

1 Introduction

Differential privacy is a framework for privacy that gives rigorous guarantees on the amount of data leakage any one person’s data can be subjected to when releasing statistical data. Since being introduced in 2006 [7], differential privacy has become the gold standard for private statistical analysis. Differentially private algorithms, whose efficacy are characterized by a “privacy cost” ϵ , primarily rely on the addition of statistical noise, ensuring that statistical results remain approximately correct while preventing any one person’s information from being revealed.

Differentially private algorithms are notoriously tricky to analyze for correctness; most famously, the Sparse Vector Technique (SVT) algorithm has gone through multiple iterations, some of which were later shown to completely fail at protecting privacy[10]. Previous implementations of differential privacy by Apple have similarly been shown to have an increase from the claimed privacy cost by a factor of up to 16 [11].

Thus, much work has been done on developing methods for automatic verification of differentially private algorithms, both in their overall privacy and in the specific privacy costs they claim to achieve. Because even for limited programs the problem of determining if a program is differentially private is undecidable[2], previous work tends to focus on semi-decidability or further restricting program models.

Recently, a line of work has emerged around **approximate liftings** [3, 5, 4, 9]. Approximate liftings are a generalization of probabilistic couplings, themselves a well-known technique in probability theory for analyzing relationships between random variables. Approximate liftings allow for a more structured proof approach to many algorithms that themselves are not conducive to a standard compositional analysis, such as SVT. Because of their structure, liftings also lend themselves to automated proof construction [1].

We first rewrite the major results of approximate liftings in $\{\text{not program logic}\}^1$. We then use approximate liftings to demonstrate that a certain limited class of programs, first described in [6], are differentially private; interestingly, we show that our class of liftings completely characterizes this class of programs. Additionally, we demonstrate that the privacy of a natural generalization of this class of programs can be proven using liftings and almost immediately follows from the privacy of the smaller class.

2 Differential Privacy

Differential privacy is a mathematically robust approach to privacy; most generally, differential privacy ensures that it is unlikely for an adversary to distinguish between whether or not one person’s data was used in a private algorithm. To do this, differentially private algorithms rely on randomization, especially through the addition of statistical noise.

More precisely then, for a fixed output σ of a private algorithm A , the probability of obtaining σ for a dataset with some individual Alex is close (measured by a multiplicative factor) to the

¹not sure how to describe this, also not sure if worth mentioning

probability of obtaining σ for the same dataset with Alex removed or Alex’s data changed.

We will consider **datasets** $\mathcal{X} \in X^n$ of size n , where X is the set of all possible rows in the dataset; each person is represented by a single row.

We next define what it means for datasets to be “similar” to each other.

Definition 2.1. Two datasets $\mathcal{X} = (x_1, \dots, x_n), \mathcal{X}' = (x'_1, \dots, x'_n) \in X^n$ are **adjacent** (notated $\mathcal{X} \sim \mathcal{X}'$) if $|\{i : x_i \neq x'_i\}| \leq 1$ ².

We thus formalize privacy under this framework as follows.

Definition 2.2 (Pure Differential Privacy). A randomized algorithm A is ε -differentially private if, for all pairs of **adjacent** datasets X and X' and all events $E \subseteq \text{im}(A)$,

$$\mathbb{P}[A(X) \in E] \leq e^\varepsilon \mathbb{P}[A(X') \in E]$$

An extremely useful property of differential privacy is that differentially private programs can be **sequentially composed** with a linear degradation in privacy:

Theorem 2.3 (Standard Composition). *If A is ε_1 -differentially private and, for all σ , $B(\sigma, \cdot)$ is ε_2 -differentially private, then $B(A(X), X)$ is $\varepsilon_1 + \varepsilon_2$ -differentially private.*

Composition therefore allows us to view privacy parameters ε as a “budget” for privacy-leaking operations in a program. Many³ common differentially private algorithms are thus built out of well-known private components combined together, which also lend themselves to straightforward analyses.

2.1 Sensitivity and the Laplace Mechanism

Because we are typically interested in analyzing *functions* of our raw dataset (for example, the average age of a town), it is often useful to examine differential privacy through a similar model - instead of comparing two adjacent datasets $X \sim X'$, we compare **queries** $f(X)$ and $f(X')$. In this world, we care about the *sensitivity* of functions: how much a function *changes* when considering adjacent inputs.

Definition 2.4. The (ℓ_1) -sensitivity of a function $f : X \rightarrow \mathbb{R}$, often denoted Δf , is defined as $\Delta f = \max_{X \sim X'} \|f(X) - f(X')\|_1$.

Given a function’s sensitivity, we can easily make it differentially private through the use of the **Laplace Mechanism**.

Definition 2.5. The Laplace distribution $\text{Lap}(\mu, b)$ with mean μ and spread parameter b is the probability distribution with probability density function $f(x) = \frac{1}{2b} \exp(-\frac{|x-\mu|}{b})$. If $\mu = 0$, we will often abbreviate $\text{Lap}(0, b)$ as $\text{Lap}(b)$.

²A common variant is to define adjacency by the removal or addition of an entry, rather than changing one

³generic platitude - reword

The Laplace Mechanism, as expected, simply adds noise sampled from the Laplace distribution to a query result.

Theorem 2.6 (Theorem 3.6 [8]). *For a function f with sensitivity Δ , $A(X) = f(X) + \text{Lap}(\frac{\Delta}{\epsilon})$ is ϵ -differentially private.*

We will consider the scenario where we are given a potentially infinite *sequence* of real-valued query functions q_0, q_1, \dots , each with sensitivity at most Δ .

2.2 Deciding Privacy

Because designing differentially private algorithms can be quite tricky, we would like to be able to automatically (i.e. algorithmically) verify whether or not a given program is private, especially for algorithms whose privacy proofs do not rely primarily on composition. Ideally, beyond just determining whether a program is private or not, if a program is private, we'd like to find a good bound on the privacy cost for the program as well.

Unfortunately, even for relatively simple programs, just the basic problem is undecidable.

Theorem 2.7 ([2]). *The problem of determining whether a program from a certain class of algorithms with assignments, conditionals, and while loops is ϵ -differentially private is undecidable⁴.*

Thus, we will derive a decision procedure for a very specific class of potentially private programs; in particular, this class of programs lends itself to a straightforward analysis by **approximate liftings**, which we now introduce.

3 Couplings and Liftings

Probabilistic couplings are a common tool in probability theory; intuitively, couplings allow for the joint analysis of nominally unrelated probabilistic processes.

Definition 3.1. A coupling between two distributions A and B is a joint distribution C such that $\pi_1(C) = A$ and $\pi_2(C) = B$, where $\pi_1(C)$ and $\pi_2(C)$ are the first and second marginals of C , respectively.

In particular, couplings can be useful when analyzing the relation between two probabilistic processes; couplings were first formulated by [check name] to analyze the behaviour of markov chains and have close connections to concepts such as total variation distance and stochastic domination.

As useful as standard couplings are, however, we must use more powerful machinery to properly reason about privacy.

Approximate liftings [4, 5, 9, 3] allow us to apply couplings to the realm of differential privacy.

⁴rephrase?

Definition 3.2. Let A_1, A_2 be two probability spaces⁵. We say a distribution μ_1 on A_1 and μ_2 on A_2 are related by the ε -**lifting** of the relation $\Psi \subseteq A_1 \times A_2$ (written $\mu_1 \Psi^{\# \varepsilon} \mu_2$) if there exist two **witness distributions** μ_L, μ_R on $A_1 \times A_2$ such that

1. $\pi_1(\mu_L) = \mu_1$ and $\pi_2(\mu_R) = \mu_2$
2. $\text{supp}(\mu_L), \text{supp}(\mu_R) \subseteq \Psi$
3. $\sup_{E \subseteq A_1 \times A_2} (\mathbb{P}_{x \leftarrow \mu_L}[x \in E] - e^\varepsilon \mathbb{P}_{x \leftarrow \mu_R}[x \in E]) \leq 0$

The similarities between the third condition and the definition of differential privacy should be clear. Indeed, there is a close connection between approximate liftings and differential privacy:

Theorem 3.3. *An algorithm $A(X)$ is ε -differentially private if and only if, for all adjacent input sequences $X \sim X'$, $A(X)(=)^{\# \varepsilon} A(X')$.*

If we are solely aiming to show that a program is private, we can instead work with the following relaxation:

Theorem 3.4. *If for all adjacent input sequences $X \sim X'$ and outputs σ of A , $A(X)\{(a, b) : a = \sigma \implies b = \sigma\}^{\# \varepsilon} A(X')$, then $A(X)$ is ε -differentially private.*

As expected, the foundational results of differential privacy can be restated in terms of liftings:

Proposition 3.5 (Laplace Mechanism for Liftings). *If $X_1 \sim \text{Lap}(\mu_1, \frac{1}{\varepsilon})$ and $X_2 \sim \text{Lap}(\mu_2, \frac{1}{\varepsilon})$, then $X_1(=)^{\# \varepsilon |\mu_1 - \mu_2|} X_2$.*

Theorem 3.6 (Composition of Liftings). *Let A_1, B_2, A_2, B_2 be distributions over S_1, T_1, S_2, T_2 , respectively and let $R_1 \subseteq S_1 \times T_1$, $R_2 \subseteq S_2 \times T_2$ be relations. If $A_1 R_1^{\# \varepsilon_1} B_1$ and $A_1 R_1 B_1 \implies A_2 R_2^{\# \varepsilon_2} B_2$, then $A_2 R_2^{\# \varepsilon_1 + \varepsilon_2} B_2$.*

The structure of theorems 3.4 and 3.6 suggests the format that coupling proofs of privacy take: given two “runs” of an algorithm on adjacent inputs, construct many smaller liftings between program variables in each run and compose these liftings together to show that a final implicative lifting between the outputs of the two runs exists.

3.1 Proving SVT with couplings

A classic algorithm that requires analysis beyond standard composition is Sparse Vector Technique (SVT). Given a possibly infinite stream of inputs and a threshold value, SVT will output if the queries are above or below the threshold (with noise on both the query and the threshold).

Unusually for differentially private algorithms, SVT can output a potentially unbounded number of “below threshold” queries before the first c “above threshold”s (or vice-versa), where c is some constant set by the user; when $c = 1$, SVT is frequently also referred to as “Above (or Below) Threshold”. Potential applications include, for example, checking that

⁵may need to formally rewrite this at some point

a series of inputs is within an expected range or, appropriately, privately determining the non-zero elements of a sparse vector.

Because SVT allows for a potentially unbounded number of “below threshold” query outputs, its analysis requires a non-standard approach; a naive composition approach that assigns a fixed cost to outputting the result of each query will immediately result in unbounded privacy cost as well. Indeed, the analysis of SVT is notoriously difficult, with multiple published attempts at privacy proofs that were later shown to be incorrect⁶.

However, re-analyzing SVT using approximate liftings can be relatively simple.

Algorithm 1 Sparse Vector Technique

Input: $\mathcal{X} \in X^n$, $T \in \mathbb{R}$, $Q = q_1, \dots \in (X^n \rightarrow \mathbb{R})^*$ with sensitivity Δ , $c \in \mathbb{N}$.

```

1:  $\varepsilon_1, \varepsilon_2 \leftarrow \frac{\varepsilon}{2}, \rho \leftarrow \text{Lap}(\frac{\Delta}{\varepsilon_1}), \text{count} \leftarrow 0$ 
2: for  $q_i \in Q$  do
3:    $z \leftarrow \text{Lap}(\frac{2c\Delta}{\varepsilon_2})$ 
4:   if  $q_i(\mathcal{X}) + z \geq T + \rho$  then
5:     output  $\top$ 
6:      $\text{count} \leftarrow \text{count} + 1$ 
7:     if  $\text{count} \geq c$  then
8:       break
9:     end if
10:  else
11:    output  $\perp$ 
12:  end if
13: end for
```

Theorem 3.7. *Sparse Vector Technique is ε -differentially private.*

Proof. Consider two runs of SVT with adjacent inputs $\mathcal{X} \sim \mathcal{X}'$, respectively. We are aiming to show that $\text{SVT}(\mathcal{X}, T, Q, c) \{ (a, b) : a = \sigma \implies b = \sigma \}^{\# \varepsilon} \text{SVT}(\mathcal{X}', T, Q, c)$ is a valid lifting.

Fix some output $\sigma \in \{\perp, \top\}^n$. Let $A = \{i : \sigma_i = \top\}$ be the indices of queries that are measured to be above the threshold. Note that $|A| = c$.

For every program variable x , let $x\langle 1 \rangle$ and $x\langle 2 \rangle$ represent the value of x in $\text{SVT}(\mathcal{X}, T, Q, c)$ and $\text{SVT}(\mathcal{X}', T, Q, c)$, respectively, so, for example, $q_i(\mathcal{X})\langle 1 \rangle = q_i(\mathcal{X})$ and $q_i(\mathcal{X})\langle 2 \rangle = q_i(\mathcal{X}')$.

Let $\tilde{T} = T + \rho$. Then $\tilde{T} \sim \text{Lap}(T, \frac{\Delta}{\varepsilon_1})$, so $\tilde{T}\langle 1 \rangle + \Delta(=) \#^{\varepsilon_1} \tilde{T}\langle 2 \rangle$.

Let $S_i = q_i(\mathcal{X}) + z_i$, so $S_i \sim \text{Lap}(q_i(\mathcal{X}), \frac{2c\Delta}{\varepsilon_2})$.

For all i such that $0 \leq i < n$, $i \notin A$, we construct the lifting $z_i\langle 1 \rangle(=) \#^0 z_i\langle 2 \rangle$.

Then note that $\tilde{T}\langle 1 \rangle + \Delta = \tilde{T}\langle 2 \rangle \wedge z_i\langle 1 \rangle = z_i\langle 2 \rangle \implies (S_i\langle 1 \rangle < \tilde{T}\langle 1 \rangle \implies S_i\langle 2 \rangle < \tilde{T}\langle 2 \rangle)$.

⁶A textbook analysis of SVT, along with a discussion of bugged versions and incorrect privacy proofs, can be found at [10]

For all $i \in A$, create the lifting $z_i\langle 1 \rangle (=)^{\# \frac{\varepsilon_2}{c}} z_i\langle 2 \rangle - q_i(\mathcal{X}) + q_i(\mathcal{X}') - \Delta$, or equivalently, $S_i\langle 1 \rangle + \Delta (=)^{\# \frac{\varepsilon_2}{c}} S_i\langle 2 \rangle$. Note that this costs $\frac{\varepsilon_2}{c}$ since $|q_i(\mathcal{X}) - q_i(\mathcal{X}')| \leq \Delta$.

Then

$$\tilde{T}\langle 1 \rangle + \Delta = \tilde{T}\langle 2 \rangle \wedge S_i\langle 1 \rangle + \Delta = S_i\langle 2 \rangle \implies (S_i\langle 1 \rangle \geq \tilde{T}\langle 1 \rangle \implies S_i\langle 2 \rangle \geq \tilde{T}\langle 2 \rangle)$$

Thus, for all i , $SVT(\mathcal{X}, T, Q, c)_i = \sigma_i \implies SVT(\mathcal{X}', T, Q, c)_i = \sigma_i$, so $SVT(\mathcal{X}, T, Q, c)\{(a, b) : a = \sigma \implies b = \sigma\}^{\# \varepsilon_1 + \varepsilon_2} SVT(\mathcal{X}', T, Q, c)$.

By Theorem 3.4, SVT is ε -differentially private. \square

4 Automatically Proving Privacy using Couplings

We begin by building up a program model for SVT-style algorithms. There are three major components of SVT: taking in a threshold value and adding Laplace noise to it, taking in input and adding Laplace noise to it, and comparing the noisy threshold to the noisy input.

TODO: add non-input transitions (still don't understand why they're useful) - very trivial since the guards of a non-input transition have to be **true**

4.1 Individual Transitions

Definition 4.1. Let Q be a finite set of program states. Then $P_Q(q) : Q \rightarrow \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$ is a function that associates each state state with two **noise parameters** $P(q) = (d_q, d'_q)$.

In our program model, we have a singular persistent real-valued variable \mathbf{x} , which can be thought of as analogous to the threshold in algorithms like SVT.

To capture the value of \mathbf{x} at different states, we pair each state with possible values of \mathbf{x} :

Definition 4.2. Given a finite set of program states Q , the set $Q \times \mathbb{R}$ is the set of **instantiated states** of Q .

Let $\mathcal{C} = \{\mathbf{true}, \mathbf{insample} < \mathbf{x}, \mathbf{insample} \geq \mathbf{x}\}$ be a set of **transition guards**.

We now can define the simplest programs, individual transitions:

Definition 4.3 (Transitions). Given a finite set of program states Q and a set of real-valued variables $\{\mathbf{x}, \mathbf{insample}, \mathbf{insample}'\}$. A transition is a tuple $t = (q, q', c, \sigma, \tau)$, where

- $q \in Q$ is the initial state.
- $q' \in Q$ is the following state.
- $c \in \mathcal{C}$ is a transition guard that determines if t is taken.
- $\sigma \in \Gamma \cup \{\mathbf{insample}, \mathbf{insample}'\}$ is the output of t .
- $\tau \in \{\mathbf{true}, \mathbf{false}\}$ is a boolean value indicating whether or not the stored value of \mathbf{x} will be updated.

Depending on context, we may also notate t as $q \rightarrow q'$.

Semantically, each transition $t = (q, q', c, \sigma, \tau)$ will first read in a real number input in , sample two random variables $z \sim \text{Lap}(0, \frac{1}{d\varepsilon})$, $z' \sim \text{Lap}(0, \frac{1}{d'\varepsilon})$, where $P(q) = (d, d')$, and then assign $\text{insample} = \text{in} + z$ and $\text{insample}' = \text{in} + z'$. If the guard c is satisfied, then we transition to state q' , outputting σ and, if $\tau = \text{true}$, reassigning $\mathbf{x} = \text{insample}$.

These transitions are primarily **structural** elements, describing in general how a program can move from state q to q' . However, we will also want to look at **individual executions** of a transition, with concrete values for \mathbf{x} .

Definition 4.4 (Instantiated Transitions). An instantiated transition is a tuple $\hat{t} = ((q, x), (q', x'), c, \sigma, \tau)$, where

- $(q, x) \in Q \times \mathbb{R}$ is the initial state paired with the initial value of \mathbf{x} at q .
- $(q', x') \in Q \times \mathbb{R}$ is the following state paired with a (possibly) new value of \mathbf{x} .
- $c \in \mathcal{C}$ is a transition guard that determines if t is taken.
- $\sigma \in \Gamma \cup \{\text{insample}, \text{insample}'\}$ is the output of t .
- $\tau \in \{0, 1\}$ is a boolean value indicating whether or not the stored value of \mathbf{x} will be updated.

