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 {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

[1] not 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 $\mathcal{X} = (x_1, \ldots, x_n), \mathcal{X}' = (x'_1, \ldots, x'_n) \in X^n$ 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 $E \subseteq \texttt{im}(A)$,

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

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. Many[3] 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 $f : X \to \mathbb{R}$, 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 $\texttt{Lap}(\mu, b)$ 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 $\texttt{Lap}(0, b)$ as $\texttt{Lap}(b)$.

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

[3] generic 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, $A(X) = f(X) + \textbf{\textit{Lap}}(\frac{\Delta}{\varepsilon})$ is ε-differentially private.*

We will consider the scenario where we are given a potentially infinite *sequence* of real-valued query functions $q_0, q_1, \ldots$, 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 undecidable*[4].

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 probablistic 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.

[4]rephrase?

Definition 3.2. Let A_1, A_2 be two probability spaces[5]. 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. $\mathtt{supp}(\mu_L), \mathtt{supp}(\mu_R) \subseteq \Psi$
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 $\varepsilon-$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 $X_1 \sim \mathtt{Lap}(\mu_1, \frac{1}{\varepsilon})$ and $X_2 \sim \mathtt{Lap}(\mu_2, \frac{1}{\varepsilon})$, 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 implicatory 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

[5] may 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 incorrect[6].

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

Algorithm 1 Sparse Vector Technique

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

$\varepsilon_1, \varepsilon_2 \leftarrow \frac{\varepsilon}{2}, \rho \leftarrow \texttt{Lap}(\frac{\Delta}{\varepsilon_1})$, $\mathit{count} \leftarrow 0$
for $q_i \in Q$ **do**
    $z \leftarrow \texttt{Lap}(\frac{2c\Delta}{\varepsilon_2})$
    if $q_i(\mathcal{X}) + z \geq T + \rho$ **then**
        output $\top$
        $\mathit{count} \leftarrow \mathit{count} + 1$
        if $\mathit{count} \geq c$ **then**
            break
        end if
    else
        output $\perp$
    end if
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 $SVT(\mathcal{X}, T, Q, c)\{(a, b) : a = \sigma \implies b = \sigma\}^{\#\varepsilon} SVT(\mathcal{X}', T, Q, c)$ 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 $SVT(\mathcal{X}, T, Q, c)$ and $SVT(\mathcal{X}', T, Q, c)$, 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 $\tilde{T} \sim \texttt{Lap}(T, \frac{\Delta}{\varepsilon_1})$, so $\tilde{T}\langle 1\rangle + \Delta (=)^{\#\varepsilon_1} \tilde{T}\langle 2\rangle$.

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

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

[6] A 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. □

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.

4.1 Individual Transitions

Definition 4.1. Let Q be a finite set of program states partitioned into input states Q_{in} and non-input states Q_{non}. Then $P_Q(q) : Q \to \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$ 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 `x`, which can be thought of as analogous to the threshold in algorithms like SVT.

To capture the value of `x` at different states, we pair each state with possible values of `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 $\mathcal{C} = \{\mathtt{true}, \mathtt{insample} < \mathtt{x}, \mathtt{insample} \geq \mathtt{x}\}$ 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 $\{\mathtt{x}, \mathtt{insample}, \mathtt{insample}'\}$. 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.
- $\sigma \in \Gamma \cup \{\mathtt{insample}, \mathtt{insample}'\}$ is the output of t.
- $\tau \in \{\mathtt{true}, \mathtt{false}\}$ is a boolean value indicating whether or not the stored value of `x` will be updated.

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

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

4.1.1 Transition Semantics

A program state is a program location coupled with a value for the program variables $\texttt{x}$, an input sequence $\texttt{in}$, and an output sequence σ. Let $S = Q \times \mathbb{R} \times \mathbb{R}^* \times (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\})^*$ be the set of all possible program states.

Then the semantics of a transition t is a function $[\![t]\!] : S \to dist(S)$ that maps an initial program state to a distribution of subsequent program states following the expected semantics. For example,

$$[\![(q_0, q_1, \texttt{insample} < \texttt{x}, \top, 1)]\!]((q_0, 0, \texttt{in}, \sigma)) = \begin{cases} (q_1, \texttt{insample}, \texttt{in}_{1\ldots}, \sigma\top) & \texttt{insample} < 0 \\ (q_{term}, -, -, \sigma) & \texttt{insample} \geq 0 \end{cases},$$

where $\texttt{insample} \sim \texttt{Lap}(\texttt{in}_0, \frac{1}{d_0\varepsilon})$.

Another formulation

Let $O \subset \mathbb{P}(\mathbb{R})$ be the set of all measurable subsets of $\mathbb{R}$. Let $t = (q, q', c, \sigma, \tau)$ be a transition, let $\texttt{in} \in \mathbb{R}$ be a valid input for t, and let $o \in \Gamma \cup O$ be a possible output of t. Let $X \sim \texttt{Lap}(\mu, \frac{1}{d_x\varepsilon})$ be a random variable sampled from a Laplace distribution. Let $P(q) = (d_q, d'_q)$.

Together, these define a random process: if, starting with $\texttt{x} = X$, the guard c is satisfied given input $\texttt{in}$ and o is output.

More precisely, let $I \sim \texttt{Lap}(\texttt{in}, \frac{1}{d_q\varepsilon})$ and $I' \sim \texttt{Lap}(\texttt{in}, \frac{1}{d'_q\varepsilon})$ be independent random variables.

For both I and I', given o, we say that I (or I', respectively) matches o if $o \subseteq \mathbb{R}$ and $I \in o$.

Then let a "success" for $(X, t, \texttt{in}, o)$ be the event where c is satisfied for $\texttt{x} = X$ and $\texttt{insample} = I$ and, if $\sigma = \texttt{insample}$ or $\sigma = \texttt{insample}'$, then I matches o or I' matches o, respectively.

We will notate the probability of such an event happening as $\mathbb{P}[X, t, \texttt{in}, o]$

Another formulation

We can describe programs as Markov processes with state space $S = Q \times \mathbb{R} \times \mathbb{R}^* \times (\Gamma \cup \mathbb{R})^*$.

A state $s \in S$ is the tuple $(q, \texttt{x}, \texttt{in}, \sigma)$. If there exists a transition $t = (q, q', c, o, 0)$, then s transitions to state $(q', \texttt{x}, \texttt{in}_{1:}, \sigma \cdot o)$ with probability equal to the probability that c is satisfied given that $\texttt{x} = \texttt{x}$ and $\texttt{in}$ is the input.

If $\texttt{in}$ is empty, then the Markov process terminates.

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 $\mathtt{x}$.

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

Additionally, if $\tau = 0$, then $x = x'$, since $\mathtt{x}$ 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 $\sigma \in \{\mathtt{insample}, \mathtt{insample}'\}$, let $o \subseteq \mathbb{R}$ (measurable)).

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

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

4.1.2 Privacy

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 $\hat{t}$.*

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

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

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

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

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

Definition 4.7 (Adjacency)**.** Two inputs $\mathtt{in} \sim_\Delta \mathtt{in}'$ are Δ-adjacent if $|\mathtt{in} - \mathtt{in}'| \leq \Delta$. If Δ is not specified, we assume that $\Delta = 1$.

Definition 4.8 (Valid inputs)**.** Let $t = (q, q', c, \sigma, \tau)$ be a transition over Q. A valid adjacent input pair to t is a pair of real numbers $(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) \in \mathbb{R}^2$ such that:

- If $q \in Q_{in}$, then $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$.
- If $q \notin Q_{in}$, then $\mathtt{in}\langle 1\rangle = \mathtt{in}\langle 2\rangle = 0$.

We will abuse notation and denote that $(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)$ is a valid adjacent input pair by also writing $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$.

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

Definition 4.9 ($d\varepsilon$-differential privacy for a transition)**.** Let $X\langle 1\rangle \sim \mathtt{Lap}(\mu\langle 1\rangle, \frac{1}{d_x\varepsilon}), X\langle 2\rangle \sim \mathtt{Lap}(\mu\langle 2\rangle, \frac{1}{d_x\varepsilon})$ be random variables representing possible initial values of `x`. Then, given $X\langle 1\rangle, X\langle 2\rangle$, a transition $t = (q, q', c, \sigma, \tau)$ is **$d\varepsilon$-differentially private** for some $d > 0$ if $\forall\varepsilon > 0$, for all valid adjacent input pairs $\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle$ and possible outputs σ of t, $\mathbb{P}[X\langle 1\rangle, t, \mathtt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[X\langle 2\rangle, t, \mathtt{in}\langle 2\rangle, \sigma]$.

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 equivalent[7], since if we are targeting ε^*-differential privacy overall, we can always take $\varepsilon = \frac{\varepsilon^*}{d}$.

