case of approximation the input is a randomized circuit, the desired approximation precision, and one of the two parameters ε, δ. In both cases, complexity is measured as function of the total input length in bits (circuit and parameters).

Previous works have studies the complexity of composing differentially private components. For any k differentially private algorithms with privacy parameters $(\varepsilon_1, \delta_1), \ldots, (\varepsilon_k, \delta_k)$ their composition is also differentially private [22, 23, 34], making composition a powerful design tool for differentially private programs. However, not all interesting differentially private programs are obtained by composing differentially private components, and a goal of our work is to understand what is the complexity of verifying that full programs are differentially private, and how this complexity differs from the one for programs which result of composing differentially private components.

Regarding the resulting parameters, the result of composing the k differentially private algorithms above results in $(\varepsilon_g, \delta_g)$-differentially private for a multitude of possible $(\varepsilon_g, \delta_g)$ pairs. Murtagh and Vadhan showed that determining the minimal ε_g given δ_g is $\#\mathbf{P}$-complete [34]. They also gave a polynomial time approximation algorithm that computes ε_g to arbitrary accuracy, giving hope that for "simple" programs deciding differential privacy or approximating of privacy parameters may be tractable. Unfortunately, our results show that this is not the case.

1.1 Contributions

Following the literature, we refer to the variant of differential privacy where $\delta = 0$ as *pure* differential privacy and to the variant where $\delta > 0$ as *approximate* differential privacy. We contribute in three directions.

- **Bounding pure differential privacy.** We show that determining whether a randomized circuit is ε-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-complete.[2] To show hardness in $\mathbf{coNP}^{\#\mathbf{P}}$ we

[2] The class $\mathbf{coNP}^{\#\mathbf{P}}$ is contained in **PSPACE** and contains the polynomial hierarchy (as, per Toda's Theorem, $\mathbf{PH} \subseteq \mathbf{P}^{\#\mathbf{P}}$).

consider a complement to the problem E-Maj-Sat [30], which is complete for $\mathbf{NP}^{\#\mathbf{P}}$ [13]. In the complementary problem, All-Min-Sat, given a formula ϕ over $n+m$ variables the task is to determine if for all allocations $\boldsymbol{x} \in \{0,1\}^n$, $\phi(\boldsymbol{x}, \boldsymbol{y})$ evaluates to true on no more than $\frac{1}{2}$ of allocations to $\boldsymbol{y} \in \{0,1\}^m$.

- **Bounding approximate differential privacy.** Turning to the case where $\delta > 0$, we show that determining whether a randomized circuit is (ε, δ)-differentially private is $\mathbf{coNP}^{\#\mathbf{P}}$-complete when the number of output bits is small relative to the total size of the circuit and otherwise between $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.
- **Approximating the parameters ε and δ.** Efficient approximation algorithms exist for optimal composition [34], and one might expect the existence of polynomial time algorithms to approximate ε or δ in randomized circuits. We show this is **NP**-hard and **coNP**-hard, and therefore an efficient algorithm does not exist (unless $\mathbf{P} = \mathbf{NP}$).

Our results show that for loop-free programs with probabilistic choice directly verifying whether a program is differentially private is intractable. These results apply to programs in any reasonable programming language supporting randomized computations. Hence, they set the limits on where to search for automated techniques for these tasks.

The relation to quantitative information flow.

Differential privacy shares similarities with quantitative information flow [17, 27], which is an entropy-based theory measuring how secure a program is. Alvim et al. [3] showed that a bound on pure differential privacy implies a bound on quantitative information flow. So, one could hope that bounding differential privacy could be easier than bounding quantitative information flow. Yasuoka and Terauchi [43] have shown that bounding quantitative information flow for loop free boolean programs with probabilistic choice is **PP**-hard (but in **PSPACE**). In contrast, our results show that bounding pure differential privacy is $\mathbf{coNP}^{\#\mathbf{P}}$-complete. Chadha et al. [11] showed the problem to be **PSPACE**-complete for boolean programs with loops and probabilistic choice (notice that this would be not true for programs with integers). We leave the analogous question for future works.

2 Preliminaries

Numbers.

By a *number given as a rational* we mean a number of the form $\frac{x}{y}$ where x, y are given as binary integers.

2.1 Loop-free Probabilistic Programs

We consider a simple loop-free imperative programming language built over Booleans, and including probabilistic choice.

$$
\begin{array}{lcll}
x & ::= & [\mathrm{a}{-}\mathrm{z}]^+ & \text{(variable identifiers)} \\
b & ::= & \texttt{true} \mid \texttt{false} \mid \texttt{random} \mid x \mid b \wedge b \mid b \vee b \mid \neg b & \text{(boolean expressions)} \\
c & ::= & \texttt{SKIP} \mid x := b \mid c;\ c \mid \texttt{if}\ b\ \texttt{then}\ c\ \texttt{else}\ c & \text{(commands)} \\
t & ::= & x \mid t, x & \text{(list of variables)} \\
p & ::= & \texttt{input}(t);\ c;\ \texttt{return}(t) & \text{(programs)}
\end{array}
$$

Probabilistic programs [29] extend standard programs with the addition of coin tosses; this is achieved by the probabilistic operation `random`, which returns either `true` or `false`

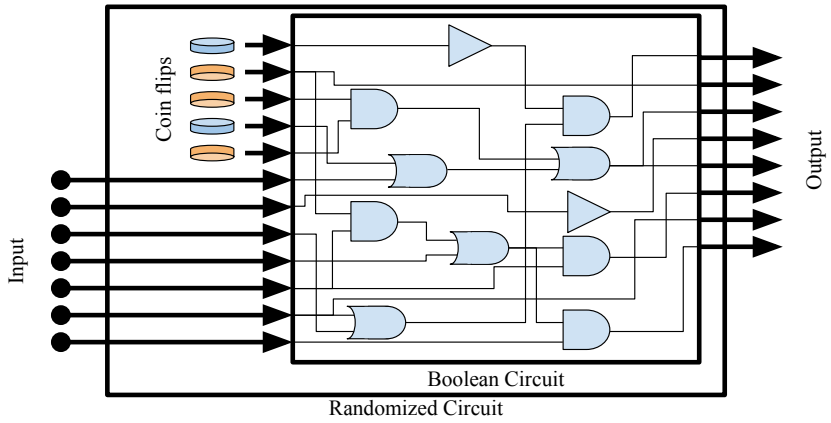

Figure 1 Example randomized circuit

with equal probability. A standard operation, sometimes denoted by $c \oplus c$, which computes one of the two expressions with probability $\frac{1}{2}$ each is achieved with `if random then` c `else` c. The notation $c \oplus c$ is avoided as $\oplus$ refers to *exclusive or* in this paper.

The semantics of the programming language are standard and straight forward. Without loss of generality, each variable assignment is final, that is, each assignment must go to a fresh variable. Looping behaviour is not permitted, although bounded looping can be encoded by unrolling the loop.

▶ Remark 1. The language is equally expressive as if it were also handle standard integer operations, where the integers are of bounded size. Further details are given in Appendix A.

2.2 Probabilistic Circuits

▶ **Definition 2.** *A Boolean circuit ψ with n inputs and ℓ outputs is a directed acyclic graph $\psi = (V, E)$ containing n input vertices with zero in-degree, labeled $X_1, \ldots, X_n$ and ℓ output vertices with zero out-degree, labeled $O_1, \ldots, O_\ell$. Other nodes are assigned a label in $\{\wedge, \vee, \neg\}$, with vertices labeled $\neg$ having in-degree one and all others having in-degree two. The size of ψ, denoted $|\psi|$, is defined to be $|V|$. A* randomized circuit *has m additional random input vertices labeled $R_1, \ldots, R_m$.*

Given an input string $\boldsymbol{x} = (x_1, \ldots, x_n) \in \{0,1\}^n$, the circuit is evaluated as follows. First, the values $x_1, \ldots, x_n$ are assigned to the nodes labeled $X_1, \ldots, X_n$. Then, m bits $\boldsymbol{r} = (r_1, \ldots, r_m)$ are sampled uniformly at random from $\{0,1\}^m$ and assigned to the nodes labeled $R_1, \ldots, R_m$. Then, the circuit is evaluated in topological order in the natural way. E.g., let v be a node labeled $\wedge$ with incoming edges $(u_1, v), (u_2, v)$ where u_1, u_2 were assigned values z_1, z_2 then v is assigned the value $z_1 \wedge z_2$. The outcome of ψ is $(o_1, \ldots, o_\ell)$, the concatenation of values assigned to the ℓ output vertices $O_1, \ldots, O_\ell$.

For input $\boldsymbol{x} \in \{0,1\}^n$ and event $E \subseteq \{0,1\}^\ell$ we have

$$\Pr[\psi(\boldsymbol{x}) \in E] = \frac{|\{\boldsymbol{r} \in \{0,1\}^m \ : \ \psi(\boldsymbol{x}, \boldsymbol{r}) \in E\}|}{2^m}.$$

▶ Remark 3. The operators, $\wedge, \vee$ and $\neg$ are functionally complete. However, we will also use $\oplus$ (exclusive or), such that $p \oplus q \iff (p \vee q) \wedge \neg(p \wedge q)$.

2.3 Equivalence of Programs and Circuits

▶ **Lemma 4.** *A loop-free probabilistic program can be converted into an equivalent probabilistic boolean circuit in linear time in the size of the program (and vice-versa).*

Proof sketch. It is clear that a probabilistic circuit can be expressed as a probabilistic program using just boolean operations by expressing a variable for each vertex after sorting the vertices in topological order.

To convert a probabilistic Boolean program into a probabilistic circuit, each of the commands can be handled using a fixed size sub-circuit, each of which can be composed together appropriately. ◀

Given the established equivalence between loop-free probabilistic programs and probabilistic circuits, the remainder of the paper will use probabilistic circuits.

2.4 Differential Privacy in Probabilistic Circuits

Let X be any input domain. An input to a differentially private analysis would generally be an array of elements from X, i.e., $\boldsymbol{x} = (x_1, \ldots, x_n) \in X^n$.

The definition of differential privacy depends on adjacency between inputs, we define *neighboring* inputs.

▶ **Definition 5.** *Inputs $\boldsymbol{x} = (x_1, \ldots, x_n)$ and $\boldsymbol{x}' = (x'_1, \ldots, x'_n) \in X^n$ are called* neighboring *if there exist $i \in [n]$ s.t. for all $j \neq i$ then $x_j = x'_j$.*

In this work, we will consider input domains with finite representation. Without loss of generality we set $X = \{0,1\}^k$ and hence an array $x = (x_1, \ldots, x_n)$ can be written as a sequence of nk bits, and given as input to a (randomized) circuit with nk inputs. Our lower bounds work already for for $k = 1$ and our upper bounds are presented using $k = 1$ but generalise to all k.