Additionally, if $\tau = 0$, then $x = x'$, since \mathbf{x} will not be updated.

To distinguish between the two notationally, all instantiated transitions will be notated \hat{t} where t is the underlying (structural) transition of \hat{t} .

As is natural when analyzing programs for differential privacy, we will often talk about **possible outputs** of a transition t . We observe that for a transition $t = (q, q', c, \sigma, \tau)$, if $\sigma \in \Gamma$, then the possible outputs of t are simply elements of Γ .

Otherwise, the possible outputs of t would be all possible real numbers; to avoid issues with probabilities of measure zero sets, we will consider all measurable subsets of real numbers instead.

Additionally, we will semantically require that in order for t to “output” a value, the guard c must be satisfied.

Definition 4.5 (is this a definition?). Let $t = (q, q', c, \sigma, \tau)$ be a transition and let o be a possible output of t (i.e. if $\sigma \in \Gamma$, let $s \in \Gamma$ and if $\sigma \in \{\text{insample}, \text{insample}'\}$, let $o \subseteq \mathbb{R}$).

Then $\mathbb{P}[t, x, \text{in}, o]$ is the **probability** that t outputs (a value in) o when reading in input $\text{in} \in \mathbb{R}$ given that $\mathbf{x} = x$.

Similarly, let $\hat{t} = ((q, x), (q', x'), c, \sigma, \tau)$ be an instantiated transition and let o be a possible output of \hat{t} .

Then $\mathbb{P}[\hat{t}, \text{in}, o]$ is the **probability** that \hat{t} outputs o when reading in input $\text{in} \in \mathbb{R}$.

Observation 4.6 (Transition Probabilities). Let $\hat{t} = ((q, x), (q', x'), c, \sigma, \tau)$ be an instantiated transition and let o be a possible output of \hat{t} .

$$\text{Then } \mathbb{P}[\hat{t}, \mathbf{in}, o] = \begin{cases} 0 & \tau = 0 \wedge x \neq x' \\ \mathbb{P}[c \text{ is satisfied}] & \tau = 0 \wedge x = x' \wedge \sigma \in \Gamma \\ \mathbb{P}[c \text{ is satisfied} \wedge \mathbf{insample} \in o] & \tau = 0 \wedge x = x' \wedge \sigma = \mathbf{insample} \\ \mathbb{P}[\mathbf{insample}' \in o] \mathbb{P}[c \text{ is satisfied}] & \tau = 0 \wedge x = x' \wedge \sigma = \mathbf{insample}' \\ \mathbb{P}[c \text{ is satisfied} \wedge \mathbf{insample} = x'] & \tau = 1 \wedge \sigma \in \Gamma \\ \mathbb{P}[c \text{ is satisfied} \wedge \mathbf{insample} \in o \wedge \mathbf{insample} = x'] & \tau = 1 \wedge \sigma = \mathbf{insample} \\ \mathbb{P}[\mathbf{insample}' \in o] \mathbb{P}[c \text{ is satisfied} \wedge \mathbf{insample} = x'] & \tau = 1 \wedge \sigma = \mathbf{insample}' \end{cases}$$

Let $X \sim \text{Lap}(\mu, \frac{1}{d_x \varepsilon})$ be a random variable representing the initial value of \mathbf{x} .

Consider a transition $t = (q, q', c, \sigma, \tau)$. Let $\hat{t}_{z, z'}$ be the instantiated transition $\hat{t}_x = ((q, z), (q, z'), c, \sigma, \tau)$. Then the probability that t outputs σ is

$$\mathbb{P}[t, X, \mathbf{in}, o] = \begin{cases} \int_{-\infty}^{\infty} \mathbb{P}[X = z] \mathbb{P}[\hat{t}_{z, z}, \mathbf{in}, o] dz & \tau = 0 \\ \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathbb{P}[\mathbf{insample} = z'] \mathbb{P}[X = z] \mathbb{P}[\hat{t}_{z, z'}, \mathbf{in}, o | \mathbf{insample} = z'] dz' dz & \tau = 1 \end{cases}$$

Trying explicitly including x

We will model programs as finite state automata, with each state of the automaton representing a possible program state. Under this paradigm, we begin with the simplest possible program: a single transition between two program states.

Definition 4.7 (Transitions). Let $\mathcal{C} = \{\mathbf{true}, \mathbf{insample} < \mathbf{x}, \mathbf{insample} \geq \mathbf{x}\}$ be a set of **transition guards**. Let Q be a set of states with each state $q \in Q$ having an associated real value for a variable \mathbf{x} . Additionally, let $P(q) : Q \rightarrow \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$ be a function that associates each state with two **noise parameters** $P(q) = (d_q, d'_q)$. Finally, let Γ be a finite alphabet of **output symbols**.

Then given two states $q, q' \in Q$, a **transition** from q to q' is defined as the tuple $t = (q, q', c, \sigma, \tau)$ where $c \in \mathcal{C}$, $\sigma \in \Gamma \cup \{\mathbf{insample}, \mathbf{insample}'\}$, and τ is a boolean value denoting whether or not the stored value of \mathbf{x} will be updated. At all times, we will assume that there exists most one transition from $q \in Q$ to $q' \in Q$.

Definition 4.8 (Taking a transition). Fix some $\varepsilon > 0$, which we will treat as a program parameter.

Consider some program state $q \in Q$ with noise parameters $P(q) = (d_q, d'_q)$. Let $z \sim \text{Lap}(0, \frac{1}{d_q \varepsilon})$ and $z' \sim \text{Lap}(0, \frac{1}{d'_q \varepsilon})$ be independent random noises sampled from Laplace distributions with spread parameters $\frac{1}{d_q \varepsilon}$ and $\frac{1}{d'_q \varepsilon}$, respectively.

For a state $q \in Q$ and input value $\mathbf{in} \in \mathbb{R}$ read at q , let $\mathbf{insample} = \mathbf{in} + z$ and $\mathbf{insample}' = \mathbf{in} + z'$ be noisy versions of the input read at q .

Let $\sigma \in \Gamma \cup \{(\mathbf{insample}, a, b), (\mathbf{insample}', a, b)\}$ be a potential output of a transition. For measurability reasons, note that we must associate real-valued outputs with an interval (a, b) . Intuitively, $\sigma = (\mathbf{insample}, a, b)$ represents the possibility that $\mathbf{insample}$ was output with a value in the interval (a, b) .

Then given a initial value $x_q \in \mathbb{R}$ for \mathbf{x} , a possible transition $t = (q, q', c, \sigma, \tau)$ from q to another state $q' \in Q$ is **taken** if c is true and, if $\sigma \in \{(\text{insample}, a, b), (\text{insample}', a', b')\}$, t outputs **insample** or **insample'** as appropriate with a value in the interval (a, b) or (a', b') , respectively.

In general, we will assume that at all states q , x_q has been sampled from some Laplace distribution $\text{Lap}(\hat{\mu}_q, \frac{1}{\hat{d}_q \epsilon})$, where \hat{d}_q is the spread parameter for **insample** at some previous state.

4.2 Privacy

Observation 4.9 (Transition probabilities). *Consider a single transition $t = (q, q', c, \sigma, \tau)$ from state $q \in Q$ to $q' \in Q$. Additionally, let $x_q \sim \text{Lap}(\hat{\mu}_q, \frac{1}{\hat{d}_q \epsilon})$ be the initial stored value of \mathbf{x} at q .*

Let σ_0 be a potential output of t

Then, given an input **in** at q , the **probability** that t is taken with output σ_0 in range $(a, b) \subseteq \mathbb{R}$ is

$$\mathbb{P}[x_q, t, \mathbf{in}, \sigma_0] = \begin{cases} \mathbb{P}[c \text{ is satisfied}] & \sigma_0 = \sigma \\ \mathbb{P}[c \text{ is satisfied} \wedge \text{insample} \in (a, b)] & \sigma_0 = (\text{insample}, a, b) \\ \mathbb{P}[\text{insample}' \in (a', b')] \mathbb{P}[c \text{ is satisfied}] & \sigma_0 = (\text{insample}', a', b') \end{cases}$$

where $\text{Lap}_{\mu, s}(x)$ is the PDF of a Laplace distribution with mean μ and spread parameter s .

Note, in particular, that c being satisfied and $\text{insample} \in (a, b)$ are not independent events.

Move this to proof section probably

More precisely, if $(u, v), (u', v') \in \mathbb{R}_\infty$ is defined as follows:

$$(u, v) = \begin{cases} (-\infty, \infty) & c = \text{true} \wedge \sigma_0 \neq (\text{insample}, a, b) \\ (a, b) & c = \text{true} \wedge \sigma_0 = (\text{insample}, a, b) \\ (-\infty, \mathbf{x}_0) & a = \text{insample} < x \wedge \sigma_0 \neq (\text{insample}, a, b) \\ (a, \min(\mathbf{x}_0, b)) & c = \text{insample} < x \wedge \sigma_0 = (\text{insample}, a, b) \\ (\mathbf{x}_0, \infty) & c = \text{insample} \geq x \wedge \sigma_0 \neq (\text{insample}, a, b) \\ (\max(\mathbf{x}_0, a), b) & c = \text{insample} \geq x \wedge \sigma_0 = (\text{insample}, a, b) \end{cases}$$

$$(u', v') = (a', b'),$$

then, given an input **in** at q , the probability that t is taken with output σ_0 can be computed to be

$$\mathbb{P}[x_q, t, \mathbf{in}, \sigma_0] = \begin{cases} \int_u^v \text{Lap}_{\mathbf{in}, \frac{1}{\hat{d}_q \epsilon}}(z) dz & \sigma_0 \neq (\text{insample}', a', b') \\ \int_{u'}^{v'} \text{Lap}_{\mathbf{in}, \frac{1}{\hat{d}_q \epsilon}}(z') dz' \int_u^v \text{Lap}_{\mathbf{in}, \frac{1}{\hat{d}_q \epsilon}}(z) dz & \sigma_0 = (\text{insample}', a', b') \end{cases}$$

where $\text{Lap}_{\mu, s}(x)$ is the PDF of a Laplace distribution with mean μ and spread parameter s .

Recall that \mathbf{in} , in reality, represents a **function** of some underlying dataset. This means that ‘closeness’ in this context is defined as follows:

Definition 4.10 (Adjacency). Two inputs $\mathbf{in} \sim_{\Delta} \mathbf{in}'$ are Δ -adjacent if $|\mathbf{in} - \mathbf{in}'| \leq \Delta$. If Δ is not specified, we assume that $\Delta = 1$.

We can now define what it means to be **differentially private**.

Definition 4.11 ($d\varepsilon$ -differential privacy for a transition). Given two initial values x_q, x'_q , a transition $t = (q, q', c, \sigma, \tau)$ is **$d\varepsilon$ -differentially private** for some $d > 0$ if $\forall \varepsilon > 0$, for all adjacent inputs $\mathbf{in} \sim \mathbf{in}'$ and possible outputs $\sigma \in \Gamma \cup \{(\mathbf{insample}, a, b), (\mathbf{insample}', a', b')\}$, $\mathbb{P}[x_q, t, \mathbf{in}, \sigma] \leq e^{d\varepsilon} \mathbb{P}[x'_q, t, \mathbf{in}', \sigma]$.

Recall that x_q and x'_q are treated as random variables drawn from a Laplace distribution.

Note that we slightly redefine ε -differential privacy as $d\varepsilon$ -differential privacy, treating ε as a universal scaling parameter that can be fine-tuned by users for their own purposes. In particular, we argue that this definition is functionally equivalent⁷, since if we are targeting ε^* -differential privacy, we can always take $\varepsilon = \frac{\varepsilon^*}{d}$.

4.2.1 Couplings

For every transition t between two states q and q' , we can show that t is differentially private using a series of liftings.

Lemma 4.12. Suppose that $x_q \langle 1 \rangle \sim \text{Lap}(\hat{\mu}_q \langle 1 \rangle, \frac{1}{d_q \varepsilon})$, $x_q \langle 2 \rangle \sim \text{Lap}(\hat{\mu}_q \langle 2 \rangle, \frac{1}{d_q \varepsilon})$ are two initial values of \mathbf{x} at state $q \in Q$.

Consider some transition $t = (q, q', c, \sigma, \tau)$ from q to $q' \in Q$. Let $P(q) = (d_q, d'_q)$.

Let $\mathbf{in} \langle 1 \rangle \sim \mathbf{in} \langle 2 \rangle$ be arbitrary and let $\sigma \langle 1 \rangle, \sigma \langle 2 \rangle$ be random variables representing possible outputs of t given inputs $\mathbf{in} \langle 1 \rangle$ and $\mathbf{in} \langle 2 \rangle$, respectively.

Then $\forall \varepsilon > 0$ and for all $\gamma_x, \gamma_q, \gamma'_q \in [-1, 1]$ that satisfy the constraints

$$\begin{cases} \gamma_q \leq \gamma_x & c = \mathbf{insample} < x \\ \gamma_q \geq \gamma_x & c = \mathbf{insample} \geq x \\ \gamma_q = 0 & \sigma = \mathbf{insample} \\ \gamma'_q = 0 & \sigma = \mathbf{insample}' \end{cases},$$

the lifting $\sigma \langle 1 \rangle \{ (a, b) : a = \sigma \implies b = \sigma \}^{\#d\varepsilon} \sigma \langle 2 \rangle$ is valid for $d = (|\hat{\mu}_q \langle 1 \rangle - \hat{\mu}_q \langle 2 \rangle + \gamma_x|)d_q + (|-\mathbf{in} \langle 1 \rangle + \mathbf{in} \langle 2 \rangle - \gamma_q|)d_q + (|-\mathbf{in} \langle 1 \rangle + \mathbf{in} \langle 2 \rangle - \gamma'_q|)d'_q$, and therefore t is $d\varepsilon$ -differentially private.

⁷[6] notes that it is not entirely clear how this differs from standard differential privacy, but that the known decidability result does not apply here - **maybe something to investigate**

Proof. Fix $\varepsilon > 0$.

We will analyze the behaviour of two different **runs** of t , one with input **in** and one with input **in'**.

At a high level, for every Laplace-distributed variable, we will couple the value of the variable in one run with its value in the other **shifted** by some amount.

We differentiate between the values of variables in the first and second run by using angle brackets $\langle k \rangle$, so, for example, we will take $x_q\langle 1 \rangle$ to be the value of \mathbf{x} at state q in the run of t with input **in** $\langle 1 \rangle$ and $x_q\langle 2 \rangle$ to be the value of \mathbf{x} in the run of t with input **in** $\langle 2 \rangle$.

We thus want to create the lifting $\sigma\langle 1 \rangle \{(a, b) : a = \sigma \implies b = \sigma\} \sigma\langle 2 \rangle$. We must guarantee two things: that if the first transition is taken, then the second is also taken and that both runs output the same value σ when taking the transition. Note that if $c = \mathbf{true}$, the first condition is trivially satisfied and when $\sigma \in \Gamma$, the second condition is trivially satisfied.

Recall that that x_q is sampled from a Laplace distribution, so $x_q\langle 1 \rangle \sim \text{Lap}(\hat{\mu}_q\langle 1 \rangle, \frac{1}{d_q\varepsilon})$ and $x_q\langle 2 \rangle \sim \text{Lap}(\hat{\mu}_q\langle 2 \rangle, \frac{1}{d_q\varepsilon})$.

Then we can first create the lifting $x_q\langle 1 \rangle + \gamma_x(=)^{\#(|\hat{\mu}_q\langle 1 \rangle - \hat{\mu}_q\langle 2 \rangle + \gamma_x|)d_q\varepsilon} x_q\langle 2 \rangle$.

Additionally, create the lifting $z\langle 1 \rangle(=)^{\#(|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_q|)d_q\varepsilon} z\langle 2 \rangle - \mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_q$.

This is equivalent to creating the lifting **insample** $\langle 1 \rangle + \gamma_q(=)^{\#(|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_q|)d_q\varepsilon}$ **insample** $\langle 2 \rangle$.

Finally, create the lifting $z'\langle 1 \rangle(=)^{\#(|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma'_q|)d'_q\varepsilon} z'\langle 2 \rangle - \mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma'_q$. As before, this is equivalent to creating the lifting **insample'** $\langle 1 \rangle + \gamma'_q(=)^{\#(|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma'_q|)d'_q\varepsilon}$ **insample'** $\langle 2 \rangle$.

If $c = \mathbf{insample} < \mathbf{x}$ and $\gamma_q \leq \gamma_x$, then

$$\begin{aligned} \mathbf{insample}\langle 1 \rangle < x_q\langle 1 \rangle &\implies \mathbf{in}\langle 1 \rangle + z\langle 1 \rangle < x_q\langle 1 \rangle \\ &\implies \mathbf{in}\langle 1 \rangle + z\langle 2 \rangle - \mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_q < x_q\langle 2 \rangle - \gamma_x \\ &\implies \mathbf{insample}\langle 2 \rangle < x_q\langle 2 \rangle \end{aligned}$$

Similarly, if $c = \mathbf{insample} \geq \mathbf{x}$ and $\gamma_q \geq \gamma_x$, then

$$\begin{aligned} \mathbf{insample}\langle 1 \rangle \geq x_q\langle 1 \rangle &\implies \mathbf{in}\langle 1 \rangle + z\langle 1 \rangle \geq x_q\langle 1 \rangle \\ &\implies \mathbf{in}\langle 1 \rangle + z\langle 2 \rangle - \mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_q \geq x_q\langle 2 \rangle - \gamma_x \\ &\implies \mathbf{insample}\langle 2 \rangle \geq x_q\langle 2 \rangle \end{aligned}$$

With these liftings, we have ensured that if the first run takes transition t , then the second run does as well.

Now, if t outputs **insample** and $\gamma_q = 0$, then clearly we have that **insample** $\langle 1 \rangle = \mathbf{insample}\langle 2 \rangle$, so $\sigma\langle 1 \rangle = (\mathbf{insample}, a, b) \implies \sigma\langle 2 \rangle = (\mathbf{insample}, a, b)$.

