# 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" $\varepsilon$, 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 [12].

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].

Our contributions:

- We develop a simple program model centred around comparing an input to a threshold value for which we show that there is a simple and, most notably, complete class of privacy proofs built from approximate liftings
  - Every program in our model is a language over an alphabet of individual program transitions
  - We can use the structure of these programs to automatically build privacy proofs for any program using approximate liftings
  - Importantly, if we cannot build a privacy proof for a program, we show that it is in fact not private at all; i.e. these privacy proofs are complete
  - We also demonstrate methods for computing the minimal privacy cost of a coupling proof directly and in approximation
- This program model exactly coincides with a previously developed, automaton-theoretic, model known as DiPA.

- We additionally demonstrate how this model, and along with it coupling proofs, can be extended to accomodate two threshold variables such that coupling proofs again characterize the model completely.

# 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 $\sigma$ of a private algorithm $A$, the probability of obtaining $\sigma$ for a dataset with some individual Alex is close (measured by a multiplicative factor) to the probability of obtaining $\sigma$ 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 $|\{i : x_i \neq x'_i\}| \leq 1$[1].

We thus formalize privacy under this framework as follows.

**Definition 2.2** (Pure Differential Privacy)**.** For some $\varepsilon > 0$, a randomized algorithm $A$ is $\varepsilon$-differentially private if, for all pairs of **adjacent** datasets $X$ and $X'$ and all events $E \subseteq \mathtt{im}(A)$,

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

The privacy parameter $\varepsilon$ is traditionally thought of as analogous to the 'privacy cost' of a program; the larger $\varepsilon$ is, the more privacy is "lost".

It is possible that some algorithms will fail to be differentially private for *any* $\varepsilon > 0$; we will often refer to algorithms or programs being "differentially private" or just "private" (without any reference to an $\varepsilon > 0$) to indicate that there exists *some* $\varepsilon > 0$ for which the algorithm is $\varepsilon$-differentially private.

We also introduce the concept of max divergence as a convenient way to "measure" this privacy cost.

**Definition 2.3** (Max Divergence)**.** For any two probability distributions $P, Q$ over a shared event space $E$, the max-divergence of $P$ and $Q$ is $D_\infty(P||Q) = \max_{e \in E} \ln\left(\frac{P(e)}{Q(e)}\right)$.

Max divergence is closely related to differential privacy; it immediately follows from definition that an algorithm $A$ is $\varepsilon$-DP if and only if for all adjacent inputs $X \sim X'$, $D_\infty(A(X)||A(X')) \leq \varepsilon$.

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

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

**Theorem 2.4** (Standard Composition)**.** *If $A$ is $\varepsilon_1$-differentially private and, for all $\sigma$, $B(\sigma, \cdot)$ is $\varepsilon_2$-differentially private, then $B(A(X), X)$ is $\varepsilon_1 + \varepsilon_2$-differentially private.*

Composition therefore allows us to view privacy parameters $\varepsilon$ as a "budget" for privacy-leaking operations in a program. Many[2] 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 2.5.** The ($\ell_1$-)sensitivity of a function $f : X \to \mathbb{R}$, often denoted $\Delta f$, 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 2.6.** The Laplace distribution $\texttt{Lap}(\mu, b)$ with mean $\mu$ 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 $\texttt{Lap}(b)$.

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

**Theorem 2.7** (Theorem 3.6 [8])**.** *For a function $f$ with sensitivity $\Delta$, $A(X) = f(X) + \texttt{Lap}(\frac{\Delta}{\varepsilon})$ is $\varepsilon$-differentially private.*

We will eventually consider a program model where we are given a potentially infinite *sequence* of real-valued query functions $q_0, q_1, \ldots$, each with sensitivity at most $\Delta$.

## 2.2 Deciding Privacy

**Sky: ¡¡this section isn't super relevant anymore - remove?¿¿**

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.

[2] generic platitude - reword

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

**Theorem 2.8** ([2])**.** *The problem of determining whether a program from a certain class of algorithms with assignments, conditionals, and while loops is $\varepsilon$-differentially private is undecidable*[3]*.*

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 are useful when analyzing the relationship between two probablistic processes. [past literature on couplings]

A previous line of work has extended the concept of couplings to reason about privacy and private processes.

In particular, objects known as **approximate liftings** [4, 5, 9, 3] allow us to apply couplings to the realm of differential privacy.

**Definition 3.2.** Let $A_1, A_2$ be two probability spaces[4]. We say a distribution $\mu_1$ on $A_1$ and $\mu_2$ on $A_2$ are related by the **$\varepsilon$-lifting** of the relation $\Psi \subseteq A_1 \times A_2$ (written $\mu_1 \Psi^{\#\varepsilon} \mu_2$) if there exist two **witness distributions** $\mu_L, \mu_R$ 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. $\sup_{E \subseteq A_1 \times A_2} (\mathbb{P}_{x \leftarrow \mu_L}[x \in E] - e^{\varepsilon} \mathbb{P}_{x \leftarrow \mu_R}[x \in E]) \leq 0$

In some sense, approximate liftings can be considered "half-couplings", where half of $\mu_L$ is coupled with half of $\mu_R$. Because of the close relationship between couplings and approximate liftings, we will often use "couplings" to generically refer to approximate liftings and couplings together.

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

[3] rephrase?

[4] may need to formally rewrite this at some point

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

However, we will generally work with the following relaxation, which still allows us to prove that an algorithm is private:

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

Almost every standard result about differential privacy can be restated in terms of couplings; for this paper, we will primarily rely on the fact that couplings can be composed and that we can couple together Laplace random variables.

**Theorem 3.5** (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 $A_2 R_2^{\#\varepsilon_1+\varepsilon_2} B_2$.*

**Proposition 3.6** (Laplace Mechanism for Liftings)**.** *If $X_1 \sim \texttt{Lap}(\mu_1, \frac{1}{\varepsilon})$ and $X_2 \sim \texttt{Lap}(\mu_2, \frac{1}{\varepsilon})$, then $X_1(=)^{\#\varepsilon|\mu_1-\mu_2|} X_2$.*

The structure of theorems 3.4 and 3.5 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

For illustrative purposes, we provide a manual proof of privacy for a notoriously tricky algorithm, the Sparse Vector Technique (SVT). **Sky: ¡¡do we need some explanation that this proof is technically original bc of the rewordings? also can remove this section for space¿¿**

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 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[5].

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 $\Delta$, $c \in \mathbb{N}$.

$\varepsilon_1, \varepsilon_2 \leftarrow \frac{\varepsilon}{2}, \rho \leftarrow \texttt{Lap}(\frac{\Delta}{\varepsilon_1})$, $count \leftarrow 0$
**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** $\top$
    $count \leftarrow count + 1$
    **if** $count \geq c$ **then**
      **break**
    **end if**
  **else**
    **output** $\bot$
  **end if**
**end for**

**Theorem 3.7.** *Sparse Vector Technique is $\varepsilon$-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 $\sigma \in \{\bot, \top\}^n$. 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 $\tilde{T}\langle 1\rangle + \Delta = \tilde{T}\langle 2\rangle \wedge z_i\langle 1\rangle = z_i\langle 2\rangle \implies (S_i\langle 1\rangle < \tilde{T}\langle 1\rangle \implies S_i\langle 2\rangle < \tilde{T}\langle 2\rangle)$.

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

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

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

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 $\varepsilon$-differentially private. □

# 4 Automatically Proving Privacy using Couplings

Our approach is to build up a program model through the lens of regular languages. In this view, a program is simply a regular language comprised of possible program paths (or executions) where each path corresponds to a word in the language. This approach allows us to simultaneously build approximate liftings for each path that prove the overall privacy of a program.

We begin with single transitions between two program locations, which correspond directly to characters in an alphabet.

**Sky: ¡¡todo: add a note about treating $\varepsilon$ as a parameter to the program¿¿**

## 4.1 Individual Transitions

Transitions in our program model are transitions between two program **locations**. Naturally then, we must fix some underlying location space for our transitions.

**Definition 4.1** (Program locations)**.** $Q$ is a finite set of program locations partitioned into input locations $Q_{in}$ and non-input locations $Q_{non}$. We will also associate each program location with two **noise parameters** with the function $P : Q \to \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$.

In our program model, we have a singular persistent real-valued variable `x`, which can be thought of as a threshold value to compare input to in algorithms like SVT. We can thus define a set of **transition guards**, boolean conditions that determine whether or not a transition should proceed.

**Definition 4.2** (Transition guards)**.** $\mathcal{C} = \{\texttt{true}, \texttt{insample} < \texttt{x}, \texttt{insample} \geq \texttt{x}\}$ is a set of (single-variable) **transition guards**.

We can now formally define individual transitions:

**Definition 4.3** (Transitions)**.** Given a finite set of program locations $Q$, a transition is a tuple $t = (q, q', c, \sigma, \tau)$, where

- $q \in Q$ is the initial location.
- $q' \in Q$ is the following location.
- $c \in \mathcal{C}$ is a transition guard that determines if $t$ is traversed.
- $\sigma \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$ is the output of $t$.
- $\tau \in \{\texttt{true}, \texttt{false}\}$ 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 $q \to q'$.

### 4.1.1 Transition Semantics

We can think of each transition as defining an extremely small program; beginning in some location $q$, a transition takes in some real valued input `in`, compares it to some threshold `x`, and, depending on the result of the comparison, moves to a location $q'$ while outputting a value $\sigma$ and possibly updating the value of `x`.

More specifically, given some threshold value `x`, each transition $t = (q, q', c, \sigma, \tau)$ will first read in a real number input `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 two variables $\texttt{insample} = \texttt{in} + z$ and $\texttt{insample}' = \texttt{in} + z'$. If the guard $c$ is satisfied, then we transition to location $q'$, outputting $\sigma$ and, if $\tau = \texttt{true}$, reassigning $\texttt{x} = \texttt{insample}$.

We can describe the semantics of a transition as a function that maps an initial program **location** and a real-valued input to a distribution of subsequent program locations and an output value.

To be precise, a program location is a tuple consisting of a program location and a value for the program variable `x`. Let $S = Q \times \mathbb{R}$ be the set of all possible program locations. As expected, every possible input is simply an element of $\mathbb{R}$. An output can either be a symbol from some finite alphabet $\Gamma$ or a real number; thus, the set of all possible output events is $\Gamma \cup \Sigma$, where $\Sigma$ is some $\sigma$-algebra over $\mathbb{R}$. In particular, we will take $\Sigma$ to be the standard $\sigma$-algebra of all Lebesgue measurable sets.

It follows that the semantics of a transition $t$ can be defined as a function $\Phi_t : S \times \mathbb{R} \to dist(S \times (\Gamma \cup \mathbb{R} \cup \lambda))$ that maps an initial program location and an input to a distribution of subsequent program locations and an output event following the expected semantics; $\lambda$ here denotes the empty string (i.e. no output).

[If we need space, just say that we use the natural semantics and move the following section to appendix]

Let $q \in Q$ be an initial location, $\texttt{x} \in \mathbb{R}$ be an initial threshold value, and $P(q) = (d_q, d'_q)$ be the distributional parameters associated with $q$. Let $t = (q, q', c, \sigma, \tau)$ be the transition whose semantics we are defining.

Let $\texttt{in} \in \mathbb{R}$ be a real-valued input and $o \in (\Gamma \cup \Sigma \cup \lambda)$ be a possible output event of $t$.

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 corresponding to `insample` and $\texttt{insample}'$.

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

Let $T$ be the event that $c$ is satisfied given `x` and $\texttt{insample} = I$ and let $O$ be the event that, if $\sigma = \texttt{insample}$, then $I$ matches $o$ and if $\sigma = \texttt{insample}'$, then $I'$ matches $o$.