[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

4.1.3 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 4.10. *Let $X\langle 1\rangle \sim \mathbf{Lap}(\mu\langle 1\rangle, \frac{1}{d_x\varepsilon}), X\langle 2\rangle \sim \mathbf{Lap}(\mu\langle 2\rangle, \frac{1}{d_x\varepsilon})$ be random variables be random variables representing possible initial values of $\boldsymbol{x}$.*

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

Let $\boldsymbol{in}\langle 1\rangle, \boldsymbol{in}\langle 2\rangle$ be an arbitrary valid adjacent input pair and let $o\langle 1\rangle$, $o\langle 2\rangle$ be random variables representing possible outputs of t given inputs $\boldsymbol{in}\langle 1\rangle$ and $\boldsymbol{in}\langle 2\rangle$, respectively.

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

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

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

Proof. Fix $\varepsilon > 0$.

We will analyze the behaviour of two different **runs** of t, one with input $\mathtt{in}\langle 1\rangle$ and one with input $\mathtt{in}\langle 2\rangle$.

Our approach to couplings will be that 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 $X\langle 1\rangle$ to be the value of $\mathtt{x}$ at state q in the run of t with input $\mathtt{in}\langle 1\rangle$ and $X\langle 2\rangle$ to be the value of $\mathtt{x}$ in the run of t with input $\mathtt{in}\langle 2\rangle$.

We thus want to create the lifting $o\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}o\langle 2\rangle$. 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 $c = \mathtt{true}$, the first condition is trivially satisfied and when $\sigma \in \Gamma$, the second condition is trivially satisfied.

We can first create the lifting $X\langle 1\rangle + \gamma_x (=)^{\#(|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d}_q\varepsilon} X\langle 2\rangle$. This is analogous to shifting the threshold values that we want to compare inputs to in an algorithm like SVT.

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

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

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

Thus, we emerge with three key statements to leverage:

- $X\langle 1\rangle + \gamma_x = X\langle 2\rangle$
- $z\langle 1\rangle = z\langle 2\rangle - \mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma_q$
- $z'\langle 1\rangle = z'\langle 2\rangle - \mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma'_q$

So if $c = \mathtt{insample} < \mathtt{x}$ and $\gamma_q \leq \gamma_x$, then

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

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

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

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

As noted, if $\sigma \in \Gamma$ and the first run taking transition t implies that the second run does as well, then $o\langle 1\rangle = \sigma \implies o\langle 2\rangle = \sigma$ trivially.

Now, if $\sigma = \mathtt{insample}$ and $\gamma_q = 0$, then clearly we have that $\mathtt{insample}\langle 1\rangle = \mathtt{insample}\langle 2\rangle$, so for all $a \in \mathbb{R}$, $o\langle 1\rangle = a \implies o\langle 2\rangle = a$.

Similarly, if $\sigma = \mathtt{insample}'$ and $\gamma'_q = 0$, we have that for all $a \in \mathbb{R}$, $o\langle 1\rangle = a \implies o\langle 2\rangle = a$.

Thus, given the constraints

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

we have shown that the lifting $o\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon} o\langle 2\rangle$ is valid, where the cost $d = (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d_q} + (|-\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma_q|)d_q + (|-\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma'_q|)d'_q$.

By an application of theorem 3.4, $\mathbb{P}[X\langle 1\rangle, t, \mathtt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[X\langle 2\rangle, t, \mathtt{in}\langle 2\rangle, \sigma]$. Because $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$ 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. In particular, we will view choices of γ_x, γ_q, and γ'_q as a **strategy** for proving that a transition is differentially private.

Definition 4.11 (Coupling strategies)**.** 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 4.12 (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

$$\begin{cases} \gamma_i \le \gamma_x^{(i)} & c_i = \texttt{insample} < \texttt{x} \\ \gamma_i \ge \gamma_x^{(i)} & c_i = \texttt{insample} \ge \texttt{x} \\ \gamma_i = 0 & \sigma_i = \texttt{insample} \\ \gamma_i' = 0 & \sigma_i = \texttt{insample}' \end{cases},$$

are all satisfied.

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

Proposition 4.13. *For a transition t_i, if there exists a valid coupling strategy $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma_i')$ given initial $\boldsymbol{x}$ 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*

$$d \le \max_{\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle} (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x^{(i)}|)\hat{d}_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_i|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_i'|)d_q'.$$

Proof. Follows immediately from lemma 4.10. □

4.1.4 Constructing an alphabet

We will consider individual transitions as part of an *alphabet*; we will show that there is an interesting subset of regular languages over an alphabet of transitions that we can apply the coupling framework to.

We will only consider certain alphabets of transitions; the ones that generate coherent programs.

Definition 4.14 (Valid Transition Alphabets)**.** Let Σ_T be a finite alphabet of transitions. We will call Σ_T **valid** if it satisfies the following conditions:

- **Initialization:** There exists some $t_{init} \in \Sigma_T$ such that $t_{init} = (q_0, q_1, \texttt{true}, \sigma, \texttt{true})$ for some $q_0, q_1 \in Q$, $\sigma \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$.
- **Determinism**: If any transition $t \in \Sigma_T$ is of the form $t = (q, q', c, \sigma, \tau)$, then no other transitions of the form $(q, q^*, c, \sigma', \tau')$ for $q, q', q^* \in Q$ exist in Σ_T. Additionally, if there exists a transition $t = (q, q', \texttt{true}, \sigma, \tau)$ such that $t \in \Sigma_T$, then transitions of the form $(q, q^*, \texttt{insample} < \texttt{x}, \sigma', \tau')$ or $(q, q^*, \texttt{insample} < \texttt{x}, \sigma', \tau')$ are not in Σ_T.
- **Output distinction**: If there exist some $\sigma, \sigma', \tau, \tau'$ such that $(q, q', \texttt{insample} < \texttt{x}, \sigma, \tau) \in \Sigma_T$ and $(q, q^*, \texttt{insample} \ge \texttt{x}, \sigma', \tau') \in \Sigma_T$, then $\sigma \ne \sigma'$. Additionally, at least one of $\sigma \in \Gamma$, $\sigma' \in \Gamma$ is true.
- **Non-input state condition**: For all states $q \in Q_{non}$, if there exists a transition $t = (q, q', c, \sigma, \tau)$ such that $t \in \Sigma_T$, then $c = \texttt{true}$.

4.2 Multiple Transitions

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

We start with *paths*, comprised of a sequence of transitions. Equivalently, paths are words comprised of letters from a (valid) transition alphabet Σ_T.

Definition 4.15 (Program paths)**.** Let Σ_T be a valid transition alphabet with underlying state space Q. A program **path** is a sequence of transitions $t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$ such that for all $i \in 0 \ldots n-1$, $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ for some c_i, σ_i, τ_i. We will often notate an path ρ as $\rho = q_0 \to q_1 \to \ldots \to q_n$.

If a path ρ is of the form $\rho = t_{init} \cdot \rho'$ for $\rho' \in \Sigma_T^*$, then we call ρ a **complete** path.

As with transitions, we can discuss a specific instance, or execution, of a path with explicit values for `x`.

Definition 4.16 (Program execution)**.** A program **execution** is a sequence of instantiated transitions $\hat{\rho} = \hat{t}_0 \cdot \hat{t}_1 \cdot \ldots \cdot \hat{t}_{n-1}$ such that for all $i \in 0 \ldots n-1$, where $\hat{t}_i = ((q_i, x_i), (q_{i+1}, x_{i+1}), c_i, \sigma_i, \tau_i)$,

- If $\tau_i = 0$, then $x_{i+1} = x_i$
- If $\tau_i = 1$, then $x_{i+1} = \texttt{insample}_i$, where $\texttt{insample}_i$ is a random variable representing a noised version of the input variable read in at t_i.

As before, we need to restrict the space of possible inputs to a path based on which states in the path actually read in user input.

Definition 4.17. For a path ρ of length n, an input sequence $\texttt{in} \in \mathbb{R}^n$ is valid if, for all q_i in ρ such that $q_i \notin Q_{in}$, $\texttt{in}_i = 0$.

We will assume that all input sequences are valid from now on.

Interestingly, the constraints on valid transition alphabets mean that outputs uniquely correspond to paths.

Proposition 4.18. *Let Σ_T be a valid transition alphabet and let Γ be the finite output alphabet associated with Σ_T. Let $O \subset (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\})^*$ be the set of all possible outputs of complete paths over Σ_T. There exists an injection $f : \Sigma_T \to t_{init}\Sigma_T^*$ from the set of all possible outputs to complete paths over Σ_T.*

Proof. Follows immediately because Σ_T satisfies determinism and output distinction. $\square$

In other words, given a valid transition alphabet, knowing an output sequence uniquely determines which path must have produced the output.

4.2.1 Privacy

As before, we can define the probability of a path outputting a particular sequence of outputs.