▶ **Definition 6** (Differential Privacy [22, 21])**.** *A probabilistic circuit ψ is (ε, δ)-differentially private if for all neighboring $\boldsymbol{x}, \boldsymbol{x}' \in X^n$ and for all $E \subseteq \{0,1\}^\ell$,*

$$\Pr[\psi(\boldsymbol{x}) \in E] \leq e^\varepsilon \cdot \Pr[\psi(\boldsymbol{x}') \in E] + \delta.$$

Following common use, we refer to the case where $\delta = 0$ as *pure* differential privacy and to the case where $\delta > 0$ as *approximate* differential privacy. When omitted, δ is understood to be zero.

2.5 Problems of deciding and approximating differential privacy

We formally define our three problems of interest.

▶ **Definition 7.** *The problem* Decide-ε-DP *asks, given ε and ψ, if ψ is ε-differentially private. We assume ε is given by the input e^ε as a rational number.*

▶ **Definition 8.** *The problem* Decide-ε, δ-DP *asks, given ε, δ and ψ, if ψ is (ε, δ)-differentially private. We assume ε is given by the input e^ε as a rational number.*

▶ **Definition 9.** *Given an approximation error $\gamma > 0$, the* Approximate-δ *problem and the* Approximate-ε *problem, respectively, ask:*

- *Given ε, find $\hat{\delta} \in [0,1]$, such that $0 \leq \hat{\delta} - \delta \leq \gamma$, where δ is the minimal value such that ψ is (ε, δ)-differentially private.*
- *Given δ, find $\hat{\varepsilon} \geq 0$, such that $0 \leq \hat{\varepsilon} - \varepsilon \leq \gamma$, where ε is the minimal value such that ψ is (ε, δ)-differentially private.*

2.6 The class coNP$^{\#\mathbf{P}}$

A language L is in **coNP**$^{\#\mathbf{P}}$ if its problem membership can be refuted using a polynomial time non-deterministic Turing machine with access to a $\#\mathbf{P}$ oracle. Alternatively, $x \in L$ iff *all* branches of the non-deterministic Turing machine accept. It is easy to see that **coNP**$^{\#\mathbf{P}}$ = **coNP**$^{\mathbf{PP}}$. Finally, **PH** $\subseteq$ **coNP**$^{\#\mathbf{P}}$ $\subseteq$ **PSPACE**, where **PH** $\subseteq$ **coNP**$^{\#\mathbf{P}}$ follows by Toda's theorem (**PH** $\subseteq \mathbf{P}^{\#\mathbf{P}}$) [40].

The following decision problem is complete for **NP**$^{\#\mathbf{P}}$ [13]:

▶ **Definition 10.** E-Maj-Sat *asks, given ϕ a quantifier free formula over $n+m$ variables if there exist an allocation $\boldsymbol{x} \in \{0,1\}^n$ such that there are strictly greater than $\frac{1}{2}$ of allocations to $\boldsymbol{y} \in \{0,1\}^m$ where $\phi(\boldsymbol{x}, \boldsymbol{y})$ evaluates to true.*

The complementary problem All-Min-Sat, is complete for **coNP**$^{\#\mathbf{P}}$: a formula ϕ is All-Min-Sat, if ϕ is not E-Maj-Sat. That is, ϕ a quantifier free formula over $n+m$ variables is All-Min-Sat if for all allocations $\boldsymbol{x} \in \{0,1\}^n$ there are no more than $\frac{1}{2}$ of allocations to $\boldsymbol{y} \in \{0,1\}^m$ where $\phi(\boldsymbol{x}, \boldsymbol{y})$ evaluates to true.

3 The complexity of deciding pure differential privacy

In this section we classify the complexity of deciding ε-differential privacy, for which we show the following theorem:

▶ **Theorem 11.** Decide-ε-DP *is* **coNP**$^{\#\mathbf{P}}$*-complete.*

It will be convenient to consider the well-known simpler reformulation of the definition of pure differential privacy in finite ranges to consider specific outcomes $\boldsymbol{o} \in \{0,1\}^\ell$ rather than events $E \subseteq \{0,1\}^\ell$.

▶ **Reformulation 1** (Pure differential privacy)**.** A probabilistic circuit ψ is ε-differentially private if and only if for all neighboring $\boldsymbol{x}, \boldsymbol{x'} \in X^n$ and for all $\boldsymbol{o} \in \{0,1\}^\ell$,

$$\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}] \leq e^\varepsilon \cdot \Pr[\psi(\boldsymbol{x'}) = \boldsymbol{o}].$$

3.1 Decide-ε-DP is in coNP$^{\#\mathbf{P}}$

We show a non-deterministic Turing machine which can 'refute' ψ being ε-differentially private in polynomial time with a $\#\mathbf{P}$ oracle. A circuit ψ is shown not to be ε-differentially private by exhibiting a combination $\boldsymbol{x}, \boldsymbol{x'}, \boldsymbol{o}$ such that $\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}] > e^\varepsilon \cdot \Pr[\psi(\boldsymbol{x'}) = \boldsymbol{o}]$. The witness to the non-deterministic Turing machine would be a sequence of $2n$ bits parsed as neighboring inputs $\boldsymbol{x}, \boldsymbol{x'} \in \{0,1\}^n$ and ℓ bits describing an output $\boldsymbol{o} \in \{0,1\}^\ell$. The constraint can then be checked in polynomial time, using the $\#\mathbf{P}$ oracle to compute $\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}]$ and $\Pr[\psi(\boldsymbol{x'}) = \boldsymbol{o}]$.

To compute $\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}]$ in $\#\mathbf{P}$ we create an instance to #CircuitSat, which will count the number of allocations to the m probabilistic bits consistent with this output. We do this by extending ψ with additional gates reducing to a single output which is true only when the input is fixed to $\boldsymbol{x}$ and the output of ψ was $\boldsymbol{o}$.

3.2 coNP$^{\#\mathbf{P}}$-hardness of Decide-ε-DP

To show **coNP**$^{\#\mathbf{P}}$-hardness of Decide-ε-DP we show a reduction from All-Min-Sat in Lemma 12; together with the inclusion result above, this entails that Decide-ε-DP is **coNP**$^{\#\mathbf{P}}$-complete (Theorem 11).

3.2.1 Randomized Response

Randomized response [42] is a technique for answering sensitive Yes/No questions by flipping the answer with probability $p < 0.5$. Setting $p = \frac{1}{1+e^\varepsilon}$ gives ε-differential privacy.

▶ **Lemma 12.** All-Min-Sat *reduces in polynomial time to* Decide-ε-DP.

Proof. We will reduce from All-Min-Sat to Decide-ε-DP using randomized response. We will take a boolean formula ϕ and create a probabilistic circuit that is ε-differentially private if and only if ϕ is All-Min-Sat.

Consider the circuit ψ which takes as input the value $z \in \{0,1\}$. It probabilistically chooses a value of $\boldsymbol{x} \in \{0,1\}^n$ and $\boldsymbol{y} \in \{0,1\}^m$ and one further random bit p_1 and computes $b = z \oplus \neg(p_1 \vee \phi(\boldsymbol{x}, \boldsymbol{y}))$. The circuit outputs $(\boldsymbol{x}, b)$.

▷ Claim 13. ψ is $\ln(3)$-differentially private if and only if ϕ is All-Min-Sat.

Suppose $\phi \in$ All-Min-Sat then, no matter the choice of $\boldsymbol{x}$,

$$0 \leq \Pr_y[\phi(\boldsymbol{x}, \boldsymbol{y}) = 1] \leq \frac{1}{2},$$

and hence

$$\frac{1}{4} \leq \Pr_{y,p_1}[\neg(p_1 \vee \phi(\boldsymbol{x}, \boldsymbol{y})) = 1] \leq \frac{1}{2}.$$

We conclude the true answer z is flipped between $\frac{1}{4}$ and $\frac{1}{2}$ of the time, recall this is exactly the region in which randomized response gives us the most privacy. In the worst case $p = \frac{1}{4} = \frac{1}{1+e^\varepsilon}$, gives $e^\varepsilon = 3$, so $\ln(3)$-differential privacy.

In the converse, suppose $\phi \in$ E-Maj-Sat, then for some $\boldsymbol{x}$

$$\frac{1}{2} < \Pr_y[\phi(\boldsymbol{x}, \boldsymbol{y}) = 1] \leq 1,$$

and then

$$\Pr_{y,p_1}[\neg(p_1 \vee \phi(\boldsymbol{x}, \boldsymbol{y})) = 1] < \frac{1}{4},$$

in which case the randomized response does not provide $\ln(3)$-differential privacy. ◀

▶ Remark 14. We skew the result so that in the positive case (when $\phi \in$ All-Min-Sat) the proportion of accepting allocations is between $\frac{1}{4}$ and $\frac{1}{2}$, resulting in the choice of $\ln(3)$-differentially privacy. Alternative skews, using more bits akin to p_1, shows hardness for other choices of ε.

3.2.2 Hardness by circuit shape

In our proof of the upper-bound we use **coNP** to resolve the non-deterministic choice of both input and output. We show this is necessary in the sense **coNP** is still required for either large input or large output. The hardness proof used in Lemma 12 shows that when $|\psi| = n$ the problem is hard for $\Omega(1)$-bit input and $\Omega(n)$-bit output.

We can also prove (Lemma 24 in Appendix B) this is hard for $\Omega(n)$-bit input and $\Omega(1)$-bit output. Intuitively a counter example to differential privacy has two choices: a pair of adjacent input and a given output upon which the relevant inequality will hold. So to "refute" All-Min-Sat the counterexample of the All choice (i.e. $\boldsymbol{x}$) can be selected in the input, rather than the output as in our case. Since the input is now non-trivial we must take care of what happens when the adjacent bit is in the choice of $\boldsymbol{x}$.

Further the problem is in $\mathbf{P}^{\#\mathbf{P}}$ for $O(\log(n))$-bit input and $O(\log(n))$-bit output, as in this case, the choices made by **coNP** can instead be checked deterministically in polynomial time. In this case we show **PP**-hardness, which applies even when there is 1-bit input and 1-bit output.

4 On the complexity of deciding approximate differential privacy

It is less clear whether deciding (ε, δ)-differential privacy can be done in $\mathbf{coNP}^{\#\mathbf{P}}$. First we consider restrictions to the shape of the circuit so that $\mathbf{coNP}^{\#\mathbf{P}}$ can be recovered, and then show that in general the problem is in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.

Recall that in the case of ε-differential privacy it was enough to consider singleton events $\{\boldsymbol{o}\}$ where $\boldsymbol{o} \in \{0,1\}^\ell$, however in the definition of (ε, δ)-differential privacy we must quantify over output events $E \subseteq \{0,1\}^\ell$. If we consider circuits with one output bit ($\ell = 1$), then the event space essentially reduces to $E \in \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$ and we can apply the same technique.

We expand this to the case when the number of outputs bits is logarithmic $\ell \leq \log(|\psi|)$. To cater to this, rather than guessing a violating $E \in \{0,1\}^\ell$, we consider a violating subset of events $E \subseteq \{0,1\}^\ell$. Given such an event E we create a circuit ψ_E on ℓ inputs and a single output which indicates whether the input is in the event E. The size of this circuit is exponential in ℓ, thus polynomial in $|\psi|$. Composing $\psi_E \circ \psi$, we check the conditions hold for this event E, with just one bit of output.