Then $\Phi_t((q, \texttt{x}), \texttt{in})$ is a distribution that assigns probabilities to output events as follows:

If $\tau = \texttt{false}$, then $\Phi_t((q, \texttt{x}), \texttt{in})$ assigns probability 0 to all events $((q', \texttt{x}'), o)$ such that $\texttt{x}' \neq \texttt{x}$. For all other events $((q', \texttt{x}), o)$. $\Phi_t((q, \texttt{x}), \texttt{in})$ assigns probability $\mathbb{P}[T \wedge O]$ to $((q', \texttt{x}), o)$ and probability $\mathbb{P}[\neg T]$ to the event $((q_{term}, \texttt{x}), \lambda)$.

Similarly, if $\tau = \texttt{true}$, then $\Phi_t((q, \texttt{x}), \texttt{in})$ assigns probability $\mathbb{P}[T \wedge O]$ to the event $((q', I), o)$ and assigns probability $\mathbb{P}[\neg T]$ to the event $((q_{term}, I), \lambda)$.

Here, $q_{term}$ is a sink or terminal location with no transitions allowed out of it.

We primarily are concerned with the probability that a transition "succeeds", that is, the probability that from location $q$, the program defined by $t$ transitions to location $q'$ and outputs a certain value.

We denote this probability as $\mathbb{P}[\texttt{x}, t, \texttt{in}, o]$, where $\texttt{x} \in \mathbb{R}$ is the initial value of $\texttt{x}$, $t$ is a transition, $\texttt{in} \in \mathbb{R}$ is a real-valued input, and $o \in \Gamma \cup \Sigma$ is a possible output of $t$. Specifically, $\mathbb{P}[\texttt{x}, t, \texttt{in}, o] = \int_{-\infty}^{\infty} \mathbb{P}[\texttt{x} \text{ is updated to have value } \texttt{x}'] \Phi_t((q, \texttt{x}), \texttt{in})((q', \texttt{x}'), o) d\texttt{x}'$.

Note that this is an aggregated probability over all possible final values of $\texttt{x}$ - we do not particularly care what the final value of the threshold is.

### 4.1.2 Constructing an alphabet

As mentioned, we will consider individual transitions as part of an *alphabet*; in particular, 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.

However, in order to ensure that these languages do in fact correspond to semantically coherent programs, we need to restrict possible transition alphabets as follows.

**Definition 4.4** (Valid Transition Alphabets)**.** Let $\Sigma_T$ be a finite alphabet of transitions. We will call $\Sigma_T$ **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 $\Sigma_T$. 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 $\Sigma_T$.
- **Output distinction**: If there exist some $\sigma, \sigma', \tau, \tau'$ such that $(q, q', \texttt{insample} < \texttt{x}, \sigma, \tau) \in \Sigma_T$ and $(q, q^*, \texttt{insample} \geq \texttt{x}, \sigma', \tau') \in \Sigma_T$, then $\sigma \neq \sigma'$. Additionally, at least one of $\sigma \in \Gamma$, $\sigma' \in \Gamma$ is true.
- **Non-input location condition**: For all locations $q \in Q_{non}$, if there exists a transition $t = (q, q', c, \sigma, \tau)$ such that $t \in \Sigma_T$, then $c = \texttt{true}$.

### 4.1.3 Couplings

We will now construct couplings for transitions with the aim of using them as building blocks for proofs of privacy. However, we refrain from defining what it means for a single

transition to be "private" in this section and focus solely on building approximate liftings. Sky: ¡¡repetitive¿¿

First, we need to adapt standard privacy definitions to our specific setting; recall that $\texttt{in}$, in reality, represents a **function** of some underlying dataset. This means that 'closeness' in this context is defined as follows:

**Definition 4.5** (Adjacency)**.** Two inputs $\texttt{in} \sim_\Delta \texttt{in}'$ are $\Delta$-adjacent if $|\texttt{in} - \texttt{in}'| \leq \Delta$. If $\Delta$ is not specified, we assume that $\Delta = 1$.

Additionally, recall that some program locations ($Q_{non}$) in our model do not read in any input; to model this, we require that whenever input is passed into a non-input location, the actual input value is always 0.

**Definition 4.6** (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 $(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) \in \mathbb{R}^2$ such that:

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

We will abuse notation and denote that $(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle)$ is a valid adjacent input pair by also writing $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$. In general, we will assume that all inputs are valid unless explicitly otherwise stated.

We are now ready to state our major coupling lemma, which defines a family of possible couplings that will be useful for proving the privacy of programs built from an alphabet of transitions.

**Lemma 4.7.** *Let $X\langle 1\rangle \sim \texttt{Lap}(\mu\langle 1\rangle, \frac{1}{d_x\varepsilon}), X\langle 2\rangle \sim \texttt{Lap}(\mu\langle 2\rangle, \frac{1}{d_x\varepsilon})$ be random variables be random variables representing possible initial values of $\boldsymbol{x}$. And let $t = (q, q^*, c, \sigma, \tau)$ be a transition 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*

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

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

*Proof.* Fix $\varepsilon > 0$.

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

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 location $q$ 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 $\sigma$ 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 $X\langle 1 \rangle + \gamma_x (=)^{\#(|\mu\langle 1 \rangle - \mu\langle 2 \rangle + \gamma_x|) d_x \varepsilon} X\langle 2 \rangle$. 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 $\sigma = \texttt{insample}'$ 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

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

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 $d = (|\mu\langle 1\rangle - \mu\langle 2\rangle + \gamma_x|)d_x + (|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma_q|)d_q + (|-\texttt{in}\langle 1\rangle + \texttt{in}\langle 2\rangle - \gamma'_q|)d'_q$.

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

We can thus think of couplings for a transition as being **parameterized** by $\gamma_x$, $\gamma_q$, and $\gamma'_q$. In particular, we will view choices of $\gamma_x, \gamma_q$, and $\gamma'_q$ as a **strategy** for proving that a transition is differentially private.

**Sky: ¡¡cut these two definitions? they're only illustrative anyway¿¿**

**Definition 4.8** (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 4.9** (Validity of a coupling strategy)**.** A coupling strategy $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i)$ for a transition $t_i$ is **valid** if the constraints

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

are all satisfied.

## 4.2 Program Paths

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

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

**Definition 4.10** (Program paths)**.** Let $\Sigma_T$ be a valid transition alphabet with underlying location 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 $c_i, \sigma_i, \tau_i$. We will often notate an path $\rho$ as $\rho = q_0 \to q_1 \to \ldots \to q_n$.

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

The length of a path $\rho$ is simply the number of transitions that are concatenated together to form $\rho$.

We define some useful notation for dealing with paths and sequences more generally.

**Definition 4.11.** Given a path (or sequence) $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$, the **tail** of $\rho$ is defined as $tail(\rho) = t_1 \cdot \ldots \cdot t_{n-1}$. We may additionally use the notation $\rho_{i:j}$ to represent the subpath (or subsequence) $q_i \to q_{i+1} \to \ldots \to q_j$ of $\rho$. Using this notation, $tail(\rho) = \rho_{1:} = \rho_{1:n}$.

Whereas an individual transition reads in one real-valued input and outputted one output value, a path reads in a **sequence** of inputs and outputs a sequence of outputs, one for each transition in the path.

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

**Definition 4.12.** For a path $\rho$ of length $n$, an input sequence $\texttt{in} \in \mathbb{R}^n$ is valid if, for all $q_i$ in $\rho$ such that $q_i \in Q_{non}$, $\texttt{in}_i = 0$.

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

Interestingly, the constraints on valid transition alphabets, specifically the constraints of determinism and output distinction, mean that outputs uniquely correspond to paths; in other words, given a valid transition alphabet, knowing an output sequence uniquely determines which path must have produced the output.

**Proposition 4.13.** *Let $\Sigma_T$ be a valid transition alphabet and let $\Gamma$ be the finite output alphabet associated with $\Sigma_T$. Let $O \subset (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\})^*$ be the set of all possible outputs of complete paths over $\Sigma_T$. There exists an injection $f : \Sigma_T \to t_{init}\Sigma_T^*$ from the set of all possible outputs to complete paths over $\Sigma_T$.*

### 4.2.1 Path Semantics

As with transitions, we can think of paths as very limited programs consisting of a series of transitions concatenated together with a persistent threshold variable `x`. Naturally, paths will now consider as input a **sequence** of real numbers, and similarly output a **sequence** of real numbers or symbols - each transition reads in an input and outputs some value.

In particular, the semantics of a path $\rho = q_0 \to \ldots \to q_n = t_0 t_1 \cdots t_{n-1}$ can be defined as the function $\Phi_\rho((q, \texttt{x}), \texttt{in}) : S \times \mathbb{R}^n \to dist(S \times (\mathbb{R} \cup \Gamma \cup \lambda)^n)$ mapping an initial program state and a input sequence to a distribution of final program states and output sequences.

$\Phi_\rho$ can be computed by composing the program semantics of individual transitions in the natural manner:

As before, let $\texttt{in} \in \mathbb{R}^n$ be a sequence of inputs and let $\sigma \in (\Sigma \cup \Gamma \cup \lambda)^n$ be a sequence of possible output events. Let $I \sim \texttt{Lap}(\texttt{in}_0, \frac{1}{d_{q^\varepsilon}})$ and $I' \sim \texttt{Lap}(\texttt{in}_0, \frac{1}{d'_{q^\varepsilon}})$ be independent random variables corresponding to `insample` and `insample`$'$.

Let $t_0 = (q, q', c_0, \sigma_0, \tau_0)$

Then

$$\Phi_\rho((q,\mathtt{x}),\mathtt{in})((q',\mathtt{x}'),\sigma) = \begin{cases} 1 & ((q',\mathtt{x}'),\sigma) = ((q_0,\mathtt{x}),\lambda) \wedge n = 0 \\ \mathbb{P}[c_0 \text{ is not satisfied}] & ((q',\mathtt{x}'),\sigma) = ((q_{term},\mathtt{x}),\lambda) \wedge \tau_0 = \mathtt{fa} \\ \mathbb{P}[c_0 \text{ is not satisfied}] & ((q',I),\sigma) = ((q_{term},\mathtt{x}),\lambda) \wedge \tau_0 = \mathtt{tru} \\ \quad \mathbb{P}[c_0 \text{ is satisfied}]* & \\ \Phi_{tail(\rho)}((q',\mathtt{x}),tail(\mathtt{in}))((q',\mathtt{x}'),tail(\sigma)) & \sigma_0 \in \Gamma \wedge \tau_0 = \mathtt{false} \\ \quad \mathbb{P}[c_0 \text{ is satisfied}]* & \\ \Phi_{tail(\rho)}((q',I),tail(\mathtt{in}))((q',\mathtt{x}'),tail(\sigma)) & \sigma_0 \in \Gamma \wedge \tau_0 = \mathtt{true} \\ \quad \mathbb{P}[c_0 \text{ is satisfied} \wedge I \text{ matches } \sigma_0]* & \\ \Phi_{tail(\rho)}((q',\mathtt{x}),tail(\mathtt{in}))((q',\mathtt{x}'),tail(\sigma)) & \sigma_0 = \mathtt{insample} \wedge \tau_0 = \mathtt{false} \\ \quad \mathbb{P}[c_0 \text{ is satisfied} \wedge I \text{ matches } \sigma_0]* & \\ \Phi_{tail(\rho)}((q',I),tail(\mathtt{in}))((q',\mathtt{x}'),tail(\sigma)) & \sigma_0 = \mathtt{insample} \wedge \tau_0 = \mathtt{true} \\ \quad \mathbb{P}[c_0 \text{ is satisfied} \wedge I' \text{ matches } \sigma_0]* & \\ \Phi_{tail(\rho)}((q',\mathtt{x}),tail(\mathtt{in}))((q',\mathtt{x}'),tail(\sigma)) & \sigma_0 = \mathtt{insample}' \wedge \tau_0 = \mathtt{false} \\ \quad \mathbb{P}[c_0 \text{ is satisfied} \wedge I' \text{ matches } \sigma_0]* & \\ \Phi_{tail(\rho)}((q',I),tail(\mathtt{in}))((q',\mathtt{x}'),tail(\sigma)) & \sigma_0 = \mathtt{insample}' \wedge \tau_0 = \mathtt{true} \\ 0 & otherwise \end{cases}$$

As before, we primarily care about the probability of a "successful" execution of a path with a particular output, which we will denote as $\mathbb{P}[\mathtt{x}_0, \rho, \mathtt{in}, \sigma]$, where $\mathtt{x} \in \mathbb{R}$ is the initial value of $\mathtt{x}$, $\rho$ is the path we are concerned about, $\mathtt{in} \in \mathbb{R}^n$ is a real-valued input sequence, and $o \in (\Gamma \cup \Sigma \cup \lambda)^n$ is a possible output sequence of $\rho$. As before, $\mathbb{P}[\mathtt{x}_0, \rho, \mathtt{in}, \sigma] = \int_{-\infty}^{\infty} \mathbb{P}[\mathtt{x} \text{ is updated to have value } \mathtt{x}']\Phi_\rho((q_0,\mathtt{x}_0),\mathtt{in})((q_n,\mathtt{x}'),\sigma)d\mathtt{x}'$, where $t_0 = (q_0, q_1, c_0, \sigma_0, \tau_0)$ is the first character of $\rho$ and $t_{n-1} = (q_{n-1}, q_n, c_{n-1}, \sigma_{n-1}, \tau_{n-1})$ is the final character of $\rho$.