Definition 4.19 (Path Probabilities)**.** Let $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$ be a path where $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ and let o be a possible output of ρ. In particular, this means that for all i, if $\sigma_i \in \Gamma$, then $o_i = \sigma_i$, and if $o_i \in \{\texttt{insample}, \texttt{insample}'\}$, then $o_i \subseteq \mathbb{R}$ (measurable).

Then let $\mathbb{P}[x, \rho, \texttt{in}, o]$ be the **probability** that t outputs (values in) o when reading in input $\texttt{in} \in \mathbb{R}^n$ given that $\texttt{x}_0 = x$.

Similarly, let $\hat{\rho} = \hat{t}_0 \cdot \hat{t}_1 \cdot \ldots \cdot \hat{t}_{n-1}$ be an instantiated transition and let o be a possible output of $\hat{\rho}$.

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

Definition 4.20. Given a path $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$, the **tail** of ρ is defined as $tail(\rho) = t_1 \cdot \ldots \cdot t_{n-1}$. Similarly, the tail of a path execution $\hat{\rho} = \hat{t}_0 \cdot \hat{t}_1 \cdot \ldots \cdot \hat{t}_{n-1}$ is $tail(\hat{\rho}) = \hat{t}_1 \cdot \ldots \cdot \hat{t}_{n-1}$

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

Observation 4.21 (Path Probabilities)**.** *Let $\hat{\rho} = \hat{t}_0 \cdot \hat{t}_1 \cdot \ldots \cdot \hat{t}_{n-1}$ be a path execution and let o be a possible output of $\hat{\rho}$. Then the probability of $\hat{\rho}$ outputting o can be defined recursively.*

So $\mathbb{P}[\hat{\rho}, \boldsymbol{in}, o] = \begin{cases} 1 & |\hat{\rho}| = 0 \\ \mathbb{P}[\hat{t}_0, \boldsymbol{in}_0, o_0]\mathbb{P}[tail(\hat{\rho}), tail(\boldsymbol{in}), tail(o)|x_1] & |\hat{\rho}| > 0 \end{cases}$

Now consider a complete path $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$.

is this meaningful?

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

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

Definition 4.22 (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 $i \in [1 \ldots n]$, $|\alpha_i - \beta_i| \leq \Delta$.

As before, if Δ is not specified, we assume that $\Delta = 1$.

Thus, we get the following definition of privacy:

Definition 4.23 ($d\varepsilon$-differential privacy for a path)**.** A complete path ρ of length n is $d\varepsilon$-differentially private for some $d > 0$ if $\forall \varepsilon > 0$, for all valid adjacent input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$ of length n and all possible output sequences σ of length n, $\mathbb{P}[\rho, \texttt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[\rho, \texttt{in}\langle 2\rangle, \sigma]$.

It will also be convenient to define a notion of privacy for sets of paths:

Definition 4.24. Let S be a set of complete paths and let O be a set of all possible outputs of paths in S. Then S is $d\varepsilon$-differentially private for some $d > 0$ if, for all paths $\rho \in S$ and outputs $\sigma \in O$, $\forall \varepsilon > 0$, for all valid adjacent input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $\mathbb{P}[\rho, \texttt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[\rho, \texttt{in}\langle 2\rangle, \sigma]$.

However, because of the path-output correspondence, the following definition is equivalent:

Definition 4.25. Let S be a set of complete paths; S is $d\varepsilon$-differentially private for some $d > 0$ if, for all paths $\rho \in S$, ρ is $d\varepsilon$-differentially private.

4.2.2 Concatenating couplings

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

In order to show that a program path ρ is differentially private, for all adjacent inputs $\alpha \sim \beta$ and all possible outputs σ, we want to create the coupling $\rho[\alpha]\{(a,b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon}\rho[\beta]$ 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 $\tau_j = \texttt{true}$). The coupling shifts for *non-assignment transitions* can thus never impact each other.

Definition 4.26 (Assignment transitions)**.** Let $A_\rho = \{t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i) : \tau_i = \texttt{true}\}$ be the set of **assignment transitions** in a path 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 $\texttt{x}$-values $\texttt{x}_0\langle 1\rangle$ and $\texttt{x}_0\langle 2\rangle$.

Thus, for an individual transition t_i of ρ. From proposition 4.13, 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 path:

Lemma 4.27. *Let $\rho = q_0 \to \ldots \to q_n$ be a complete path of length n. Let $\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle$ 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 $\boldsymbol{in}\langle 1\rangle$ and $\boldsymbol{in}\langle 2\rangle$, 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

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

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

Proof. From the proof of lemma 4.10, we know that we can create the couplings $\texttt{insample}_i\langle 1\rangle + \gamma_i (=)^{\#(|-\texttt{in}_i\langle 1\rangle+\texttt{in}_i\langle 2\rangle-\gamma_i|)d_i\varepsilon} \texttt{insample}_i\langle 2\rangle$ and $\texttt{insample}'_i\langle 1\rangle + \gamma'_i (=)^{\#(|-\texttt{in}_i\langle 1\rangle+\texttt{in}_i\langle 2\rangle-\gamma'_i|)d'_i\varepsilon} \texttt{insample}'_i\langle 2\rangle$ for all q_i in ρ.

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

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

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

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

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 $\texttt{insample}$ and $\texttt{insample}$) for each q_i. Thus, the total coupling cost associated with each q_i is at most $(|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i$, which gives us an overall coupling cost of $d = \sum_{i=0}^{n-1}(|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma'_i|)d'_i$. ☐

Definition 4.28. For a complete path ρ of length n and adjacent input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, a **coupling strategy** is two functions $\boldsymbol{\gamma}(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) : \mathbb{R}^n \times \mathbb{R}^n \to [-1,1]^n$ and $\boldsymbol{\gamma}'(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) : \mathbb{R}^n \times \mathbb{R}^n \to [-1,1]^n$ that produce shifts for each transition of ρ dependent on the input sequences.

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

Definition 4.29. For a complete path ρ of length n, a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is **valid** if $\forall \texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $\boldsymbol{\gamma}(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle)$ and $\boldsymbol{\gamma}'(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle)$ satisfy the constraints

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

4.2.3 Optimizing Privacy

Definition 4.30. For a complete path ρ of length n, the **cost** of a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is

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

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

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

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

Proposition 4.31. *If $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is valid, then ρ is $cost(C_\rho)\varepsilon$-differentially private.*

Proof. Follows immediately from lemma 4.27. □

Corollary 4.32. *Any complete path ρ is $cost(\rho)\varepsilon$-differentially private. Further, for all complete paths ρ, $cost(\rho) < \infty$.*

Proof. The first claim follows immediately from definitions.

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

Proposition 4.33. *The cost of a coupling strategy over a fixed path ρ is maximized when, for every transition, the difference between the input values is 1. In other words,*

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

Proof. □

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

4.3 Branching

Definition 4.34 (Branching program)**.** Let Σ_T be a valid transition alphabet, a branching program B is a finite set of complete paths over Q.

Equivalently, a branching program is a language over Σ_T that can be represented by a regular expression only using concatenation and finite union; every word in the language must also be of the form $t_{init}\Sigma_T^*$.

4.3.1 Privacy

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

Definition 4.35 (Coupling strategies)**.** A (branched program) coupling strategy C for a branching program B is a collection of (path) coupling strategies where each complete path $\rho \in B$ is assigned a coupling strategy C_ρ.

Definition 4.36. A coupling strategy C for a branching program B is valid if, for every constituent path coupling strategy C_ρ, C_ρ is valid.

Definition 4.37. The cost of a coupling strategy C for a branched program B is

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

Proposition 4.38. *Optimal cost is dependent on path (Vishnu)*

4.4 Loops

Definition 4.39. A looping branch L is a (possibly infinite) set of complete paths such that L is the language described by a single union-free regular expression over a valid transition alphabet Σ_T.

For a looping branch L, we will use R_L to denote the minimal union-free regular expression that defines L.

Definition 4.40 (Coupling strategy for a looping branch)**.** Let r be the union-free regular expression describing paths in a looping branch L. Let T_r be the set of all transitions that appear in r. Then a coupling strategy $C = (\gamma, \gamma')$ for a looping branch is a function $C : T_r \times [-1, 1] \to [-1, 1] \times [-1, 1]$ that computes shifts for each transition in L as a function of the difference between two adjacent inputs.