▷ Claim 15. DECIDE-ε, δ-DP, restricted to circuits ψ with ℓ bit outputs where $\ell \leq \log(|\psi|)$, is in $\mathbf{coNP}^{\#\mathbf{P}}$ (and hence $\mathbf{coNP}^{\#\mathbf{P}}$-complete).

The claim trivially extends to $\ell \leq c \cdot \log(|\psi|)$ for any *fixed* $c > 0$.

4.1 DECIDE-ε, δ-DP is in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$

We now show that DECIDE-ε, δ-DP in the most general case can be solved in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$. We will assume $e^\varepsilon = \alpha$ is given as a rational, with $\alpha = \frac{u}{v}$ for some integers u and v. While we will use non-determinism to choose inputs leading to a violating event, unlike in Section 3 it would not be used for finding a violating event E, as an (explicit) description of such an event may be of super-polynomial length. It would be useful for us to use a reformulation of approximate differential privacy, using a sum over potential individual outcomes.

▶ **Reformulation 2** (Pointwise differential privacy [7]). A probabilistic circuit ψ is (ε, δ)-differentially private if and only if for all neighboring $\boldsymbol{x}, \boldsymbol{x}' \in X^n$ and for all $\boldsymbol{o} \in \{0,1\}^\ell$,

$$\sum_{\boldsymbol{o} \in \{0,1\}^\ell} \delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o}) \leq \delta,$$

where

$$\delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o}) = \max\left(\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}] - e^\varepsilon \cdot \Pr[\psi(\boldsymbol{x}') = \boldsymbol{o}], 0\right).$$

We define $\mathcal{M}$, a non-deterministic Turing Machine with access to a $\#\mathbf{P}$-oracle, and where each execution branch runs in polynomial time. On inputs a probabilistic circuit ψ and neighboring $\boldsymbol{x}, \boldsymbol{x}' \in X^n$ the number of accepting executions of $\mathcal{M}$ would be proportional to $\sum_{\boldsymbol{o} \in \{0,1\}^\ell} \delta_{\boldsymbol{x},\boldsymbol{x}'}(\boldsymbol{o})$.

In more detail, on inputs ψ, $\boldsymbol{x}$ and $\boldsymbol{x}'$, $\mathcal{M}$ chooses $\boldsymbol{o} \in \{0,1\}^\ell$ and an integer $C \in \{1, 2, \ldots, 2^{m+\lceil \log(v) \rceil}\}$ (this requires choosing $l + m + \lceil \log(v) \rceil$ bits). Through a call to the $\#\mathbf{P}$ oracle, $\mathcal{M}$ computes

$$a = |\{\boldsymbol{r} \in \{0,1\}^m : \psi(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{o}\}|$$

and

$$b = |\{\boldsymbol{r} \in \{0,1\}^m : \psi(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{o}\}| .$$

Finally, $\mathcal{M}$ accepts if $v \cdot a - u \cdot b \geq C$ and otherwise rejects.

▶ **Lemma 16.** *Given two inputs $\boldsymbol{x}, \boldsymbol{x}' \in X^n$, $\mathcal{M}(\psi, \boldsymbol{x}, \boldsymbol{x}')$ has exactly $v \cdot 2^m \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \delta_{\boldsymbol{x}, \boldsymbol{x}'}(\boldsymbol{o})$ accepting executions.*

Proof. Let $\mathbb{1}\{X\}$ be the indicator function, which is one if the predicate X holds and zero otherwise.

$$\begin{aligned}
v \cdot 2^m \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \delta_{\boldsymbol{x}, \boldsymbol{x}'}(\boldsymbol{o}) &= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} v \cdot 2^m \max\left(\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}] - \alpha \Pr[\psi(\boldsymbol{x}') = \boldsymbol{o}], 0\right) \\
&= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} v 2^m \max\left(\frac{1}{2^m} \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{o}\} - \alpha \frac{1}{2^m} \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \\
&= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \max\left(v \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{o}\} - v\alpha \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \\
&= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \max\left(v \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{o}\} - u \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \\
&= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \sum_{C=1}^{2^{\lceil \log(v) \rceil + m}} \mathbb{1}\left\{\max\left(v \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = \boldsymbol{o}\} - u \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = \boldsymbol{o}\}, 0\right) \geq C\right\} \\
&= \text{number of accepting executions in } \widehat{\mathcal{M}}
\end{aligned}$$

◀

We can now describe our $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ procedure for DECIDE-ε, δ-DP. The procedure takes as input a probabilistic circuit ψ.

1. Non-deterministically choose neighboring $\boldsymbol{x}$ and $\boldsymbol{x}' \in \{0,1\}^n$ (i.e., $2n$ bits).
2. Let $\mathcal{M}$ be the non-deterministic Turing Machine with access to a $\#\mathbf{P}$-oracle as described above. Create a machine $\widehat{\mathcal{M}}$ with no input that executes $\mathcal{M}$ on $\psi, \boldsymbol{x}, \boldsymbol{x}'$.
3. Make an $\#\mathbf{P}^{\#\mathbf{P}}$ oracle call for the number of accepting executions $\widehat{\mathcal{M}}$ has.
4. Reject if the number of accepting executions is greater than $v \cdot 2^m \cdot \delta$ and otherwise accept.

By Lemma 16, there is a choice $\boldsymbol{x}, \boldsymbol{x}'$ on which the procedure rejects if and only if ψ is not (ε, δ)-differentially private.

4.2 Hardness

Theorem 11 shows that DECIDE-ε-DP is $\mathbf{coNP}^{\#\mathbf{P}}$-complete, in particular $\mathbf{coNP}^{\#\mathbf{P}}$-hard and since DECIDE-$\varepsilon$-DP is a special case of DECIDE-ε, δ-DP, this is also $\mathbf{coNP}^{\#\mathbf{P}}$-hard. Nevertheless the proof is based on particular values of ε and we provide an alternative proof of hardness based on δ (Theorem 29 in Appendix C). This proof result will apply for any ε (even for $\varepsilon = 0$) and for a large range of δ (but not $\delta = 0$).

The proof proceeds by first considering the generalisation of All-Min-Sat to the version where *minority*, i.e. less than $\frac{1}{2}$ of the assignments, is replaced with another threshold. This problem is also $\mathbf{coNP}^{\#\mathbf{P}}$-hard for a range of thresholds. Note however, if this threshold is exactly 1 the problem is true for all circuits, and if the threshold is 0 the problem is simply asks if the formula is unsatisfiable (a **coNP** problem).

This generalised problem can then be reduced to deciding Decide-ε, δ-DP, where the threshold corresponds exactly to δ. It will turn out in the resulting circuit ε does not change the status of differential privacy, i.e. it is (ε, δ)-differentially private for all ε, or not.

The proof shows hardness for $\Omega(n)$-input bits and 1-output bit; the case in which there also exists a $\mathbf{coNP}^{\#\mathbf{P}}$ upper-bound. To show hardness in a higher complexity class, e.g., $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$, it would be required to use a circuit with more output bits.

5 Inapproximability of the privacy parameters ε, δ

Given the difficulty of deciding if a circuit is differentially private, one might naturally consider whether approximating ε or δ could be efficient. We show that these tasks are both **NP**-hard and **coNP**-hard.

We show that distinguishing between (ε, δ), and (ε', δ')-differential privacy is **NP**-hard, by reduction from a problem we call Not-Constant which we also show is **NP**-hard. A boolean formula is in Not-Constant if it is satisfiable and not also a tautology.

▶ **Lemma 17.** Not-Constant *is* **NP***-hard. (hence* Constant *is* **coNP***-hard).*

Proof. Clearly, Not-Constant $\in$ **NP**, the witness being a pair of satisfying and non-satisfying assignments. We reduce 3-SAT to Not-Constant. Given a Boolean formula ϕ over variables $x_1, \ldots, x_n$ let $\phi'(x_1, \ldots, x_n, x_{n+1}) = \phi(x_1, \ldots, x_n) \wedge x_{n+1}$. Note that ϕ' is never a tautology as $\phi'(x_1, \ldots, x_n, 0) = 0$. Furthermore, ϕ' is satisfiable iff ϕ is. ◀

In Section 3.2.1 we used randomized response in the pure differential privacy setting. We now consider the approximate differential privacy variant $RR_{\varepsilon,\delta} : \{0,1\} \to \{\top, \bot\} \times \{0,1\}$ defined as follows:

$$RR_{\varepsilon,\delta}(x) = \begin{cases} (\top, x) & \text{w.p. } \delta \\ (\bot, x) & \text{w.p. } (1-\delta)\frac{\alpha}{1+\alpha} \\ (\bot, \neg x) & \text{w.p. } (1-\delta)\frac{1}{1+\alpha} \end{cases} \quad \text{where} \quad \alpha = e^{\varepsilon}$$

I.e., with probability δ, $RR_{\varepsilon,\delta}(x)$ reveals x and otherwise it executes $RR_{\varepsilon}(x)$. The former is marked with "$\top$" and the latter with "$\bot$". This mechanism is equivalent to the one described in [34] and is (ε, δ)-differentially private.

▶ **Definition 18.** *Let* $0 \leq \varepsilon \leq \varepsilon'$, $0 \leq \delta \leq \delta' \leq 1$, *with either* $\varepsilon < \varepsilon'$ *or* $\delta < \delta'$. *The problem* Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP *takes as input a circuit* ψ, *guaranteed to be either* (ε, δ)*-differentially private, or* (ε', δ')*-differentially private. The problem asks whether* ψ *is* (ε, δ)*-differentially private or* (ε', δ')*-differentially private.*

▶ **Lemma 19.** Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP *is* **NP***-hard (and* **coNP***-hard).*

Proof. We reduce Not-Constant to Distinguish-$(\varepsilon, \delta), (\varepsilon', \delta')$-DP. Given the boolean formula $\phi(\boldsymbol{x})$ on n bits, we create a probabilistic circuit ψ. The input to ψ consists of the n bits $\boldsymbol{x}$ plus a single bit y. The circuit ψ has four output bits (o_1, o_2, o_3, o_4) such that $(o_1, o_2) = RR_{\varepsilon,\delta}(y)$ and $(o_3, o_4) = RR_{\varepsilon',\delta'}(\phi(\boldsymbol{x}))$.

Observe that $(o_1, o_2) = RR_{\varepsilon,\delta}(y)$ is always (ε, δ) differentially private. As for $(o_3, o_4) = RR_{\varepsilon',\delta'}(\phi(\boldsymbol{x}))$, if $\phi \in$ Not-Constant then there are adjacent $\boldsymbol{x}, \boldsymbol{x}'$ such that $\phi(\boldsymbol{x}) \neq \phi(\boldsymbol{x}')$. In this case, $(o_3, o_4) = RR_{\varepsilon',\delta'}(\phi(\boldsymbol{x}))$ is (ε', δ')-differentially private, and, because $(\varepsilon, \delta) < (\varepsilon', \delta')$, so is ψ . On the other hand, if $\phi \notin$ Not-Constant then $\phi(\boldsymbol{x})$ does not depend on $\boldsymbol{x}$ and hence (o_3, o_4) does not affect privacy, in which case we get that ψ is (ε, δ) differentially private.