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

### 4.2.2 Privacy

By leveraging the construction of couplings for individual transitions, we can construct a set of approximate liftings for entire paths

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

**Definition 4.14** (Adjacency for a sequence of inputs)**.** Two input sequences $\{\alpha_i\}_{i=1}^n, \{\beta_i\}_{i=1}^n$ of length $n$ are $\Delta$-adjacent (notated $\alpha \sim_\Delta \beta$) if, for all $i \in [1 \dots n]$, $|\alpha_i - \beta_i| \leq \Delta$.

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

Thus, we get the following definition of privacy for complete paths:

**Definition 4.15** ($d\varepsilon$-differential privacy for a path)**.** A complete path $\rho$ 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 $\sigma$ of length $n$, $\mathbb{P}[\rho, \texttt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[\rho, \texttt{in}\langle 2\rangle, \sigma]$.

Because, under our model, a program is simply a collection of paths, it will also be convenient to define a notion of privacy for sets of (complete) paths:

**Definition 4.16.** 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 $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $\mathbb{P}[\rho, \texttt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[\rho, \texttt{in}\langle 2\rangle, \sigma]$.

We observe that because of the path-output correspondence, the following definition of privacy is equivalent:

**Definition 4.17.** Let $S$ be a set of complete paths; $S$ is $d\varepsilon$-differentially private for some $d > 0$ if, for all paths $\rho \in S$, $\rho$ is $d\varepsilon$-differentially private.

Note that we slightly redefine $\varepsilon$-differential privacy as $d\varepsilon$-differential privacy, treating $\varepsilon$ 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[6], since if we are targeting $\varepsilon^*$-differential privacy overall, we can always take $\varepsilon = \frac{\varepsilon^*}{d}$.

### 4.2.3 Concatenating couplings

Just as individual transitions can be concatenated to form program paths, we can compose together couplings associated with each transition to produce a coupling proof of privacy for an entire path.

If $\rho[\texttt{in}]$ is a random variable representing the output of $\rho$ given input sequence $\texttt{in}$, then in order to show that a program path $\rho$ is differentially private we want to create the coupling $\rho[\texttt{in}\langle 1\rangle]\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon}\rho[\texttt{in}\langle 2\rangle]$ for some $d > 0$ for all adjacent inputs $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$ and all possible outputs $\sigma$.

As it turns out, naively composing together the couplings from lemma 4.7 are essentially 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 $\rho$ (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 4.18** (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 $\rho$. Additionally, for every transition $t_i$ in $\rho$, let $t_{at(i)}$ be the most recent assignment transition in $\rho$; i.e., $at(i) = \max\{j < i : t_j \in A_\rho\}$. If such a $j$ does not exist, we set $at(i) = -1$.

[6] [6] note that it is not entirely clear how this differs from standard differential privacy, but that the known decidability result does not apply here

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

Thus, for an individual transition $t_i$ of $\rho$, 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 4.19.** *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 $\sigma$ of $\rho$ of length $n$ and let $\sigma\langle 1\rangle$, $\sigma\langle 2\rangle$ be random variables representing possible outputs of $\rho$ 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*

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

*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 4.7, we know that we can create the couplings $\mathtt{insample}_i\langle 1\rangle + \gamma_i (=)^{\#(|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i|)d_i\varepsilon} \mathtt{insample}_i\langle 2\rangle$ and $\mathtt{insample}_i'\langle 1\rangle + \gamma_i' (=)^{\#(|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i'|)d_i'\varepsilon} \mathtt{insample}_i'\langle 2\rangle$ for all $q_i$ in $\rho$.

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

$$\begin{cases} \gamma_i \le \gamma_x & c_i = \mathtt{insample} < \mathtt{x} \\ \gamma_i \ge \gamma_x & c_i = \mathtt{insample} \ge \mathtt{x} \\ \gamma_i = 0 & \sigma_i = \mathtt{insample}_i \\ \gamma_i' = 0 & \sigma_i = \mathtt{insample}_i' \end{cases},$$

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

$$\begin{cases} \gamma_i \le \gamma_{at(i)} & c_i = \mathtt{insample} < \mathtt{x} \\ \gamma_i \ge \gamma_{at(i)} & c_i = \mathtt{insample} \ge \mathtt{x} \\ \gamma_i = 0 & \sigma_i = \mathtt{insample}_i \\ \gamma_i' = 0 & \sigma_i = \mathtt{insample}_i' \end{cases}$$

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 `insample` and `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 $d = \sum_{i=0}^{n-1}(|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i'|)d_i'$. □

As with individual transitions, lemma 4.19 implicitly defines an entire family of possible coupling proofs that demonstrate the privacy of a path; we call instantiations of these coupling proofs **coupling strategies**.

**Definition 4.20.** For a complete path $\rho$ of length $n$, **coupling strategy** is a tuple of 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 $\rho$ for every possible pair of adjacent input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$. 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 4.21.** For a complete path $\rho$ 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

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

Thus, if we have a **valid** coupling strategy $C$ for a path $\rho$, then immediately by lemma 4.19, we have a proof that $\rho$ is $d\varepsilon$-differentially private.

**Sky: ¡¡how necessary are these following defs/props¿¿**

**Definition 4.22.** For a complete path $\rho$ of length $n$, the **cost** of a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is

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

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

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

Naturally, the existence of a valid coupling strategy bounds the privacy cost of any path.

**Proposition 4.23.** *If $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is valid, then $\rho$ is $cost(C_\rho)\varepsilon$-differentially private.*

*Proof.* Follows immediately from lemma 4.19. □

## 4.3 Branching

considering cutting this section - the major function it serves is to emphasize that, excepting loops, paths should all have their own coupling strategies, which could just be explained in the program section

**Definition 4.24** (Branching program)**.** Let $\Sigma_T$ 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 $\Sigma_T$ 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 $t_{init}\Sigma_T^*$ and satisfy the path condition.

Branching programs exemplify our conception of programs as simply collections of possible program executions; indeed, we will demonstrate that for the purposes of privacy, we can do no better than treating them as collections of paths.

### 4.3.1 Privacy

We first extend the definition of coupling strategies to sets of paths in the natural manner.

**Definition 4.25** (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 $C_\rho$.

**Definition 4.26.** A coupling strategy $C$ for a branching program $B$ is valid if, for every constituent path coupling strategy $C_\rho$, $C_\rho$ is valid.

**Definition 4.27.** The cost of a coupling strategy $C$ for a branched program $B$ is $\max_{\rho \in B} cost(\rho)$.

Notably, for a general branching program, there is no "smart" way to combine coupling strategies together; that is, in order to obtain the optimal coupling cost, we must find different coupling strategies for each path in a branching program. We provide a simple counterexample.

**Proposition 4.28.** *Optimal cost is dependent on path. There exists a valid transition alphabet $\Sigma_T$, a location space $Q$, and a branching program $B$ for which the optimal cost of a coupling strategy $C$ for $B$ is dependent on the path $\rho$.*

*In other words, the optimal strategy $C$ must assign different coupling strategies to occurances of the same transition in different paths.*

*Proof.* Let $Q = \{q_0, q_1, q_2, q_3\}$ consist only of input locations, each of which have both noise

parameters equal to 1. Let $\Sigma_T = \{t_{init}, t_{geq1}, t_{leq1}, t_{leq2}, t_{geq2}\}$ where

$$\begin{aligned}
t_{init} &= (q_0, q_1, \texttt{true}, \bot, 1) \\
t_{geq1} &= (q_1, q_2, \texttt{insample} \geq \texttt{x}, \top, 0) \\
t_{leq1} &= (q_1, q_3, \texttt{insample} < \texttt{x}, \bot, 0) \\
t_{geq2} &= (q_2, q_2, \texttt{insample} \geq \texttt{x}, \top, 0) \\
t_{leq2} &= (q_3, q_3, \texttt{insample} < \texttt{x}, \bot, 0)
\end{aligned}$$

and let $B = \{t_{init}t_{geq1}t_{geq2}^n, t_{init}t_{leq1}t_{leq2}^n\}$ be the branching program consisting of two paths, each of which have $n$ repetitions of the cycle transitions $t_{geq2}$ and $t_{leq2}$, respectively.

Let $\rho_1 = t_{init}t_{geq1}t_{geq2}^n$ and $\rho_2 = t_{init}t_{leq1}t_{leq2}^n$ be the two paths in $B$. Notice the following:

- The cost of any coupling strategy for $B$ must be at least 2.

  Let $C_{\rho_1}$ be a coupling strategy for $\rho_1$. We can bound its cost as follows:

$$\begin{aligned}
cost(C_{\rho_1}) &= \max_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} \sum_{i=0}^{n+2} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i(\texttt{in}_i\langle 1\rangle, \texttt{in}_i\langle 2\rangle)) \\
&\qquad + (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i'(\texttt{in}_i\langle 1\rangle, \texttt{in}_i\langle 2\rangle)|) \\
&\geq \max_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} \sum_{i=0}^{n+2} (|-\texttt{in}_i\langle 1\rangle + \texttt{in}_i\langle 2\rangle - \gamma_i(\texttt{in}_i\langle 1\rangle, \texttt{in}_i\langle 2\rangle)|) \\
&= \max_{\Delta \in [-1,1]^{n+2}} \sum_{i=0}^{n+2} (|\Delta_i - \gamma_i(0, \Delta_i)|) \\
&\geq |1 - \gamma_0(0,1)| + \sum_{i=1}^{n+2} |-1 - \gamma_i(0,-1)| \\
&= 1 - \gamma_0(0,1) + \sum_{i=1}^{n+2} (1 + \gamma_i(0,-1)) \\
&= 1 - \gamma_0(0,1) + (n+2) + \sum_{i=1}^{n+2} \gamma_i(0,-1) \\
&\geq 1 - \gamma_0(0,1) + (n+2) + \sum_{i=1}^{n+2} \gamma_0(0,1) \qquad \text{(privacy constraint)} \\
&= (n+3) + (n+1)\gamma_0(0,1) \\
&\geq 2
\end{aligned}$$

  and by a similar argument, $cost(C_{\rho_2}) \geq 2$ for any coupling strategy $C_{\rho_2}$.

- There exists a coupling strategy $C^*$ for $B$ such that $cost(C^*) = 2$.