Definition 4.41 (Induced Coupling Strategy)**.** not sure how necessary this is Given a coupling strategy $C = (\gamma, \gamma')$ for a looping branch L and a specific path $\rho \in L$, the coupling strategy for $\rho = q_0 \to \ldots \to q_n$ induced by C is the pair of functions $\gamma_\rho, \gamma'_\rho$ such that

$$\gamma_\rho(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) = (\gamma(q_0 \to q_1)(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle), \gamma(q_1 \to q_2)(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle), \ldots, \gamma(q_{n-1} \to q_n)(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle))$$
$$\gamma'_\rho(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) = (\gamma'(q_0 \to q_1)(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle), \gamma'(q_1 \to q_2)(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle), \ldots, \gamma'(q_{n-1} \to q_n)(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle))$$

One natural question that arises is whether it could be more efficient with regards to cost to

Proposition 4.42. *If there exists a valid coupling strategy C_ρ with cost $cost(C_\rho)$ for every path ρ of an path class $[\rho]$ in a program A and $\sup_{\rho\in[\rho]} cost(C_\rho) < \infty$, then there exists a class coupling strategy C' for $[\rho]$ of A that is valid such that $cost(C') \leq \sup_{\rho\in[\rho]} cost(C_\rho)$.*

Note that, because of the introduction of stars (or, equivalently, cycles) to our model, it is possible for a looping branch to fail to be private for *any* $d > 0$; i.e. every coupling strategy for a looping branch has infinite cost. We can characterize whether or not a coupling strategy has infinite cost through another constraint:

Lemma 4.43. *For a looping branch L, a valid coupling strategy $C = (\gamma, \gamma')$ has finite cost $cost(C) < \infty$ if and only if the following constraint applies for all i:*

- *If t_i is contained within a star in R_L (i.e. t_i is in a cycle), then $\gamma_i = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$ and $\gamma'_i = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$.*

In other words, a finite-cost coupling strategy must assign shifts such that every cycle transition has 0 privacy cost.

Proof. ($\Longleftarrow$)

Let T be the set of transitions t_i in L such that t_i is **not** found under a star in R_L.

Fix a complete path $\rho \in L$ and let C_ρ be the coupling strategy for ρ induced by C.

Let D_ρ be the set of transitions $t_i \in \rho$ such that t_i is under a star in R_L, i.e., $t_i \notin T$.

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

So

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

Thus, $cost(C) \leq |T| \max_{i:t_i\in T}(2d_i + 2d'_i) < \infty$.

($\Longrightarrow$)

Let t_i be a transition in L under a star in R_L such that $\gamma_i \neq -\mathtt{in}\langle 1\rangle_i + \mathtt{in}\langle 2\rangle_i$ or $\gamma'_i \neq -\mathtt{in}\langle 1\rangle_i + \mathtt{in}\langle 2\rangle_i$. Thus, $\exists \mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$ such that $(|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma'_i|)d'_i > 0$. Fix such a $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$.

Then there exists some complete path ρ in L of the form $a(bt_ic)^*d$ for some $a, b, c, d \in \Sigma_T^*$.

Let $\rho_k = a(bt_ic)^k d$ be the corresponding complete path in L with (bt_ic) iterated k times. This is equivalent to iterating the cycle containing t_i k times. Then for all $k \in \mathbb{N}$,

$$cost(\rho_k) \geq k((|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i'|)d_i'),$$

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

In particular, we can combine this constraint that gives us *finite cost* class coupling strategies with the four constraints that ensure that coupling strategies are valid.

Definition 4.44 (Privacy Constraint System)**.** Let L be a looping branch over a valid transition alphabet Σ_T. If, for a candidate coupling strategy $C_L = (\gamma, \gamma')$ for L, the following constraints are satisfied for all i:

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

then we say that C satisfies the privacy constraint system for L.

This provides a useful tool for the decision problem of privacy, which we will discuss in more depth later.

Proposition 4.45. *If there exists a coupling strategy C for a looping branch L that satisfies the privacy constraint system, then there exists a finite $d > 0$ such that L is $d\varepsilon$-differentially private.*

4.5 Programs

Definition 4.46. A program P is a finite union of looping branches over a valid finite alphabet of transitions.

Naturally, this means that P is a language described by a regular expression in union normal form, where each term defines a looping branch.

Lemma 4.47. *If, for every looping branch L in P, there exists a valid coupling strategy C_L, then P is* $(\max_{L \in P} cost(C_L))\varepsilon$*-differentially private.*

4.6 Deciding Privacy

We are primarily(?) concerned with the boolean question of privacy; that is, deciding whether or not there exists *any* finite $d > 0$ such that a program is $d\varepsilon$-differentially private.

Clearly, we can algorithmically show that at least some subset of differentially private programs are private through the use of couplings and coupling strategies.

Lemma 4.48. *If, for every looping branch $L \subseteq P$ in a program P, there exists a coupling strategy C_L that satisfies the privacy constraint system, then there exists some finite $d > 0$ such that P is $d\varepsilon$-differentially private.*

Proof. Follows immediately from proposition 4.45. □

Perhaps surprisingly though, we show that coupling proofs are **complete** for programs of this form; every differentially private program can be proved to be private using couplings.

Lemma 4.49. *If, for some looping branch $L \subseteq P$ in a program P, there does not exist a coupling strategy C_L that satisfies the privacy constraint system, then there does not exist any finite $d > 0$ such that P is $d\varepsilon$-differentially private.*

4.7 DiPA

We now discuss a previously defined program model, DiPA, which turns out to be exactly equivalent to our own program model.

Definition 4.50 ([6])**.** A Differentially Private Automaton (DiPA) A is an 8-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
- $C = \{\texttt{true}, \texttt{insample} < x, \texttt{insample} \geq x\}$ is a set of guard conditions
- Γ is a finite output alphabet
- $q_{init} \in Q$ is the initial state
- $X = \{\texttt{x}, \texttt{insample}, \texttt{insample}'\}$ is a set of variables
- $P : Q \to \mathbb{Q} \times \mathbb{Q}^{\geq 0} \times \mathbb{Q} \times \mathbb{Q}^{\geq 0}$ is a parameter function that assigns sampling parameters for the Laplace distribution for each state
- $\delta : (Q \times C) \to (Q \times (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\}) \times \{\texttt{true}, \texttt{false}\})$ is a partial transition function.

In addition, δ must satisfy some additional conditions:

- **Determinism:** For any state $q \in Q$, if $\delta(q, \texttt{true})$ is defined, then $\delta(q, \texttt{insample} < x)$ and $\delta(q, \texttt{insample} \geq x)$ are not defined.
- **Output Distinction:** For any state $q \in Q$, if $\delta(q, \texttt{insample} \geq x) = (q_1, o_1, b_1)$ and $\delta(q, \texttt{insample} < x) = (q_2, o_2, b_2)$, 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 $\delta(q_0, \texttt{true}) = (q, o, \texttt{true})$.
- **Non-input transition:** From any $q \in Q_{non}$, if $\delta(q, c)$ is defined, then $c = \texttt{true}$.

Definition 4.51. Let ρ be a path in a DiPA A, let $\texttt{in}$ be a valid input sequence and let o be a possible output of ρ. In particular, if $o \in \{\texttt{insample}, \texttt{insample}'\}$, then we require that o is an *interval* $(a, b) \subseteq \mathbb{R}$, rather than simply a measurable set as before. Then $\texttt{Pr}[x, \rho, \texttt{in}, o]$ is the probability of ρ being taken with input sequence $\texttt{in}$ and outputting o. If the first state in ρ is q_{init}, then $\texttt{Pr}[x, \rho, \texttt{in}, o]$ may be shortened to $\texttt{Pr}[\rho, \texttt{in}, o]$, since the initial value of $\texttt{x}$ is irrelevant.

For a full numerical definition, we refer back to the original work.

Definition 4.52. A DiPA A is $d\varepsilon$-differentially private for some $d > 0$ if for all paths ρ in A, for all possible outputs o of ρ and valid adjacent input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$,

$$\mathbb{P}[\rho, \texttt{in}\langle 1\rangle, o] \leq e^{d\varepsilon}\mathbb{P}[\rho, \texttt{in}\langle 2\rangle, o]$$

We can draw a direct equivalence between programs and DiPAs.

Proposition 4.53. *Every path ρ through a DiPA A is represented by a complete path $\hat{\rho}$ comprised of transitions from a valid transition alphabet Σ_T; further, the set of all possible paths through A is a regular language over Σ_T.*

Proof. Let $\rho = q_0 \to q_1 \to \ldots \to q_n$ be a path in $A = (Q, \mathbb{R}, C, \Gamma, q_0, X, P, \delta)$.

For all $i \in 0 \ldots n-1$, there must be some c_i such that $\delta(q_i, c_i) = (q_{i+1}, \sigma_i, \tau_i)$. Let $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ and let $\Sigma_\rho = \{t_i : i \in 0 \ldots n-1\}$ be the set of all such transitions. Note that because δ satisfies the conditions of determinism, output distinction, initialization, and non-input transition, Σ_ρ must as well. Then let $\hat{\rho} = t_0 \cdot t_1 \cdot \ldots \cdot t_{i-1}$ be the representation of ρ as a word over Σ_ρ.

Let $\Sigma_T = \bigcup_{\rho \in A} \Sigma_\rho$. Note that Σ_T must have finite size because A is a finite automaton and must still be a valid transition alphabet.