Similarly, if $\gamma'_q = 0$, we have that $\sigma\langle 1 \rangle = (\mathbf{insample}', a, b) \implies \sigma\langle 2 \rangle = (\mathbf{insample}', a, b)$.

Thus, if t outputs a real number in the interval (a, b) in the first run, it must also output a real number in the same interval in the second run.

Thus, given the constraints

$$\begin{cases} \gamma_q \leq \gamma_x & c = \text{insample} < \mathbf{x} \\ \gamma_q \geq \gamma_x & c = \text{insample} \geq \mathbf{x} \\ \gamma_q = 0 & \sigma = \text{insample} \\ \gamma'_q = 0 & \sigma = \text{insample}' \end{cases},$$

this is sufficient to create the lifting $\sigma\langle 1 \rangle \{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon} \sigma\langle 2 \rangle$, where the cost $d = (|\hat{\mu}_q\langle 1 \rangle - \hat{\mu}_q\langle 2 \rangle + \gamma_x|)\hat{d}_q + (|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_q|)d_q + (|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma'_q|)d'_q$.

By an application of theorem 3.4, $\mathbb{P}[x_q\langle 1 \rangle, t, \mathbf{in}\langle 1 \rangle, \sigma] \leq e^{d\varepsilon} \mathbb{P}[x_q\langle 2 \rangle, t, \mathbf{in}\langle 2 \rangle, \sigma]$. Because $\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$ are arbitrary adjacent inputs and σ is an arbitrary possible output of t , this implies that t is $d\varepsilon$ -differentially private. \square

We can thus think of couplings for a transition as being parameterized by γ_x , γ_q , and γ'_q .

Definition 4.13 (Coupling strategies for a transition). A **coupling strategy** for a transition $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ is a tuple $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i) \in [-1, 1]^3$.

Definition 4.14 (Validity of a coupling strategy). A coupling strategy $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i)$ for a transition t_i is **valid** if the constraints

$$\begin{cases} \gamma_i \leq \gamma_x^{(i)} & c_i = \text{insample} < \mathbf{x} \\ \gamma_i \geq \gamma_x^{(i)} & c_i = \text{insample} \geq \mathbf{x} \\ \gamma_i = 0 & \sigma_i = \text{insample} \\ \gamma'_i = 0 & \sigma_i = \text{insample}' \end{cases},$$

are all satisfied.

In particular, a valid coupling proof gives an upper bound on the privacy cost of any individual transition.

Proposition 4.15. *For a transition t_i , if there exists a valid coupling strategy $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i)$ given initial \mathbf{x} values centred at $\hat{\mu}_q\langle 1 \rangle$ and $\hat{\mu}_q\langle 2 \rangle$, then t_i is $d\varepsilon$ -differentially private for some*

$$d \leq \max_{\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle} (|\hat{\mu}_q\langle 1 \rangle - \hat{\mu}_q\langle 2 \rangle + \gamma_x^{(i)}|)\hat{d}_q + (|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma_i|)d_q + (|-\mathbf{in}\langle 1 \rangle + \mathbf{in}\langle 2 \rangle - \gamma'_i|)d'_q.$$

Proof. Follows immediately from lemma 4.12. \square

4.3 Multiple Transitions

Of course, in practice we would like to analyze the behaviour of programs with more than two program states.

We start with *executions*, comprised of a sequence of transitions.

Definition 4.16 (Program executions). Consider some set of program states Q . A program **execution** is a sequence of transitions t_0, t_1, \dots, t_{n-1} such that for all $i \in 0 \dots n-1$, $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ for some c_i, σ_i, τ_i . We will often notate an execution ρ as $\rho = q_0 \rightarrow q_1 \rightarrow \dots \rightarrow q_n$.

Additionally, if $\tau_0 = \text{true}$ and $c_0 = \text{true}$ (i.e. the first transition in any execution always assigns its noised input value into \mathbf{x}), then ρ is a **complete execution**. We call this condition the **initialization** condition.

Definition 4.17 (Taking a execution). Consider some complete execution $\rho = q_0 \rightarrow \dots \rightarrow q_n$ and let \mathbf{x}_i for all $i \in 1 \dots n$ be the value of \mathbf{x} at each program state q_i . Note that we can safely ignore \mathbf{x}_0 because of the initialization condition. As before, for a program state $q_i \in Q$ with noise parameters $P(q_i) = (d_{q_i}, d'_{q_i})$, let $z_i \sim \text{Lap}(0, \frac{1}{d_{q_i}\epsilon})$ and $z'_i \sim \text{Lap}(0, \frac{1}{d'_{q_i}\epsilon})$. Further, given an input $\text{in} \in \mathbb{R}^n$ and possible output $\sigma \in (\Gamma \cup \{(\text{insample}, a, b), (\text{insample}', a, b)\})^n$, let $\text{insample}_i = \text{in}_i + z_i$ and $\text{insample}'_i = \text{in}_i + z'_i$. \mathbf{x} will be updated with insample at a

transition t_i if and only if $\tau_i = \text{true}$. In other words, $\mathbf{x}_i = \begin{cases} \mathbf{x}_{i-1} & \tau_{i-1} = \text{false} \\ \text{insample}_{i-1} & \tau_{i-1} = \text{true} \end{cases}$.

Then ρ is **taken** if, $\forall i \in 0 \dots n-1$, c_i is satisfied for insample_i and \mathbf{x}_i and, if t_i outputs a real value, it is of the form $(\text{insample}_i, a, b)$ or $(\text{insample}'_i, a', b')$, as appropriate.

4.3.1 Privacy

We can naturally extend the definition of probability on a single transition to an entire execution recursively.

Definition 4.18. Given a execution $\rho = q_0 \rightarrow q_1 \rightarrow \dots \rightarrow q_n$, the **tail** of ρ is defined as $\text{tail}(\rho) = q_1 \rightarrow q_2 \rightarrow \dots \rightarrow q_n$.

We may additionally use the notation $\rho_{i:j}$ to represent the subexecution $q_i \rightarrow q_{i+1} \rightarrow \dots \rightarrow q_j$ of ρ . Using this notation, $\text{tail}(\rho) = \rho_i = \rho_{i:n}$.

Definition 4.19 (Execution Probabilities). Consider a execution $\rho = q_0 \rightarrow q_1 \rightarrow \dots \rightarrow q_n$ of length n . Let $t_0 = (q_0, q_1, c_0, \sigma_0, \tau_0)$ between states q and q' . Additionally, let \mathbf{x}_0 be the initial value of \mathbf{x} at q_0 .

If $\sigma_0 \in \{\text{insample}, \text{insample}'\}$, then let $\sigma_0 = (\text{insample}, a, b)$ or $\sigma_0 = (\text{insample}', a', b')$ as appropriate.

Additionally, let $(u, v), (u', v') \in \mathbb{R}_\infty$ be defined as follows:

$$(u, v) = \begin{cases} (-\infty, \infty) & c_0 = \text{true} \wedge \sigma_0 \neq (\text{insample}, a, b) \\ (a, b) & c_0 = \text{true} \wedge \sigma_0 = (\text{insample}, a, b) \\ (-\infty, \mathbf{x}_0) & c_0 = \text{insample} < x \wedge \sigma_0 \neq (\text{insample}, a, b) \\ (c, \min(\mathbf{x}_0, d)) & c_0 = \text{insample} < x \wedge \sigma_0 = (\text{insample}, a, b) \\ (\mathbf{x}_0, \infty) & c_0 = \text{insample} \geq x \wedge \sigma_0 \neq (\text{insample}, a, b) \\ (\max(\mathbf{x}_0, c), d) & c_0 = \text{insample} \geq x \wedge \sigma_0 = (\text{insample}, a, b) \end{cases}$$

$$(u', v') = (a', b')$$

Then, given an initial \mathbf{x} -value \mathbf{x}_0 , an input sequence $\mathbf{in} \in \mathbb{R}^n$, and an output sequence $\sigma \in (\Gamma \cup \mathbb{R})^n$, the probability of ρ is defined as

$$\mathbb{P}[\mathbf{x}_0, \rho, \mathbf{in}, \sigma] = \begin{cases} 1 & n = 0 \\ \int_u^v \text{Lap}_{\mathbf{in}_0, \frac{1}{d_0\epsilon}}(z) dz \mathbb{P}[\mathbf{x}_0, \rho_{1:}, \mathbf{in}_{1:}, \sigma_{1:}] & \sigma_0 \neq \text{insample}' \wedge \tau_0 = \text{false} \\ \int_u^v \text{Lap}_{\mathbf{in}_0, \frac{1}{d_0\epsilon}}(z) \mathbb{P}[z, \rho_{1:}, \mathbf{in}_{1:}, \sigma_{1:}] dz & \sigma_0 \neq \text{insample}' \wedge \tau_0 = \text{true} \\ \int_{u'}^{v'} \text{Lap}_{\mathbf{in}_0, \frac{1}{d_0\epsilon}}(z') dz' \int_u^v \text{Lap}_{\mathbf{in}_0, \frac{1}{d_0\epsilon}}(z) dz \mathbb{P}[\mathbf{x}_0, \rho_{1:}, \mathbf{in}_{1:}, \sigma_{1:}] & \sigma_0 = \text{insample}' \wedge \tau_0 = \text{false} \\ \int_{u'}^{v'} \text{Lap}_{\mathbf{in}_0, \frac{1}{d_0\epsilon}}(z') dz' \int_u^v \text{Lap}_{\mathbf{in}_0, \frac{1}{d_0\epsilon}}(z) \mathbb{P}[z, \rho_{1:}, \mathbf{in}_{1:}, \sigma_{1:}] dz & \sigma_0 = \text{insample}' \wedge \tau_0 = \text{true} \end{cases}$$

where $\text{Lap}_{\mu, s}(x)$ is the PDF of a Laplace distribution with mean μ and spread parameter s .

For a complete execution ρ , note that the initial value of \mathbf{x} is irrelevant, so we will shorthand $\mathbb{P}[\mathbf{x}_0, \rho, \mathbf{in}, \sigma]$ to $\mathbb{P}[\rho, \mathbf{in}, \sigma]$.

Additionally, because we now read in a *sequence* of real-valued inputs, we need to slightly modify our definition of adjacency.

Definition 4.20 (Adjacency for a sequence of inputs). Two input sequences $\{\alpha_i\}_{i=1}^n, \{\beta_i\}_{i=1}^n$ of length n are Δ -adjacent (notated $\alpha \sim_\Delta \beta$) if, for all $i \in [1 \dots n]$, $|\alpha_i - \beta_i| \leq \Delta$.

If Δ is not specified, we assume that $\Delta = 1$.

Thus, we get the following definition of privacy:

Definition 4.21 ($d\epsilon$ -differential privacy for an execution). A complete execution ρ of length n is $d\epsilon$ -differentially private for some $d > 0$ if $\forall \epsilon > 0$, for all adjacent input sequences $\alpha \sim \beta$ of length n and all possible output sequences σ of length n , $\mathbb{P}[\rho, \alpha, \sigma] \leq e^{d\epsilon} \mathbb{P}[\rho, \beta, \sigma]$.

4.3.2 Concatenating couplings

Let $\rho[\mathbf{in}]$ be a random variable representing the output of ρ given input sequence \mathbf{in} .

In order to show that a program execution ρ is differentially private, for all adjacent inputs $\alpha \sim \beta$ and all possible outputs σ , we want to create the coupling $\rho[\alpha]\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\epsilon} \rho[\beta]$ for some $d > 0$.

Ideally, we would like to simply create couplings for each individual transition in ρ as before and compose them together to create this overall coupling. Indeed, this approach is almost sufficient; the constraints imposed upon shifts for a coupling for transition t_i depend solely on the shift at the most recent **assignment transition** in ρ (i.e. the most recent transition t_j such that $\tau_j = \mathbf{true}$). The coupling shifts for *non-assignment transitions* can thus never impact each other.

Definition 4.22 (Assignment transitions). Let $A_\rho = \{t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i) : \tau_i = \mathbf{true}\}$ be the set of **assignment transitions** in a execution ρ .

For every transition t_i in ρ , let $t_{at(i)}$ be the most recent assignment transition in ρ ; i.e., $at(i) = \max\{j < i : t_j \in A_\rho\}$. If such a j does not exist, we set $at(i) = -1$.

In particular, note that for transition t_i , $\gamma_x = \gamma_{at(i)}$, where γ_{-1} is the shift applied to the initial \mathbf{x} -values $\mathbf{x}_0\langle 1 \rangle$ and $\mathbf{x}_0\langle 2 \rangle$.

Thus, for an individual transition t_i of ρ . From proposition 4.15, we have a family of valid coupling strategies $C_i(\gamma_{at(i)}, \gamma_i, \gamma'_i)$.

We can merge these coupling strategies together to create a proof of privacy for the entire execution:

Lemma 4.23. *Let $\rho = q_0 \rightarrow \dots \rightarrow q_n$ be a complete execution of length n . Let $\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$ be arbitrary adjacent input sequences of length n . Additionally, fix some potential output σ of ρ of length n and let $\sigma\langle 1 \rangle, \sigma\langle 2 \rangle$ be random variables representing possible outputs of ρ given inputs $\mathbf{in}\langle 1 \rangle$ and $\mathbf{in}\langle 2 \rangle$, respectively. Additionally, for all q_i , let $P(q_i) = (d_i, d'_i)$.*

Then $\forall \varepsilon > 0$ and for all $\{\gamma_i, \gamma'_i\}_{i=0}^{n-1}$ that, for all i , satisfy the constraints

$$\begin{cases} \gamma_i \leq \gamma_{at(i)} & c_i = \mathbf{insample} < x \\ \gamma_i \geq \gamma_{at(i)} & c_i = \mathbf{insample} \geq x \\ \gamma_i = 0 & \sigma_i = \mathbf{insample} \\ \gamma'_i = 0 & \sigma_i = \mathbf{insample}' \end{cases},$$

the lifting $\sigma\langle 1 \rangle \{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon} \sigma\langle 2 \rangle$ is valid for $d = \sum_{i=0}^{n-1} (|-\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle - \gamma_i|)d_i + (|-\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle - \gamma'_i|)d'_i$, and therefore t is $d\varepsilon$ -differentially private.

Proof. From the proof of lemma 4.12, we know that we can create the couplings $\mathbf{insample}_i\langle 1 \rangle + \gamma_i(=)^{\#(|-\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle - \gamma_i|)d_i\varepsilon} \mathbf{insample}_i\langle 2 \rangle$ and $\mathbf{insample}'_i\langle 1 \rangle + \gamma'_i(=)^{\#(|-\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle - \gamma'_i|)d'_i\varepsilon} \mathbf{insample}'_i\langle 2 \rangle$ for all q_i in ρ .

Additionally, for some fixed q_i in ρ , if we have the coupling $\mathbf{x}_i\langle 1 \rangle + \gamma_x(=)^{\#(|\hat{\mu}_i\langle 1 \rangle - \hat{\mu}_i\langle 2 \rangle + \gamma_x|)d_i\varepsilon} x_i\langle 2 \rangle$, where $\mathbf{x}_i\langle 1 \rangle \sim \text{Lap}(\hat{\mu}_i\langle 1 \rangle, \frac{1}{d_i\varepsilon})$ and $\mathbf{x}_i\langle 2 \rangle \sim \text{Lap}(\hat{\mu}_i\langle 2 \rangle, \frac{1}{d_i\varepsilon})$, then subject to the constraints

$$\begin{cases} \gamma_i \leq \gamma_x & c_i = \mathbf{insample} < x \\ \gamma_i \geq \gamma_x & c_i = \mathbf{insample} \geq x \\ \gamma_i = 0 & \sigma_i = \mathbf{insample}_i \\ \gamma'_i = 0 & \sigma_i = \mathbf{insample}'_i \end{cases},$$

the coupling $\sigma_i \langle 1 \rangle \{(a, b) : a = \sigma_i \implies b = \sigma_i\}^{\#d\varepsilon} \sigma_i \langle 2 \rangle$ is valid for some d .

Indeed, note that for all i , $\mathbf{x}_i = \mathbf{insample}_{at(i)}$ by definition. Thus, we have that $\mathbf{x}_i \langle 1 \rangle + \gamma_x (=) \#(|-\mathbf{in}_{at(i)} \langle 1 \rangle + \mathbf{in}_{at(i)} \langle 2 \rangle + \gamma_{at(i)}|)^{d_{at(i)}\varepsilon} x_i \langle 2 \rangle$, and we must satisfy the constraints

$$\begin{cases} \gamma_i \leq \gamma_{at(i)} & c_i = \mathbf{insample} < \mathbf{x} \\ \gamma_i \geq \gamma_{at(i)} & c_i = \mathbf{insample} \geq \mathbf{x} \\ \gamma_i = 0 & \sigma_i = \mathbf{insample}_i \\ \gamma'_i = 0 & \sigma_i = \mathbf{insample}'_i \end{cases}$$

for all i .

Thus, we can put all of these couplings together to show that the coupling $\sigma_i \langle 1 \rangle \{(a, b) : a = \sigma_i \implies b = \sigma_i\}^{\#d\varepsilon} \sigma_i \langle 2 \rangle$ is valid for some $d > 0$.

In particular, note that we have created at most one pair of couplings (for $\mathbf{insample}$ and $\mathbf{insample}'$) for each q_i . Thus, the total coupling cost associated with each q_i is at most $(|-\mathbf{in}_i \langle 1 \rangle + \mathbf{in}_i \langle 2 \rangle - \gamma_i|)d_i + (|-\mathbf{in}_i \langle 1 \rangle + \mathbf{in}_i \langle 2 \rangle - \gamma'_i|)d'_i$, which gives us an overall coupling cost of $d = \sum_{i=0}^{n-1} (|-\mathbf{in}_i \langle 1 \rangle + \mathbf{in}_i \langle 2 \rangle - \gamma_i|)d_i + (|-\mathbf{in}_i \langle 1 \rangle + \mathbf{in}_i \langle 2 \rangle - \gamma'_i|)d'_i$. \square

Definition 4.24. For a complete execution ρ of length n and adjacent input sequences $\mathbf{in} \langle 1 \rangle \sim \mathbf{in} \langle 2 \rangle$, a **coupling strategy** is two functions $\gamma(\mathbf{in} \langle 1 \rangle, \mathbf{in} \langle 2 \rangle) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow [-1, 1]^n$ and $\gamma'(\mathbf{in} \langle 1 \rangle, \mathbf{in} \langle 2 \rangle) : \mathbb{R}^n \times \mathbb{R}^n \rightarrow [-1, 1]^n$ that produce shifts for each transition of ρ dependent on the input sequences.