  We will first describe $C^*_{\rho_1} = (\gamma, \gamma')$. Since no transition outputs `insample`, we can set

$\gamma_i'(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) = \mathtt{in}\langle 2\rangle - \mathtt{in}\langle 1\rangle$ for all $i$ with no privacy cost. Define

$$\begin{aligned}\gamma_0(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= -1\\ \gamma_i(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= \mathtt{in}_i\langle 2\rangle - \mathtt{in}_i\langle 1\rangle \qquad \text{for all } i > 0\end{aligned}$$

We see that $C^*_{\rho_1}$ is valid, since $\gamma_i \geq \gamma_0$ for all $i > 0$. Further, we see that

$$\begin{aligned}cost(C^*_{\rho_1}) &= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=0}^{n+2} (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i(\mathtt{in}_i\langle 1\rangle, \mathtt{in}_i\langle 2\rangle))\\ &\qquad + (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i'(\mathtt{in}_i\langle 1\rangle, \mathtt{in}_i\langle 2\rangle)|)\\ &= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} |-\mathtt{in}_0\langle 1\rangle + \mathtt{in}_0\langle 2\rangle - \gamma_0(\mathtt{in}_0\langle 1\rangle, \mathtt{in}_0\langle 2\rangle)|\\ &= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} |-\mathtt{in}_0\langle 1\rangle + \mathtt{in}_0\langle 2\rangle + 1|\\ &\leq 2\end{aligned}$$

showing that $cost(C^*_{\rho_1}) = 2$. Similarly, there is a coupling strategy $C^*_{\rho_2}$ for which $cost(C^*_{\rho_2}) = 2$. This shows that there is a coupling strategy $C^*$, consisting of $C^*_{\rho_1}$ and $C^*_{\rho_2}$, for which $cost(C^*) = 2$.

- Any coupling strategy $C$ that assigns the same coupling strategy to $t_{init}$ in both $\rho_1$ and $\rho_2$ must have cost $> 2$.

  Let $C$ be as described, and assume that $C$ has optimal cost, ie. $cost(C) = 2$. If $\gamma_0(0,1) \neq -1$ in $C_{\rho_1}$, then $cost(C_{\rho_1}) > 2$ in the same method as the above, a contradiction.

  Thus, $\gamma_0(0,1) = -1$ in $C_{\rho_1}$, which by hypothesis, means that $\gamma_0(0,1) = -1$ in $C_{\rho_2}$. We have the privacy constraint $\gamma_i \leq \gamma_0$ on $\rho_2$, which also means that $\gamma_i = -1$ identically for all $i > 0$. However, this means that

$$\begin{aligned}cost(C_{\rho_2}) &= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=0}^{n+2} (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i(\mathtt{in}_i\langle 1\rangle, \mathtt{in}_i\langle 2\rangle))\\ &\qquad + (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i'(\mathtt{in}_i\langle 1\rangle, \mathtt{in}_i\langle 2\rangle)|)\\ &\geq \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=0}^{n+2} (|-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle - \gamma_i(\mathtt{in}_i\langle 1\rangle, \mathtt{in}_i\langle 2\rangle)|)\\ &\geq \max_{\Delta \in [-1,1]^{n+2}} |\Delta_0 - \gamma_i(0, \Delta_0)| + \sum_{i=1}^{n+2} (|\Delta_i - \gamma_i(0, \Delta_i)|)\\ &\geq |1 - \gamma_i(0,1)| + \sum_{i=1}^{n+2} (|1+1|)\\ &= 2 \cdot (n+2)\end{aligned}$$

  showing that $C$ is not optimal, a contradiction. Therefore $cost(C) > 2$.

The above observations show that the optimal coupling strategy for $B$ must necessarily assign different coupling strategies to $t_{init}$ in $\rho_1$ and $\rho_2$. □

Indeed, there exist a family of counterexamples such that the relative "cost" of choosing a unified coupling strategy for a branching program increases quadratically in the size of the branching program. **Sky: ¡¡Is this technically correct @ Vishnu?¿¿ Vishnu: ¡¡Yes!¿¿**

The proposition below shows that the cost of a coupling strategy that assigns the same shift to each transition (path-independent strategies) can be arbitrarily larger than the optimal cost.

**Proposition 4.29.** *There exist a family of branching programs $\{B_n\}_{n\in\mathbb{N}}$ for which the cost of any path-independent coupling strategy $C$ for $B_n$ is in $\Omega(n^2)$, but for which there exists a path-dependent coupling strategy $C'$ for $B_n$ with cost in $O(n)$.*

*Proof.* Construct $B_n$ as follows. Consider a path $\rho = t^{(1)}_{true}t^{(1)}_{geq}\left(t^{(1)}_{leq}\right)^n \dots t^{(n)}_{true}t^{(n)}_{geq}\left(t^{(n)}_{leq}\right)^n$ where:

- $t^{(i)}_{true}$ are assignment transitions with guard `true`.
- $t^{(i)}_{geq}$ are assignment transitions with guard `insample` $\geq x$.
- $t^{(i)}_{leq}$ are non-assignment transitions with guard `insample` $< x$.
- The noise parameters for all transitions are 1.
- None of the transitions output `insample`.

For each $i$, construct also the looping branch $L_i = L\left(t^{(n+i)}_{true}\left(t^{(i)}_{loop}\right)^* t^{(i)}_{geq}\right)$ such that each transition $t^{(i)}_{geq}$ is preceded by an arbitrary number of transitions with guard `insample` $< x$.

Let $C$ be a path-independent coupling strategy for $B_n$ – this menas that $C$ must assign the same shift to each transition in $B_n$.

From the privacy constraints on the looping branches $L_i$, we see that $\gamma_{t^{n+i}_{true}}(1,0) = 1$ from the same method as in 4.28, which then implies from the constraint $\gamma_{t^{n+i}_{true}} \leq \gamma_{t^{(i)}_{geq}}$ that $\gamma_{t^{(i)}_{geq}}(1,0) = 1$.

Since the preceding assignment transition for $t^{(i)}_{leq}$ is given by $t^{(i)}_{geq}$, for which we have the privacy constraint $\gamma_{t^{(i)}_{geq}} \leq \gamma_{t^{(i)}_{leq}}$, we see that $\gamma_{t^{(i)}_{leq}}(1,0) = 1$ as well.

Computing the cost of the coupling strategy assigned to $\rho$, which has $n \cdot (n+2)$ transitions, we get

$$
\begin{aligned}
cost(C_\rho) &\geq \max_{\Delta=\mathtt{in}\langle 2\rangle-\mathtt{in}\langle 1\rangle} \sum_{i=1}^{n} \big(|\Delta_{t^{(i)}_{true}} - \gamma_{t^{(i)}_{true}}(-\Delta_{t^{(i)}_{true}}, 0)| + |\Delta_{t^{(i)}_{geq}} - \gamma_{t^{(i)}_{geq}}(-\Delta_{t^{(i)}_{geq}}, 0)| \\
&\qquad + n \cdot |\Delta_{t^{(i)}_{leq}} - \gamma_{t^{(i)}_{leq}}(-\Delta_{t^{(i)}_{leq}}, 0)|\big) \\
&\geq \sum_{i=1}^{n} \Big(|-1-\gamma_{t^{(i)}_{true}}(1,0)| + |-1-\gamma_{t^{(i)}_{geq}}(1,0)| + n \cdot |-1-\gamma_{t^{(i)}_{leq}}(1,0)|\Big) \\
&= \sum_{i=1}^{n} \Big(|-1-\gamma_{t^{(i)}_{true}}(1,0)| + |-1-1| + n \cdot |-1-1|\Big) \\
&\geq \sum_{i=1}^{n} (2n+1) \\
&= n \cdot (2n+1)
\end{aligned}
$$

whereas there exists another coupling strategy $C^*_\rho = (\gamma^*, \gamma'^*)$ for $\rho$ that would assign

$$
\begin{aligned}
\gamma^*_{t^{(i)}_{true}}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= -\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle \\
\gamma^*_{t^{(i)}_{geq}}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= 1 \\
\gamma^*_{t^{(i)}_{leq}}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= -\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle
\end{aligned}
$$

which satisfies the privacy constraints

$$
\gamma^*_{t^{(i)}_{true}} \leq \gamma^*_{t^{(i)}_{geq}} \geq \gamma^*_{t^{(i)}_{leq}}
$$

for all $i$, which has cost $2n$.

For the looping branches $L_i$, the coupling strategy $C^*_{L_i} = (\gamma^*, \gamma'^*)$ assigns

$$
\begin{aligned}
\gamma^*_{t^{(n+i)}_{true}}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= 1 \\
\gamma^*_{t^{(i)}_{loop}}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= -\mathtt{in}\langle 1\rangle + \mathtt{in}\langle 2\rangle \\
\gamma^*_{t^{(i)}_{geq}}(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle) &= 1
\end{aligned}
$$

which satisfies the privacy constraints

$$
\gamma^*_{t^{(i)}_{geq}} \geq \gamma^*_{t^{(n+i)}_{true}} \leq \gamma^*_{t^{(i)}_{loop}}
$$

for all $i$, and has cost 4.

Putting these strategies together, we have a coupling startegy $C^*$ for $B_n$ with cost $2n$, as opposed to at least $n(2n+1)$ for any path-independent coupling strategy $C$. □

## 4.4 Loops

We now introduce loops into our program model through the introduction of a star operator.

**Definition 4.30.** 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 $\Sigma_T$.

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

Intuitively, a looping branch is a single, straight-line path except that we allow for cycles to be inserted within the path. Naturally, looping branches are closely related to general "star-dot" or union-free regular languages and their associated 1-cycle-free-path-automata (see, for example, [11]).

We will associate entire looping branches with a **single** coupling strategy.

**Definition 4.31** (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 $C : T_r \times \mathbb{R} \times \mathbb{R} \to [-1, 1] \times [-1, 1]$ that computes shifts for each transition in $L$ as a function of two adjacent inputs.

We will sometimes call these coupling strategies "class" coupling strategies if necessary to distinguish them from coupling strategies for individual paths.

Observe that a class coupling strategy implicitly defines a coupling strategy for each path in $L$.

**Definition 4.32** (Induced Coupling Strategy)**.** **Sky: ¡¡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 $\rho = t_0 \cdots t_{n-1}$ induced by $C$ is the pair of functions $\gamma_\rho, \gamma'_\rho$ such that

$$\gamma_\rho(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) = (\gamma(t_0)(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle), \gamma(t_1)(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle), \ldots, \gamma(t_{n-1})(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle))$$
$$\gamma'_\rho(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) = (\gamma'(t_0)(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle), \gamma'(t_1)(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle), \ldots, \gamma'(t_{n-1})(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle))$$

Perhaps surprisingly, we show that it is in fact optimal to only consider a single coupling strategy for an entire looping branch; not only is finding individual coupling strategies for every single path in a looping branch intractable, but it also does not lead to a better overall privacy cost.

**Proposition 4.33.** *If there exists a valid coupling strategy $C_\rho$ with cost $cost(C_\rho)$ for every path $\rho$ 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 (i.e. cycles) to our model, it is possible for a looping branch to fail to be private for *any* $d > 0$; in other words, 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 4.34.** *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 $\gamma'_i = -\boldsymbol{in}\langle 1\rangle_i + \boldsymbol{in}\langle 2\rangle_i$.*

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

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. This helps us solve the *decision* problem of privacy: does there **exist** any finite $d > 0$ such that a looping branch is $d\varepsilon$-differentially private?

**Definition 4.35** (Privacy Constraint System)**.** Let $L$ be a looping branch over a valid transition alphabet $\Sigma_T$. 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 $L$.

Naturally, if there exists a solution to the privacy constraint system for a looping branch $L$, then $L$ is private in the sense of the decision problem.

**Proposition 4.36.** *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.*

Perhaps surprisingly, we will also later show that the privacy constraint system is complete; that is, if there does not exist a solution to the privacy constraint system, we can prove that there also does not exist any finite $d > 0$ such that $L$ is $d\varepsilon$-differentially private.

## 4.5 Programs

We are now equipped to introduce our complete program model.

**Definition 4.37.** A program $P$ is a finite union of looping branches over a valid finite alphabet of transitions.

Naturally, this equivalently means that $P$ is a language described by a regular expression in union normal form such that each individual term describes a looping branch.

As illustrated by branching programs, we cannot do any better with regards to coupling strategies than simply choosing one coupling strategy per looping branch.