The same argument also gives **coNP**-hardness. ◀

Notice that the above theorem holds when $\delta = \delta'$ and $\varepsilon < \varepsilon'$ (similarly, $\varepsilon = \varepsilon'$ and $\delta < \delta'$), which entails the following theorem:

▶ **Theorem 20.** *Assuming* $\mathbf{P} \neq \mathbf{NP}$*, for any approximation error* $\gamma > 0$*, there does not exist a polynomial time approximation algorithm that given a probabilistic circuit* ψ *and* δ *computes some* $\hat{\varepsilon}$*, where* $|\hat{\varepsilon} - \varepsilon| \leq \gamma$ *and* ε *is the minimal such that* ψ *is* (ε, δ)*-differentially private within error* γ*. Similarly, given* ε*, no such* $\hat{\delta}$ *can be computed polynomial time where* $|\hat{\delta} - \delta| \leq \gamma$ *and* δ *is minimal.*

▶ Remark 21. The result also applies when approximating within a given ratio $\rho > 1$ (e.g. in the case of approximating ε, to find $\hat{\varepsilon}$ such that $\frac{\hat{\varepsilon}}{\varepsilon} \leq \rho$). Moreover, the result also holds when approximating pure differential privacy, that is when $\delta = 0$.

6 Related work

Differential privacy was introduced in [22]. It is a definition of privacy in the context of data analysis capturing the intuition that information specific to an individuals is protected if every single user's input has a bounded influence on the computation's outcome distribution, where the bound is specified by two parameters, usually denoted by ε, δ. Intuitively, these parameters set an upperbound on privacy loss, where the parameter ε limits the loss and the parameter δ limits the probability in which the loss may exceed ε.

Extensive work has occurred in the computer-assisted or automated of verification of differential privacy. Early work includes, PINQ [32] and Airavat [39] which are systems that keep track of the privacy budgets (ε and δ) using trusted privacy primitives in SQL-like and MapReduce-like paradigms respectively. In other work, programming languages were developed, that use the type system to keep track of the sensitivity and ensure the correct level of noise is added [38, 9, 16, 8]. Another line of work uses proof assistants to help prove that an algorithm is differentially private [7]; although much of this work is not automated, recent work has gone in this direction [2, 45].

These techniques focuses on 'soundness', rather than 'completeness' thus are not amenable to complexity analysis. In the constrained case of verifying differential privacy on probabilistic automata and Markov chains there are bisimulation based techniques [41, 12]. Towards complexity analysis; [15] shows that computing the optimal value of δ for a finite labelled Markov chain is undecidable. Further [14] and [15] provides distances, which are (necessarily) not tight, but can be computed in polynomial time with an **NP** oracle and a weaker bound in polynomial time. Recent works have focused on developing techniques for finding violations of differential privacy [19, 10]. The methods proposed so far have been based on some form of testing. Our result limits also the tractability of these approaches. Finally, [5] proposes an automated technique for proving differential privacy or finding counterexamples. This paper studies a constrained class of programs extending the language we presented here, and provides a 'complete' procedure for deciding differential privacy for them. The paper does

not provide any complexity guarantee for the proposed method and we expect our results to apply also in their setting.

As we already discussed, Murtagh and Vadhan [34] showed that finding the optimal values for the privacy parameters when composing different algorithms in a black-box way is #**P**-complete, but also that approximating the optimal values can be done efficiently. In contrast, our results show that when one wants to consider programs as white-box, as often needed to achieve better privacy guarantees (e.g. in the case of the sparse vector technique), the complexity is higher.

Several works have explored different property testing related to differential privacy [20, 28, 26], including verification [26]. In the standard model used in property testing, a user has only black-box access to the function and the observable outputs are the ones provided by a privacy mechanism. In contrast, our work is based on the program description and aim to provide computational limits to the design of techniques for program analyses for differential privacy.

We already discussed some works on quantitative information flow. In addition to those, it was shown that comparing the quantitative information flow of two programs on inputs coming from the uniform distribution is #**P**-hard [44]. However, when quantifying over all distributions the question is **coNP**-complete [44].

7 Conclusions and future work

Verifying differential privacy of loop-free probabilistic boolean programs.

We have shown the difficulty of verifying differential privacy in loop-free probabilistic boolean programs through their correspondence with probabilistic circuits. Deciding ε-differential privacy is $\mathbf{coNP}^{\#\mathbf{P}}$-complete and (ε, δ)-differential privacy is $\mathbf{coNP}^{\#\mathbf{P}}$-hard and in $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ (a gap that we leave for future work). Both problems are positioned in the counting hierarchy, in between the polynomial hierarchy **PH** and **PSPACE**.

Verifying differential privacy of probabilistic boolean programs.

One interesting question that our work leaves open is the characterization of the complexity of deciding differential privacy problems for probabilistic boolean programs, including loops. Similarly to the works on quantitative information flow [11], we expect these problems to be decidable and we expect them to be in **PSPACE**. However, this question requires some further investigation that we leave for future work.

Solvers mixing non-determinism and counting.

Returning to our motivation for this work—developing practical tools for verifying differential privacy—our results seem to point to a deficiency in available tools for model checking. The model checking toolkit includes well established SAT solvers for **NP** (and **coNP**) problems, solvers for further quantification in **PH**, solvers for #SAT (and hence for #**P** problems[3]). However to the best of our knowledge, there are currently no solvers that are specialized for mixing the polynomial hierarchy **PH** and counting problems #**P**, in particular $\mathbf{coNP}^{\#\mathbf{P}}$ and $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$.

[3] See, for example, `http://beyondnp.org/pages/solvers/`, for a range of solvers

Approximating the differential privacy parameters.

We show that distinguishing (ε, δ)-differential privacy from (ε', δ') differential privacy where $(\varepsilon, \delta) < (\varepsilon', \delta')$ is both **NP**- and **coNP**-hard. We leave refining the classification of this problem as an open problem.

References

1 John M Abowd. The us census bureau adopts differential privacy. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2867–2867. ACM, 2018.

2 Aws Albarghouthi and Justin Hsu. Synthesizing coupling proofs of differential privacy. *Proceedings of the ACM on Programming Languages*, 2(POPL):58, 2017.

3 Mário S Alvim, Miguel E Andrés, Konstantinos Chatzikokolakis, and Catuscia Palamidessi. On the relation between differential privacy and quantitative information flow. In *International Colloquium on Automata, Languages, and Programming*, pages 60–76. Springer, 2011.

4 Apple. Apple differential privacy technical overview. URL: `https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf`.

5 Gilles Barthe, Rohit Chadha, Vishal Jagannath, A Prasad Sistla, and Mahesh Viswanathan. Automated methods for checking differential privacy. *arXiv preprint arXiv:1910.04137*, 2019.

6 Gilles Barthe, Marco Gaboardi, Emilio Jesús Gallego Arias, Justin Hsu, Aaron Roth, and Pierre-Yves Strub. Higher-order approximate relational refinement types for mechanism design and differential privacy. In *POPL*, pages 55–68, 2015. `doi:10.1145/2676726.2677000`.

7 Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. In *2016 31st Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, pages 1–10. IEEE, 2016.

8 Gilles Barthe, Marco Gaboardi, Justin Hsu, and Benjamin Pierce. Programming language techniques for differential privacy. *ACM SIGLOG News*, 3(1):34–53, 2016.

9 Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella Beguelin. Probabilistic relational reasoning for differential privacy. *ACM SIGPLAN Notices*, 47(1):97–110, 2012.

10 Benjamin Bichsel, Timon Gehr, Dana Drachsler-Cohen, Petar Tsankov, and Martin T. Vechev. Dp-finder: Finding differential privacy violations by sampling and optimization. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 508–524, 2018. `doi:10.1145/3243734.3243863`.

11 Rohit Chadha, Dileep Kini, and Mahesh Viswanathan. Quantitative information flow in boolean programs. In *International Conference on Principles of Security and Trust*, pages 103–119. Springer, 2014.

12 Konstantinos Chatzikokolakis, Daniel Gebler, Catuscia Palamidessi, and Lili Xu. Generalized bisimulation metrics. In *CONCUR*, pages 32–46. Springer, 2014.

13 Dmitry Chistikov, Rayna Dimitrova, and Rupak Majumdar. Approximate counting in smt and value estimation for probabilistic programs. *Acta Informatica*, 54(8):729–764, 2017.

14 Dmitry Chistikov, Andrzej S Murawski, and David Purser. Bisimilarity distances for approximate differential privacy. In *International Symposium on Automated Technology for Verification and Analysis*, pages 194–210. Springer, 2018.

15 Dmitry Chistikov, Andrzej S Murawski, and David Purser. Asymmetric distances for approximate differential privacy. In *30th International Conference on Concurrency Theory (CONCUR 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

16 Loris D'Antoni, Marco Gaboardi, Emilio Jesús Gallego Arias, Andreas Haeberlen, and Benjamin Pierce. Sensitivity analysis using type-based constraints. In *Proceedings of the 1st annual workshop on Functional programming concepts in domain-specific languages*, pages 43–50. ACM, 2013.

17 Dorothy E. Denning. *Cryptography and Data Security*. Addison-Wesley, 1982.

18 Differential Privacy Team, Apple. Learning with privacy at scale. 2017. URL: https://machinelearning.apple.com/docs/learning-with-privacy-at-scale/appledifferentialprivacysystem.pdf.

19 Zeyu Ding, Yuxin Wang, Guanhong Wang, Danfeng Zhang, and Daniel Kifer. Detecting violations of differential privacy. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 475–489, 2018. doi:10.1145/3243734.3243818.

20 Kashyap Dixit, Madhav Jha, Sofya Raskhodnikova, and Abhradeep Thakurta. Testing the lipschitz property over product distributions with applications to data privacy. In *Theory of Cryptography - 10th Theory of Cryptography Conference, TCC 2013, Tokyo, Japan, March 3-6, 2013. Proceedings*, pages 418–436, 2013. doi:10.1007/978-3-642-36594-2_24.

21 Cynthia Dwork, Krishnaram Kenthapadi, Frank McSherry, Ilya Mironov, and Moni Naor. Our data, ourselves: Privacy via distributed noise generation. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 486–503. Springer, 2006.

22 Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.

23 Cynthia Dwork, Guy N. Rothblum, and Salil P. Vadhan. Boosting and differential privacy. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 51–60, 2010. doi:10.1109/FOCS.2010.12.

24 Úlfar Erlingsson, Vasyl Pihur, and Aleksandra Korolova. Rappor: Randomized aggregatable privacy-preserving ordinal response. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 1054–1067. ACM, 2014.

25 Matthew Fredrikson and Somesh Jha. Satisfiability modulo counting: A new approach for analyzing privacy properties. In *Proceedings of the Joint Meeting of the Twenty-Third EACSL Annual Conference on Computer Science Logic (CSL) and the Twenty-Ninth Annual ACM/IEEE Symposium on Logic in Computer Science (LICS)*, page 42. ACM, 2014.

26 Anna C Gilbert and Audra McMillan. Property testing for differential privacy. In *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 249–258. IEEE, 2018.