Let $D = (Q, \Sigma_T, \delta_D, q_0, F = Q)$ be an NFA defined over the set of program locations Q such that δ_D is defined as follows: Let $q \in Q$ be an arbitrary state. If $\delta(q, c) = (q', \sigma, \tau)$ is defined for some $c \in C$, let $\delta_D(q, (q, q', c, \sigma, \tau)) = q'$.

Then clearly every path in A is also a path in D and vice versa; since every state in D is an accepting state, $\mathcal{L}(D) = \{\hat{\rho} : \rho \in A\}$. Thus, the set of (representations of) all paths in A must be a regular language. □

Proposition 4.54. *For all input sequences $\textit{\textbf{in}}$ and possible outputs σ, and for all paths ρ in a DiPA A, $\mathbb{P}[\rho, \textit{\textbf{in}}, \sigma] = \mathbb{P}[\hat{\rho}, \textit{\textbf{in}}, \sigma]$.*

Proof. Follows immediately from definitions. □

Proposition 4.55. *For every program P over a valid transition alphabet Σ_T, there exists a corresponding DiPA A such that there exists a path ρ in A if and only if its representation is in P.*

Proof. The DiPA can be directly constructed from P. $\square$

Interestingly, the privacy of any DiPA A is completely characterized by four graph-theoretic structures (we omit the precise definitions here).

Theorem 4.56 ([6])**.** *A DiPA A does not have a leaking cycle, leaking pair, disclosing cycle, or privacy violating path if and only if there exists some $d > 0$ such that for all $\varepsilon > 0$, A is $d\varepsilon$-differentially private.*

Corollary 4.57. *If the corresponding DiPA A to a program P contains a leaking cycle, leaking pair, disclosing cycle, or privacy violating path, then does not exist a finite $d > 0$ such that P is $d\varepsilon$-differentially private.*

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 4.58. For a complete path ρ in a program P, let A_ρ be the set of all assignment transitions in ρ.

Definition 4.59. A **segment** of a (complete) path $\rho = q_0 \to \ldots \to q_n$ is a subpath $q_i \to q_{i+1} \to \ldots \to q_j$ 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 `x` at a time.

Proposition 4.60 (Vishnu's segment graph overview here)**.**

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 4.61. *Introduce linear program here - maybe this is a definition?*

Proposition 4.62. *Introduce approximate version here*

Proposition 4.63. *Optimal linear program always has cost at worst approx version; there is indeed a separation between the optimal + approx*

Proposition 4.64. *Approximate version is an approximation of the optimal by a factor linear in the number of segments.*

Proposition 4.65. *Approximate version can be solved in polytime*

Conjecture 4.66. Optimal coupling cost $\leq$ cost given by original DiPA analysis. [Vishnu says he almost certainly has a proof of this]

Conjecture 4.67. Optimal coupling cost = "true" optimal privacy cost

5 Program Model Extensions

The class of programs we have defined is rather limited, leading to the natural question of whether our results can be extended to more powerful program models.

Some 'natural' extensions of DiPA end up reducing directly to DiPA. For example, we considered an extension of DiPA that included a single integral "counter" variable n; this program model also allowed for branching conditional on n being greater than a threshold value. We discovered that every DiPA equipped with a counter can be rewritten as a standard DiPA by using a power set-style automata construction.

However, there are other, non-trivial, extensions that warrant further study. In particular, we describe one extension of DiPA that allows for two real-valued program variables $\mathtt{x}, \mathtt{y}$.

5.1 Two-Threshold Programs: GDiPA

5.1.1 Multivariable Transitions

notation for guards

Definition 5.1 (2v-transitions)**.** A two-variable transition (2v-transition) is a tuple (q, q', c, σ, τ) where

- $q \in Q$ is the initial state
- $q' \in Q$ is the destination state
- $c \in \{\mathtt{true}, \mathtt{insample} < \mathtt{x}, \mathtt{insample} \geq \mathtt{x}\} \times \{\wedge, \vee\} \times \{\mathtt{true}, \mathtt{insample} < \mathtt{y}, \mathtt{insample} \geq \mathtt{y}\}$ is a transition guard.
- $\sigma \in \Gamma \cup \{\mathtt{insample}^{(x)}, \mathtt{insample}^{(x)\prime}, \mathtt{insample}^{(y)}, \mathtt{insample}^{(y)\prime}\}$ is the output of the transition
- $\tau \in \{0, 1, 2\}$ indicates whether to assign into no variable, $\mathtt{x}$, or $\mathtt{y}$. In particular, note that only a single variable can be assigned into at a time.

Semantics here

- in particular, the noise for the input compared to each program variable is fresh/independent

Definition 5.2 (Valid Transition Alphabets)**.** A finite 2v-transition alphabet Σ_T is valid if it satisfies the following conditions:

- **Initialization:** There exist some $t^{(x)}_{init}, t^{(y)}_{init} \in \Sigma_T$ such that $t^{(x)}_{init} = (q_0, q_1, \texttt{true}, \sigma_0, 1)$ and $t^{(y)}_{init} = (q_1, q_2, \texttt{true}, \sigma_1, 2)$ for some $q_0, q_1, q_2 \in Q$, $\sigma_0, \sigma_1 \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$.
- **Determinism**: If Σ_T contains transition t, t' such that $t = (q, q', c, \sigma, \tau)$ and $t' = (q, q^*, c', \sigma', \tau')$, then c and c' must be disjoint events.
- **Output distinction**: If there exist some $\sigma, \sigma', \tau, \tau'$ such that $(q, q', \texttt{insample} < \texttt{x}, \sigma, \tau) \in \Sigma_T$ and $(q, q^*, \texttt{insample} \geq \texttt{x}, \sigma', \tau') \in \Sigma_T$, then $\sigma \neq \sigma'$. Additionally, at least one of $\sigma \in \Gamma$, $\sigma' \in \Gamma$ is true.
- **Non-input state condition**: For all states $q \in Q_{non}$, if there exists a transition $t = (q, q', c, \sigma, \tau)$ such that $t \in \Sigma_T$, then $c = \texttt{true}$.

Definition 5.3. Let Σ_T be a valid 2v-transition alphabet. A 2v-path over Σ_T is a sequence of transitions $t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$ such that for all i there exists a state $q_i \in Q$ such that $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$. A 2v-path is **complete** if it is in the form $t^{(x)}_{init} t^{(y)}_{init} \cdot \rho$ for some $\rho \in \Sigma_T^*$.

5.1.2 Multivariable Couplings

- separating variables

Definition 5.4. Let C be a

Proposition 5.5. *If there exist coupling strategies C_x, C_y that each independently satisfy the privacy constraint system for a 2v-transition t, then there exists a valid 2-coupling strategy C for t such that $cost(C) = cost(C_x) + cost(C_y)$.*

- combining independent couplings together

- new type of coupling

5.1.3 Cross-CouplingsCHANGE NAME

We introduce a new type of coupling strategy for a single (2v-)transition:

Valid coupling strategies

- coupling strategy is a series of shifts -

Lemma 5.6. *Let $X\langle 1\rangle \sim \textbf{\textit{Lap}}(\mu_x\langle 1\rangle, \frac{1}{d_x\varepsilon}), X\langle 2\rangle \sim \textbf{\textit{Lap}}(\mu_x\langle 2\rangle, \frac{1}{d_x\varepsilon})$ be random variables be random variables representing possible initial values of $\boldsymbol{x}$, respectively, and $Y\langle 1\rangle \sim \textbf{\textit{Lap}}(\mu_y\langle 1\rangle, \frac{1}{d_y\varepsilon}), Y\langle 2\rangle \sim \textbf{\textit{Lap}}(\mu_y\langle 2\rangle, \frac{1}{d_y\varepsilon})$ be random variables be random variables representing possible initial values of $\boldsymbol{x}$, respectively.*

Let $t \in \Sigma_T$ be a (2v-)transition $t = (q, q^*, c, \sigma, \tau)$ from q to $q^* \in Q$. Let $P(q) = (d_q, d'_q)$.

Let $\boldsymbol{in}\langle 1\rangle$, $\boldsymbol{in}\langle 2\rangle$ be an arbitrary valid adjacent input pair for t and let $o\langle 1\rangle$, $o\langle 2\rangle$ be random variables representing possible outputs of t given inputs $\boldsymbol{in}\langle 1\rangle$ and $\boldsymbol{in}\langle 2\rangle$, respectively.

Then for all $\gamma_T^{(x)}, \gamma_q^{(x)}, \gamma_q^{(x)\prime}, \gamma_T^{(y)}, \gamma_q^{(y)}, \gamma_q^{(y)\prime} \in [-1,1]$ *that satisfy the constraints*

$$\begin{cases} \gamma_q^{(x)} = 0 & \sigma = \boldsymbol{insample}^{(x)} \\ \gamma_q^{(x)\prime} = 0 & \sigma = \boldsymbol{insample}^{(x)\prime} \\ \gamma_q^{(y)} = 0 & \sigma = \boldsymbol{insample}^{(y)} \\ \gamma_q^{(y)\prime} = 0 & \sigma = \boldsymbol{insample}^{(y)\prime} \end{cases},$$

The following four statements hold:

1. *If* $c = \boldsymbol{insample} < \boldsymbol{x} \wedge \boldsymbol{insample} \geq \boldsymbol{y}$, *then* $\forall \varepsilon > 0$, *the lifting* $o\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}^{\# d\varepsilon} o\langle 2\rangle$ *is valid for* $d = |\min(\mu_y\langle 1\rangle - \mu_x\langle 1\rangle, 0)| + (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d}_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q'|)d_q'$.