**Lemma 4.38.** *If, for every looping branch $L$ in $P$, there exists a valid coupling strategy $C_L$, then $P$ is $(\max_{L \subseteq P} cost(C_L))\varepsilon$-differentially private.*

*Proof.* Follows immediately from definitions. □

## 4.6 Deciding Privacy

In this section, we discuss the boolean or decision problem 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 4.39.** *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 4.36. □

As previously mentioned, 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 4.40.** *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.*

To complete the proof, we introduce a previously analyzed program model known as DiPA, which we claim captures the same class of programs as our program model.

## 4.7 DiPA

We now discuss a previously defined program model, DiPA, which turns out to be exactly equivalent to our own program model. Additionally, the equivalence between DiPA and programs allows us to construct our completeness proof.

**Definition 4.41** ([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 locations partitioned into input locations $Q_{in}$ and non-input locations $Q_{non}$.
- $\Sigma = \mathbb{R}$ is the input alphabet
- $C = \{\texttt{true}, \texttt{insample} < x, \texttt{insample} \geq x\}$ is a set of guard conditions
- $\Gamma$ is a finite output alphabet

- $q_{init} \in Q$ is the initial location
- $X = \{\texttt{x}, \texttt{insample}, \texttt{insample}'\}$ 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 location
- $\delta : (Q \times C) \to (Q \times (\Gamma \cup \{\texttt{insample}, \texttt{insample}'\}) \times \{\texttt{true}, \texttt{false}\})$ is a partial transition function.

In addition, $\delta$ must satisfy some additional conditions:

- **Determinism:** For any location $q \in Q$, 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 location $q \in Q$, 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 location $q_0$ 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 $c = \texttt{true}$.

A DiPA operates as follows:

- At each location, a real-valued input $\texttt{in}$ is read in and two variables $\texttt{insample} \sim \texttt{Lap}(\texttt{in}, d\varepsilon)$ and $\texttt{insample}' \sim \texttt{Lap}(\texttt{in}, d\varepsilon)$ are sampled.
- $\texttt{insample}$ is compared to the stored variable $\texttt{x}$, and depending on the guards of the transitions out of the current location, changes the current location and outputs a value. This value can either be $\texttt{insample}, \texttt{insample}'$, or a symbol from $\Gamma$.
- Finally, the value of $\texttt{x}$ is optionally updated with the value of $\texttt{insample}$.

We first establish notation for discussing the probabilities of different paths in a DiPA, which allows us to define $d\varepsilon$-differential privacy.

**Definition 4.42.** Let $\rho$ be a path in a DiPA $A$, let $\texttt{in}$ be a valid input sequence and let $o$ be a possible output of $\rho$. In particular, if $\sigma_i \in \{\texttt{insample}, \texttt{insample}'\}$, then we require that $o_i$ 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 $\rho$ being taken with input sequence $\texttt{in}$ and outputting $o$. If the first location in $\rho$ 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 definition of DiPA semantics, we refer back to the original work.

**Definition 4.43.** A DiPA $A$ is $d\varepsilon$-differentially private for some $d > 0$ if for all paths $\rho$ in $A$, for all possible outputs $o$ of $\rho$ and valid adjacent input sequences $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$,

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

Notably, we show that we can draw a direct equivalence between programs and DiPAs.

**Proposition 4.44.** *Every path $\rho$ through a DiPA $A$ is represented by a complete path $\hat{\rho}$ comprised of transitions from a valid transition alphabet $\Sigma_T$; further, the set of all possible paths through $A$ is a regular language over $\Sigma_T$.*

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

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 $\delta$ satisfies the conditions of determinism, output distinction, initialization, and non-input transition, $\Sigma_\rho$ must as well. Then let $\hat{\rho} = t_0 \cdot t_1 \cdot \ldots \cdot t_{i-1}$ be the representation of $\rho$ as a word over $\Sigma_\rho$.

Let $\Sigma_T = \bigcup_{\rho \in A} \Sigma_\rho$. Note that $\Sigma_T$ 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 $\delta_D$ is defined as follows: Let $q \in Q$ be an arbitrary location. 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 location in $D$ is an accepting location, $\mathcal{L}(D) = \{\hat{\rho} : \rho \in A\}$. Thus, the set of (representations of) all paths in $A$ must be a regular language. □

**Proposition 4.45.** *For every program $P$ over a valid transition alphabet $\Sigma_T$, there exists a corresponding DiPA $A$ such that there exists a path $\rho$ in $A$ if and only if its representation is in $P$.*

*Proof.* The DiPA can be directly constructed from $P$. □

As expected, the probability of a path "succeeding" in a program is the same as the probability of a path being traversed in a DiPA.

**Proposition 4.46.** *For all paths $\rho$ in a DiPA $A$ and for all input sequences $\boldsymbol{in}$ and possible outputs $\sigma$ of $\rho$, $\mathbb{P}[\rho, \boldsymbol{in}, \sigma] = \mathbb{P}[\hat{\rho}, \boldsymbol{in}, \sigma]$.*

*Proof.* Follows immediately from definitions. □

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

**Theorem 4.47** ([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.*

This provides us with a method of demonstrating that a program is not $d\varepsilon$-differentially private for any $d > 0$.

**Corollary 4.48.** *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.*

Our key result relates the privacy constraint system (see definition 4.35) and these graph-theoretic structures.

**Lemma 4.49.** *If, for a looping branch $L$, there does not exist a coupling strategy $C$ that satisfies the privacy constraint system for $L$, then the corresponding DiPA to $L$ must contain either a leaking cycle, a leaking pair, a disclosing cycle, or a privacy violating path.*

## 4.8 An algorithm for deciding privacy

We now introduce a polynomial-time **Sky: ¡¡linear?¿¿** algorithm for solving the decision problem of privacy.

**Sky: ¡¡note: vishnu is restructuring/removing the following ¿¿**

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

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

**Definition 4.50.** For a complete path $\rho$ in a program $P$, let $A_\rho$ be the set of all assignment transitions in $\rho$. **Vishnu: ¡¡Unless this is used later, we can probably remove this definition.¿¿**

**Definition 4.51.** 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 $\rho$ 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 $\rho$ 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.

**Definition 4.52.** Given segments $s_i$ and $s_j$ of a path $\rho$, we say that $s_j$ follows $s_i$ (written $s_i \hookrightarrow s_j$) if $s_j$ is a subpath of $\rho$ occurring immediately after $s_i$ in $\rho$.

**Definition 4.53** (Segment graph of a program)**.** Given a DiPA $\mathcal{A}$, the segment graph $G_P = (V, E)$ of $\mathcal{A}$ is a directed graph where:

- For every segment $s_i \in seg(\mathcal{A})$, there is a vertex $v_i \in V$ representing the coupling shift on the assignment transition of $s_i$.
- For every pair of segments $(s_i, s_j)$ which have the privacy constraint $\gamma_i \leq \gamma_j$, there is an edge $(v_i, v_j) \in E$.
- There are nodes $\mathbf{1}, -\mathbf{1} \in V$ such that:
  - The edges $(-\mathbf{1}, v)$ exist for all $v \in V$.

- The edges $(v, \mathbf{1})$ exist for all $v \in V$.
- For every segment $s_i$ with the privacy constraint $\gamma_i = 1$, there is an edge $(\mathbf{1}, v_i) \in E$.
- For every segment $s_i$ with the privacy constraint $\gamma_i = -1$, there is an edge $(v_i, -\mathbf{1}) \in E$.

**Lemma 4.54.** *Let there exist a path $v_1 \to \cdots \to v_k$ in the segment graph of $\mathcal{A}$, and let the corresponding segments be $s_{i_1}, \ldots, s_{i_k}$. Then we either have that*

$$s_{i_1} \hookrightarrow s_{i_2} \hookrightarrow \cdots \hookrightarrow s_{i_k} \qquad \textit{and} \qquad \textit{guard}(s_{i_k}) =<$$

*or*

$$s_{i_k} \hookrightarrow s_{i_{k-1}} \hookrightarrow \cdots \hookrightarrow s_{i_1} \qquad \textit{and} \qquad \textit{guard}(s_{i_k}) =\geq$$

*Proof.* We will use induction on $k$ with base case $k = 2$. If we have $v_1 \to v_2$ in the segment graph, then we either have that $s_{i_1} \hookrightarrow s_{i_2}$ with $\textit{guard}(s_{i_2}) =<$, or that $s_{i_2} \hookrightarrow s_{i_1}$ with $\textit{guard}(s_{i_2}) =\geq$. For a path with length $k+1$ in the segment graph, let us assume the desired result. Then apply the base case on $s_{i_k} \to s_{i_{k+1}}$ to conclude. □

**Theorem 4.55.** *A DiPA $\mathcal{A}$ is differentially private if and only if there does not exist a path from $\mathbf{1}$ to $-\mathbf{1}$ in the segment graph of $\mathcal{A}$.*

*Proof.* ( $\implies$ ) Suppose that $\mathcal{A}$ is differentially private. Then there exists a shift-coupling proof of privacy $\Gamma$ with finite cost. Assume that there exists a path $\mathbf{1} \to v_1 \to \cdots \to v_k \to -\mathbf{1}$ in the segment graph of $\mathcal{A}$. Without loss of generality, we have the sequence of segments $s_{i_1} \hookrightarrow \cdots \hookrightarrow s_{i_k}$ corresponding to the path by Lemma 4.54. We can then extend this to a sequence of segments $s_1 \to \cdots \to s_{i_1} \to \cdots \to s_{i_k} \to \cdots \to s_n$ that begins with the initialization segment and ends with a terminal segment.

If we restrict the segment-shift graph to only these segments, we find that any assignment of shifts $\gamma$ would not be valid, as we would be able to construct a sequence of inequalities to conclude that $1 \leq -1$, which is false. Thus, there is no valid assignment of shifts $\gamma$ for this sequence of segments with finite cost, and so $\Gamma$ is not a shift-coupling proof of privacy with finite cost. This is a contradiction.

( $\impliedby$ ) This direction follows from the fact that there is *some* valid assignment of shifts $\gamma$ with finite cost on each segment, and so we can construct a shift-coupling proof of privacy $\Gamma$ with finite cost. □

We can construct the segment graph of a DiPA in linear time, and check for a path from $\mathbf{1}$ to $-\mathbf{1}$ in linear time using a breadth-first search. Thus, we can decide privacy in linear time.

## 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. We can partition a program into a finite set of looping branches, and then optimize the privacy cost of each looping branch individually.

**Proposition 4.56.** *Let $L$ be a looping branch in a program $P$ consisting of $n$ transitions. The cost $opt(L)$ of the optimal coupling strategy for $L$ can be computed by the following optimization problem:*

$$\begin{aligned} opt(L) = \max_{\Delta \in [-1,1]^n} \min_{\gamma,\gamma' \in [-1,1]^n} & \sum_{i=1}^{n} (|\Delta_i - \gamma_i| d_i + |\Delta_i - \gamma'_i| d'_i) \\ \textit{subject to} \quad & \gamma_{at(i)} \leq \gamma_i \textit{ if } c_i = \texttt{\textit{insample}} \geq x, \\ & \gamma_{at(i)} \geq \gamma_i \textit{ if } c_i = \texttt{\textit{insample}} < x, \\ & \gamma_i = 0 \textit{ if } \sigma_i = \texttt{\textit{insample}}, \\ & \gamma'_i = 0 \textit{ if } \sigma_i = \texttt{\textit{insample}}' \\ & \gamma_i = \gamma'_i = \Delta_i \textit{ if } t_i \textit{ is in a cycle} \end{aligned}$$

*If a looping branch $L$ is not differentially private, a solution to this optimization problem does not exist, and we write $opt(L) = \infty$.*

*Proof.* Assume that $L$ is differentially private. Then there exists a valid coupling strategy for $L$ with finite cost, showing that the constraints above are feasible, and a finite solution to the optimization problem exists.