If $\mathbf{in} \langle 1 \rangle$ and $\mathbf{in} \langle 2 \rangle$ are clear from context, we will often shorthand notating a coupling strategy as γ and γ' .

Definition 4.25. For a complete execution ρ of length n , a coupling strategy $C_\rho = (\gamma, \gamma')$ is **valid** if $\forall \mathbf{in} \langle 1 \rangle \sim \mathbf{in} \langle 2 \rangle$, $\gamma(\mathbf{in} \langle 1 \rangle, \mathbf{in} \langle 2 \rangle)$ and $\gamma'(\mathbf{in} \langle 1 \rangle, \mathbf{in} \langle 2 \rangle)$ satisfy the constraints

$$\begin{cases} \gamma_i \leq \gamma_{at(i)} & c_i = \mathbf{insample} < \mathbf{x} \\ \gamma_i \geq \gamma_{at(i)} & c_i = \mathbf{insample} \geq \mathbf{x} \\ \gamma_i = 0 & \sigma_i = \mathbf{insample} \\ \gamma'_i = 0 & \sigma_i = \mathbf{insample}' \end{cases}.$$

4.3.3 Optimizing Privacy

Definition 4.26. For a complete execution ρ of length n , the **cost** of a coupling strategy $C_\rho = (\gamma, \gamma')$ is

$$\text{cost}(C_\rho) = \max_{\mathbf{in} \langle 1 \rangle \sim \mathbf{in} \langle 2 \rangle} \sum_{i=0}^{n-1} (|-\mathbf{in}_i \langle 1 \rangle + \mathbf{in}_i \langle 2 \rangle - \gamma_i|)d_i + (|-\mathbf{in}_i \langle 1 \rangle + \mathbf{in}_i \langle 2 \rangle - \gamma'_i|)d'_i.$$

Additionally, let G be the set of all valid coupling strategies $C_\rho = (\gamma, \gamma')$ for ρ . Then the **coupling cost** of ρ is

$$\text{cost}(\rho) = \min_{(\gamma, \gamma') \in G} \text{cost}((\gamma, \gamma')).$$

As before, the existence of a valid coupling strategy upper bounds the privacy cost of any execution.

Proposition 4.27. *If $C_\rho = (\gamma, \gamma')$ is valid, then ρ is $\text{cost}(C_\rho)\varepsilon$ -differentially private.*

Proof. Follows immediately from lemma 4.23. \square

Corollary 4.28. *Any complete execution ρ is $\text{cost}(\rho)\varepsilon$ -differentially private. Further, for all complete executions ρ , $\text{cost}(\rho) < \infty$.*

Proof. The first claim follows immediately from definitions.

The second claim follows by considering a coupling strategy (γ, γ') where $\forall \mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle, \gamma = \gamma' = \mathbf{0}$. Note that (γ, γ') is trivially valid. Since $\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$, $\text{cost}(\rho) \leq \text{cost}(C_\rho(\mathbf{0})) \leq \sum_{i=0}^{n-1} (d_i + d'_i)$, which is finite for all fixed ρ . \square

Proposition 4.29. *The cost of a coupling strategy over a fixed execution ρ is maximized when, for every transition, the difference between the input values is 1. In other words,*

$$\begin{aligned} & \max_{\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle} \sum_{i=0}^{n-1} (| - \mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle - \gamma_i |) d_i + (| - \mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle - \gamma'_i |) d'_i \\ &= \max_{\Delta \in \{-1, 1\}} \sum_{i=0}^{n-1} (|\Delta - \gamma_i|) d_i + (|\Delta - \gamma'_i|) d'_i \end{aligned}$$

Proof. There's a massive \square

Note then, that the process of finding the optimal coupling cost of a execution ρ can be formulated as a linear program.

4.4 Branching

Definition 4.30 (Branching program). Given a set of program states Q , a branching program B is a finite set of complete executions over Q such that for all $\rho = q_0 \rightarrow \dots \rightarrow q_n, \rho' = q'_0 \rightarrow \dots \rightarrow q'_m \in B, q_0 = q'_0$. B must also satisfy the following properties:

1. If any transition t in any execution $\rho \in B$ is of the form (q, q', c, σ, τ) , then no other transitions of the form $(q, q', c, \sigma', \tau')$ for $q, q', c \in Q$ exist in any execution of B . Additionally, if any transition t in any execution $\rho \in B$ is of the form $(q, q', \mathbf{true}, \sigma, \tau)$, then transitions of the form $(q, q', \mathbf{insample} < \mathbf{x}, \sigma', \tau')$ or $(q, q', \mathbf{insample} < \mathbf{x}, \sigma', \tau')$ do not exist in any execution of B .
2. If a transition of the form $(q, q', \mathbf{insample} < \mathbf{x}, \sigma, \tau)$ exists in any execution in B and a transition of the form $(q, q', \mathbf{insample} \geq \mathbf{x}, \sigma', \tau')$ exists in any (potentially different) execution in B , then $\sigma \neq \sigma'$. Additionally, at least one of $\sigma \in \Gamma, \sigma' \in \Gamma$ is true.

We will refer to the first condition as **determinism** and the second condition as **output distinction**.

4.4.1 Privacy

Definition 4.31 (DP for branching programs). A branching program is DP if [standard def]

Proposition 4.32. *A branching program B is $d\varepsilon$ -differentially private for $d > 0$ if and only if for every complete execution ρ in B , ρ is $d\varepsilon$ -differentially private.*

Proof. From the conditions of output distinction and determinism, note that for every single possible output σ of B , there is exactly one possible execution ρ that could have output σ . Thus, the probability that B outputs σ is exactly the probability that ρ outputs σ ; the result follows immediately. \square

We can do no better than assigning coupling strategies for each execution independently.

Definition 4.33 (Coupling strategies). A (branched program) coupling strategy C for a branching program B is a collection of (execution) coupling strategies where each complete execution $\rho \in B$ is assigned a coupling strategy C_ρ .

Definition 4.34. A coupling strategy C for a branching program B is valid if, for every constituent execution coupling strategy C_ρ , C_ρ is valid.

Definition 4.35. The cost of a coupling strategy C for a branched program B is

$$\max_{\rho \in B} \text{cost}(\rho)$$

Proposition 4.36. *Optimal cost is dependent on execution*

Proof. TODO: Vishnu \square

is this still necessary?

As previously noted, we use a finite union of complete executions to model an underlying program with branching, in part due to proposition 4.36. We now formally define that underlying model:

Definition 4.37. The underlying graph [NAME VERY TENTATIVE] of a branching program B is the directed acyclic graph $G = (V, E)$ such that $V = Q$ and $(q, q') \in E$ if, for some execution $\rho \in B$, there exists a transition $t \in \rho$ such that $t = (q, q', c, \sigma, \tau)$ for some c, σ, τ .

need also to force this to be acyclic

4.5 Loops

We will illustrate applying coupling strategies to loops through a simple **lasso** model.

Definition 4.38 (Lassos). Consider a complete execution $\rho = q_0 \rightarrow \dots \rightarrow q_n$ such that for some $i < n$, $q_i = q_n$. Suppose also that $\forall j \neq i, q_j \neq q_n$. A lasso L_ρ is the infinite set of executions $\{q_0 \rightarrow \dots \rightarrow q_{i-1} \rightarrow (q_i \rightarrow \dots \rightarrow q_{n-1})^k \rightarrow q_n \mid k \in \mathbb{N}\}$, where $(q_i \rightarrow \dots \rightarrow q_{n-1})^k$ means that the subexecution $(q_i \rightarrow \dots \rightarrow q_{n-1})$ is iterated k times. Additionally, L_ρ must satisfy the properties of **determinism** and **output distinction** as with branching programs.

Definition 4.39 (Lasso privacy). A lasso L_ρ is $d\varepsilon$ -differentially private for $d > 0$ if for all complete executions ρ' in L_ρ , for all possible outputs $\sigma_{\rho'}$ of ρ' and input sequences $\text{in}\langle 1 \rangle \sim \text{in}\langle 2 \rangle$, $\forall \varepsilon > 0, \mathbb{P}[\rho', \sigma_{\rho'}, \text{in}\langle 1 \rangle] \leq e^{d\varepsilon} \mathbb{P}[\rho', \sigma_{\rho'}, \text{in}\langle 2 \rangle]$.

Proposition 4.40. A lasso L_ρ is $d\varepsilon$ -differentially private for $d > 0$ if and only if for all complete executions ρ' in L_ρ , ρ' is $d\varepsilon$ -differentially private.

Definition 4.41 (Coupling strategies). A **coupling strategy** for a lasso L_ρ , where $\rho = q_0 \rightarrow \dots \rightarrow q_n$ is an assignment of shifts γ_i, γ'_i for each transition in ρ as a function of the input differences.

Definition 4.42 (Validity). A coupling strategy $C_{L_\rho} = (\gamma, \gamma')$ for a lasso L_ρ is **valid** if, for all executions ρ' of A , the following constraints hold for all adjacent input sequences $\alpha \sim \beta$ and for all i :

- If $c_i = \text{insample} < \mathbf{x}$, then $\gamma_i \leq \gamma_{at(i)}$
- If $c_i = \text{insample} \geq \mathbf{x}$, then $\gamma_i \geq \gamma_{at(i)}$
- If $\sigma_i = \text{insample}$, then $\gamma_i = 0$
- If $\sigma_i = \text{insample}'$, then $\gamma'_i = 0$

Note that given a complete execution ρ' in a lasso L_ρ , a coupling strategy C_A induces a coupling strategy $C_{\rho'}$ for ρ' .

Definition 4.43. The **cost** of a coupling strategy $C_{L_\rho} = (\gamma, \gamma')$ for a lasso L_ρ is

$$\begin{aligned} \text{cost}(C_{L_\rho}) &= \sup_{\rho' \in L_\rho} \max_{\text{in}\langle 1 \rangle \sim \text{in}\langle 2 \rangle} \sum_{i=0}^{|\rho'|-1} (| -\text{in}_i\langle 1 \rangle + \text{in}_i\langle 2 \rangle - \gamma_i |) d_i + (| -\text{in}_i\langle 1 \rangle + \text{in}_i\langle 2 \rangle - \gamma'_i |) d'_i \\ &= \sup_{\rho' \in L_\rho} \text{cost}(C_{\rho'}) \end{aligned}$$

Further, let G_{L_ρ} be the set of all valid coupling strategies for L_ρ . Then the **coupling cost** of L_ρ is

$$\text{cost}(L_\rho) = \min_{C_{L_\rho} \in G_{L_\rho}} \text{cost}(C_{L_\rho})$$

Proposition 4.44. Given a valid coupling strategy C_{L_ρ} for a lasso L_ρ , L_ρ is $\text{cost}(C_{L_\rho})\varepsilon$ -differentially private.

Proof. Let ρ' be a complete execution in L_ρ and let $C_{\rho'}$ be the coupling strategy for ρ' derived from C_A . We know from lemma 4.23 that, if C_A is valid, then the lifting $\sigma\langle 1 \rangle\{(a, b) :$

$a = \sigma \implies b = \sigma\}^{\#cost(C_{\rho'})\varepsilon}\sigma\langle 2 \rangle$ is valid, and therefore for all possible outputs σ of ρ' and adjacent inputs $\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$, $\forall \varepsilon > 0, \mathbb{P}[\rho', \sigma, \mathbf{in}\langle 1 \rangle] \leq e^{cost(C_{\rho'})\varepsilon} \mathbb{P}[\rho', \sigma, \mathbf{in}\langle 2 \rangle] \leq e^{cost(C_{L_\rho})\varepsilon} \mathbb{P}[\rho', \sigma, \mathbf{in}\langle 2 \rangle]$. \square

Corollary 4.45. *A lasso L_ρ is $cost(L_\rho)\varepsilon$ -differentially private.*

Lemma 4.46. *For a lasso L_ρ , given adjacent input sequences $\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$, a valid coupling strategy $C_{L_\rho} = (\gamma, \gamma')$ has finite cost $cost(C_{L_\rho}) < \infty$ if and only if the following constraint applies for all i :*

- If t_i is in a cycle, then $\gamma_i = -\mathbf{in}\langle 1 \rangle_i + \mathbf{in}\langle 2 \rangle_i$ and $\gamma'_i = -\mathbf{in}\langle 1 \rangle_i + \mathbf{in}\langle 2 \rangle_i$.

Proof. (\Leftarrow)

Let T be the set of transitions t_i in L_ρ such that t_i is **not** in a cycle. Note that T is independent of any choice of execution(s) through L_ρ .

Fix a complete execution ρ' in L_ρ and let C'_ρ be the coupling strategy for ρ' induced by C_{L_ρ} .

Let D_ρ be the set of transitions t_i in ρ such that t_i is in a cycle in L_ρ , i.e., $t_i \notin T$.

If the given constraint holds, then we know that $\max_{\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle} \sum_{i:t_i \in D_\rho} (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma_i|)d_i + (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma'_i|)d'_i = 0$

So

$$\begin{aligned} cost(C_\rho) &= \max_{\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle} \sum_{i:t_i \in D_\rho} (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma_i|)d_i + (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma'_i|)d'_i \\ &\quad + \sum_{i:t_i \notin D_\rho} (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma_i|)d_i + (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma'_i|)d'_i \\ &= \max_{\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle} \sum_{i:t_i \in T} (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma_i|)d_i + (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma'_i|)d'_i \\ &\leq \sum_{i:t_i \in T} (2d_i + 2d'_i) \\ &\leq |T| \max_{i:t_i \in T} (2d_i + 2d'_i) \end{aligned}$$

(\implies)

Suppose that some transition t_i is in a cycle C in A and $\gamma_i \neq -\mathbf{in}\langle 1 \rangle_i + \mathbf{in}\langle 2 \rangle_i$ or $\gamma'_i \neq -\mathbf{in}\langle 1 \rangle_i + \mathbf{in}\langle 2 \rangle_i$. Then $\exists \mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$ such that $(|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma_i|)d_i + (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma'_i|)d'_i > 0$.

Fix such a $\mathbf{in}\langle 1 \rangle \sim \mathbf{in}\langle 2 \rangle$.

Let ρ_k be a complete execution in A with C iterated k times. Then for all $k \in \mathbb{N}$,

$$cost(\rho_k) \geq k((|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma_i|)d_i + (|\mathbf{in}\langle 1 \rangle_i - \mathbf{in}\langle 2 \rangle_i| + |\gamma'_i|)d'_i),$$

so for all $M \in \mathbb{R}$, $\exists \rho_k$ such that $cost(\rho_k) > M$. \square

move these two lemmas away from this section (appendix or closer to usage in main theorem)

Lemma 4.47. *If a coupling strategy $C = (\gamma, \gamma')$ for a lasso L_ρ is valid and has finite cost, then the following must hold for all i :*

1. *If t_i is in a cycle and $c_i = \text{insample} < \mathbf{x}$, then $\gamma_i = -\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle$ and $\gamma_{at(i)} = 1$.*
2. *If t_i is in a cycle and $c_i = \text{insample} \geq \mathbf{x}$, then $\gamma_i = -\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle$ and $\gamma_{at(i)} = -1$.*

Proof. We will show (1). (2) follows symmetrically.

Consider some t_i in a cycle where $c_i = \text{insample} < \mathbf{x}$. Because C has finite cost, we know from lemma 4.46 that $\gamma_i = -\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle$ for all $\mathbf{in}_i\langle 1 \rangle \sim \mathbf{in}_i\langle 2 \rangle$. In particular, when $-\mathbf{in}_i\langle 1 \rangle + \mathbf{in}_i\langle 2 \rangle = 1$, then $\gamma_i = 1$.

Further, because $\gamma_{at(i)}$ must be greater than γ_i for all $\mathbf{in}_i\langle 1 \rangle \sim \mathbf{in}_i\langle 2 \rangle$ for C to be valid, we must have that $\gamma_{at(i)} = 1$. \square

Lemma 4.48. *If a valid finite cost coupling strategy $C = (\gamma, \gamma')$ exists for a lasso L_ρ , then there exists a valid finite cost coupling strategy $C^* = (\gamma^*, \gamma'^*)$ such that for all $i \in AT(L_\rho)$, $\gamma_i^* \in \{-1, 0, 1\}$.*

Proof. Note that because C is valid and has finite cost, there cannot be any assignment transitions in the cycle in ρ .

Further, there cannot be both a transition with guard $\text{insample} < \mathbf{x}$ and guard $\text{insample} \geq \mathbf{x}$ in the cycle in ρ by the previous lemma.

Let $t_{at(k)}$ be the final assignment transition before the cycle in ρ .

By the previous lemma, $\gamma_{at(k)} \in \{-1, 1\}$ already.

- every previous assignment transition can either be changed without caring about $\gamma_{at(k)}$ or

\square

4.6 Programs

Definition 4.49. Given a finite state space Q , a program A is a set of complete executions ρ such that for all $\rho = q_0 \rightarrow \dots \rightarrow q_n, \rho' = q'_0 \rightarrow \dots \rightarrow q'_m \in B$, $q_0 = q'_0$ and A satisfies the following closure property:

- If a complete execution $\rho \in A$ such that $\rho = q_0 \rightarrow \dots \rightarrow q_i \rightarrow \dots \rightarrow q_j \rightarrow q_i \rightarrow \dots \rightarrow q_n$, then, for all $k \in \mathbb{N}$, the execution $\rho_k \in A$, where $\rho_k = q_0 \rightarrow \dots \rightarrow (q_i \rightarrow \dots \rightarrow q_j)^k \rightarrow q_i \rightarrow \dots \rightarrow q_n$.

Additionally, A must satisfy the properties of **determinism** and **output distinction**, as with branching programs.

Intuitively, this closure condition means that if a execution ρ with the cycle $q_i \rightarrow \dots \rightarrow q_j \rightarrow q_i$ is in A , then every execution that is identical to ρ except with the cycle iterated a different number of times must be in A as well.

Definition 4.50 (DP). A program A is $d\varepsilon$ -differentially private for some $d > 0$ if for all adjacent input sequences $\text{in}\langle 1 \rangle \sim \text{in}\langle 2 \rangle$ and for all possible outputs σ of A , $\mathbb{P}[\sigma, \text{in}\langle 1 \rangle] \leq e^{d\varepsilon} \mathbb{P}[\sigma, \text{in}\langle 2 \rangle]$.