2. *If* $c = \boldsymbol{insample} \geq \boldsymbol{x} \wedge \boldsymbol{insample} < \boldsymbol{y}$, *then* $\forall \varepsilon > 0$, *the lifting* $o\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}^{\# d\varepsilon} o\langle 2\rangle$ *is valid for* $d = |\min(\mu_x\langle 1\rangle - \mu_y\langle 1\rangle, 0)| + (|\hat{\mu_q}\langle 1\rangle - \hat{\mu_q}\langle 2\rangle + \gamma_x|)\hat{d}_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q'|)d_q'$.

3. *If* $c = \boldsymbol{insample} < \boldsymbol{x} \vee \boldsymbol{insample} \geq \boldsymbol{y}$, *then* $\forall \varepsilon > 0$, *the lifting* $o\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}^{\# d\varepsilon} o\langle 2\rangle$ *is valid for* $d =$.

4. *If* $c = \boldsymbol{insample} \geq \boldsymbol{x} \vee \boldsymbol{insample} < \boldsymbol{y}$, *then* $\forall \varepsilon > 0$, *the lifting* $o\langle 1\rangle\{(a,b) : a = \sigma \implies b = \sigma\}^{\# d\varepsilon} o\langle 2\rangle$ *is valid for* $d =$.

Proof. For convenience, we will rewrite X and Y as $X = \mu_x + \zeta_x$ and $Y = \mu_y + \zeta_y$, where $\zeta_x \sim \mathtt{Lap}(0, \frac{1}{d_x\varepsilon})$ and $\zeta_y \sim \mathtt{Lap}(0, \frac{1}{d_x\varepsilon})$. As before, we also write $\mathtt{insample}^{(x)} = \mathtt{in} + z^{(x)}$, where $z \sim \mathtt{Lap}(0, \frac{1}{d_q\varepsilon})$ and $\mathtt{insample}^{(x)\prime} = \mathtt{in} + z^{(x)\prime}$, where $z' \sim \mathtt{Lap}(0, \frac{1}{d_q'\varepsilon})$ (and symmetrically for $\mathtt{insample}^{(y)}$, $\mathtt{insample}^{(y)\prime}$).

We show cases (1) and (3). Cases (2) and (4) follow symmetrically.

As in the single variable case, suppose that we have the liftings

- $X\langle 1\rangle + \gamma_T^{(x)} (=)^{\#(|\mu_x\langle 1\rangle - \mu_x\langle 2\rangle + \gamma_T^{(x)}|) d_x\varepsilon} X\langle 2\rangle$.
- $\mathtt{insample}^{(x)}\langle 1\rangle + \gamma_q^{(x)} (=)^{\#(|-\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma_q^{(x)}|) d_q\varepsilon} \mathtt{insample}^{(x)}\langle 2\rangle$
- $\mathtt{insample}^{(x)\prime}\langle 1\rangle + \gamma_q^{(x)\prime} (=)^{\#(|-\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma_q^{(x)\prime}|) d_q'\varepsilon} \mathtt{insample}^{(x)\prime}\langle 2\rangle$
- $Y\langle 1\rangle + \gamma_T^{(y)} (=)^{\#(|\mu_y\langle 1\rangle - \mu_y\langle 2\rangle + \gamma_T^{(y)}|) d_y\varepsilon} Y\langle 2\rangle$.
- $\mathtt{insample}^{(y)}\langle 1\rangle + \gamma_q^{(y)} (=)^{\#(|-\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma_q^{(y)}|) d_q\varepsilon} \mathtt{insample}^{(y)}\langle 2\rangle$
- $\mathtt{insample}^{(y)\prime}\langle 1\rangle + \gamma_q^{(y)\prime} (=)^{\#(|-\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle - \gamma_q^{(y)\prime}|) d_q'\varepsilon} \mathtt{insample}^{(y)\prime}\langle 2\rangle$

Case 1:

Create the lifting $\zeta_x\langle 1\rangle + \gamma_{xy} (=)^{\#|\gamma_{xy}|\varepsilon} \zeta_y\langle 1\rangle$ where $\gamma_{xy} = -\min(\mu_y\langle 1\rangle - \mu_x\langle 1\rangle, 0)$.

Then observe that if $\mu_x\langle 1\rangle \geq \mu_y\langle 1\rangle$,

$$\begin{aligned}X\langle 1\rangle &= \mu_x\langle 1\rangle + \zeta_x\langle 1\rangle \\ &= \mu_x\langle 1\rangle + \zeta_y\langle 1\rangle + \mu_y\langle 1\rangle - \mu - x\langle 1\rangle \\ &= Y\langle 1\rangle\end{aligned}$$

Otherwise, if $\mu_x\langle 1\rangle < \mu_y\langle 1\rangle$, then

$$\begin{aligned}X\langle 1\rangle &= \mu_x\langle 1\rangle + \zeta_x\langle 1\rangle \\ &\leq \mu_y\langle 1\rangle + \zeta_y\langle 1\rangle \\ &= Y\langle 1\rangle\end{aligned}$$

so $X\langle 1\rangle \leq Y\langle 1\rangle$.

Further, create the lifting $z^{(x)}\langle 1\rangle(=)^{\#0}z^{(y)}\langle 1\rangle$. Note that then $\texttt{insample}^{(x)}\langle 1\rangle = \texttt{in}_i\langle 1\rangle + z_i^{(x)}\langle 1\rangle = \texttt{in}_i\langle 1\rangle + z_i^{(y)}\langle 1\rangle = \texttt{insample}^{(y)}\langle 1\rangle$.

Then because $x\langle 1\rangle \leq y\langle 1\rangle$, $\texttt{insample}^{(x)}\langle 1\rangle < x\langle 1\rangle \wedge \texttt{insample}^{(y)}\langle 1\rangle \geq y\langle 1\rangle$ must be false. Thus, $\texttt{insample}^{(x)}\langle 1\rangle < x\langle 1\rangle \wedge \texttt{insample}^{(y)}\langle 1\rangle \geq y\langle 1\rangle \implies \texttt{insample}^{(x)} < x\langle 2\rangle \wedge \texttt{insample}^{(y)} \geq y\langle 2\rangle$.

This suffices to show that if the transition is taken in run $\langle 1\rangle$, then it is also taken in run $\langle 2\rangle$. Thus, if any of

- $\sigma \in \Gamma$
- $\sigma = \texttt{insample}^{(x)}$ and $\gamma_q^{(x)} = 0$
- $\sigma = \texttt{insample}^{(x)\prime}$ and $\gamma_q^{(x)\prime} = 0$
- $\sigma = \texttt{insample}^{(y)}$ and $\gamma_q^{(y)} = 0$
- $\sigma = \texttt{insample}^{(y)\prime}$ and $\gamma_q^{(y)\prime} = 0$

are true, then the lifting holds as desired.

Case 3:

Create the lifting $\zeta_x\langle 1\rangle + \gamma_{xy}(=)^{\#|\gamma_{xy}|\varepsilon}\zeta_y\langle 1\rangle$ where $\gamma_{xy} = -\max(0, \mu_y\langle 1\rangle + \gamma_T^{(y)} - \mu_x\langle 1\rangle - \gamma_T^{(x)})$

Then if $\mu_x\langle 1\rangle + \gamma_T^{(x)} \leq \mu_y\langle 1\rangle + \gamma_T^{(y)}$,

$$\begin{aligned}X\langle 2\rangle &= X\langle 1\rangle + \gamma_T^{(x)} \\ &= \mu_x\langle 1\rangle + \zeta_x\langle 1\rangle + \gamma_T^{(x)} \\ &= \mu_x\langle 1\rangle + \zeta_y\langle 1\rangle + \mu_y\langle 1\rangle + \gamma_T^{(y)} - \mu_x\langle 1\rangle - \gamma_T^{(x)} + \gamma_T^{(x)} \\ &= Y\langle 1\rangle + \gamma_T^{(y)} \\ &= Y\langle 2\rangle\end{aligned}$$

Otherwise, if $\mu_x\langle 1\rangle + \gamma_T^{(x)} > \mu_y\langle 1\rangle + \gamma_T^{(y)}$, then

$$\begin{aligned} X\langle 2\rangle &= X\langle 1\rangle + \gamma_T^{(x)} \\ &= \mu_x\langle 1\rangle + \zeta_x\langle 1\rangle + \gamma_T^{(x)} \\ &\geq \mu_y\langle 1\rangle + \gamma_T^{(y)} + \zeta_y\langle 1\rangle \\ &= Y\langle 1\rangle + \gamma_T^{(y)} \\ &= Y\langle 2\rangle \end{aligned}$$

Thus, $X\langle 2\rangle \geq Y\langle 2\rangle$.