We will specify a valid coupling strategy $C^* = (\gamma^*, \gamma'^*)$ for $L$ with the cost stated above, and then show it is optimal. Define $\Gamma : [-1,1]^n \to [-1,1]^n \times [-1,1]^n$ as follows, where $\Gamma(\Delta)$ is a pair $(\gamma, \gamma')$:

$$\begin{aligned} \Gamma(\Delta) = \operatorname*{arg\,min}_{\gamma,\gamma' \in [-1,1]^n} & \sum_{i=1}^{n} (|\Delta_i - \gamma_i| d_i + |\Delta_i - \gamma'_i| d'_i) \\ \text{subject to} \quad & \gamma_{at(i)} \leq \gamma_i \text{ if } c_i = \texttt{insample} \geq x, \\ & \gamma_{at(i)} \geq \gamma_i \text{ if } c_i = \texttt{insample} < x, \\ & \gamma_i = 0 \text{ if } \sigma_i = \texttt{insample}, \\ & \gamma'_i = 0 \text{ if } \sigma_i = \texttt{insample}' \\ & \gamma_i = \gamma'_i = \Delta_i \text{ if } t_i \text{ is in a cycle} \end{aligned}$$

Then define

$$(\gamma^*(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle), \gamma'^*(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle)) = \Gamma(\texttt{in}\langle 1\rangle - \texttt{in}\langle 2\rangle)$$

Notice the following:

1. $C^*$ is a valid coupling strategy for $L$, since the privacy constraints on $\gamma^*$ and $\gamma'^*$ are satisfied by construction.

2. $C^*$ has the cost given by the solution to the optimization problem, since

$$\begin{aligned}
cost(C^*) = \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=1}^{n} & |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma_i^*(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d_i \\
& + |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma'^*_i(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d'_i \\
= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=1}^{n} & |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \Gamma_1(\mathtt{in}\langle 1\rangle - \mathtt{in}\langle 2\rangle)_i|d_i \\
& + |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \Gamma_2(\mathtt{in}\langle 1\rangle - \mathtt{in}\langle 2\rangle)_i|d'_i \\
= \max_{\Delta \in [-1,1]^n} \sum_{i=1}^{n} & |\Delta_i - \Gamma_1(\Delta)_i|d_i + |\Delta_i - \Gamma_2(\Delta)_i|d'_i \\
= \max_{\Delta \in [-1,1]^n} \min_{\gamma,\gamma' \in [-1,1]^n} \sum_{i=1}^{n} & |\Delta_i - \gamma_i|d_i + |\Delta_i - \gamma'_i|d'_i
\end{aligned}$$

3. $C^*$ is optimal, since for any valid coupling strategy $C = (\delta, \delta')$ for $L$, we have

$$\begin{aligned}
cost(C) = \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} \sum_{i=1}^{n} & |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \delta_i(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d_i \\
& + |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \delta'_i(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d'_i \\
\geq \max_{\Delta \in [-1,1]^n} \sum_{i=1}^{n} & |\Delta_i - \delta_i(0, \Delta_i)|d_i + |\Delta_i - \delta'_i(0, \Delta_i)|d'_i \\
\geq \max_{\Delta \in [-1,1]^n} \min_{\gamma,\gamma' \in [-1,1]^n} \sum_{i=1}^{n} & (|\Delta_i - \gamma_i|d_i + |\Delta_i - \gamma'_i|d'_i) \\
= cost(C^*) &
\end{aligned}$$

which shows that the optimization problem computes the optimal cost of a coupling strategy for $L$ that satisfies the privacy constraints. □

Note: the inner problem is convex, and so the outer problem is that of convex maximization.

**Definition 4.57.** Define the approximate privacy cost of a differentially private looping branch $L$ to be as follows. Let $I$ be the set of transitions in $L$ that do *not* appear in a cycle.

$$\begin{aligned}
approx(L) = & \sum_{t_i \text{outputs } \mathtt{insample}} d'_i + \min_{\gamma \in [-1,1]^n} \sum_{i \in I} (1 + |\gamma_i|)\, d_i \\
\text{subject to } & \gamma_{at(i)} \leq \gamma_i \text{ if } c_i = \mathtt{insample} \geq x, \\
& \gamma_{at(i)} \geq \gamma_i \text{ if } c_i = \mathtt{insample} < x, \\
& \gamma_i = 0 \text{ if } \sigma_i = \mathtt{insample}, \\
& \gamma_i = 1 \text{ if } t_i \text{ is in a cycle and has } c_i = \mathtt{insample} < x, \\
& \gamma_i = -1 \text{ if } t_i \text{ is in a cycle and has } c_i = \mathtt{insample} \geq x
\end{aligned}$$

which is the cost of a coupling strategy in which $\gamma$ and $\gamma'$ are constant with respect to $\texttt{in}\langle 1\rangle$ and $\texttt{in}\langle 2\rangle$ on non-cyclic transitions.

**Proposition 4.58.** *Given a differentially private looping branch $L$, there exists a valid coupling strategy $C_L$ for $L$ such that $cost(C_L) = approx(L)$.*

*Moreover, if $\gamma \in [-1, 1]^n$ satisfies the approximate privacy constraints, then there is a valid coupling strategy $C_L = (\gamma^*, \gamma'^*)$ for $L$ such that $\gamma_i^*(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) = \gamma_i$ for all $t_i$ that do not appear in cycles.*

*Proof.* Let

$$\gamma = \underset{\gamma \in [-1,1]^n}{\arg\min} \sum_{i \in I} (1 + |\gamma_i|)\, d_i$$

subject to the constraints above. Define $C_L = (\gamma^*, \gamma'^*)$ where

$$\gamma_i^*(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle) = \begin{cases} \texttt{in}\langle 1\rangle_i - \texttt{in}\langle 2\rangle_i & \text{if } t_i \text{ is in a cycle} \\ \gamma_i & \text{otherwise} \end{cases}$$

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

Notice the following:

- $C_L$ satisfies the privacy constraints, and so is valid.

  If $t_i$ is in a cycle with $c_i = \texttt{insample} < x$, then the constraints on $\gamma$ require that $\gamma_i = 1$, and so $1 = \gamma_i \leq \gamma_{at(i)} = 1$. As a result, we will satisfy the privacy constraint $\gamma^*_{at(i)} \geq \gamma^*_i$:

  $$\gamma_i^* = \texttt{in}\langle 1\rangle_i - \texttt{in}\langle 2\rangle_i \leq 1 = \gamma^*_{at(i)}$$

  A similar argument holds for if $t_i$ is in a cycle with $c_i = \texttt{insample} \geq x$.

  All other privacy constraints are satisfied by construction.

- $C_L$ has the cost given by the solution to the optimization problem, since

$$
\begin{aligned}
cost(C_L) = \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} & \sum_{i=1}^{n} |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma_i^*(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d_i \\
& \qquad + |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma'^*_i(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d'_i \\
= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} & \left( \sum_{i \in I} |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma_i^*(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d_i \right) \\
& \qquad + \left( \sum_{t_i \text{outputs } \mathtt{insample}} |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma'^*_i(\mathtt{in}\langle 1\rangle, \mathtt{in}\langle 2\rangle)|d'_i \right) \\
= \max_{\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle} & \left( \sum_{i \in I} |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle - \gamma_i|d_i \right) \\
& \qquad + \left( \sum_{t_i \text{outputs } \mathtt{insample}} |\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle|d'_i \right) \\
= & \sum_{i \in I} (1 + |\gamma_i|)d_i + \sum_{t_i \text{outputs } \mathtt{insample}} d'_i \\
= & \; approx(L)
\end{aligned}
$$

□

**Proposition 4.59.** *The approximate privacy cost of $L$ can be computed in time polynomial in $n$, the number of transitions in $L$.*

*Proof.* To compute the solution to the minimization problem, we can set up the following linear program:

$$
\begin{aligned}
\min_{\gamma, A_i \in [-1,1]^n} & \sum_{i=1}^{n} (1 + A_i)\, d_i \\
\text{subject to } & \gamma_{at(i)} \le \gamma_i \text{ if } c_i = \mathtt{insample} < x, \\
& \gamma_{at(i)} \ge \gamma_i \text{ if } c_i = \mathtt{insample} \ge x, \\
& \gamma_i = 0 \text{ if } \sigma_i = \mathtt{insample}, \\
& \gamma_i = 1 \text{ if } t_i \text{ is in a cycle and has } c_i = \mathtt{insample} < x, \\
& \gamma_i = -1 \text{ if } t_i \text{ is in a cycle and has } c_i = \mathtt{insample} \ge x, \\
& \gamma_i \le A_i, -\gamma_i \le A_i \text{ for all } i \in \{1, \ldots, n\}
\end{aligned}
$$

This program can be solved using the ellipsoid method in polynomial time. □

**Proposition 4.60.** *For a looping branch $L$ with $n$ distinct transitions, we have*

$$
opt(L) \le approx(L) \le opt(L) + \sum_{i \in I}^{n} d_i + \sum_{t_i \, outputs \; \mathtt{insample}} d'_i
$$

*where $I$ is the set of transitions in $L$ that do not appear in a cycle.*

*Proof.* We have $opt(L) \le approx(L)$ by Proposition 4.56. Let $I$ be the set of transitions in $L$ that do *not* appear in a cycle. Then we have

$$\begin{aligned}
opt(L) &= \max_{\Delta\in[-1,1]^n} \min_{\gamma,\gamma'\in[-1,1]^n} \sum_{i=1}^{n} \left(|\Delta_i - \gamma_i| d_i + |\Delta_i - \gamma_i'| d_i'\right) \\
&= \max_{\Delta\in[-1,1]^n} \min_{\gamma,\gamma'\in[-1,1]^n} \sum_{i\in I} \left(|\Delta_i - \gamma_i| d_i + |\Delta_i - \gamma_i'| d_i'\right) \\
&= \max_{\Delta\in[-1,1]^n} \min_{\gamma,\gamma'\in[-1,1]^n} \left(\sum_{i\in I} \left(|\Delta_i - \gamma_i| d_i\right) + \sum_{t_i \text{outputs } \texttt{insample}} |\Delta_i| d_i'\right) \\
&\ge \max_{\Delta\in[-1,1]^n} \min_{\gamma,\gamma'\in[-1,1]^n} \left(\sum_{i\in I} \left(|\gamma_i| - |\Delta_i|\right) d_i + \sum_{t_i \text{outputs } \texttt{insample}} |\Delta_i| d_i'\right) \\
&= \max_{\Delta\in[-1,1]^n} \left(-\sum_{i\in I} |\Delta_i| d_i + \sum_{t_i \text{outputs } \texttt{insample}} |\Delta_i| d_i' + \min_{\gamma,\gamma'\in[-1,1]^n} \sum_{i\in I} |\gamma_i| d_i\right) \\
&\ge \min_{\gamma,\gamma'\in[-1,1]^n} \sum_{i\in I} |\gamma_i| d_i \\
&= approx(L) - \sum_{t_i \text{outputs } \texttt{insample}} d_i' - \sum_{i\in I} d_i'
\end{aligned}$$

showing the second inequality. □

**Conjecture 4.61.** The optimal coupling cost is the "true" privacy cost of a looping branch. That is,

$$opt(L) = \sup_{\rho\in L} \sup_{\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle} D_\infty(\mathbb{P}[\rho, \texttt{in}\langle 1\rangle, o] \mid\mid \mathbb{P}[\rho, \texttt{in}\langle 2\rangle, o])$$

representing the worst-case privacy loss over all possible paths in $L$ and all possible adjacent inputs.

## 4.10 Deciding Privacy (Graph, no segments)

We can check whether a looping branch is differentially private by checking whether the approximate privacy cost of the looping branch is finite. This is true if and only if the *approximate privacy constraints* are feasible.

We will give an algorithm to check whether the approximate privacy constraints are feasible. Although this is equivalent to solving a 2SAT instance, we will encode these constraints in a graph which preserves some of the structure of our program.