As with branching programs, the conditions of determinism and output distinction mean that knowing the output of a program means that we know what execution the program must have taken.

Proposition 4.51. *For every complete execution ρ of a program A , ρ is $d\varepsilon$ -differentially private if and only if A is differentially private.*

As expected, then, we can use coupling strategies for executions to prove privacy.

Proposition 4.52. *If, for every complete execution ρ of a program A , there is a coupling strategy C_ρ such that $\text{cost}(C_\rho) < d$, then A is $d\varepsilon$ -differentially private.*

However, this naive approach is not actually tractable, since there are possibly an unbounded number of executions in a single program A .

What we will show instead is that, inspired by our work with lassos, we can choose a coupling strategy for each of a finite number of execution classes of A , which will induce coupling strategies for every execution of A .

ALTERNATIVE DEFINITION FROM SUBGRAPHS

Definition 4.53. For an execution ρ of a program A , let $\text{graph}(\rho)$ be the subgraph of the underlying graph of A that is traversed by ρ . That is, if the underlying graph of A is $G = (V, E)$, then $\text{graph}(\rho) = (V', E')$, where $V' \subseteq V$, $E' \subseteq E$, and for all transitions $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$, $q_i, q_{i+1} \in V$ and $(q_i, q_{i+1}) \in E$.

Note that the relation between complete executions \sim_G where $\rho \sim_G \rho'$ if and only if $\text{graph}(\rho) = \text{graph}(\rho')$ is clearly an equivalence relationship; let $[\rho]_G$ be the equivalence class of ρ defined by this relationship for any complete execution ρ of a program A .

Definition 4.54 (Class Coupling Strategies). Consider an execution class $[\rho]_G$ of a program A and let $G = (V, E) = \text{graph}(\rho)$ be the underlying subgraph of $[\rho]_G$. A [name tentative] class coupling strategy $C_{[\rho]_G}$ for $[\rho]_G$ is a function $C_{[\rho]_G} : E \rightarrow (\mathbb{R} \times \mathbb{R} \rightarrow [-1, 1]) \times (\mathbb{R} \times \mathbb{R} \rightarrow [-1, 1])$ mapping edges of G to shift functions (γ, γ') .

Observe that there are a finite possible number of execution classes of a program A since A must have a finite number of states; additionally, a class coupling strategy for an execution class $[\rho]_G$ directly induces execution coupling strategies for every complete execution in the class.

Definition 4.55 (Execution Descriptions). Given a complete execution $\rho = q_0 \rightarrow \dots \rightarrow q_n$ of a program A , let the description $\text{desc}(\rho)$ of ρ be the cycle-free substring of ρ . still don't

like this definition

Note that the relation between complete executions \sim where $\rho \sim \rho'$ if and only if $\text{desc}(\rho) = \text{desc}(\rho')$ is clearly an equivalence relationship; let $[\rho]$ be the equivalence class of ρ defined by this relationship for any complete execution ρ of a program A .

Definition 4.56 (Execution Class). An execution class $[\rho]$ of a program A is the equivalence class of an execution ρ of A under the aforementioned equivalence relation.

Observation 4.57. *There are a finite number of execution classes of a program A because there are a finite number of simple paths through any finite graph.*

Definition 4.58 (Class Coupling Strategies). Given an execution class $[\rho]$ of a program A , let T be the set of all transitions of any path in $[\rho]$. Note that T is finite. Then a (class) coupling strategy for $[\rho]$ is a pair of functions $C = (\gamma, \gamma')$ such that $\gamma : T \rightarrow (\mathbb{R} \times \mathbb{R} \rightarrow [-1, 1])$ and $\gamma' : (\mathbb{R} \times \mathbb{R} \rightarrow [-1, 1])$ are two shift functions that map each transition in $[\rho]$ to a shift in $[-1, 1]$.

Definition 4.59 (Induced Coupling Strategy). not sure how necessary this is Given a class coupling strategy $C = (\gamma, \gamma')$ for an execution class $[\rho]$ and a specific execution $\rho \in [\rho]$, the coupling strategy for $\rho = q_0 \rightarrow \dots \rightarrow q_n$ induced by C is the pair of functions $\gamma_\rho, \gamma'_\rho$ such that

$$\begin{aligned}\gamma_\rho(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle) &= (\gamma(q_0 \rightarrow q_1)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle), \gamma(q_1 \rightarrow q_2)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle), \dots, \gamma(q_{n-1} \rightarrow q_n)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle)) \\ \gamma'_\rho(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle) &= (\gamma'(q_0 \rightarrow q_1)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle), \gamma'(q_1 \rightarrow q_2)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle), \dots, \gamma'(q_{n-1} \rightarrow q_n)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle))\end{aligned}$$

Definition 4.60. The cost of a class coupling strategy C for an execution class $[\rho]$ is $\text{cost}(C) = \sup_{\rho \in [\rho]} \text{cost}(C_\rho)$, where C_ρ is the execution coupling strategy for ρ' induced by C .

Proposition 4.61. *If, for every complete execution class $[\rho]$ of a program A , there is a coupling strategy $C_{[\rho]}$ such that $\text{cost}(C_{[\rho]}) < d$, then A is $d\varepsilon$ -differentially private.*

As with lassos, we primarily care about making sure that cycle transitions are costless.

Proposition 4.62. *A valid class coupling strategy $C = (\gamma, \gamma')$ for an execution class $[\rho]$ has finite coupling cost if and only if the following constraint holds for all i :*

- *If t_i is in a cycle in $[\rho]$ and $c_i \neq \text{true}$, then $\gamma_i = -\text{in}\langle 1 \rangle_i + \text{in}\langle 2 \rangle_i$ and $\gamma'_i = -\text{in}\langle 1 \rangle_i + \text{in}\langle 2 \rangle_i$.*

Indeed, collapsing the infinite collection of executions in a program to the finite set of execution classes does not cost us anything.

Proposition 4.63. *If there exists a valid coupling strategy C_ρ with cost $\text{cost}(C_\rho)$ for every execution ρ of an execution class $[\rho]$ in a program A and $\sup_{\rho \in [\rho]} \text{cost}(C_\rho) < \infty$, then there exists a class coupling strategy C' for $[\rho]$ of A that is valid such that $\text{cost}(C') \leq \sup_{\rho \in [\rho]} \text{cost}(C_\rho)$.*

Proof. Make sure to put this proof after the DiPA counterexample proof

Because $\sup_{\rho \in [\rho]} \text{cost}(C_\rho) < \infty$, we can assume that there are no leaking cycles, disclosing cycles, leaking pairs, or privacy violating paths in $[\rho]$.

For a given execution ρ and a coupling strategy C_ρ , recall that we effectively assign each transition t_i in ρ the cost $\max_{\Delta \in \{-1, 0, 1\}} |\Delta - \gamma_i(\Delta)| + |\Delta' - \gamma'_i(\Delta)|$. For convenience, we will shorthand this quantity as $\delta(\rho, t_i) + \delta'(\rho, t_i)$.

For all $n \in \mathbb{N}$, let ρ_n be the path in $[\rho]$ with every cycle in ρ_n repeated n times.

Let $\text{cycle}([\rho])$ be the set of all transitions in $[\rho]$ that are contained within a cycle in $[\rho]$. Observe that for all $t \in \text{cycle}([\rho])$,

$$\lim_{n \rightarrow \infty} \inf_{t_i \in \rho_n: t_i = t} \delta(\rho_n, t_i) = 0$$

Informally, for every cycle transition t in $[\rho]$, if the cycle it is contained in is iterated enough times, there must be some iteration t_i of t that is assigned costs approaching 0.

This can be shown by considering a transition t in a cycle in $[\rho]$ whose minimum coupling cost is non-zero (i.e. $\inf_{\rho \in [\rho]: t_i = t} \delta(t_i) > 0$). Then for any finite $d > 0$, there exists an execution ρ_n where $n > \lceil \frac{d}{\inf \delta(t_i)} \rceil + 1$. Then $\text{cost}(C_{\rho^*}) > d$, which implies that $\sup_{\rho \in [\rho]} \text{cost}(C_\rho) = \infty$, so the observation must hold.

Let t_i be a transition in a cycle in $[\rho]$ and let \mathcal{C}_i be the cycle containing t_i .

Then in particular, if \mathcal{C}_i contains a transition with guard **insample** $< \mathbf{x}$, then for all $\psi > 0$, there exists $n \in \mathbb{N}$ such that for $\rho_n \in [\rho]$, $\gamma_{at(i)} > 1 - \psi$ and if \mathcal{C}_i contains a transition with guard **insample** $< \mathbf{x}$, then for all $\psi > 0$, $\gamma_{at(i)} > -1 + \psi$. Informally, assignment transitions before an L-cycle have shifts that approach 1 and assignment transitions before a G-cycle have shifts that approach -1 in ρ_n as $n \rightarrow \infty$.

Because we know that all coupling strategies C_ρ are valid, this may also imply that other assignment transitions also have shifts that approach 1 or -1.

Further, if the shifts for an assignment transition t_i approach 1, then the shifts for a transition t_j such that $at(j) = i$ and $c_j = \text{insample} \geq \mathbf{x}$ must also approach 1; symmetrically, if the shifts for an assignment transition t_i approach -1, then the shifts for a transition t_j such that $at(j) = i$ and $c_j = \text{insample} < \mathbf{x}$ must also approach -1.

Let T_1 and T_{-1} be the sets of assignment transitions in $[\rho]$ that approach 1 and -1, respectively.

Note that every other transition in $[\rho]$ is a non-cycle transition. Consider such a transition t in $[\rho]$. Then for every execution $\rho \in [\rho]$ and its corresponding coupling strategy C_ρ , there is exactly one shift assignment for t because t is not in a cycle.

Let the class coupling strategy $C' = (\gamma, \gamma')$ be partially defined as follows:

$$\begin{aligned}\gamma(t_i)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle) &= \begin{cases} 1 & t_i \in T_1 \\ -1 & t_i \in T_{-1} \\ \text{in}\langle 1 \rangle - \text{in}\langle 2 \rangle & c_i = \text{insample} < \mathbf{x} \wedge t_{at(i)} \in T_1 \\ \text{in}\langle 1 \rangle - \text{in}\langle 2 \rangle & c_i = \text{insample} \geq \mathbf{x} \wedge t_{at(i)} \in T_{-1} \\ 1 & c_i = \text{insample} \geq \mathbf{x} \wedge t_{at(i)} \in T_1 \\ -1 & c_i = \text{insample} < \mathbf{x} \wedge t_{at(i)} \in T_{-1} \end{cases} \\ \gamma'(t_i)(\text{in}\langle 1 \rangle, \text{in}\langle 2 \rangle) &= \begin{cases} 0 & t_i \text{ outputs } \text{insample}' \\ \text{in}\langle 1 \rangle - \text{in}\langle 2 \rangle & \text{otherwise} \end{cases}\end{aligned}$$

Let T_{un} be the set of transitions in $[\rho]$ that are not assigned by γ so far. Note that all transitions in T_{un} are not in cycles.

Let $C^* = (\gamma^*, \gamma'^*)$ be the minimal-cost valid class coupling strategy such that for all $t \notin T_{un}$, $\gamma^*(t) = \gamma(t)$.

In other words, $\text{cost}(C^*) = \inf_{\text{all such possible valid class coupling strategies } C} \text{cost}(C)$. Note that C^* is valid.

We additionally claim that $\text{cost}(C^*) \leq \sup_{\rho \in [\rho]} \text{cost}(C_\rho)$. The cost of C^* can be separated into costs attributed to γ'^* , costs attributed to all transitions not in T_{un} by γ^* , and costs attributed to all transitions in T_{un} by γ^* .

First, note that the coupling cost attributed to γ'^* in C^* must be at most the maximum coupling cost attributed to γ' over all execution-specific coupling strategies. From before, we additionally know that the cost attributed to all transitions $t \notin T_{un}$ by γ^* is at most the supremum of the costs attributed to t over all executions in $[\rho]$, since we take the limit of all such shifts for ρ_n as $n \rightarrow \infty$.

Finally, since all execution-specific coupling strategies are valid, taking the remaining transition shifts to minimize the overall cost while retaining a valid coupling strategy suffices.

If i have time, come back to this argument - expressed poorly right now □

Perhaps surprisingly, we prove that coupling proofs are **complete** for programs of this form; while producing a valid and finite cost coupling proof is clearly sufficient for a program to be private, it is also necessary for such a coupling proof to exist for privacy.

Theorem 4.64. *There exists a valid and finite cost coupling strategy for a program P if and only if P is $d\varepsilon$ -differentially private for some $d > 0$.*

Lemma 4.65. *If there exists a valid coupling strategy with cost $d > 0$ for a program P , then P is $d\varepsilon$ -differentially private.*

4.7 DiPA

We now discuss a previously defined program model, DiPA, which is perhaps surprisingly exactly equivalent to our defined model.

Definition 4.66 ([6]). A Differential Private Automaton (DiPA) A is a 7-tuple $(Q, \Sigma, C, \Gamma, q_{init}, X, P, \delta)$ where

- Q is a finite set of states partitioned into input states Q_{in} and non-input states Q_{non} .
- $\Sigma = \mathbb{R}$ is the input alphabet
- $C = \{\mathbf{true}, \mathbf{insample} < x, \mathbf{insample} \geq x\}$ is a set of guard conditions
- Γ is a finite output alphabet
- $q_{init} \in Q$ is the initial state
- $X = \{\mathbf{x}, \mathbf{insample}, \mathbf{insample}'\}$ is a set of variables
- $P : Q \rightarrow \mathbb{Q} \times \mathbb{Q}^{\geq 0} \times \mathbb{Q} \times \mathbb{Q}^{\geq 0}$ is a parameter function that assigns sampling parameters for the Laplace distribution for each state
- $\delta : (Q \times C) \rightarrow (Q \times (\Gamma \cup \{\mathbf{insample}, \mathbf{insample}'\}) \times \{\mathbf{true}, \mathbf{false}\})$ is a partial transition function.

In addition, δ must satisfy some additional conditions:

- **Determinism:** For any state $q \in Q$, if $\delta(q, \mathbf{true})$ is defined, then $\delta(q, \mathbf{insample} < x)$ and $\delta(q, \mathbf{insample} \geq x)$ are not defined.
- **Output Distinction:** For any state $q \in Q$, if $\delta(q, \mathbf{insample} \geq x) = (q_1, o_1, b_1)$ and $\delta(q, \mathbf{insample} < x) = (q_2, o_2, b_2)$, then $o_1 \neq o_2$ and at least one of $o_1 \in \Gamma$ and $o_2 \in \Gamma$ is true.
- **Initialization:** The initial state q_0 has only one outgoing transition of the form $\delta(q_0, \mathbf{true}) = (q, o, \mathbf{true})$.
- **Non-input transition:** From any $q \in Q_{non}$, if $\delta(q, c)$ is defined, then $c = \mathbf{true}$.

Definition 4.67. A DiPA A is $d\varepsilon$ -differentially private if [copy this def]

The privacy of a DiPA can be characterized by the presenece of certain graph-theoretic structures

Proposition 4.68 (informal). *There is a bijection between the set S_P of all possible programs P and the set of all possible DiPAs S_D . [that preserves coupling proofs etc]*

4.7.1 Deciding Privacy

Definition 4.69 (Leaking Pair, etc from DiPA). [...]

Theorem 4.70 (from DiPA). *If a DiPA A has a leaking cycle, leaking pair, disclosing cycle, or privacy violating path, then A is not $d\varepsilon$ -differentially private for any $d > 0$.*

Note that for any path in A , we have a list of constraints that a coupling strategy $C = (\gamma, \gamma')$ can satisfy

1. If $c_i = \text{insample} < \mathbf{x}$, then $\gamma_i \leq \gamma_{at(i)}$
2. If $c_i = \text{insample} \geq \mathbf{x}$, then $\gamma_i \geq \gamma_{at(i)}$
3. If $\sigma_i = \text{insample}$, then $\gamma_i = 0$
4. If $\sigma_i = \text{insample}'$, then $\gamma'_i = 0$
5. If t_i is in a cycle, then $\gamma_i = -\text{in}\langle 1 \rangle_i + \text{in}\langle 2 \rangle_i$
6. If t_i is in a cycle, then $\gamma'_i = -\text{in}\langle 1 \rangle_i + \text{in}\langle 2 \rangle_i$

If all constraints are satisfied, then A is private.

Lemma 4.71. *If there does not exist a valid coupling strategy for a DiPA A with finite cost, then A is not differentially private.*

Proof. Consider a “maximially” satisfied coupling strategy $C = (\gamma, \gamma')$; i.e. there is no other coupling strategy C' for A such that C' satisfies more constraints than C . By lemma [tbd], we are allowed to only consider coupling strategies $C = (\gamma, \gamma')$ such that, for all $i \in AT(A)$, $\gamma_i \in \{-1, 0, 1\}$.

Fix some execution class ρ in A such that at least one constraint is not satisfied by C as applied to ρ .

By assumption, at least one constraint is unsatisfied by C . We will show that in every case, A must contain at least one of a leaking cycle, leaking pair, disclosing cycle, or privacy violating path. By theorem 4.70, this is sufficient to show that A is not $d\varepsilon$ -differentially private for any $d > 0$.

If I have time (low priority): rewrite this using a few helper lemmas to compress (e.g. $\gamma_{at(i)} = 1 \implies \text{L-cycle}$)

Case 1: (1) is unsatisfied for γ_i

In this case, $c_i = \text{insample} < \mathbf{x}$ and $\gamma_i > \gamma_{at(i)}$. Note that $\gamma_{at(i)} \neq 1$.

We can assume that for all assignment transitions $t_{at(k)}$ in ρ that $t_{at(k)}$ is not in a cycle, since otherwise there would be a leaking cycle in A .

Case 1.1: t_i is in a cycle

In this case, we can suppose that t_i is not an assignment transition and t_i does not output `insample` or `insample'`, since otherwise either a leaking cycle or a disclosing cycle would clearly exist in A . We can thus additionally assume that constraint (5) is satisfied for γ_i .

Note that the cycle containing t_i is also an L-cycle by definition.

Then attempting to resolve (1) for γ_i by setting $\gamma_{at(i)} = 1$ must violate another constraint. In particular, either constraint (1) or (3) for $\gamma_{at(i)}$ or constraint (2) for some γ_j such that

$at(j) = at(i)$ must be newly violated. Note that constraint (5) for $\gamma_{at(i)}$ cannot be violated since we assumed that $t_{at(i)}$ is not in a cycle.