27 J. W. Gray. Probabilistic interference. In *Proceedings. 1990 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 170–179, May 1990. doi:10.1109/RISP.1990.63848.

28 Madhav Jha and Sofya Raskhodnikova. Testing and reconstruction of lipschitz functions with applications to data privacy. *SIAM J. Comput.*, 42(2):700–731, 2013. doi:10.1137/110840741.

29 Dexter Kozen. Semantics of probabilistic programs. *J. Comput. Syst. Sci.*, 22(3):328–350, 1981. doi:10.1016/0022-0000(81)90036-2.

30 Michael L Littman, Judy Goldsmith, and Martin Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998.

31 Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *PVLDB*, 10(6):637–648, 2017. URL: http://www.vldb.org/pvldb/vol10/p637-lyu.pdf, doi:10.14778/3055330.3055331.

32 Frank D McSherry. Privacy integrated queries: an extensible platform for privacy-preserving data analysis. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of data*, pages 19–30. ACM, 2009.

33 Ilya Mironov. On significance of the least significant bits for differential privacy. In *CCS*. ACM, 2012.

34 Jack Murtagh and Salil Vadhan. The complexity of computing the optimal composition of differential privacy. In *Theory of Cryptography Conference*, pages 157–175. Springer, 2016.

35 Roelof Maarten Marie Oberman. *Digital circuits for binary arithmetic*. Macmillan International Higher Education, 1979.

36 Federico Olmedo, Friedrich Gretz, Nils Jansen, Benjamin Lucien Kaminski, Joost-Pieter Katoen, and Annabelle McIver. Conditioning in probabilistic programming. *ACM Trans. Program. Lang. Syst.*, 40(1):4:1–4:50, 2018. doi:10.1145/3156018.

37 Nicolas Papernot, Martín Abadi, Ulfar Erlingsson, Ian Goodfellow, and Kunal Talwar. Semi-supervised knowledge transfer for deep learning from private training data. *arXiv preprint arXiv:1610.05755*, 2016.

38 Jason Reed and Benjamin C Pierce. Distance makes the types grow stronger: a calculus for differential privacy. In *ACM Sigplan Notices*, volume 45, pages 157–168. ACM, 2010.

39 Indrajit Roy, Srinath TV Setty, Ann Kilzer, Vitaly Shmatikov, and Emmett Witchel. Airavat: Security and privacy for mapreduce. In *NSDI*, volume 10, pages 297–312, 2010.

40 Seinosuke Toda. PP is as hard as the polynomial-time hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

41 Michael Carl Tschantz, Dilsun Kaynar, and Anupam Datta. Formal verification of differential privacy for interactive systems. *ENTCS*, 276:61–79, 2011.

42 Stanley L Warner. Randomized response: A survey technique for eliminating evasive answer bias. *Journal of the American Statistical Association*, 60(309):63–69, 1965.

43 Hirotoshi Yasuoka and Tachio Terauchi. On bounding problems of quantitative information flow. In *European Symposium on Research in Computer Security*, pages 357–372. Springer, 2010.

44 Hirotoshi Yasuoka and Tachio Terauchi. Quantitative information flow-verification hardness and possibilities. In *2010 23rd IEEE Computer Security Foundations Symposium*, pages 15–27. IEEE, 2010.

45 Danfeng Zhang and Daniel Kifer. Lightdp: towards automating differential privacy proofs. In *ACM SIGPLAN Notices*, volume 52, pages 888–901. ACM, 2017.

A Equivalence and expressivity of circuits and programs

This section will prove Lemma 4, but first let us expand on Remark 1 to prove Lemma 4 for an apparently more general language. We observe in Remark 22 that they are equivalent.

Let us consider the following consider programs expressed by the following language. This is an extension to the language defined in Section 2.1, supporting integer operations.

$$
\begin{array}{rcll}
x & ::= & [\mathrm{a}{-}\mathrm{z}]^+ & \text{(variable identifiers)} \\
i & ::= & 0 \mid 1 \mid \ldots \mid 2^k - 1 & \text{(constants)} \\
b & ::= & \texttt{true} \mid \texttt{false} \mid \texttt{random} \mid x \mid b \wedge b \mid b \vee b \mid \neg b & \text{(boolean expressions)} \\
 & & \quad \mid e > e \mid e \geq e \mid e = e & \\
e & ::= & i \mid x \mid e + e \mid e \times e \mid e - e \mid (e) \mid b \mid \texttt{sample} & \text{(integer expressions)} \\
c & ::= & \texttt{SKIP} \mid x := e \mid c;\ c \mid \texttt{if } b \texttt{ then } c \texttt{ else } c & \text{(commands)} \\
t & ::= & x \mid t, x & \text{(list of variables)} \\
p & ::= & \texttt{input}(t);\ c;\ \texttt{return}(t) & \text{(programs)}
\end{array}
$$

We define the programming language to operate over integers of a fixed size, parametrised by k. Integer expressions, denoted e, take on values from the field $F_{2^k} = \{0, \ldots, 2^k - 1\}$, where each integer can be represented with k bits. Formally we consider the operations working on unsigned integers, where overflows wrap around. Similarly one could equally consider signed integers operating over $\{-2^{k-1}, \ldots, 2^{k-1} - 1\}$. Boolean values can also be interpreted as the integers 0 and 1. The `sample` command uniformly chooses from $\{0, \ldots, 2^k - 1\}$; this is syntactic sugar for $2^{k-1} \times \texttt{random} + \cdots + 2^0 \times \texttt{random}$.

▶ Remark 22. This language is equally expressive without integers, as Boolean operations can be used to simulated them—although programming in such a language may be somewhat tedious. To see this note that the following proof shows that programs in these extended

language can be converted into probabilistic circuits, and similarly converted back into the Boolean fragment of the program. Both of these conversions occur in *linear* time.

Proof of Lemma 4. As noted in the proof sketch, it is clear that probabilistic boolean circuits can be converted into probabilistic loop-free Boolean programs.

For the reverse conversion, we show any probabilistic loop-free program *with Booleans and bounded integers* can be converted into a probabilistic *Boolean circuit* by showing that every expression can be constructed as a fixed sub-circuit.

- Boolean operations $\wedge, \vee, \neg$ are built into the circuit.
- `random` is achieved by introducing a fresh random input to the circuit for each call to `random`.
- In the evaluation of expressions e, the value of each sub-expression can be stored as the output at most k gates. The standard operations $+, \times, -, \geq, >, =$ can each be computed in a circuit (see e.g. [35]). Multiplication of two k-bit numbers may require $2k$-bits; since the language is defined over the field, these are renormalised back into the field by taking the k least significant bits. Similarly adding two k-bit numbers results in $k+1$-bits, which must similarly be transformed back into the field. Note that it is acceptable to have up to $2k$ bits temporarily in the circuit.
- Assignment $x := e$. Denote the k gates as the output of expression e as the variable x. References to x then use these k output bits.
- Branching `if` b `then` c_L `else` c_R simulated in the following way: Create a circuit for each branch, taking two copies of each variable that can be assigned in either c_L or c_R. At the end the true copy of each variable can be set. Let $(x)_i$ be the i^{th} bit of variable x then set x by $(x)_i \leftarrow ((x_L)_i \wedge b) \vee ((x_R)_i \wedge (\neg b))$, so that $x = x_L$ if b is true and $x = x_R$ if b is false.
- Constants are achieved by expressing the number in binary and using k gates to represent the value. Any gate can be forced to be one by $x \vee \neg x$ and forced to be zero by $x \vee \neg x$ for any input or random bit.
- `return`$(x_1, \ldots, x_m)$ is achieved by dedicating the $m \times k$ gates as output gates. ◀

▶ **Example 23.** Consider the following loop-free probabilistic program, with $k = 4$. It can be transformed into the circuit shown in Figure 2 through the procedure given in the proof of Lemma 4.

```
input(a,b)
x = 3 × a
y : = x + b
if random then z := x else z := y
return(z)
```

B Hardness of DECIDE-ε-DP by number of input/output bits

▶ **Lemma 24.** *Given a circuit ψ, we show the the following hardness results for large and small number of input and output bits:*

# Input Bits	*# Output Bits*	*Hardness*
$\Omega(n)$	*1*	**coNP**$^{\#\mathbf{P}}$*-hard*
1	$\Omega(n)$	**coNP**$^{\#\mathbf{P}}$*-hard*
1	*1*	**PP***-hard*

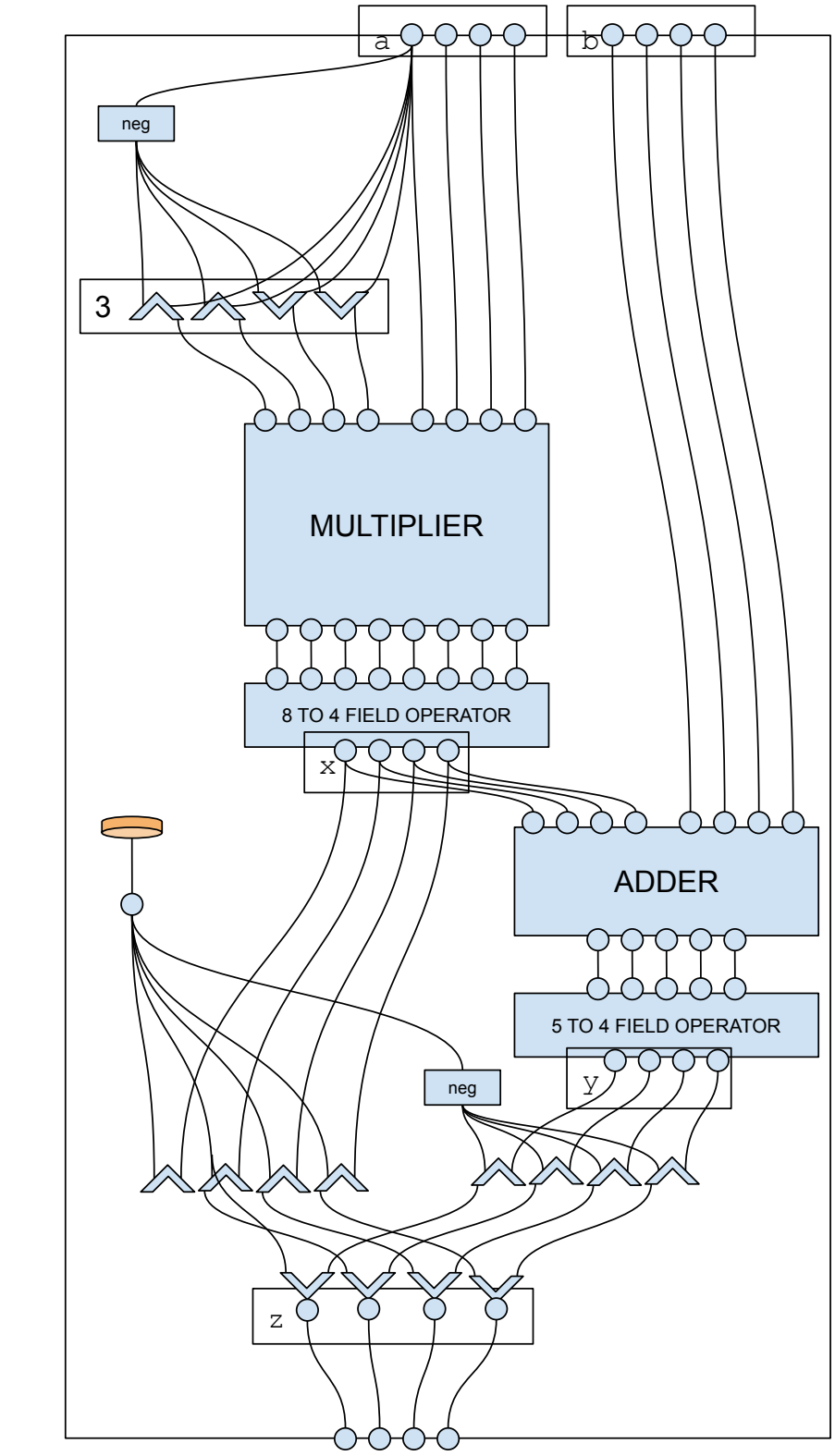

Figure 2 Example Circuit Transformation from Example 23

▶ Remark 25. Note that the hardness results entail hardness for any larger number of input and output bits; for example $\Theta(\log n)$-input,$\Theta(\log n)$-output is **PP**-hard and $\Theta(n)$-input,$\Theta(n)$-output is $\mathbf{coNP}^{\#\mathbf{P}}$-hard.