Further, create the lifting $z^{(x)}\langle 1\rangle (=)^{\#0} z^{(y)}\langle 1\rangle$. Note that then $\texttt{insample}^{(x)}\langle 1\rangle = \texttt{in}_i\langle 1\rangle + z_i^{(x)}\langle 1\rangle = \texttt{in}_i\langle 1\rangle + z_i^{(y)}\langle 1\rangle = \texttt{insample}^{(y)}\langle 1\rangle$.

Then because $X\langle 2\rangle \leq Y\langle 2\rangle$, $\texttt{insample}^{(x)}\langle 2\rangle < x\langle 2\rangle \vee \texttt{insample}^{(y)}\langle 2\rangle \geq y\langle 2\rangle$ must be true. Thus, $\texttt{insample}^{(x)}\langle 1\rangle < x\langle 1\rangle \vee \texttt{insample}^{(y)}\langle 1\rangle \geq y\langle 1\rangle \implies \texttt{insample}^{(x)} < x\langle 2\rangle \vee \texttt{insample}^{(y)} \geq y\langle 2\rangle$.

As before, if any of

- $\sigma \in \Gamma$
- $\sigma = \texttt{insample}^{(x)}$ and $\gamma_q^{(x)} = 0$
- $\sigma = \texttt{insample}^{(x)\prime}$ and $\gamma_q^{(x)\prime} = 0$
- $\sigma = \texttt{insample}^{(y)}$ and $\gamma_q^{(y)} = 0$
- $\sigma = \texttt{insample}^{(y)\prime}$ and $\gamma_q^{(y)\prime} = 0$

are true, then the lifting holds as desired.

$\square$

Paths: sanity check that you can't derive contradictory inequalities

Lemma 5.7. *Let $\rho = q_0 \to \ldots \to q_n$ be a complete path of length n. Let $\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle$ be arbitrary adjacent valid 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 $\boldsymbol{in}\langle 1\rangle$ and $\boldsymbol{in}\langle 2\rangle$, 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 following constraints:

1. *If $c_i(x) = \boldsymbol{insample} < \boldsymbol{x}$, then at least one of the following is true:*
   - $\gamma_i^{(x)} \leq \gamma_{at_x(i)}^{(x)}$
   - *$c_i = \boldsymbol{insample} < \boldsymbol{x} \vee \boldsymbol{insample} \geq \boldsymbol{y}$ and $\gamma_{(at_x(i), at_y(i))} = -\min(\boldsymbol{in}_{at_y(i)}\langle 1\rangle - \boldsymbol{in}_{at_x(i)}\langle 1\rangle, 0)$*
   - *$c_i = \boldsymbol{insample} < \boldsymbol{x} \wedge \boldsymbol{insample} \geq \boldsymbol{y}$ and $\gamma_{(at_x(i), at_y(i))} = -\max(0, \boldsymbol{in}_{at_y(i)}\langle 1\rangle + \gamma_{at_y(i)}^{(y)} - \boldsymbol{in}_{at_x(i)}\langle 1\rangle - \gamma_{at_x(i)}^{(x)})$*

2. *If $c_i(x) = \texttt{insample} \geq \texttt{x}$, then at least one of the following is true:*
   - $\gamma_i^{(x)} \geq \gamma_{at_x(i)}^{(x)}$
   - $c_i = \texttt{insample} \geq \texttt{x} \vee \texttt{insample} < \texttt{y}$ *and* $\gamma_{(at_x(i),at_y(i))} = -\min(\texttt{in}_{at_x(i)}\langle 1\rangle - \texttt{in}_{at_y(i)}\langle 1\rangle, 0)$
   - $c_i = \texttt{insample} \geq \texttt{x} \wedge \texttt{insample} < \texttt{y}$ *and* $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_x(i)}\langle 1\rangle + \gamma_{at_x(i)}^{(x)} - \texttt{in}_{at_y(i)}\langle 1\rangle - \gamma_{at_y(i)}^{(y)})$
3. *If $c_i(y) = \texttt{insample} < \texttt{y}$, then at least one of the following is true:*
   - $\gamma_i^{(y)} \leq \gamma_{at_y(i)}^{(y)}$
   - $c_i = \texttt{insample} \geq \texttt{x} \vee \texttt{insample} < \texttt{y}$ *and* $\gamma_{(at_x(i),at_y(i))} = -\min(\texttt{in}_{at_x(i)}\langle 1\rangle - \texttt{in}_{at_y(i)}\langle 1\rangle, 0)$
   - $c_i = \texttt{insample} \geq \texttt{x} \wedge \texttt{insample} < \texttt{y}$ *and* $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_x(i)}\langle 1\rangle + \gamma_{at_x(i)}^{(x)} - \texttt{in}_{at_y(i)}\langle 1\rangle - \gamma_{at_y(i)}^{(y)})$
4. *If $c_i(y) = \texttt{insample} \geq \texttt{x}$, then at least one of the following is true:*
   - $\gamma_i^{(x)} \geq \gamma_{at_x(i)}^{(x)}$
   - $c_i = \texttt{insample} < \texttt{x} \vee \texttt{insample} \geq \texttt{y}$ *and* $\gamma_{(at_x(i),at_y(i))} = -\min(\texttt{in}_{at_y(i)}\langle 1\rangle - \texttt{in}_{at_x(i)}\langle 1\rangle, 0)$
   - $c_i = \texttt{insample} < \texttt{x} \wedge \texttt{insample} \geq \texttt{y}$ *and* $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_y(i)}\langle 1\rangle + \gamma_{at_y(i)}^{(y)} - \texttt{in}_{at_x(i)}\langle 1\rangle - \gamma_{at_x(i)}^{(x)})$
5. *If $\sigma_i = \texttt{insample}^{(x)}$, then $\gamma_i^{(x)} = 0$*
6. *If $\sigma_i = \texttt{insample}^{(x)\prime}$, then $\gamma_i^{(x)\prime} = 0$*
7. *If $\sigma_i = \texttt{insample}^{(y)}$, then $\gamma_i^{(y)} = 0$*
8. *If $\sigma_i = \texttt{insample}^{(y)\prime}$, then $\gamma_i^{(y)\prime} = 0$*

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

5.1.4 Couplings for GDiPA

Definition 5.8. A GDiPA A is a tuple ().

Theorem 5.9. *A GDiPA A is $d\varepsilon$-differentially private for some $d > 0$ if and only if there exists a valid and finite cost coupling strategy for every looping branch of A.*

Lemma 5.10. *If there does not exist a single variable coupling strategy with respect to a variable $\texttt{x}$ for a looping branch ρ that satisfies the privacy constraint conditions for $\texttt{x}$, then ρ must contain either a leaking cycle, leaking pair, disclosing cycle, or privacy violating path in $\texttt{x}$.*

Lemma 5.11. *If there does not exist a valid and finite cost coupling strategy for every looping branch of A, then there must exist either a leaking cycle, disclosing cycle, or privacy violating path in a single variable or a non-cancelling leaking pair.*

Lemma 5.12. *If there exists a leaking cycle, disclosing cycle, or privacy violating path in a single variable or a non-cancelling leaking pair in a single variable in a GDiPA A, then A is not $d\varepsilon$-differentially private for any $d > 0$.*

6 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.

7 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 heuristiccally 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 K ¨ Opf, Federico Olmedo, and Santiago Zanella-B ´ Eguelin. 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.

8 Appendix

8.1 Proofs for section 4

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

Lemma 8.1. *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 $c_i = \texttt{insample} < \texttt{x}$, then $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ and $\gamma_{at(i)} = 1$.*
2. *If t_i is in a cycle and $c_i = \texttt{insample} \geq \texttt{x}$, then $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ and $\gamma_{at(i)} = -1$.*

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

Consider some t_i in a cycle where $c_i = \texttt{insample} < \texttt{x}$. Because C is has finite cost, we know from lemma 4.43 that $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ for all $\texttt{in}_i\langle 1\rangle \sim \texttt{in}_i\langle 2\rangle$. In particular, when $-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle = 1$, then $\gamma_i = 1$.

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

Lemma 8.2. *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 $C^* = (\gamma^*, \gamma^{*\prime})$ 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 $\texttt{insample} < \texttt{x}$ and guard $\texttt{insample} \geq \texttt{x}$ 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

$\square$

Proof of proposition 4.42. Make sure to put this proof after the DiPA counterexample proof

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

For a given path 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 $cycle([\rho])$ be the set of all transitions in $[\rho]$ that are contained within a cycle in $[\rho]$. Observe that for all $t \in cycle([\rho])$,

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

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 path ρ_n where $n > \lceil \frac{d}{\inf \delta(t_i)} \rceil + 1$. Then $cost(C_{\rho^*}) > d$, which implies that $\sup_{\rho \in [\rho]} cost(C_\rho) = \infty$, 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 $\texttt{insample} < \texttt{x}$, 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 $\texttt{insample} < \texttt{x}$, 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 $n \to \infty$.