Case 1.1.1: setting $\gamma_{at(i)} = 1$ violates constraint (1) for $\gamma_{at(i)}$

Let $t_{at(k)}$ be the earliest assignment transition before $t_{at(i)}$ such that, for all $at(k) \leq at(l) < at(i)$, $\gamma_{at(l)} < 1$ and $c_{at(l)} = \text{insample} < \mathbf{x}$. Then there must be *some* $\gamma_{at(l)}$ such that setting $\gamma_{at(l)} = 1$ would violate constraint (2) for some $\gamma_{l'}$ such that $at(l') = at(l)$.

Observe that $c_{l'} = \text{insample} \geq \mathbf{x}$ and there is an AL-path from $t_{l'}$ to t_i .

Then setting $\gamma_{l'} = 1$ must violate either constraint (3) or constraint (5) for $\gamma_{l'}$. If constraint (3) is violated, then $\gamma_{l'}$ is a transition with guard $\text{insample} \geq \mathbf{x}$ that outputs insample , so there is a privacy violating path from $t_{l'}$ to t_i . Otherwise if constraint (5) is violated, then $\gamma_{l'}$ is in a G-cycle, so there is a leaking pair composed of the cycles containing $t_{l'}$ and t_i , respectively.

Case 1.1.2: Setting $\gamma_{at(i)} = 1$ would violate (3) for $\gamma_{at(i)}$

Then $\gamma_{at(i)}$ is an assignment transition that outputs insample . Further, the path from $t_{at(i)}$ to t_i is an AL-path, since there are no transitions on it. Thus, there is a privacy violating path from $t_{at(i)}$ to t_i .

Case 1.1.3: Setting $\gamma_{at(i)} = 1$ would violate (2) for some γ_j such that $at(j) = at(i)$

Note that, if $i < j$, the path from t_i to t_j (or vice versa, if $j < i$) is both an AL and AG-path.

Setting $\gamma_j = 1$ must violate either constraint (3) or constraint (5) for γ_j .

If constraint (3) is violated, then γ_j is a transition with guard $\text{insample} \geq \mathbf{x}$ that outputs insample . Thus if $i < j$, there is a privacy violating path from t_i to t_j and if $j < i$, there is a privacy violating path from t_j to t_i .

Otherwise if constraint (5) is violated, then γ_j is in a G-cycle, so there is a leaking pair composed of the cycle containing t_j and the cycle containing t_i if $j < i$ or vice versa if $j > i$.

Case 1.2: t_i is not in a cycle

Note that t_i must either be an assignment transition or output insample or both, since otherwise, setting $\gamma_i = \gamma_{at(i)}$ would resolve constraint (1) for γ_i without violating any other constraint.

Case 1.2.1: t_i outputs insample and t_i is an assignment transition

In this case, attempting to resolve constraint (1) without violating constraint (3) for γ_i by setting $\gamma_i = \gamma_{at(i)} = 0$ must violate some other constraint. In particular, setting $\gamma_{at(i)} = 0$ can newly violate constraint (1) for $\gamma_{at(i)}$ or constraint (2) for some γ_j such that $at(j) = at(i)$; note that setting $\gamma_{at(i)} = 0$ cannot *newly* violate constraint (1) for some γ_j such that $at(j) = at(i)$. Alternatively, setting $\gamma_i = 0$ could potentially newly violate either constraint (1) or constraint (2) for some γ_j such that $at(j) = i$.

Case 1.2.2.1: Setting $\gamma_{at(i)} = 0$ violates constraint (1) for $\gamma_{at(i)}$

Let $t_{at(k)}$ be the earliest assignment transition before $t_{at(i)}$ such that, for all $at(k) \leq at(l) < at(i)$, $\gamma_{at(l)} = -1$ and $c_{at(l)} = \text{insample} < \mathbf{x}$. Then there must be *some* $\gamma_{at(l)}$ such that setting $\gamma_{at(l)} = 0$ would violate constraint (2) for some $\gamma_{l'}$ such that $at(l') = at(l)$. Additionally, note that setting $\gamma_{l'} = \gamma_{at(l)} = 0$ can only violate constraint (5) for $\gamma_{l'}$, since $\gamma_{l'}$ cannot be an assignment transition.

Thus, $t_{l'}$ is in a cycle, so the cycle containing $t_{l'}$ is a G-cycle. Note that the path from $t_{l'}$ to t_i is an AL-path. Therefore, there is a privacy violating path from $t_{l'}$ to t_i .

Case 1.2.2.2: Setting $\gamma_{at(i)} = 0$ violates constraint (2) for some γ_j such that $at(j) = at(i)$

Note that $j \neq i$, meaning that t_j is not an assignment transition. Then setting $\gamma_j = \gamma_{at(i)} = 0$ must violate constraint (5) for γ_j ; this means that t_j is in a G-cycle.

If $i < j$, then the path from t_i to t_j is an AL-path, so it is also a privacy violating path.

Otherwise illustratef $j < i$, then the path from t_j to t_i is an AG path, so it is also a privacy violating path.

Case 1.2.2.3: Setting $\gamma_i = 0$ violates constraint (1) for some γ_j such that $at(j) = i$

If γ_j is not an assignment transition, then setting $\gamma_j = \gamma_i = 0$ must violate constraint (5) for γ_j , so t_j is in an L-cycle. Then there is a privacy violating path from t_i to t_j , since the path from t_{i+1} to t_j is an AL-path by virtue of not containing any assignment transitions.

Otherwise if t_j is an assignment transition, then γ_j must originally be set to 1. Let $t_{at(k)}$ be the latest assignment after t_i such that, for all $i \leq at(l) < at(k)$, $\gamma_{at(l)} = 1$ and $c_{at(l)} = \text{insample} < \mathbf{x}$. Then there must be *some* $\gamma_{at(l)}$ such that setting $\gamma_{at(l)} = 0$ would violate constraint (1) for some $\gamma_{l'}$ such that $at(l') = at(l)$. Additionally, note that setting $\gamma_{l'} = \gamma_{at(l)} = 0$ can only violate constraint (5) for $\gamma_{l'}$, since $\gamma_{l'}$ cannot be an assignment transition.

Then $\gamma_{l'}$ must be in an L-cycle. Since the path from t_i to $t_{l'}$ is an AL-path, there is a privacy violating path from t_i to $t_{l'}$.

Case 1.2.2.4: Setting $\gamma_i = 0$ violates constraint (2) for some γ_j such that $at(j) = i$

This case is exactly symmetric to case 1.2.2.3.

Case 1.2.2: t_i outputs insample and t_i is not an assignment transition

We can assume that $\gamma_{at(i)} = -1$ originally, since otherwise, setting $\gamma_i = 0$ would resolve constraint (1) without violating any additional ones.

Thus attempting to resolve constraint (1) while preserving constraint (3) for γ_i by setting $\gamma_{at(i)} = \gamma_i = 0$ must violate constraint (1) for $\gamma_{at(i)}$.

Let $t_{at(k)}$ be the earliest assignment transition before $t_{at(i)}$ such that, for all $at(k) \leq at(l) < at(i)$, $\gamma_{at(l)} = -1$ and $c_{at(l)} = \text{insample} < \mathbf{x}$. Then there must be *some* $\gamma_{at(l)}$ such that setting $\gamma_{at(l)} = 0$ would violate constraint (2) for some $\gamma_{l'}$ such that $at(l') = at(l)$. Additionally, note that setting $\gamma_{l'} = \gamma_{at(l)} = 0$ can only violate constraint (5) for $\gamma_{l'}$, since $\gamma_{l'}$ cannot be an assignment transition.

Thus, $t_{l'}$ is in a cycle, so the cycle containing $t_{l'}$ is a **G**-cycle. Note that the path from $t_{l'}$ to t_i is an **AL**-path. Therefore, there is a privacy violating path from $t_{l'}$ to t_i .

Case 1.2.3: t_i does not output insample and t_i is an assignment transition

In this case, attempting to resolve (1) by setting $\gamma_{at(i)} = 1$ must violate either constraint (1) or (3) for $\gamma_{at(i)}$, or constraint (2) for some γ_j such that $at(j) = at(i)$.

Additionally, note that $\gamma_{at(i)} \in \{0, -1\}$.

Case 1.2.3.1: $\gamma_{at(i)} = 0$

Since originally, $\gamma_i > \gamma_{at(i)} \implies \gamma_i = 1$, we know that setting $\gamma_i = \gamma_{at(i)} = 0$ must violate constraint (1) for some γ_j such that $at(j) = i$. If t_j is not an assignment transition, then setting $\gamma_j = 0$ can only violate constraint (5) for γ_j , meaning that t_j is in an **L**-cycle.

Otherwise, if t_j is an assignment transition, let $t_{at(k)}$ be the latest assignment transition after t_i such that for all $j \leq at(l) < at(k)$, $\gamma_{at(l)} = 1$ and $c_{at(k)} = \text{insample} < \mathbf{x}$. Then there must exist some $at(l), j \leq at(l) < at(k)$ such that setting $\gamma_{at(l)} = 0$ would violate constraint (1) for some non-assignment $\gamma_{l'}$ where $at(l') = at(l)$.

Further, setting $\gamma_{l'} = 0$ must then violate constraint (5) for $\gamma_{l'}$, so $t_{l'}$ is in an **L**-cycle.

Therefore, there exists a **AL**-path from t_i to some transition t in an **L**-cycle.

Case 1.2.3.1.1: Setting $\gamma_{at(i)} = 1$ would violate constraint (1) for $\gamma_{at(i)}$

Let $t_{at(j)}$ be the earliest assignment transition before $t_{at(i)}$ such that for all $at(j) \leq at(k) < at(i)$, $\gamma_{at(k)} = 0$ and $c_{at(k)} = \text{insample} < \mathbf{x}$. Then there must exist some $at(k), at(j) \leq at(k) < at(i)$ such that setting $\gamma_{at(k)} = 1$ would violate constraint (2) for some non-assignment γ_l where $at(l) = at(k)$, so $c_l = \text{insample} \geq \mathbf{x}$

Note that there is an **AL**-path from t_l to t_i , and therefore an **AL**-path from t_l to some transition t_o in an **L**-cycle.

Further, setting $\gamma_l = \gamma_{at(k)} = 1$ must then violate either constraint (3) or (5) for γ_l .

If constraint (3) is violated, then t_l outputs **insample**, so there is a privacy violating from t_l to t_o .

If constraint (5) is violated, then t_l is in a **G**-cycle, so there is a leaking pair consisting of the cycle containing t_l and the cycle containing t_o .

Case 1.2.3.1.2: Setting $\gamma_{at(i)} = 1$ would violate constraint (3) for $\gamma_{at(i)}$

Note that there is an **AL** path from $t_{at(i)}$ to some transition t_j such that t_j is in an **L**-cycle.

Then $t_{at(i)}$ is an assignment transition that outputs **insample**, so there is a privacy violating path from $t_{at(i)}$ to t_j .

Case 1.2.3.1.3: Setting $\gamma_{at(i)} = 1$ would violate constraint (2) for some γ_j such that $at(j) = at(i)$

As before, note that there is an **AL** path from t_j to some transition t_k such that t_k is in an **L-cycle**.

Then trying to set $\gamma_j = \gamma_{at(i)} = 1$ must violate either constraint (3) or constraint (5) for γ_j . If constraint (3) is violated, then t_j outputs **insample**, so there is a privacy violating from t_j to t_k . If constraint (5) is violated, then t_j is in a **G-cycle**, so there is a leaking pair consisting of the cycle containing t_j and the cycle containing t_k .

Case 1.2.3.2: $\gamma_{at(i)} = -1$

Note that $\gamma_i \in \{0, 1\}$.

First, if $\gamma_i = 0$, then setting $\gamma_i = -1$ must newly violate constraint (1) for some γ_j where $at(j) = i$. If t_j is not an assignment transition, then setting $\gamma_j = -1$ can only newly violate constraint (3) for γ_j , meaning that t_j outputs **insample**.

Otherwise, if t_j is an assignment transition, let $t_{at(k)}$ be the latest assignment transition after t_i such that for all $j \leq at(l) < at(k)$, $\gamma_{at(l)} = 0$ and $c_{at(k)} = \text{insample} < \mathbf{x}$. Then there must exist some $at(l), j \leq at(l) < at(k)$ such that setting $\gamma_{at(l)} = -1$ would newly violate constraint (1) for some non-assignment $\gamma_{l'}$ where $at(l') = at(l)$; as before, this means that $t_{l'}$ outputs **insample**.

Otherwise, if $\gamma_i = 1$, then setting $\gamma_i = -1$ must newly violate constraint (1) for some γ_j where $at(j) = i$. If t_j is not an assignment transition, then setting $\gamma_j = -1$ can only newly violate constraint (5) for γ_j , meaning that t_j is in an **L-cycle**.

Otherwise, if t_j is an assignment transition, let $t_{at(k)}$ be the latest assignment transition after t_i such that for all $j \leq at(l) < at(k)$, $\gamma_{at(l)} = 0$ and $c_{at(k)} = \text{insample} < \mathbf{x}$. Then there must exist some $at(l), j \leq at(l) < at(k)$ such that setting $\gamma_{at(l)} = -1$ would newly violate constraint (1) for some non-assignment $\gamma_{l'}$ where $at(l') = at(l)$; as before, this means that $t_{l'}$ is in an **L-cycle**.

Thus, if $\gamma_i = 0$, then there is **AL**-path from t_i to some other transition that has guard **insample** $< \mathbf{x}$ and outputs **insample**. Otherwise, if $\gamma_i = 1$, there is an **AL**-path from t_i to some other transition that is in an **L-cycle**.

Case 1.2.3.2.1: Setting $\gamma_{at(i)} = \gamma_i$ would violate constraint (1) for $\gamma_{at(i)}$

Let $t_{at(j)}$ be the earliest assignment transition before $t_{at(i)}$ such that for all $at(j) \leq at(k) < at(i)$, $\gamma_{at(k)} = -1$ and $c_{at(k)} = \text{insample} < \mathbf{x}$. Then there must exist some $at(k), at(j) \leq at(k) < at(i)$ such that setting $\gamma_{at(k)} = \gamma_i$ would newly violate constraint (2) for some non-assignment γ_l where $at(l) = at(k)$.

First note that $c_l = \text{insample} \geq \mathbf{x}$ and there is an **AL**-path from t_l to t_i .

If $\gamma_i = 0$, then setting $\gamma_l = \gamma_{at(k)} = \gamma_i = 0$ can only newly violate constraint (5) for γ_l . Thus, γ_l is in a **G-cycle**. Since $\gamma_i = 0$, there exists some $t_{l'}$ such that there is an **AL**-path from t_i to $t_{l'}$ and $t_{l'}$ has guard **insample** $< \mathbf{x}$ and outputs **insample**. Thus, there is an **AL** path from t_l to $t_{l'}$, and so there is a privacy violating path from t_l to $t_{l'}$.

If $\gamma_i = 1$, then setting $\gamma_l = \gamma_{at(k)} = \gamma_i = 1$ can newly violate constraints (3) or (5) for γ_l .

Further, since $\gamma_i = 1$, there exists some $t_{l'}$ such that there is an AL-path from t_i to $t_{l'}$ and $t_{l'}$ is in an L-cycle. Thus, there is an AL path from t_l to $t_{l'}$.

If constraint (3) is newly violated, then t_l is a transition with guard `insample` $\geq x$ that outputs `insample`. Thus, there is a privacy violating path from t_l to $t_{l'}$.

If constraint (5) is newly violated, then t_l is in a G-cycle. Thus, there is a leaking pair composed of the cycles containing t_l and $t_{l'}$, respectively.

Case 1.2.3.2.2: Setting $\gamma_{at(i)} = \gamma_i$ would violate constraint (3) for $\gamma_{at(i)}$

First, $t_{at(i)}$ is an assignment transition that outputs `insample`. Since $\gamma_i = 1$, there exists some t_j such that there is an AL-path from $t_{at(i)}$ to t_j and t_j is in an L-cycle. Then there is a privacy violating path from $t_{at(i)}$ to t_j .

Case 1.2.3.2.3: Setting $\gamma_{at(i)} = \gamma_i$ would violate constraint (2) for some γ_j such that $at(j) = at(i)$

Observe that t_j is not an assignment transition and has guard `insample` $\geq x$. Additionally, there is an AL-path from t_j to t_i since there are no assignments between t_j and t_i .

If $\gamma_i = 0$, then setting $\gamma_j = \gamma_{at(i)} = 0$ can only newly violate constraint (5) for γ_j . Thus, γ_j is in a G-cycle. Since $\gamma_i = 0$, there exists some t_k such that there is an AL-path from t_i to t_k . Thus, there is an AL path from t_j to t_k , and so there is a leaking pair composed of the cycles containing t_j and t_k , respectively.

If $\gamma_i = 1$, then setting $\gamma_j = \gamma_{at(i)} = 1$ can newly violate constraints (3) or (5) for γ_l . Further, since $\gamma_i = 1$, there exists some t_k such that there is an AL-path from t_i to t_k and t_k is in an L-cycle. Thus, there is an AL path from t_j to t_k .

If constraint (3) is newly violated, then t_j is a transition with guard `insample` $\geq x$ that outputs `insample`. Thus, there is a privacy violating path from t_j to t_k .

If constraint (5) is newly violated, then t_j is in a G-cycle. Thus, there is a leaking pair composed of the cycles containing t_j and t_k , respectively.

Case 2: (2) is unsatisfied for γ_i

This case is exactly symmetric to case (1).

Case 3: (3) is unsatisfied for γ_i

First note that if t_i is in a cycle, then that cycle will be a disclosing cycle because t_i outputs `insample`. Thus, we will assume that t_i is not in a cycle.

Because C is maximal, setting $\gamma_i = 0$ must violate at least one of constraints (1) or (2) for γ_i or (1) for some γ_l such that $at(l) = i$.

Case 3.1: Satisfying (3) for γ_i would violate (1) for γ_i

This means that $\gamma_{at(i)} < 0 \implies \gamma_{at(i)} = -1$. Further, $c_i = \text{insample} < x$. Then changing $\gamma_{at(i)} = 0$ can newly violate constraints (1) or (5) for $\gamma_{at(i)}$ or constraint (2) for some γ_j such that $at(j) = at(i)$.