Proof for large input small output. Given $\phi(\boldsymbol{x}, \boldsymbol{y})$, we reduce $\phi \in$ All-Min-Sat to Decide-ε-DP. Our resulting circuit ψ will have 1 output bit but $n+1$ input bits

Let $\psi(\boldsymbol{x}, z) = (z \vee p_1) \wedge (\neg z \vee (p_2 \vee (p_3 \wedge p_4 \wedge \phi(\boldsymbol{x}, \boldsymbol{r}))))$, with $p_1, \ldots, p_4, \boldsymbol{r}$ determined randomly. This circuit has the property:

- If $z = 0$ return 1 w.p. $\frac{1}{2}$.
- If $z = 1$ return 1 w.p. $\frac{1}{2} + \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1]$

▷ Claim 26. $\phi \in$ All-Min-Sat $\iff$ $\ln(\frac{4}{3})$-differential privacy holds.

If $\phi \notin$ All-Min-Sat then for some $\boldsymbol{x}$ with $\Pr[\phi(\boldsymbol{x}) = 1] > \frac{1}{2}$, $\Pr[\phi(\boldsymbol{x}) = 0] < \frac{1}{2}$

$$\begin{aligned}
\frac{\Pr[\psi(\boldsymbol{x}, 0) = 0]}{\Pr[\psi(\boldsymbol{x}, 1) = 0]} &= \frac{\frac{1}{2}}{1 - \frac{1}{2} - \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4} - \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1])} \\
&== \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}(1 - \Pr[\phi(\boldsymbol{x}) = 1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}(1 - \Pr[\phi(\boldsymbol{x}) = 1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}(\Pr[\phi(\boldsymbol{x}) = 0])} \\
&> \frac{\frac{1}{2}}{\frac{1}{4} + \frac{1}{4}\frac{1}{2}} = \frac{4}{3} \approx 1.3
\end{aligned}$$

If $\phi \in$ All-Min-Sat then for all $\boldsymbol{x}$ we have $\Pr[\phi(\boldsymbol{x}) = 1] \leq \frac{1}{2}$, $\Pr[\phi(\boldsymbol{x}) = 0] \geq \frac{1}{2}$

$$\frac{\Pr[\psi(\boldsymbol{x}, 0) = 0]}{\Pr[\psi(\boldsymbol{x}, 1) = 0]} \leq \frac{4}{3} \approx 1.3$$

$$\begin{aligned}
\frac{\Pr[\psi(\boldsymbol{x}, 1) = 0]}{\Pr[\psi(\boldsymbol{x}, 0) = 0]} &= \frac{1 - \frac{1}{2} - \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1]}{\frac{1}{2}} \\
&= \frac{\frac{1}{4} + \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 0]}{\frac{1}{2}} \leq 1
\end{aligned}$$

$$\frac{\Pr[\psi(\boldsymbol{x}, 1) = 1]}{\Pr[\psi(\boldsymbol{x}, 0) = 1]} = \frac{\frac{1}{2} + \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1]}{\frac{1}{2}} \leq \frac{\frac{1}{2} + \frac{1}{4}\frac{1}{2}}{\frac{1}{2}} = 1.25$$

$$\frac{\Pr[\psi(\boldsymbol{x}, 0) = 1]}{\Pr[\psi(\boldsymbol{x}, 1) = 1]} = \frac{\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1]} \leq 1$$

$$\frac{\Pr[\psi(\boldsymbol{x}, 0) = 1]}{\Pr[\psi(\boldsymbol{x}', 0) = 1]} = \frac{\frac{1}{2}}{\frac{1}{2}} = 1$$

$$\frac{\Pr[\psi(\boldsymbol{x}, 1) = 1]}{\Pr[\psi(\boldsymbol{x}', 1) = 1]} = \frac{\frac{1}{2} + \frac{1}{4}\Pr[\phi(\boldsymbol{x}) = 1]}{\frac{1}{2} + \frac{1}{4}\Pr[\phi(\boldsymbol{x}') = 1]} \leq \frac{\frac{1}{2} + \frac{1}{4}\frac{1}{2}}{\frac{1}{2} + \frac{1}{4}0} = 1.25$$

$$\frac{\Pr[\psi(\boldsymbol{x},1)=0]}{\Pr[\psi(\boldsymbol{x}',1)=0]} = \frac{1-\frac{1}{2}-\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1]}{1-\frac{1}{2}-\frac{1}{4}\Pr[\phi(\boldsymbol{x}')=1]} \leq \frac{\frac{1}{2}+\frac{1}{4}0}{\frac{1}{2}-\frac{1}{4}\frac{1}{2}} = \frac{4}{3}$$

◀

Proof for small input large output. Given $\phi(\boldsymbol{x},\boldsymbol{y})$, we reduce $\phi \in$ All-Min-Sat to Decide-ε-DP. Our resulting circuit ψ will have 1 input bit but $n+1$ output bits.

Let $\psi(z) = (\boldsymbol{x}, (z \vee p_1) \wedge (\neg z \vee (p_2 \vee (p_3 \wedge p_4 \wedge \phi(\boldsymbol{x},\boldsymbol{r})))))$, with $p_1,\ldots,p_4,\boldsymbol{x},\boldsymbol{r}$ all chosen randomly. Then the circuit has the property:

- Choose and output some $\boldsymbol{x}$ and,
- If $z=0$ return 1 w.p. $\frac{1}{2}$.
- If $z=1$ return 1 w.p. $\frac{1}{2}+\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1]$

▷ Claim 27. $\phi \in$ All-Min-Sat $\iff$ $\ln(\frac{4}{3})$-differential privacy holds.

If $\phi \notin$ All-Min-Sat then for some $\boldsymbol{x}$ with $\Pr[\phi(\boldsymbol{x})=1] > \frac{1}{2}$, $\Pr[\phi(\boldsymbol{x})=0] < \frac{1}{2}$

$$\begin{aligned}
\frac{\Pr[\psi(0)=(\boldsymbol{x},0)]}{\Pr[\psi(1)=(\boldsymbol{x},0)]} &= \frac{\frac{1}{2}}{1-\frac{1}{2}-\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}-\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}(1-\Pr[\phi(\boldsymbol{x})=1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}(1-\Pr[\phi(\boldsymbol{x})=1])} \\
&= \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}(\Pr[\phi(\boldsymbol{x})=0])} \\
&> \frac{\frac{1}{2}}{\frac{1}{4}+\frac{1}{4}\frac{1}{2}} = \frac{4}{3} \approx 1.3
\end{aligned}$$

If $\phi \in$ All-Min-Sat then for all $\boldsymbol{x}$ we have $\Pr[\phi(\boldsymbol{x})=1] \leq \frac{1}{2}$, $\Pr[\phi(\boldsymbol{x})=0] \geq \frac{1}{2}$

$$\frac{\Pr[\psi(0)=(\boldsymbol{x},0)]}{\Pr[\psi(1)=(\boldsymbol{x},0)]} \leq \frac{4}{3} \approx 1.3$$

$$\begin{aligned}
\frac{\Pr[\psi(1)=(\boldsymbol{x},0)]}{\Pr[\psi(0)=(\boldsymbol{x},0)]} &= \frac{1-\frac{1}{2}-\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1]}{\frac{1}{2}} \\
&= \frac{\frac{1}{4}+\frac{1}{4}\Pr[\phi(\boldsymbol{x})=0]}{\frac{1}{2}} \leq 1
\end{aligned}$$

$$\frac{\Pr[\psi(1)=(\boldsymbol{x},1)]}{\Pr[\psi(0)=(\boldsymbol{x},1)]} = \frac{\frac{1}{2}+\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1]}{\frac{1}{2}} \leq \frac{\frac{1}{2}+\frac{1}{4}\frac{1}{2}}{\frac{1}{2}} = 1.25$$

$$\frac{\Pr[\psi(0)=(\boldsymbol{x},1)]}{\Pr[\psi(1)=(\boldsymbol{x},1)]} = \frac{\frac{1}{2}}{\frac{1}{2}+\frac{1}{4}\Pr[\phi(\boldsymbol{x})=1]} \leq 1$$

◀

Proof for small input small output. Given $\phi(\boldsymbol{x})$, we reduce $\phi \in$ Maj-Sat to Decide-ε-DP. Our resulting circuit ψ will have 1 output bit and 1 input bits

Let $\psi(z) = (p_1 \wedge z) \vee (\neg p_1 \wedge (z \oplus \phi(\boldsymbol{r})))$, where p_1 and $\boldsymbol{r}$ are chosen randomly. Then the circuit has the property:

- probability $\frac{1}{2}$ output z.
- probability $\frac{1}{2}$ output $z \oplus \phi(\boldsymbol{r})$. (Output z, flipped proportionally to the number of accepting allocations to ϕ.)

▷ Claim 28. $\phi \in$ Maj-Sat $\iff$ ψ is ln(3)-differentially private

The probabilities behave as follows, where each case is also bound by $\frac{1}{2}$ in the direction consistent with the probability shown.

Output ↓	Input →	1	0	Max-Ratio
0	Maj	$> \frac{1}{4}$	$< \frac{3}{4}$	> 3
	Min	$\leq \frac{1}{4}$	$\geq \frac{3}{4}$	≤ 3
1	Maj	$< \frac{3}{4}$	$> \frac{1}{4}$	> 3
	Min	$\geq \frac{3}{4}$	$\leq \frac{1}{4}$	≤ 3

Then in the either Maj case we have the ratio is greater than 3 (violating privacy) and for both Min case the ratio is bounded by 3 (satisfying privacy). ◀

C Direct Proof that Decide-ε, δ-DP is coNP$^{\#\mathbf{P}}$-hard

We prove that Decide-ε, δ-DP is **coNP**$^{\#\mathbf{P}}$-hard, even when there is just one output bit and for every ε.