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 $c_j = \texttt{insample} \geq \texttt{x}$ 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 $c_j = \texttt{insample} < \texttt{x}$ 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 path $\rho \in [\rho]$ 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:

$$\gamma(t_i)(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) = \begin{cases} 1 & t_i \in T_1 \\ -1 & t_i \in T_{-1} \\ \texttt{in}\langle 1\rangle - \texttt{in}\langle 2\rangle & c_i = \texttt{insample} < \texttt{x} \wedge t_{at(i)} \in T_1 \\ \texttt{in}\langle 1\rangle - \texttt{in}\langle 2\rangle & c_i = \texttt{insample} \geq \texttt{x} \wedge t_{at(i)} \in T_{-1} \\ 1 & c_i = \texttt{insample} \geq \texttt{x} \wedge t_{at(i)} \in T_1 \\ -1 & c_i = \texttt{insample} < \texttt{x} \wedge t_{at(i)} \in T_{-1} \end{cases}$$

$$\gamma'(t_i)(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) = \begin{cases} 0 & t_i \text{ outputs } \texttt{insample}' \\ \texttt{in}\langle 1\rangle - \texttt{in}\langle 2\rangle & \text{otherwise} \end{cases}$$

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 $C^* = (\gamma^*, \gamma^{*\prime})$ be the minimal-cost valid class coupling strategy such that for all $t \notin T_{un}$, $\gamma^*(t) = \gamma(t)$.

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

We additionally claim that $cost(C^*) \leq \sup_{\rho \in [\rho]} cost(C_\rho)$. The cost of C^* can be separated into costs attributed to $\gamma^{*\prime}$, 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 $\gamma^{*\prime}$ in C^* must be at most the maximimum coupling cost attributed to γ' over all path-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 paths in $[\rho]$, since we take the limit of all such shifts for ρ_n as $n \to \infty$.

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

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

Proof of lemma 4.49. Let $[\rho]$ be a path class in P that does not have a coupling strategy that satisfies the privacy constraint system.

Consider a "maximially" satisfied coupling strategy $C = (\gamma, \gamma')$ for $[\rho]$; i.e. there is no other coupling strategy C' for $[\rho]$ 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 path 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 4.56, 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. $\gamma_{at(i)} = 1 \implies \mathtt{L}-cycle$)

Case 1: (1) is unsatisfied for γ_i

In this case, $c_i = \mathtt{insample} < \mathtt{x}$ 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 $\mathtt{insample}$ or $\mathtt{insample}'$, 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.

Noe that the cycle containing t_i is also an $\mathtt{L}$-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 $c_{at(l)} = \mathtt{insample} < \mathtt{x}$. 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 $c_{l'} = \texttt{insample} \geq \texttt{x}$ 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 $\texttt{insample} \geq \texttt{x}$ 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, repectively.

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 $t_i at(i)$ 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 $\texttt{insample} \geq \texttt{x}$ 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 $c_{at(l)} = \texttt{insample} < \texttt{x}$. 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 $c_{at(l)} =$ `insample` $<$ `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 $c_{at(l)} =$ `insample` $<$ `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 $c_{at(k)} = \texttt{insample} < \texttt{x}$. 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 $c_{at(k)} = \texttt{insample} < \texttt{x}$. 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 $c_l = \texttt{insample} \geq \texttt{x}$

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 $c_{at(k)} = \texttt{insample} < \texttt{x}$. 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 $c_{at(k)} = \texttt{insample} < \texttt{x}$. 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 $\texttt{insample} < \texttt{x}$ 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 $c_{at(k)} = \texttt{insample} < \texttt{x}$. 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 $c_l = \texttt{insample} \geq \texttt{x}$ 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 $\texttt{insample} < \texttt{x}$ 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 $\texttt{insample} \geq \texttt{x}$ 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 $\texttt{insample} \geq \texttt{x}$. 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 $\texttt{insample} \geq \texttt{x}$ 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, $c_i = \texttt{insample} < \texttt{x}$. 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 $c_i = \texttt{insample} < \texttt{x}$. 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 $c_{at(i)} = \texttt{insample} < \texttt{x}$. 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 $t_{at(i)}$, $c_{at(k)} = \texttt{insample} < \texttt{x}$ 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 $c_j = \texttt{insample} \geq \texttt{x}$.

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 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 $t_{at(m)}$, $c_{at(k)} = \texttt{insample} \geq \texttt{x}$ 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 $c_l = \texttt{insample} < \texttt{x}$.

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 $c_{at(j)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(j)} < 1$ and, for all assignment transitions $t_{at(k)}$ between t_l and $t_{at(j)}$, $c_{at(k)} = \texttt{insample} < \texttt{x}$ 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 $\gamma_i = -\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle$ 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 $c_i = \texttt{insample} < \texttt{x}$ 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, $c_i = \texttt{insample} < \texttt{x}$, 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 $c_{at(i)} \neq \texttt{insample} < \texttt{x}$, 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 $c_i = \texttt{insample} < \texttt{x}$. Then the path from $t_{at(i)}$ (an assignment transition that outputs `insample`) to t_i is a privacy violating path.

If $c_{at(i)} = \texttt{insample} < \texttt{x}$, then let $c_{at(j)}$ be the earliest assignment transition such that $c_{at(j)} = \texttt{insample} < \texttt{x}$ and $\gamma_{at(j)} < 1$ and, for all assignment transitions $t_{at(k)}$ between $t_{at(j)}$ and t_i, $c_{at(k)} = \texttt{insample} < \texttt{x}$ 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 $\gamma'_i = -\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle$ 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. □

Definition 8.3 (Leaking Cycles [6])**.** A path $\rho = q_0 \to \ldots \to q_n$ in a DiPA A is a leaking path if there exist indices i, j where $0 \leq i < j < n$ such that the i'th transition $q_i \to q_{i+1}$ in ρ is an assignment transition and the guard of the transition $q_j \to q_{j+1}$ is not `true`. If ρ is also a cycle, then we call it a leaking cycle.

Definition 8.4 ([6])**.** A cycle ρ in a DiPA A is an `L`-cycle if for some transition $q_i \to q_{i+1}$ in ρ, $\mathtt{guard}(q_i \to q_{i+1}) = \mathtt{insample} < \mathtt{x}$. Similarly, ρ is a `G`-cycle if for some transition $q_i \to q_{i+1}$ in ρ, $\mathtt{guard}(q_i \to q_{i+1}) = \mathtt{insample} \geq \mathtt{x}$.

Additionally, a path ρ of a DiPA A is an `AL`-path (respectively, `AG`-path) if all assignment transitions in ρ have guard $\mathtt{insample} < \mathtt{x}$ (respectively, $\mathtt{insample} \geq \mathtt{x}$)

Definition 8.5 (Leaking Pairs [6])**.** A pair of cycles (C, C') is called a leaking pair if one of the following two conditions is satisfied.

1. C is an `L`-cycle, C' is a `G`-cycle and there is an `AG`-path from a state in C to a state in C'.

2. C is a `G`-cycle, C' is an `L`-cycle and there is an `AL`-path from a state in C to a state in C'.

Definition 8.6 (Disclosing Cycles [6])**.** A cycle $C = q_0 \to \ldots \to q_n \to q_0$ of a DiPA A is a disclosing cycle if there is an i, $0 \leq i < |C|$ such that $q_i \in Q_{in}$ and the transition $q_i \to q_{i+1}$ that outputs either `insample` or `insample`'.

Definition 8.7 (Privacy Violating Paths [6])**.** We say that a path $\rho = q_0 \to \ldots \to q_n$ of a DiPA A is a privacy violating path if one of the following conditions hold:

- $tail(\rho)$ is an `AG`-path (resp., `AL`-path) such that $last(\rho)$ is in a `G`-cycle (resp., `L`-cycle) and the 0th transition $q_0 \to q_1$ is an assignment transition that outputs `insample`.
- ρ is an `AG`-path (resp., `AL`-path) such that q_n is in a `G`-cycle (resp., `L`-cycle) and the first transition $q_0 \to q_1$ has guard $\texttt{insample} < \texttt{x}$ (resp., $\texttt{insample} \geq \texttt{x}$) and outputs `insample`
- ρ is an `AG`-path (resp., `AL`-path) such that q_0 is in an `L`-cycle (resp., `G`-cycle) and the last transition $q_{n-1} \to q_n$ has guard $\texttt{insample} \geq \texttt{x}$ (resp., $\texttt{insample} < \texttt{x}$) and outputs `insample`