**Definition 4.62** (Privacy constraint graph)**.** Let $L$ be a looping branch in a program $P$. The **privacy constraint graph** $G_L = (V, E)$ of $L$ is a directed graph where:

- For every transition $t_i$ in $L$, there is a vertex $v_i \in V$ representing the shift $\gamma_i$ on $t_i$.
- For every pair of transitions $(t_i, t_j)$ which have the privacy constraint $\gamma_i \leq \gamma_j$, there is an edge $(v_i, v_j) \in E$.
- There are nodes $\mathbf{1}, -\mathbf{1} \in V$ such that:
  - The edges $(-\mathbf{1}, v)$ exist for all $v \in V$.
  - The edges $(v, \mathbf{1})$ exist for all $v \in V$.
  - For every transition $t_i$ with the privacy constraint $\gamma_i = 1$, there is an edge $(\mathbf{1}, v_i) \in E$.
  - For every transition $t_i$ with the privacy constraint $\gamma_i = -1$, there is an edge $(v_i, -\mathbf{1}) \in E$.

**Proposition 4.63.** *A looping branch $L$ is differentially private if and only if there does not exist a path from $\mathbf{1}$ to $-\mathbf{1}$ in the privacy constraint graph of $L$.*

*Proof.* ( $\implies$ ) Let $L$ be differentially private. Then $approx(L) < \infty$, and so there exists $\gamma \in [-1, 1]^n$ such that the approximate privacy constraints are satisfied by $\gamma$. Aiming for a contradiction, assume that there exists a path $\mathbf{1} \to v_{i_1} \to \cdots \to v_{i_k} \to -\mathbf{1}$ in the privacy constraint graph of $L$. This corresponds to the sequence of privacy constraint inequalities

$$1 \leq \gamma_{i_1} \leq \cdots \leq \gamma_{i_k} \leq -1$$

which is a contradiction, showing that no such $\gamma$ could exist. Therefore, there is no path from $\mathbf{1}$ to $-\mathbf{1}$ in the privacy constraint graph of $L$.

( $\impliedby$ ) Let there exist no path from $\mathbf{1}$ to $-\mathbf{1}$ in the privacy constraint graph of $L$. Define

$$\gamma_i = \begin{cases} 1 & \text{if there exists a path from } \mathbf{1} \text{ to } v_i \text{ in } G_L \\ -1 & \text{otherwise} \end{cases}$$

We claim that the approximate privacy constraints are satisfied by $\gamma$. **Vishnu: ¡¡Need to do a better job of stating what approximate privacy constraints are and why they only depend on $\gamma \in [-1, 1]^n$ and not some function of $\texttt{in}\langle 1 \rangle$ and $\texttt{in}\langle 2 \rangle$¿¿.**

- Consider the approximate privacy constraint $\gamma_i \leq \gamma_j$. This corresponds to the edge $(v_i, v_j)$ in $G_L$. If $\gamma_i = 1$, there is a path from $\mathbf{1}$ to $v_i$, and so there is a path from $\mathbf{1}$ to $v_j$, and so $\gamma_j = 1$, satisfying the constraint. If $\gamma_i = -1$, then any assignment of $\gamma_j$ satisfies the constraint.
- Consider the constraint $\gamma_i = 1$. This corresponds to the edge $(\mathbf{1}, v_i)$ in $G_L$. Since there is a path from $\mathbf{1}$ to $v_i$, we have $\gamma_i = 1$, satisfying the constraint.
- Consider the constraint $\gamma_i = -1$. This corresponds to the edge $(v_i, -\mathbf{1})$ in $G_L$. Since there is no path from $\mathbf{1}$ to $-\mathbf{1}$ in $G_L$, there must be no path from $\mathbf{1}$ to $v_i$. Thus, $\gamma_i = -1$, satisfying the constraint.

Thus, the approximate privacy constraints are satisfied by $\gamma$, which means that $approx(L) < \infty$ and $L$ is differentially private. □

We have specified a linear time algorithm only to check whether a given looping branch is private. However, we can check whether *all* looping branches are private at once in linear time by combining the privacy constraint graphs for each looping branch into a single graph.

**Definition 4.64.** The privacy constraint graph $G_P$ of a program $P$ is the union of the privacy constraint graphs of each looping branch in $P$.

**Proposition 4.65.** *Let $v_{i_0}, \ldots, v_{i_k}$ be vertices in $G_P$ corresponding to transitions $t_{i_0}, \ldots, t_{i_k}$ in $P$. If $v_{i_0} \to \cdots \to v_{i_k}$ is a path in $G_P$, then there exists a path $\rho \in P$ such that*

$$t_{i_0} \cdots t_{i_k} \text{ is a subsequence of } \rho \text{ with } guard(t_{i_j}) = \texttt{insample} \geq x \text{ for all } j \in \{1, \ldots, k\}$$

*or*

$$t_{i_k} \cdots t_{i_0} \text{ is a subsequence of } \rho \text{ with } guard(t_{i_j}) = \texttt{insample} < x \text{ for all } j \in \{k-1, \ldots, 0\}$$

*Proof.* We will use induction on the length of the path $v_{i_0} \to \cdots \to v_{i_k}$ in $G_P$.

- Base Case ($k = 1$)

  If $k = 1$, then the path $v_{i_0} \to v_{i_1}$ comprises of a single edge $(v_{i_0}, v_{i_1})$ in $G_P$. So, there is a looping branch $L$ for which $(v_{i_0}, v_{i_1}) \in G_L$, for which there is the privacy constraint $\gamma_{i_0} \leq \gamma_{i_1}$.

  We either have that $i_0 = at(i_1)$ and $c_{i_1} = \texttt{insample} \geq x$, or $i_1 = at(i_0)$ and $c_{i_0} = \texttt{insample} < x$. In the first case, we have that $t_{i_0} t_{i_1}$ is a subsequence of some path $\rho$ in $L$ with $guard(t_{i_1}) = \texttt{insample} \geq x$. In the second case, we have that $t_{i_1} t_{i_0}$ is a subsequence of some path $\rho$ in $L$ with $guard(t_{i_0}) = \texttt{insample} < x$.

- Inductive Step ($k > 1$)

  By the inductive hypothesis, we have one of the following cases:

  1. There exists a path $\rho_1 \in P$ such that $t_{i_0} \cdots t_{i_{k-1}}$ is a subsequence of $\rho_1$ with $guard(t_{i_j}) = \texttt{insample} \geq x$ for all $j \in \{1, \ldots, k-1\}$.

     Since we have the edge $(v_{i_{k-1}}, v_{i_k})$ in $G_P$, there exists a looping branch $L$ for which $(v_{i_{k-1}}, v_{i_k}) \in G_L$, for which there is the privacy constraint $\gamma_{i_{k-1}} \leq \gamma_{i_k}$.

     We either have that $i_{k-1} = at(i_k)$ and $c_{i_k} = \texttt{insample} \geq x$ ($t_{i_{k-1}}$ precedes $t_{i_k}$ in $L$), or $i_k = at(i_{k-1})$ and $c_{i_{k-1}} = \texttt{insample} < x$ ($t_{i_k}$ precedes $t_{i_{k-1}}$ in $L$). Notice, however, that we cannot have that $t_{i_k}$ precedes $t_{i_{k-1}}$, since we have assumed $c_{i_{k-1}} = \texttt{insample} \geq x$.

     So, there exists a path $\rho_2 \in L$ such that $t_{i_{k-1}} t_{i_k}$ is a subsequence of $\rho_2$ with $guard(t_{i_k}) = \texttt{insample} \geq x$.

     Let $j_1$ be the index at which $t_{i_{k-1}}$ appears in $\rho_1$, and $j_2$ be the index at which it appears in $\rho_2$. Then, the path $\rho_1[: j_1]\rho_2[j_2 :]$ is a path in $P$ such that $t_{i_0} \cdots t_{i_k}$

is a subsequence of $\rho_1[:j_1]\rho_2[j_2:]$ with $guard(t_{i_j}) = \texttt{insample} \geq x$ for all $j \in \{1, \ldots, k\}$.

2. There exists a path $\rho_1 \in P$ such that $t_{i_{k-1}} \cdots t_{i_0}$ is a subsequence of $\rho_1$ with $guard(t_{i_j}) = \texttt{insample} < x$ for all $j \in \{k-2, \ldots, 0\}$.

   Similar to the argument above, we either have that $t_{i_{k-1}}$ precedes $t_{i_k}$, or $t_{i_k}$ precedes $t_{i_{k-1}}$ in some looping branch. We cannot have that $t_{i_{k-1}}$ precedes $t_{i_k}$, and so $c_{i_{k-1}}$ is forced to be $\texttt{insample} < x$. We can then construct a path $\rho \in P$ such that $t_{i_k} \cdots t_{i_0}$ is a subsequence of $\rho$ with $guard(t_{i_j}) = \texttt{insample} < x$ for all $j \in \{k-1, \ldots, 0\}$.

This completes the proof. □

**Corollary 4.66.** *The path $v_{i_0} \to \cdots \to v_{i_k}$ is in $G_P$ if and only if there is a looping branch $L$ in $P$ such that $v_{i_0} \to \cdots \to v_{i_k}$ is a path in $G_L$.*

**Proposition 4.67.** *A program $P$ is differentially private if and only if there does not exist a path from $\mathbf{1}$ to $-\mathbf{1}$ in the privacy constraint graph of $P$.*

*Proof.* There is a path from $\mathbf{1}$ to $-\mathbf{1}$ in the privacy constraint graph of $P$ if and only if there is a path from $\mathbf{1}$ to $-\mathbf{1}$ in the privacy constraint graph of some looping branch $L$ in $P$ by Proposition 4.65. This is true if and only if there exists some $L$ which is not differentially private, which is true if and only if $P$ is not differentially private. □

**Theorem 4.68.** *Given a program $P$, we can check whether $P$ is differentially private in linear time in the size of $P$.*

*Proof.* Constructing the privacy constraint graph $G_P$ takes linear time in the size of $P$, on which we can perform a breadth first search to check whether there is a path from $\mathbf{1}$ to $-\mathbf{1}$. This can be done in linear time in the size of $G_P$, which is linear in the size of $P$. □

## 4.11 Deciding Privacy (SAT)

**Proposition 4.69.** *We can check whether the approximate privacy constraints are feasible in linear time in $n$, the number of transitions in the looping branch.*

*Proof.* (Sketch) (If there is a transition in a cycle that outputs `insample`, then the approximate privacy constraints are not feasible, and we can return false. There is a way to encode this also into 2-SAT, but is it really worth it? So we'll only consider $\gamma$ in this sketch).

We will construct a reduction to 2-SAT.

- For every transition $t_i$ in the looping branch, construct the variable $x_i$, which represents whether $\gamma_i = 1$ ($x_i = 1$) or $\gamma_i = -1$ ($x_i = 0$).
- For every constraint $\gamma_j \leq \gamma_i$, construct

$$\neg x_j \vee x_i \tag{1}$$

essentially encoding $(\gamma_j = 1) \implies (\gamma_i = 1)$.

- For every constraint $\gamma_i = 1$, construct the clause

$$x_i \tag{2}$$

and for every constraint $\gamma_i = -1$, construct the clause

$$\neg x_i \tag{3}$$

We can check whether the constraints are feasible by checking whether the resulting 2-SAT instance is satisfiable. This can be done in linear time in $n$ using the algorithm described in (TODO: Cite 2-SAT algorithm). □

In order to check whether a program $P$ is private, we could check whether each looping branch in $P$ is private. However, the number of looping branches in $P$ could be exponential in the size of $P$. Instead, we can check whether *all* looping branches are private at once in linear time by combining the approximate privacy constraints for each looping branch into a single set of constraints.

**Theorem 4.70.** *We can check whether $P$ is private in linear time in the size of $P$.*

*Proof.* We will construct a reduction to 2-SAT. Each looping branch $L$ in $P$ has a set of clauses $D$ constructed above from the approximate privacy constraints. Note that the privacy constraints on a transition or pair of transitions in $P$ *do not* depend on which looping branch $L$ they appear in, only on the structure of the program $P$. Thus, we can combine the clauses $D_L$ for each looping branch $L$ into a single set of clauses $D$.