If constraint (5) is newly violated, then $t_{at(i)}$ is in a cycle. In particular, the cycle must be a leaking cycle; if t_i and $t_{at(i)}$ are both contained in a cycle, then it must be leaking because $c_i = \text{insample} < \mathbf{x}$. Otherwise, there still must be some transition in the cycle containing $t_{at(i)}$ that has a non-true guard since otherwise a path from $t_{at(i)}$ to t_i could not exist.

By similar reasoning, we can assume that for every assignment transition $t_{at(j)}$ before $t_{at(i)}$ on a complete path to t_i , $t_{at(j)}$ is not in a cycle.

If constraint (1) is newly violated for $\gamma_{at(i)}$, then $c_{at(i)} = \text{insample} < \mathbf{x}$. Let $t_{at(j)}$ be the earliest assignment transition before $t_{at(i)}$ such that $\gamma_{at(l)} = -1$ and for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and $t_{at(i)}$, $c_{at(k)} = \text{insample} < \mathbf{x}$ and $\gamma_{at(k)} = -1$.

Then there must exist some assignment transition $t_{at(k)}$, $at(j) \leq at(k) \leq at(i)$ between $t_{at(j)}$ and $t_{at(i)}$ such that setting $\gamma_{at(k)} = 0$ would newly violate constraint (2) for some l where $at(l) = at(k)$. In particular, this must be because t_l is in a cycle and setting $\gamma_l = 0$ would violate constraint (5). Thus, t_l is in a G-cycle. Then there is an AL-path from t_l to t_i , creating a privacy violating path from t_l to t_i .

If changing $\gamma_{at(i)}$ from -1 to 0 means that constraint (2) would be newly violated for some γ_j such that $at(j) = at(i)$, note that $\gamma_j < 0$ and $c_j = \text{insample} \geq \mathbf{x}$.

So setting $\gamma_j = 0$ can violate either (2) for some γ_l where $at(l) = j$ or (5) for γ_j .

If setting $\gamma_j = 0$ would violate constraint (2) for some γ_l where $at(l) = j$, then let $t_{at(m)}$ be the latest assignment transition after $t_{at(j)}$ such that $\gamma_{at(m)} = -1$ and for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and $t_{at(m)}$, $c_{at(k)} = \text{insample} \geq \mathbf{x}$ and $\gamma_{at(k)} = -1$.

Then there must exist some assignment transition $t_{at(k)}$, $at(j) \leq at(k) \leq at(m)$ between $t_{at(j)}$ and $t_{at(m)}$ such that setting $\gamma_{at(k)} = 0$ would newly violate constraint (2) for some l' where $at(l') = at(k)$. In particular, this must be because $t_{l'}$ is in a cycle and setting $\gamma_l = 0$ would violate constraint (5). Thus, t_l is in a G-cycle. Then there is an AG-path from t_i to t_l , creating a privacy violating path from t_i to t_l .

Otherwise, if setting $\gamma_j = 0$ would violate constraint (5) for γ_j , then t_j is in a G-cycle. We can assume that $j \neq i$ because otherwise, the cycle containing t_j would be a disclosing cycle. Additionally, note that there are no assignment transitions between t_i and t_j or vice versa, since $at(j) = at(i)$. Thus, if $j < i$, then there is an AL-path from t_j to t_i , which forms a privacy violating path. Symmetrically, if $i < j$, then there is an AG-path from t_i to t_j , which again forms a privacy violating path.

Case 3.2: Satisfying (3) for γ_i would violate (2) for γ_i

This case is exactly symmetric to case (3a).

Case 3.3: Satisfying (3) for γ_i would violate (1) for some γ_l where $at(l) = i$

Note that t_i must be an assignment transition. Further, we know that $\gamma_l > 0$ and $c_l = \text{insample} < \mathbf{x}$.

Because C is maximal, setting $\gamma_l = 0$ would now violate either constraint (1) for some $\gamma_{l'}$ where $at(l') = l$ or constraint (5) for γ_l . Note that because $\gamma_l > 0$, constraint (2) cannot be

newly violated for some $\gamma_{l'}$ where $at(l') = l$.

If constraint (5) would be newly violated for γ_l , then γ_l is in an L-cycle. Additionally, note that the path from t_{i+1} to t_l is an AL-path, so there is a privacy violating path from t_i to t_l .

Otherwise, if setting $\gamma_l = 0$ would violate constraint (1) for some $\gamma_{l'}$ where $at(l') = l$, let $t_{at(j)}$ be the latest assignment transition such that $c_{at(j)} = \text{insample} < \mathbf{x}$ and $\gamma_{at(j)} < 1$ and, for all assignment transitions $t_{at(k)}$ between t_l and $t_{at(j)}$, $c_{at(k)} = \text{insample} < \mathbf{x}$ and $\gamma_{at(k)} < 1$.

If $at(j) = l$, then l' is not an assignment transition. Then, setting $\gamma_{l'} = 0$ could only violate constraint (5). In this case, as before, there is a privacy violating path from t_i to t_l .

Otherwise, since C is maximal, we cannot set $\gamma_{at(k)} = 0$ for any $l < at(k) \leq at(j)$ without violating another constraint. In particular, there must be some $at(k)$ such that setting $\gamma_{at(k)} = 0$ would violate constraint (1) for some $\gamma_{k'}$ such that $at(k') = at(k)$. Note that there must be an AL-path from t_i to $t_{k'}$. Then, as before, there must be a privacy violating path from t_i to $t_{k'}$.

Case 3.4: Satisfying (3) for γ_i would violate (2) for some γ_l where $at(l) = i$

This case is exactly symmetric to case (3c).

Case 4: (4) is unsatisfied for γ'_i

Because C is maximal, setting $\gamma'_i = 0$ must violate some other constraint. In particular, this must mean that constraint (6) is now violated. However, this would imply that t_i is in a cycle, and so the cycle containing t_i would be a disclosing cycle.

Case 5: (5) is unsatisfied for t_i : Because C is maximal, we know that if $\gamma_i = -\text{in}_i\langle 1 \rangle + \text{in}_i\langle 2 \rangle$ then another constraint must be violated. In particular, at least one of constraints (1), (2), or (3) must be violated for γ_i .

Case 5.1: Satisfying (5) for t_i would violate (1)

If (1) is now violated, then either t_i is an assignment transition or $c_i = \text{insample} < \mathbf{x}$ and $\gamma_{at(i)} < 1$. If t_i is an assignment transition, then the cycle containing t_i has a transition with a non-true guard (t_i) and an assignment transition, so it must be a leaking cycle.

Otherwise, if t_i is not an assignment transition, $c_i = \text{insample} < \mathbf{x}$, and constraint (1) is violated for γ_i , we must have that $\gamma_{at(i)} < 1$ due to other constraints.

Consider all assignment transitions in ρ before t_i . Note that if any such assignment transition is in a cycle, then that cycle must be a leaking cycle since either the assignment transition is in the same cycle as t_i or there must be some non-true transition in the cycle because otherwise t_i is unreachable. **make sure to add a condition to programs so that this is bad**

So assume that all assignment transitions in ρ before t_i are not in a cycle. Then if $c_{at(i)} \neq \text{insample} < \mathbf{x}$, because C is maximal, this must mean that $t_{at(i)}$ outputs **insample**. Note that the path from $t_{at(i)+1}$ to t_i is an AL-path (since there are no assignment transitions on it) and t_i is in an L-cycle since t_i is in a cycle and $c_i = \text{insample} < \mathbf{x}$. Then the path from $t_{at(i)}$ (an assignment transition that outputs **insample**) to t_i is a privacy violating path.

If $c_{at(i)} = \text{insample} < \mathbf{x}$, then let $c_{at(j)}$ be the earliest assignment transition such that $c_{at(j)} = \text{insample} < \mathbf{x}$ and $\gamma_{at(j)} < 1$ and, for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and t_i , $c_{at(k)} = \text{insample} < \mathbf{x}$ and $\gamma_{at(k)} < 1$. Note that such an $t_{at(j)}$ must exist.

If $t_{at(j)} = t_{at(i)}$, then setting $\gamma_{at(i)} = 1$ must violate either constraint (2) for some other γ_l such that $at(l) = at(i)$, or constraint (3) for $\gamma_{at(i)}$. Without loss of generality, we will assume that $l \neq i$. If constraint (3) would be violated, then as before, there exists a privacy violating path from $t_{at(j)}$ to t_i . If constraint (2) would be violated for some γ_l such that $at(l) = at(i)$, then either t_l must output `insample` or t_l must be in a cycle.

Suppose that $i < l$; then that the path from t_i to t_l is both an **AG**-path and an **AL**-path (since there are no assignment transitions on it). Thus, if t_l outputs `insample`, there exists a privacy violating path from t_i to t_l and if t_l is in a cycle, then the cycle containing t_i and the cycle containing t_l together make up a leaking pair, since the cycle containing t_l is a **G**-cycle by definition. Symmetrically, if $l > i$, then either the path from t_l to t_i is a privacy violating path or the cycle containing t_l and the cycle containing t_i make up a leaking pair.

Otherwise, note that the path from $t_{at(j)}$ to t_i is an **AL**-path. Since C is maximal, we cannot set $\gamma_{at(k)} = 1$ for $\gamma_{at(j)}$ or for any of the other assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and t_i without violating another constraint. In particular, there must be some $t_{at(k)}$ where $at(j) \leq at(k) < i$ such that setting $\gamma_{at(k)} = 1$ would mean that either constraint (2) for some γ_l such that $at(l) = at(k)$ or constraint (3) would be violated for $\gamma_{at(k)}$. If constraint (3) would be violated for $\gamma_{at(k)}$ then $t_{at(k)}$ outputs `insample`, so as before, there is a privacy violating path from $t_{at(k)}$ to t_i . Otherwise if constraint (2) would be violated for some γ_l such that $at(l) = at(k)$, then as before, γ_l must either output `insample` or t_l is in a cycle. Just like before, this means that there must be either a privacy violating path from t_l to t_i or the cycle containing t_l and the cycle containing t_i together make up a leaking pair.

Case 5.2: Satisfying (5) for t_i would violate (2)

This case is exactly symmetric to case (5a).

Case 5.3: Satisfying (5) for t_i would violate (3)

If (3) would be violated, then t_i must output `insample`. Then the cycle containing t_i must be a disclosing cycle.

Case 6: (6) is unsatisfied for t_i : Because C is maximal, we know that if $\gamma'_i = -\text{in}_i\langle 1 \rangle + \text{in}_i\langle 2 \rangle$ then another constraint must be violated for γ'_i . In particular, constraint (4) must be violated, since no other constraint involves γ'_i . Then t_i is a transition in a cycle that outputs `insample'`, so A has a disclosing cycle. \square

4.8 An algorithm for deciding privacy

Observe that the constraints imposed on valid coupling strategies for a complete path ρ only depend on the shifts associated with *assignment transitions* in ρ .

In particular, this lends itself to conceptualizing complete paths by splitting them up based on assignment transitions.

Definition 4.72. A **segment** of a (complete) path $\rho = q_0 \rightarrow \dots \rightarrow q_n$ is a subpath $q_i \rightarrow q_{i+1} \rightarrow \dots \rightarrow q_j$ of ρ such that $t_i \in A_\rho$; for all $i < k < j$, $t_k \notin A_\rho$; and either $j = n$ or $t_j \in A_\rho$.

In other words, a segment is a subpath of ρ between two consecutive assignment transitions (or between the last assignment transition and the end of the path). Splitting up a path into segments thus allows us to think about a single value of \mathbf{x} at a time.

Definition 4.73. Segments following each other

Proposition 4.74 (Vishnu’s segment graph overview here). *TODO: Vishnu*

4.9 Minimizing a privacy budget

If we have a differentially private program, we’d also like to optimize its privacy cost. We can do so via couplings:

Proposition 4.75. *Introduce linear program here - maybe this is a definition? TODO: Vishnu*

Proposition 4.76. *Introduce approximate version here TODO: Vishnu*

Proposition 4.77. *Optimal linear program always has cost at worst approx version; there is indeed a separation between the optimal + approx TODO: Vishnu*

Proposition 4.78. *Approximate version is an approximation of the optimal by a factor linear in the number of segments. TODO: Vishnu*

Proposition 4.79. *Approximate version can be solved in polytime TODO: Vishnu*

Conjecture 4.80 (IF I HAVE TIME LOOK INTO THIS). Optimal coupling cost \leq cost given by original DiPA analysis. [Vishnu says he almost certainly has a proof of this] *TODO: Vishnu*

Conjecture 4.81. Optimal coupling cost = “true” optimal privacy cost *TODO: Vishnu*

5 Extensions

- brief word about counter, maybe the simplified multivariable case

5.1 Two-Threshold Programs

6 DiPA

We now introduce the final extension to our program model.

Definition 6.1 ([6]). A DiP⁸ Automaton (DiPA) A is a 7-tuple $(Q, \Sigma, C, \Gamma, q_{init}, X, P, \delta)$ where

- Q is a finite set of states partitioned into input states Q_{in} and non-input states Q_{non} .
- $\Sigma = \mathbb{R}$ is the input alphabet
- $C = \{\mathbf{true}, \mathbf{insample} < x, \mathbf{insample} \geq x\}$ is a set of guard conditions
- Γ is a finite output alphabet
- $q_{init} \in Q$ is the initial state
- $X = \{\mathbf{x}, \mathbf{insample}, \mathbf{insample}'\}$ is the set of variables
- $P : Q \rightarrow \mathbb{Q} \times \mathbb{Q}^{\geq 0} \times \mathbb{Q} \times \mathbb{Q}^{\geq 0}$ is a parameter function that assigns sampling parameters for the Laplace distribution for each state
- $\delta : (Q \times C) \rightarrow (Q \times (\Gamma \cup \{\mathbf{insample}, \mathbf{insample}'\}) \times \{\mathbf{true}, \mathbf{false}\})$ is a partial transition function.

In addition, δ must satisfy some additional conditions:

- **Determinism:** For any state $q \in Q$, if $\delta(q, \mathbf{true})$ is defined, then $\delta(q, \mathbf{insample} < x)$ and $\delta(q, \mathbf{insample} \geq x)$ are not defined.
- **Output Distinction:** For any state $q \in Q$, if $\delta(q, \mathbf{insample} \geq x) = (q_1, o_1, b_1)$ and $\delta(q, \mathbf{insample} < x) = (q_2, o_2, b_2)$, then $o_1 \neq o_2$ and at least one of $o_1 \in \Gamma$ and $o_2 \in \Gamma$ is true.
- **Initialization:** The initial state q_0 has only one outgoing transition of the form $\delta(q_0, \mathbf{true}) = (q, o, \mathbf{true})$.
- **Non-input transition:** From any $q \in Q_{non}$, if $\delta(q, c)$ is defined, then $c = \mathbf{true}$.

A DiPA A functions almost identically to the previous automata we have defined, except that instead of only the first transition of A assigning into \mathbf{x} , now any **assignment transition** can re-assign the value of \mathbf{x} to be $\mathbf{insample}$. In addition, DiPAs are no longer dependent on an initial value for \mathbf{x} , since the first transition must always be a \mathbf{true} guard assignment transition.

Note that we can treat a DiPA A as a (possibly infinite, if an assignment transition is in a cycle) set of segment automata by “splitting” at every assignment transition in A .

Definition 6.2. For a path ρ in a DiPA A , let $segments(\rho)$ be the set of subpaths of ρ created by splitting $\rho = r_0 \rightarrow \dots \rightarrow r_n$ at each assignment transition.

More precisely, let I be the ordered set of indices such that $\mathbf{trans}(r_{I_i})$ for $i \in [|I|]$ is an assignment transition. Then $segments(\rho) = \{r_0 \rightarrow \dots \rightarrow r_{I_0}, \dots, r_0 \rightarrow \dots \rightarrow r_{I_k}\}$.

Definition 6.3. Let A be a DiPA and let P be the set of all paths through A .

Then $branches(A) = \bigcup_{\rho \in P} \bigcup_{\eta \in segments(\rho)} acyclic(\eta)$ be the set of branches in A .

⁸Differentially Private

Note that every path in A can be broken up into a sequence of branches, each from its own segment.

6.1 Joining Coupling Strategies

Naively, the fact that we can separate a DiPA A into a set of segment automata would suggest an approach of assigning a coupling strategy to each branch of A as before and simply combining these coupling strategies in sequence. However, it is not always possible to arbitrarily join coupling strategies together.

This is primarily because we can no longer freely decide the initial value for \mathbf{x} at the beginning of each segment, as we could with segment automata. Recall that for a fixed path $\rho = r_0 \rightarrow \dots \rightarrow r_n$, it was important that the initial transition guard $\text{guard}(r_0)$ is satisfied. Notably, whether or not the initial guard is satisfied is dependent on the initialized value of \mathbf{x} ; since we had no additional restrictions on \mathbf{x} , we were able to (for example) arbitrarily couple $x_0\langle 1 \rangle + 1 (=)^{\#d_{x\varepsilon}} x_0\langle 2 \rangle$.

However, now that we have joined multiple segment automata together, the initial value x_0 for a segment automata is actually the assigned value \mathbf{x} from the *previous* segment automata. This means that $x_0\langle 1 \rangle$ and $x_0\langle 2 \rangle$ will already be coupled together, perhaps in a way such that coupling $\text{insample}_0\langle 1 \rangle$ and $\text{insample}_0\langle 2 \rangle$ will be impossible to both ensure the correct shift of \mathbf{x} for the new coupling strategy and so that the initial guard is satisfied.

For example, consider two paths $\rho = r_0 \rightarrow \dots \rightarrow r_n$ and $\rho' = q_1 \rightarrow \dots \rightarrow q_m$ with the assignment transition $r_n \rightarrow q_1$ connecting them, where $\text{guard}(r_n) = \text{insample} \geq x$. We are aiming to choose one coupling strategy for each of ρ and ρ' .

If S^L is chosen for ρ , then at the connecting assignment transition, x_0 is coupled such that $x_0\langle 1 \rangle + 1 = x_1\langle 1 \rangle$. Further, we would like to couple insample_0 such that $\text{insample}_0\langle 1 \rangle = \text{insample}_0\langle 2 \rangle + 1$. However, trying to create this lifting while also ensuring that the implication $\text{insample}_0\langle 1 \rangle \geq x_0\langle 1 \rangle \implies \text{insample}_0\langle 2 \rangle \geq x_0\langle 2 \rangle$ holds is impossible.

Thus, we cannot choose S^G for a segment that follows a segment with coupling strategy S^L if the connecting assignment transition has guard $\text{insample} \geq x$. There are a series of similar restrictions on how coupling strategies can be connected for adjacent segments, which along with previous restrictions on coupling strategies (e.g. we cannot choose S^G or S^L if the first transition of a segment outputs insample).