▶ **Theorem 29.** Decide-ε, δ-DP *is* **coNP**$^{\#\mathbf{P}}$*-hard.*

We could show Decide-ε, δ-DP by reduction from All-Min-Sat, which would entail that $(\varepsilon, \frac{1}{2})$-differential privacy is **coNP**$^{\#\mathbf{P}}$-hard. To show hardness for a large range of δ we first generalise All-Min-Sat to All-Frac-f-Sat, showing this is also hard. We will then reduce All-Frac-f-Sat to All-Min-Sat.

C.1 Generalising All-Min-Sat

Let us first generalise All-Min-Sat to All-Frac-f-Sat, which rather than requiring that the minority (half) of allocations to y give true, rather no more than a fraction f. Similarly we generalise E-Maj-Sat to E-Frac-f-Sat.

▶ **Definition 30.** *A formula* $\phi(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{y} \in \{0,1\}^m$ *and* $f \in [0,1] \cap \mathbb{Q}$ *is* All-Frac-f-Sat *if for every* $\boldsymbol{x} \in \{0,1\}^n$

$$\frac{|\{\boldsymbol{y} \in \{0,1\}^m \mid \phi(\boldsymbol{x}, \boldsymbol{y}) \text{ is true}\}|}{2^m} \leq f$$

▶ Remark 31. For $f = 0$, we require that for all $\boldsymbol{x}$, no input of $\boldsymbol{y}$ gives true, therefore we require ϕ is unsatisfiable. For $f = 1$, we have essentially no restriction and for $f = \frac{2^m - 1}{2^m}$ we require that ϕ is not a tautology. All-Min-Sat is then when $f = \frac{1}{2}$.

▶ **Definition 32.** *A formula* ϕ *is* E-Frac-f-Sat *if it is not* All-Frac-f-Sat.

This means a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$, $\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{y} \in \{0,1\}^n$ and $f \in [0,1] \cap \mathbb{Q}$ is E-Frac-f-Satif there exists an allocation $\boldsymbol{x}$ that more than f fraction of allocations to $\boldsymbol{y}$ result in $\phi(\boldsymbol{x}, \boldsymbol{y})$ being true. That is there exists $\boldsymbol{x} \in \{0,1\}^n$ such that $\frac{|\{\boldsymbol{y} \in \{0,1\}^m \mid \phi(\boldsymbol{x}, \boldsymbol{y}) \text{ is true}\}|}{2^m} > f$.

Towards showing All-Frac-f-Sat is **coNP**$^{\#\mathbf{P}}$-hard, we show E-Frac-f-Satis **NP**$^{\#\mathbf{P}}$-hard, entailing Corollary 34.

▶ **Lemma 33.** E-Frac-f-Sat*is* **NP**$^{\#\mathbf{P}}$*-hard for* $f \in [\frac{1}{2^m}, \frac{2^m - 1}{2^m})$.

▶ **Corollary 34.** ALL-FRAC-f-SAT *is* $\mathbf{coNP}^{\#\mathbf{P}}$*-hard for* $f \in [\frac{1}{2^m}, \frac{2^m-1}{2^m})$.

Proof of Lemma 33 for f of the form $\frac{1}{2^{k+1}}$. We reduce E-MAJ-SAT, given a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$ to E-FRAC-$\frac{1}{2^{k+1}}$-SAT.

(We assume $\frac{1}{2^{k+1}}$ takes $O(k)$ bits.)

Define a formula $\phi'(\boldsymbol{x}, \boldsymbol{w})$, with $\boldsymbol{w} \in \{0,1\}^{m+k}$. Let $\boldsymbol{w} = (y_1, \ldots, y_m, z_1 \ldots z_k)$, where $\boldsymbol{y} = (y_1, \ldots, y_m)$.

$\phi'(\boldsymbol{x}, \boldsymbol{w}) = z_1 \wedge \cdots \wedge z_k \wedge \phi(\boldsymbol{x}, y_1, \ldots, y_m)$

For $\boldsymbol{x}$ fixed if g allocations to $y_1, \ldots, y_m$ satisfy ϕ then each of these satisfy ϕ' only when $z_1 = \cdots = z_k = 1$. All remaining times are unsatisfied.

Suppose $\frac{g}{2^m}$ of $\boldsymbol{y}$'s satisfy $\phi(\boldsymbol{x}, \boldsymbol{y})$ then $\frac{g}{2^m \cdot 2^k}$ $\boldsymbol{w}$'s satisfy $\phi'(\boldsymbol{x}, \boldsymbol{w})$.

That is we have:

$$\frac{g}{2^m \cdot 2^k} > \frac{1}{2^{k+1}} \iff \frac{g}{2^m} > \frac{1}{2}.$$

◀

Proof of Lemma 33 for f of the form $\frac{a}{2^{k+1}}$. We reduce E-MAJ-SAT, given a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$ to E-FRAC-$\frac{1}{2^{k+1}}$-SAT. Assume a odd (otherwise, half and take $\frac{a/2}{2^k}$) and greater than 1 (otherwise use above).

Define a formula $\phi'(\boldsymbol{x}, \boldsymbol{w})$, with $\boldsymbol{w} \in \{0,1\}^{m+k}$. Let $\boldsymbol{w} = (y_1, \ldots, y_m, z_1 \ldots z_k)$, where $\boldsymbol{y} = (y_1, \ldots, y_m)$. Let $b = \frac{a-1}{2}$ (b is always between 1 and $2^k - 1$).

Let $\chi_{\frac{b}{2^k}}(z_1, \ldots, z_k))$ be a circuit on k bits, which is true for $\frac{b}{2^k}$ of its inputs of $z_1 \ldots z_k$, but not true for $z_1 = \cdots = z_k = 1$ when $b < 2^k$. [4]

Then we let $\phi'(\boldsymbol{x}, \boldsymbol{w}) = (z_1 \wedge \cdots \wedge z_k \wedge \phi(\boldsymbol{x}, y_1, \ldots, y_m)) \vee \chi_{\frac{b}{2^k}}(z_1, \ldots, z_k)$.

That is the formula ϕ' is true whenever $z_1 = \cdots = z_k = 1$ and ϕ is true, or on the $\frac{b}{2^k}$ choices of $z_1 \ldots z_k$.

Suppose $\frac{g}{2^m}$ of $\boldsymbol{y}$'s satisfy $\phi(\boldsymbol{x}, \boldsymbol{y})$ then $\frac{g}{2^m \cdot 2^k} + \frac{b}{2^k} = \frac{g}{2^m \cdot 2^k} + \frac{a-1}{2^{k+1}}$ $\boldsymbol{w}$'s satisfy $\phi'(\boldsymbol{x}, \boldsymbol{w})$.

That is we have:

$$\frac{g}{2^m \cdot 2^k} + \frac{a-1}{2^{k+1}} > \frac{a}{2^{k+1}} \iff \frac{g}{2^m \cdot 2^k} > \frac{1}{2^{k+1}} \iff \frac{g}{2^m} > \frac{1}{2}$$

◀

Proof of Lemma 33 for f of the form $\frac{a}{b}$. Let m be the number such that $\boldsymbol{y} \in \{0,1\}^m$, the number of $\boldsymbol{y}$ bits of the formula, or the number of 'MAJ' bits. Let z be such that $\frac{z}{2^m} < \frac{a}{b} < \frac{z+1}{2^m}$. We reduce E-FRAC-$\frac{z}{2^m}$-SAT to E-FRAC-$\frac{a}{b}$-SAT, by simply taking ϕ unchanged.

Suppose $\frac{g}{2^m}$ of $\boldsymbol{y}$'s satisfy $\phi(\boldsymbol{x}, \boldsymbol{y})$ then

$$\frac{g}{2^m} > \frac{z}{2^m} \iff \frac{g}{2^m} \geq \frac{z+1}{2^m} \iff \frac{g}{2^m} > \frac{a}{b}.$$

◀

C.2 Main Proof of Theorem 29

Proof. Assume we are given an instance of ALL-FRAC-f-SAT, a formula $\phi(\boldsymbol{x}, \boldsymbol{y})$ for $\boldsymbol{x} \in \{0,1\}^n, \boldsymbol{y} \in \{0,1\}^m$ and $f \in [0,1]$.

We define a circuit ψ, with inputs $\boldsymbol{x} \in \{0,1\}^{n+1}$, we write as $(z, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$; matching the inputs $\boldsymbol{x}$ and an additional bit z. There are m probabilistic bits $\boldsymbol{r} \in \{0,1\}^m$, matching $\boldsymbol{y}$.

[4] Given b, k such that $0 < \frac{b}{2^k} < 1$, we create a formula, over $2k$ bits, which given two k-bit integers m, n, return whether $m \leq n$ (such a formula is of size polynomial in k). By fixing n to b, we have a circuit on m input bits which decides if $m \leq b$. Instead sample over the m bits of m producing a circuit $\chi_{\frac{b}{2^k}}$ which is true on $\frac{b}{2^k}$ of its inputs.

There is one output bit $\boldsymbol{o} \in \{0,1\}^1$. The circuit ψ will behave like ϕ when $z = 1$ and simply output 0 when $z = 0$; i.e. $\boldsymbol{o}_1 = z \wedge \phi(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n, \boldsymbol{r}_1, \ldots, \boldsymbol{r}_m)$.

▷ Claim 35. $\phi \in$ All-Frac-f-Sat if and only if ψ is (ε, δ)-differentially private, for $\delta = f$ and any choice of ε (including zero).

Direction: if $\phi \notin$ All-Frac-f-Sat then not (ε, δ)-differentially private.

Given $\phi \notin$ All-Min-Sat then there exists $\boldsymbol{x} \in \{0,1\}^n$ such that $\phi(\boldsymbol{x}, \boldsymbol{y})$ is on more than f portion of $\boldsymbol{y} \in \{0,1\}^m$. We show the differential privacy condition is violated exactly using this $\boldsymbol{x}$, let $\boldsymbol{x} = (1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$ and $\boldsymbol{x'} = (0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$. Let us consider the probability of the event $o_1 = 1$.

Then we have $\Pr[\psi(1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = 1] > f$ and $\Pr[\psi(0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = 1] = 0$. Violating differential privacy since,

$$\Pr[\psi(1, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = 1] - e^{\varepsilon} \Pr[\psi(0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = 1] > f - 0 = \delta.$$

Direction: if $\phi \in$All-Frac-f-Sat then (ε, δ)-differentially private.

Since $\phi \in$ All-Frac-f-Sat then for all $\boldsymbol{x} \in \{0,1\}^n$ we have $\phi(\boldsymbol{x}, \boldsymbol{y})$ true for less or equal f proportion of the allocations to $\boldsymbol{y} \in \{0,1\}^m$. Equivalently the more than f of $\boldsymbol{y} \in \{0,1\}^m$ with $\phi(\boldsymbol{x}, \boldsymbol{y})$ false.