6.2 A Constraint System for Valid Couplings

Based on the restrictions on joining segments together, we can construct a set of constraints on coupling strategies for a DiPA A that, if solved, will give us a valid proof of $d\varepsilon$ -differential privacy.

Construction 6.4. Consider a DiPA A .

For each branch $s_i \in \text{branches}(A)$, we can assign one of three coupling strategies $S_i \in \{S^L, S^G, S^N\}$. We would like to find an assignment of coupling strategies for each segment

of each variable, subject to the following constraints:

1. Constraints for valid couplings

- (a) For all s_i , if $\text{trans}(s_i)$ outputs insample , then $S_i = S^N$.
- (b) For all s_i, s_j such that s_i is directly followed by s_j ,
 - i. If $\text{guard}(s_j) = \text{insample} < x$ and $S_i = S^G$, then $S_j = S^G$.
 - ii. If $\text{guard}(s_j) = \text{insample} \geq x$ and $S_i = S^L$, then $S_j = S^L$.
 - iii. If $\text{guard}(s_j) = \text{insample} < x$ and $S_i = S^N$, then $S_j \neq S^L$.
 - iv. If $\text{guard}(s_j) = \text{insample} \geq x$ and $S_i = S^N$, then $S_j \neq S^G$.

2. Constraints for finite cost

- (a) For all s_i , no cycle in s_i has a transition that outputs insample or $\text{insample}'$.
- (b) For all s_i , if s_i has an L-cycle, then $S_i = S^L$.
- (c) For all s_i , if s_i has a G-cycle, then $S_i = S^G$.
- (d) For all segments s_i , there is no transition $\text{trans}(a_k)$ in s_i that is *faulty*, i.e.:
 - i. If s_i contains a $\text{insample} < x$ transition that outputs insample , then $S_i \neq S^G$.
 - ii. If s_i contains a $\text{insample} \geq x$ transition that outputs insample , then $S_i \neq S^L$.

Theorem 6.5. *Let A be a DiPA. If $|\text{branches}(A)| < \infty$ and for each branch $B \in \text{branches}(A)$, there exists an assignment of coupling strategies to each segment that satisfies the constraint system defined in 6.4, then A is $d\varepsilon$ -differentially private for*

$$d = \sum_{s_i \in \text{branches}(A)} d_{s_i},$$

where d_{s_i} is the cost of the satisfying assignment strategy S_i for a branch s_i .

Proof. Let ρ be a path through A that is composed of branches $s_{x_1} \rightarrow \dots \rightarrow s_{x_k}$ and let S_{x_1}, \dots, S_{x_k} be the associated coupling strategies from the satisfying assignment.

Because of constraints (1a-1b), we can actually compose each coupling strategy in sequence to create a valid lifting $A(X)\{(a, b) : a = \sigma \implies b = \sigma\}A(X')$ for all adjacent datasets $X \sim X'$ and possible outputs σ of ρ .

In addition, we claim that the cost of the satisfying assignments d_{s_i} is bounded for all s_i . There are two ways for a branch cost d_{s_i} to be unbounded: either a cycle transition has non-zero cost, or a transition has to be faulty with respect to the chosen coupling strategy. From constraint (2d), we know that no transitions can be faulty, and we know from constraints (2a-2c) that every cycle transition has to have zero cost.

Finally, the fact that $|branches(A)| < \infty$ and every path can go through each branch at most one time gives us the bound on d . \square

6.3 Well-formedness

Similar to before, [6] define four structures in the graphs of DiPAs that characterize privacy for DiPAs. Within DiPAs, these graph structures can be defined as follows:

- **Leaking Cycles:** A cycle C is a leaking cycle if one of the transitions in C is an assignment transition.
- **Leaking Pair:** A pair of cycles C, C' are a leaking pair if C is an L-cycle, C' is a G-cycle, and there exists a path from C to C' such that every assignment transition on the path has guard `insample` $\geq x$; or, symmetrically, C is an G-cycle, C' is a L-cycle, and there exists a path from C to C' such that every assignment transition on the path has guard `insample` $< x$.
- **Disclosing Cycle:** A cycle C is a disclosing cycle if a transition in C outputs either `insample` or `insample'`.
- **Privacy Violating Path:** A path $\rho = q_0 \rightarrow \dots \rightarrow q_n$ is a privacy violating path if any of the following conditions hold:
 - $q_1 \rightarrow \dots q_n$ is a path whose assignment transitions all have guard `insample` $\geq x$ (resp. `insample` $< x$) such that q_n is part of a G-cycle (resp. L-cycle) and `trans`(q_0) is an assignment transition that outputs `insample`.
 - ρ is a path whose assignment transitions all have guard `insample` $\geq x$ (resp. `insample` $< x$) such that q_n is part of a G-cycle (resp. L-cycle), `guard`(q_0) = `insample` $< x$ (resp. `insample` $\geq x$), and `trans`(q_0) outputs `insample`.
 - ρ is a path whose assignment transitions all have guard `insample` $\geq x$ (resp. `insample` $< x$) such that q_0 is part of a L-cycle (resp. G-cycle), `guard`(q_{n-1}) = `insample` $\geq x$ (resp. `insample` $< x$), and `trans`(q_{n-1}) outputs `insample`.

Definition 6.6. A DiPA A is **well-formed** if it does not have a leaking cycle, leaking pair, disclosing cycle, or privacy violating path.

Importantly, [6] have shown that the existence of these graph structures completely decides whether or not a DiPA is $d\varepsilon$ -differentially private.

Theorem 6.7 ([6]). *A DiPA A is $d\varepsilon$ -differentially private for some $d > 0$ if and only if A is well-formed. Further, the well-formedness of an automaton A can be decided in linear time in the size of A .*

We will leverage this result to show that the existence of valid coupling strategies is also a complete characterization of DiPAs.

Lemma 6.8. *For a DiPA A , $|branches(A)| < \infty$ if and only if A has no leaking cycles.*

Lemma 6.9. *Suppose a DiPA A has a finite number of branches. If constraint system 6.4 is not satisfiable, there must exist a leaking cycle, a leaking pair, a disclosing cycle, or a privacy violating path in the graph of A .*

Proof. TBD, want to briefly discuss the best way to do this.

high level overview: very similar to the previous, one-segment, version of this, but we can chain constraints across **AG**– and **AL**–paths.

The reasoning gets a bit convoluted though, because there are a bunch of cases / you have to keep a bunch of assumptions hanging around. \square

Theorem 6.10. *A DiPA A is $d\varepsilon$ -differentially private if and only if A has a finite number of branches and there exists a valid assignment of coupling strategies over all branches of A .*

Proof. Follows from theorem 6.5 and lemmas 6.8 and 6.9. \square

7 Bounds on Privacy

So far, we have focused on the binary question of whether a DiPA is private or not for *any* finite $d > 0$. An obvious question remaining thus is how tight of a privacy cost bound can we obtain using couplings?

Recall that S^G , S^L , and S^N are primarily characterized by **shifts**; that is, offsetting variables from each other.

Indeed, we can generalize our discrete set of coupling strategies to a continuous family of coupling strategies by treating the offsets of each variable as parameters. By rephrasing in this way, we can derive a linear system:

[insert vishnu’s linear program here]

If solvable, this linear system will thus give the best possible privacy cost obtainable using this family of coupling strategies.

Proposition 7.1. *An optimal solution to the linear program provides the minimal cost coupling over all shift coupling strategies. If the linear program is infeasible, then the DiPA is not $d\varepsilon$ -differentially private for any ε .*

Theorem 7.2 (optimistically). *If a DiPA A is ε -differentially private, then there exists some valid assignment of coupling strategies over all segments that has a total cost of ε (i.e. coupling cost is tight).*

8 Multivariable DiPA

In this section, we explore a natural extension of the DiPA model to demonstrate the utility of using couplings as proof infrastructure. Specifically, we introduce Generalized DiPAs (GDIPAs), which allow for an arbitrary finite set of variables and an expanded alphabet where

multiple variables can be compared to the input simultaneously to determine transitions. This would, for example, allow for SVT-style algorithms that check for membership between or outside of two thresholds or indeed, membership within any finite union of intersections of halves⁹ of \mathbb{R} .

8.1 GDiPA

A GDiPA A is a generalization of DiPA whose primary characteristics are that:

- At every state, a (real-valued) input is read. Additionally, Laplace noise (with parameters set by the user) is added to the input.
- The automaton has a stored finite set of (real-valued) variables \mathcal{X} . At each transition, the value of each variable can be updated with the noisy input value read in.
- A transition can optionally assign the (noisy) input value read at the previous state into at most one program variable.
- The automaton will take transitions based on a boolean combination of comparisons between the noisy input and a subset of the stored variables. Importantly, noise is added independently to the input for each separate variable comparison. Alternatively, there can be exactly one guaranteed (**true**) transition out of a state.
- At every transition, the automaton will output either
 1. A symbol from a pre-defined finite alphabet (Γ)
 2. The noisy input value as compared to one of the input variables (**insample**)
 3. The input value with fresh Laplace noise (**insample'**)

As with DiPAs, the output sequence of any GDiPA must uniquely determine a path through the automaton.

8.2 Combining Separate Variable Couplings

Consider a GDiPA A with program variables \mathcal{X} . For each program variable $x \in \mathcal{X}$, we can define coupling strategies similar to with single variables. That is, for two runs of the automaton with adjacent inputs, we must ensure that if the first run takes a certain transition, the other run does as well. In particular, for real outputs, we must ensure that the outputs are equal.

For each variable in A , we can consider a “shadow” automaton in a single variable that only has a single program variable. Such a shadow automaton would have the same underlying graph structure as A , but would only contain the transition guards and assignments that pertained to a single variable. For example, if we were considering such an automaton A_x with respect to the variable x , a transition with guard “**insample** $< x$ and **insample** $\geq y$ ”

⁹think there’s a term for this I’m forgetting

would correspond to a transition with guard `insample` $< x$ in A_x ; a transition with guard “`insample` $\geq y$ ” would correspond to a transition with guard `true` in A_x .

Note that these automata are not, strictly speaking, DiPAs, since it is possible for a state to have multiple transitions with guard `true` leaving it.

Thus, each segment can be assigned a coupling strategy S^N, S^L , or S^G as before based on these “shadow” automata, with similar constraints.

Construction 8.1. Consider a GDiPA A with program variables \mathcal{X} . Let $\{s_i^{(x)}\}$ be the segments of A for each variable $x \in \mathcal{X}$.

For each $x \in \mathcal{X}$ and segment $s_i^{(x)}$ for x , we can assign one of three coupling strategies $S_i^{(x)} \in \{S^L, S^G, S^N\}$. We would like to find an assignment of coupling strategies for each segment of each variable, subject to the following constraints:

1. Constraints for valid couplings

- (a) For all $s_i^{(x)}$, if `trans`($s_i^{(x)}$) outputs `insample`, then $S_i^{(x)} = S^N$.
- (b) For all $s_i^{(x)}, s_j^{(x)}$ such that $s_i^{(x)}$ is immediately followed by $s_j^{(x)}$,
 - i. If `guard`($s_j^{(x)}$) = `insample` $< x$ and $S_i^{(x)} = S^G$, then $S_j^{(x)} = S^G$.
 - ii. If `guard`($s_j^{(x)}$) = `insample` $\geq x$ and $S_i^{(x)} = S^L$, then $S_j^{(x)} = S^L$.
 - iii. If `guard`($s_j^{(x)}$) = `insample` $< x$ and $S_i^{(x)} = S^N$, then $S_j^{(x)} \neq S^L$.
 - iv. If `guard`($s_j^{(x)}$) = `insample` $\geq x$ and $S_i^{(x)} = S^N$, then $S_j^{(x)} \neq S^G$.
- (c) For all segments $s_i^{(x)}$, there is no transition `trans`(a_k) in $s_i^{(x)}$ that is *faulty*, i.e.:
 - i. If $s_i^{(x)}$ contains a `insample` $< x$ transition that outputs `insample`, then $S_i^{(x)} \neq S^G$.
 - ii. If $s_i^{(x)}$ contains a `insample` $\geq x$ transition that outputs `insample`, then $S_i^{(x)} \neq S^L$.
- (d) If any transition in $s_i^{(x)}$ outputs x , then $S_i^{(x)} = S^N$.

2. Constraints for finite cost

- (a) For all $s_i^{(x)}$, no cycle in $s_i^{(x)}$ has a transition that outputs `insample`, `insample'`.
- (b) For all $s_i^{(x)}$, if $s_i^{(x)}$ has an L-cycle with respect to x , then $S_i^{(x)} = S^L$.
- (c) For all $s_i^{(x)}$, if $s_i^{(x)}$ has a G-cycle with respect to x , then $S_i^{(x)} = S^G$.

This constraint system is **satisfiable** for a GDiPA A if, for all variables $x \in \mathcal{X}$, there exists an assignment of all $S_i^{(x)}$ such that all constraints are satisfied.

In particular, we can independently resolve constraints for each variable and combine them together.

Theorem 8.2. *For a GDiPA \mathcal{A} , for all variables x , if there exist a finite number of segments s_i , and for all segments s_i , the constraint system 6.4 is satisfied by an assignment S_x , then all S_x assignments together induce a valid coupling proof that \mathcal{A} is ε -DP.*

Proof. Fix some variable $x \in \mathcal{X}$. From before, we know that if 8.1 is satisfied with respect to x , then the coupling

$$\mathcal{A}(X)(\mathcal{A}(X) \text{ takes transitions } T \text{ wrt } x \implies \mathcal{A}(X') \text{ takes transitions } T \text{ wrt } x)^{\#(\varepsilon_x, 0)} \mathcal{A}(X')$$

is valid for some finite ε_x .

Because the noises on `insample` are independent, and $((a \implies c) \wedge (b \implies d)) \implies ((a \wedge b) \implies (c \wedge d))$ and $((a \implies c) \vee (b \implies d)) \implies ((a \vee b) \implies (c \vee d))$, for every combined guard, we can construct the lifting

$$\mathcal{A}(X)(\mathcal{A}(X) \text{ takes transitions } T \implies \mathcal{A}(X') \text{ takes transitions } T)^{\#(\sum_{x \in \mathcal{X}} \varepsilon_x, 0)} \mathcal{A}(X')$$

which gives us the lifting

$$\mathcal{A}(X)(\mathcal{A}(X) = \sigma \implies \mathcal{A}(X') = \sigma)^{\#(\sum_{x \in \mathcal{X}} \varepsilon_x, 0)} \mathcal{A}(X')$$

□

Theorem 8.3 (less optimistically). *If, for each variable x , there exists a valid assignment of coupling strategies over all segments of a DiPA A with respect to x , then A is $d\varepsilon$ -differentially private.*

Theorem 8.4 (optimistically). *A GDiPA A is $d\varepsilon$ -differentially private if and only if, for each variable x , there exists a valid assignment of coupling strategies over all segments of A with respect to x .*

9 Conclusion

We have shown how to use coupling techniques to prove privacy for a class of SVT-like programs first defined in [6] and discovered that couplings additionally characterize this class. We additionally showed that this can be done tractably, and that couplings can help provide lower bounds on privacy costs of these algorithms.

Future work most naturally would focus on extensions of the program model. For the model, potential areas include removing the requirement for output to be deterministic of a path through the automaton, which would allow for algorithms such as Report Noisy Max to be captured by the model. Similarly, the alphabet of the automaton could be expanded to incorporate more than comparisons between two real numbers. Such extensions would naturally also require extensions of the class of couplings we define here, which are limited to “shifts”.

Additionally, we believe that couplings should completely characterize GDiPAs as well as DiPAs; proving this requires showing that a lack of well-formedness in any single variable

generates a counterexample to privacy. In this vein, we would like to explore using couplings to *disprove* privacy; the fact that shift couplings completely characterize DiPAs hints at the possibility of “anti-couplings” to generate counterexamples.

10 Related Work

The DiPA model and counterexamples to privacy are drawn from [6]. Approximate liftings were developed in [5, 4] and applied to algorithms such as SVT in [3]. A full exploration of approximate liftings can be found in [9]. [1] uses couplings; and in particular the “shift” couplings family we use, to create a heuristically successful program for proving the correctness of possible differentially private algorithms.

need to reformat some citations at some point

References

- [1] Aws Albarghouthi and Justin Hsu. Synthesizing coupling proofs of differential privacy. 58:30, 2018.
- [2] Gilles Barthe, Rohit Chadha, Vishal Jagannath, A. Prasad Sistla, and Mahesh Viswanathan. Deciding differential privacy for programs with finite inputs and outputs. In *Proceedings of the 35th Annual ACM/IEEE Symposium on Logic in Computer Science*, LICS ’20, page 141–154, New York, NY, USA, 2020. Association for Computing Machinery.
- [3] Gilles Barthe, Marco Gaboardi, Benjamin Grégoire, Justin Hsu, and Pierre-Yves Strub. Proving differential privacy via probabilistic couplings. *Proceedings - Symposium on Logic in Computer Science*, 05-08-July-2016:749–758, 1 2016.
- [4] Gilles Barthe and Federico Olmedo. Beyond differential privacy: Composition theorems and relational logic for f-divergences between probabilistic programs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 7966 LNCS:49–60, 2013.
- [5] Gilles Barthe, Boris Köpf, Federico Olmedo, and Santiago Zanella-Béguelin. N probabilistic relational reasoning for differential privacy.
- [6] Rohit Chadha, A. Prasad Sistla, and Mahesh Viswanathan. On Linear Time Decidability of Differential Privacy for Programs with Unbounded Inputs, April 2021.
- [7] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pages 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [8] Cynthia Dwork and Aaron Roth. The algorithmic foundations of differential privacy. *Found. Trends Theor. Comput. Sci.*, 9(3–4):211–407, aug 2014.

- [9] Justin Hsu. Probabilistic couplings for probabilistic reasoning. *CoRR*, abs/1710.09951, 2017.
- [10] Min Lyu, Dong Su, and Ninghui Li. Understanding the sparse vector technique for differential privacy. *Proc. VLDB Endow.*, 10(6):637–648, feb 2017.
- [11] Jun Tang, Aleksandra Korolova, Xiaolong Bai, Xueqiang Wang, and XiaoFeng Wang. Privacy loss in apple’s implementation of differential privacy on macos 10.12. *CoRR*, abs/1709.02753, 2017.