To show privacy we consider all adajcent inputs and all output event. The output events are $E \subseteq \{0,1\}^1$, giving $\{\}, \{0\}, \{1\}, \{0,1\}$. The probability of 'no output' $\{\}$ is zero for all inputs, so cannot violate differential privacy. The probability of 'output anything' $\{0,1\}$ is one for all inputs, so does cannot violate differential privacy. Thus we argue that events $\{0\}$ and $\{1\}$ do not violate differential privacy, for all adjacent inputs.

Inputs take the form $\boldsymbol{x} = (z, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$, and $\boldsymbol{x}, \boldsymbol{x'}$ can be adjacent either with fixed $\boldsymbol{x}$ and differing z or, fixed z and $\boldsymbol{x}$ differing in one position; in each case we show $\Pr[\psi(\boldsymbol{x}) = E] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x'}) = E] \leq \delta$ and $\Pr[\psi(\boldsymbol{x'}) = E] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x}) = E] \leq \delta$

Let z be fixed to zero.

Hence we have $\boldsymbol{x}, \boldsymbol{x'}$ with $\boldsymbol{x}$'s differing in one position. For $z = 0$ the circuit outputs zero in all cases, independently of $\boldsymbol{x}$, thus does not violate differential privacy since $\Pr[\psi(\boldsymbol{x}) = E] = \Pr[\psi(\boldsymbol{x'}) = E]$.

Let z be fixed to one.

Hence we have $\boldsymbol{x}, \boldsymbol{x'}$ with $\boldsymbol{x}$'s differing in one position. Without loss of generality suppose the difference is x_j.

For the event $E = \{1\}$ we have the probability being $\leq f$ for each input; that is regardless of $\boldsymbol{x}$ we have $\Pr[\psi(\boldsymbol{x}) = 1] \leq f$. So $\Pr[\psi(\boldsymbol{x}) = E] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x'}) = E] \leq \Pr[\psi(\boldsymbol{x}) = E] \leq f = \delta$ and $\Pr[\psi(\boldsymbol{x'}) = E] - e^{\varepsilon} pr[\psi(\boldsymbol{x}) = E] \leq \Pr[\psi(\boldsymbol{x'}) = E] \leq f = \delta$.

For the event $E = \{0\}$ we have the probability being $\geq 1 - f$ for each input; that is regardless of $\boldsymbol{x}$ we have $\Pr[\psi(\boldsymbol{x}) = 0] \geq 1 - f$. So $\Pr[\psi(\boldsymbol{x}) = E] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x'}) = E] \leq 1 - \Pr[\psi(\boldsymbol{x'}) = E] \leq f = \delta$ and $\Pr[\psi(\boldsymbol{x'}) = E] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x}) = E] \leq 1 - \Pr[\psi(\boldsymbol{x}) = E] \leq f = \delta$.

Let $x_1, \ldots, x_n$ be fixed.

We have $x_1, \ldots, x_n$ fixed and the case $z = 0$ and $z = 1$, hence we have $\boldsymbol{x} = (1, x_1, \ldots, x_n)$ and $\boldsymbol{x'} = (0, x_1, \ldots, x_n)$.

For the event $\{1\}$, when $z = 1$, we have $\Pr[\psi(\boldsymbol{x}) = 1] \leq f$, but for $z = 0$ the circuit is always 0, thus $\Pr[\psi(\boldsymbol{x'}) = 1] = 0$.

Then $\Pr[\psi(\boldsymbol{x}) = 1] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x'}) = 1] = \Pr[\psi(\boldsymbol{x}) = 1] \leq f = \delta$ and $\Pr[\psi(\boldsymbol{x'}) = 1] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x}) = 1] \leq 0 \leq \delta$.

For the event $\{0\}$ we have then $\Pr[\psi(\boldsymbol{x}) = 0] \geq 1 - f$ and $\Pr[\psi(\boldsymbol{x'}) = 0] = 1$ Then $\Pr[\psi(\boldsymbol{x}) = 0] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x'}) = 0] \leq 1 - e^{\varepsilon} \leq 0 \leq \delta$ and $\Pr[\psi(\boldsymbol{x'}) = 0] - e^{\varepsilon} \Pr[\psi(\boldsymbol{x}) = 0] \leq 1 - \Pr[\psi(\boldsymbol{x}) = 0] \leq f = \delta$. ◀

D Conditioning

Conditioning allows the run of a program to fail, so that the probability associated with failing runs is renormalised over all other runs (see e.g. [36]). We show our decision procedures are robust to this notion, for which we simulate failure with an additional output bit and incorporate the renationalisation into our decision procedures. Naturally our lower bounds apply to this more general notion.

Conditioning can be encoded in a circuit by assuming a bit which indicates whether the run has succeeded; in the case of failure we can assume all other output bits are false (denoted $\mathbf{0}$). Thus for 'proper' events in $\{0,1\}^{\ell}$, the circuit formally outputs from $\{0,1\}^{\ell+1}$. We redefine our probability of an event as

$$\Pr[\psi(\boldsymbol{x}) \in E] = \frac{|\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o}) \text{ with } \boldsymbol{o} \in E\}|}{|\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) \neq (\bot, \mathbf{0})\}|}.$$

Note that $|\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) \neq (\bot, \mathbf{0})\}|$ is independent of the choice of E.

D.1 Decide-ε-DP

We generalise the procedure for Decide-ε-DP, maintaining $\mathbf{coNP}^{\#\mathbf{P}}$.

1. Guess $\boldsymbol{x}, \boldsymbol{x'} \in \{0,1\}^n, \boldsymbol{o} \in \{0,1\}^l$
   a. Compute $a = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o})\}|$
   b. Compute $b = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x'}, \boldsymbol{r}) = (\top, \boldsymbol{o})\}|$
   c. Compute $D_1 = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) \neq (\bot, \mathbf{0})\}|$
   d. Compute $D_2 = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x'}, \boldsymbol{r}) \neq (\bot, \mathbf{0})\}|$
   e. Reject if D_1 or D_2 is zero.
   f. Accept if $a \cdot D_2 \leq \exp(\varepsilon) \cdot b \cdot D_2$ and otherwise reject.

▷ Claim. ψ is ε-differentially private $\iff$ Decide-ε-DP accepts on all branches

D.2 Decide-ε, δ-DP

We generalise the procedure for Decide-ε, δ-DP, maintaining $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$. Recall $e^{\varepsilon} = \alpha = \frac{u}{v}$. In more detail, on inputs ψ, $\boldsymbol{x}$ and $\boldsymbol{x'}$, $\mathcal{M}$ does the following:

1. Choose $\boldsymbol{o} \in \{0,1\}^{\ell}$ and an integer $C \in \{1, 2, \ldots, 2^{2m+\lceil \log(v) \rceil}\}$ (this requires choosing $l + 2m + \lceil \log(v) \rceil$ bits).
2. Through a call to the $\#\mathbf{P}$ oracle, $\mathcal{M}$ computes
   - $a = |\{\boldsymbol{r} \in \{0,1\}^m : \psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o})\}|$
   - $b = |\{\boldsymbol{r} \in \{0,1\}^m : \psi(\boldsymbol{x'}, \boldsymbol{r}) = (\top, \boldsymbol{o})\}|$.

- $D_1 = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) \neq (\bot, \mathbf{0})\}|$
- $D_2 = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}', \boldsymbol{r}) \neq (\bot, \mathbf{0})\}|$

3. $\mathcal{M}$ accepts if $v \cdot a \cdot D_2 - u \cdot b \cdot D_1 \geq C$ and otherwise rejects.

▶ **Lemma 36.** *Given two inputs $\boldsymbol{x}, \boldsymbol{x}' \in X^n$, $\mathcal{M}(\psi, \boldsymbol{x}, \boldsymbol{x}')$ has exactly $v \cdot D_1 \cdot D_2 \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \delta_{\boldsymbol{x}, \boldsymbol{x}'}(\boldsymbol{o})$ accepting executions.*

Proof. Let $\mathbb{1}\{X\}$ be the indicator function, which is one if the predicate X holds and zero otherwise.

$$vD_1D_2 \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \delta_{\boldsymbol{x}, \boldsymbol{x}'}(\boldsymbol{o}) = \sum_{\boldsymbol{o} \in \{0,1\}^\ell} vD_1D_2 \max\left(\Pr[\psi(\boldsymbol{x}) = \boldsymbol{o}] - \alpha \Pr[\psi(\boldsymbol{x}') = \boldsymbol{o}], 0\right)$$

$$= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} vD_1D_2 \max\left(\frac{1}{D_1} \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o})\} - \alpha \frac{1}{D_2} \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = (\top, \boldsymbol{o})\}, 0\right)$$

$$= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \max\left(vD_2 \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o})\} - v\alpha D_1 \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = (\top, \boldsymbol{o})\}, 0\right)$$

$$= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \max\left(vD_2 \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o})\} - uD_1 \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = (\top, \boldsymbol{o})\}, 0\right)$$

$$= \sum_{\boldsymbol{o} \in \{0,1\}^\ell} \sum_{C=1}^{2^{2m + \lceil \log(v) \rceil}} \mathbb{1}\left\{\max\left(vD_2 \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}, \boldsymbol{r}) = (\top, \boldsymbol{o})\} - uD_1 \sum_{\boldsymbol{r} \in \{0,1\}^m} \mathbb{1}\{\psi(\boldsymbol{x}', \boldsymbol{r}) = (\top, \boldsymbol{o})\}, 0\right) \geq C\right\}$$

$$= \text{number of accepting executions in } \widehat{\mathcal{M}}$$

◀

We can now describe our $\mathbf{coNP}^{\#\mathbf{P}^{\#\mathbf{P}}}$ procedure for DECIDE-ε, δ-DP. The procedure takes as input a probabilistic circuit ψ.

1. Non-deterministically choose neighboring $\boldsymbol{x}$ and $\boldsymbol{x}' \in \{0,1\}^n$ (i.e., $2n$ bits)
2. Let $\mathcal{M}$ be the non-deterministic Turing Machine with access to a $\#\mathbf{P}$-oracle as described above. Create a machine $\widehat{\mathcal{M}}$ with no input that executes $\mathcal{M}$ on $\psi, \boldsymbol{x}, \boldsymbol{x}'$
3. Make an $\#\mathbf{P}^{\#\mathbf{P}}$ oracle call for the number of accepting executions $\widehat{\mathcal{M}}$ has.
4. Make an $\#\mathbf{P}$ oracle call for
   - $D_1 = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}, \boldsymbol{r}) = (\bot, \mathbf{0})\}|$
   - $D_2 = |\{\boldsymbol{r} \in \{0,1\}^m \mid \psi(\boldsymbol{x}', \boldsymbol{r}) = (\bot, \mathbf{0})\}|$
5. Reject if D_1 or D_2 is zero.
6. Reject if the number of accepting executions of $\widehat{\mathcal{M}}$ is greater than $v \cdot D_1 \cdot D_2 \cdot \delta$ and otherwise accept.

▷ Claim. ψ is (ε, δ)-differentially private $\iff$ DECIDE-ε, δ-DP accepts on all branches