Claim: $D$ is satisfiable if and only if $D_L$ is satisfiable for all looping branches $L$ in $P$.

*Proof.* ( $\implies$ ) Suppose that $D$ is satisfiable. Then there exists an assignment of variables $x_1, \ldots, x_n$ that satisfies $D$. Let $L$ be a looping branch in $P$. Then the assignment to $x_1, \ldots, x_n$ also satisfies $D_L$, since the clauses in $D_L$ are a subset of the clauses in $C$. Thus, $D_L$ is satisfiable.

( $\impliedby$ ) Suppose that $D$ is not satisfiable. There exists □

□

# 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. **Sky: ¡¡is more elaboration here necessary?¿¿**

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

## 5.1 Two-Variable Programs: GDiPA

In this section, we introduce programs with two threshold variables.

### 5.1.1 Multivariable Transitions

In this section, we introduce two-variable programs in a similar manner to single variable programs by constructing a transition alphabet for two-variable programs.

**Definition 5.1** (Two-variable guards)**.** Let $\texttt{x}, \texttt{y}$ be real-valued program variables. Then a transition **guard** is a boolean statement $c = c(x) \oplus c(y)$ where

- $c(x) \in \{\texttt{true}, \texttt{insample} < \texttt{x}, \texttt{insample} \geq \texttt{x}\}$
- $c(y) \in \{\texttt{true}, \texttt{insample} < \texttt{y}, \texttt{insample} \geq \texttt{y}\}$
- $\oplus \in \{\wedge, \vee\}$

As expected, if, for example, $c = c(x) \wedge \texttt{true}$, we will shorthand $c$ to $c(x)$ and if $c = c(x) \vee \texttt{true}$, we will shorthand $c$ to $\texttt{true}$. In general, we will use $c(x)$ and $c(y)$ to notate the $\texttt{x}$ and $\texttt{y}$ components of a two-variable guard.

Let $\mathcal{C}^{(2)}$ be the set of all possible guards with two variables $\texttt{x}$ and $\texttt{y}$.

We can now define two variable transitions, which differ from single variable transitions only in their guard and in the ability to choose which variable to optionally assign into.

**Definition 5.2** (2v-transitions)**.** A two-variable transition (2v-transition) is a tuple $(q, q', c, \sigma, \tau)$ where

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

### 5.1.2 Two Variable Program Semantics

Two variable transitions semantically operate extremely similarly to single variable transitions: given some threshold values $\mathtt{x}$ and $\mathtt{y}$, each transition $t = (q, q', c, \sigma, \tau)$ will first read in a real number input $\mathtt{in}$, sample two random variables $z^{(x)} \sim \mathtt{Lap}(0, \frac{1}{d\varepsilon})$ and $z^{(x)\prime} \sim \mathtt{Lap}(0, \frac{1}{d'\varepsilon})$ for comparing the input to $\mathtt{x}$ and two more random variables $z^{(y)} \sim \mathtt{Lap}(0, \frac{1}{d\varepsilon})$ and $z^{(y)\prime} \sim \mathtt{Lap}(0, \frac{1}{d'\varepsilon})$ for comparing the input to $\mathtt{y}$. Using these noise variables, the transition then assigns four variables $\mathtt{insample}^{(x)} = \mathtt{in}+z^{(x)}$, $\mathtt{insample}^{(x)\prime} = \mathtt{in}+z^{(x)\prime}$, $\mathtt{insample}^{(y)} = \mathtt{in}+z^{(y)}$, and $\mathtt{insample}^{(x)\prime} = \mathtt{in} + z^{(y)\prime}$. If the guard $c$ is satisfied when comparing $\mathtt{insample}^{(x)}$ to $\mathtt{x}$ and $\mathtt{insample}^{(y)}$ to $\mathtt{y}$, then we transition to location $q'$, outputting $\sigma$ and, depending on $\tau$, optionally reassigning $\mathtt{x} = \mathtt{insample}^{(x)}$ or $\mathtt{x} = \mathtt{insample}^{(y)}$.

In particular, note that the noisy input that the threshold variables are compared to is independent for $\mathtt{x}$ and $\mathtt{y}$.

More formally, a program location is a tuple consisting of a program location and values for $\mathtt{x}$ and $\mathtt{y}$. Let $S = Q \times \mathbb{R} \times \mathbb{R}$ be the set of all possible program locations. As expected, every possible input is simply an element of $\mathbb{R}$. As before, the set of all possible output events is $\Gamma \cup \Sigma$, where $\Sigma$ is the standard $\sigma$-algebra of all Lebesgue measurable sets.

Then the semantics of a 2v-transition $t$ can be defined as a function $\Phi_t((q, \mathtt{x}, \mathtt{y}), \mathtt{in}) : S \to dist(S \times (\Gamma \cup \mathbb{R} \cup \lambda))$ that maps an initial program state and an input to a distribution of subsequent program states and an output event following the expected semantics; $\lambda$ here denotes the empty string (i.e. no output).

The precise semantics are defined exactly analogously to the single variable case.

We again denote the probability that a transition $t = (q, q', c, \sigma, tau)$ "succeeds" as $\mathbb{P}[\mathtt{x}, \mathtt{y}, t, \mathtt{in}, o] = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \mathbb{P}[\mathtt{x} \leftarrow x']\mathbb{P}[\mathtt{y} \leftarrow y']\Phi_t((q, \mathtt{x}, \mathtt{y}))((q', x', y'), o)dx'dy'$, where $\mathtt{x}, \mathtt{y} \in \mathbb{R}$ are the initial values of $\mathtt{x}$ and $\mathtt{y}$, respectively, $\mathtt{in} \in \mathbb{R}$ is a real-valued input, and $o \in \Gamma \cup \Sigma$ is a possible output of $t$.

**Definition 5.3** (Valid Transition Alphabets)**.** A finite 2v-transition alphabet $\Sigma_T$ is valid if it satisfies the following conditions:

- **Initialization:** There exist some $t^{(x)}_{init}, t^{(y)}_{init} \in \Sigma_T$ such that $t^{(x)}_{init} = (q_0, q_1, \mathtt{true}, \sigma_0, 1)$ and $t^{(y)}_{init} = (q_1, q_2, \mathtt{true}, \sigma_1, 2)$ for some $q_0, q_1, q_2 \in Q$, $\sigma_0, \sigma_1 \in \Gamma \cup \{\mathtt{insample}, \mathtt{insample}'\}$.
- **Determinism**: If $\Sigma_T$ 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 $(q, q', c, \sigma, \tau) \in \Sigma_T$ and $(q, q^*, c', \sigma', \tau') \in \Sigma_T$ for distinct guards $c, c'$, then $\sigma \neq \sigma'$. Additionally, at least one of $\sigma \in \Gamma$, $\sigma' \in \Gamma$ is true.
- **Non-input location condition**: For all locations $q \in Q_{non}$, if there exists a transition $t = (q, q', c, \sigma, \tau)$ such that $t \in \Sigma_T$, then $c = \mathtt{true}$.
- **Assignment spread parameter condition**: There exists some constant $d_{at} > 0$ such that for all transitions $t = (q, q', c, \sigma, \tau) \in \Sigma_T$ where $\tau \in \{1, 2\}$, $P(q) = (d_{at}, d'_q)$.

For technical reasons that will become clear later, we introduce a new validity condition (the "assignment spread parameter condition") that requires that all assignment transitions have the same spread parameter on their Laplace noise.

### 5.1.3 Multivariable Couplings

In this section, we introduce two methods for constructing two variable couplings for 2v-transitions; one method combines couplings in parallel and the other combines couplings for different variables together.

We first show that if we can create couplings for each variable in isolation, then we can immediately create a coupling for a combined 2v-transition.

**Lemma 5.4.** *Let* $X\langle 1\rangle \sim \texttt{Lap}(\mu_x\langle 1\rangle, \frac{1}{d_x\varepsilon}), X\langle 2\rangle \sim \texttt{Lap}(\mu_x\langle 2\rangle, \frac{1}{d_x\varepsilon})$ *be random variables be random variables representing possible initial values of* $\boldsymbol{x}$*, respectively, and* $Y\langle 1\rangle \sim \texttt{Lap}(\mu_y\langle 1\rangle, \frac{1}{d_y\varepsilon}), Y\langle 2\rangle \sim \texttt{Lap}(\mu_y\langle 2\rangle, \frac{1}{d_y\varepsilon})$ *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* $\forall \varepsilon > 0$ *and 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*

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

*the lifting* $o\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon} o\langle 2\rangle$ *is valid for* $d = (|\mu_x\langle 1\rangle - \mu_x\langle 2\rangle + \gamma_T^{(x)}|)d_T^{(x)} + (|\mu_y\langle 1\rangle - \mu_y\langle 2\rangle + \gamma_T^{(y)}|)d_T^{(y)} + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q^{(x)}| + |-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q^{(y)}|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q^{(x)\prime}| + |-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q^{(y)\prime}|)d'_q$*, and therefore* $t$ *is* $d\varepsilon$*-differentially private.*

*Proof.* From lemma 4.7, we know that if these constraints are satisfied, we can create liftings such that

- $c(x)$ is satisfied in $\langle 1\rangle \implies c(x)$ is satisfied in $\langle 2\rangle$
- $c(y)$ is satisfied in $\langle 1\rangle \implies c(y)$ is satisfied in $\langle 2\rangle$

Further, $c$ being satisfied is equivalent to either $c(x) \wedge c(y)$ or $c(x) \vee c(y)$. In either case, we can immediately show that $c$ being satisfied in $\langle 1\rangle \implies c$ is satisfied in $\langle 2\rangle$.

Finally, as before, if $\sigma \in \{\texttt{insample}^{(x)}, \texttt{insample}^{(x)\prime}, \texttt{insample}^{(y)}, \texttt{insample}^{(y)\prime}\}$, then for all $\sigma$, $o\langle 1\rangle = \sigma \implies o\langle 2\rangle = \sigma$. By simply adding up the costs from lemma 4.7, we complete the proof. □

This provides an extremely straightforward method of combining coupling strategies for different variables together; indeed, this specific result is immediately extensible to an arbitrary number of variables.

**Corollary 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 $cost(C) = cost(C_x) + cost(C_y)$.*

**Cross-Couplings**

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

**Lemma 5.6.** *Let $X\langle 1\rangle \sim \texttt{Lap}(\mu_x\langle 1\rangle, \frac{1}{d_x\varepsilon}), X\langle 2\rangle \sim \texttt{Lap}(\mu_x\langle 2\rangle, \frac{1}{d_x\varepsilon})$ be random variables be random variables representing possible initial values of $\boldsymbol{x}$, respectively, and $Y\langle 1\rangle \sim \texttt{Lap}(\mu_y\langle 1\rangle, \frac{1}{d_y\varepsilon}), Y\langle 2\rangle \sim \texttt{Lap}(\mu_y\langle 2\rangle, \frac{1}{d_y\varepsilon})$ 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 $d = |\min(\mu_y\langle 1\rangle - \mu_x\langle 1\rangle, 0)| + (|\mu\langle 1\rangle - \mu\langle 2\rangle + \gamma_x|)d_x + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma'_q|)d'_q$.*
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 $d = |\min(\mu_x\langle 1\rangle - \mu_y\langle 1\rangle, 0)| + (|\mu\langle 1\rangle - \mu\langle 2\rangle + \gamma_x|)d_x + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma'_q|)d'_q$.*
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 $d = |\max(0, \mu_y\langle 1\rangle + \gamma_T^{(y)} - \mu_x\langle 1\rangle - \gamma_T^{(x)})| + (|\mu\langle 1\rangle - \mu\langle 2\rangle + \gamma_x|)d_x + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma'_q|)d'_q$.*

4. *If* $c = \textbf{\textit{insample}} \geq \boldsymbol{x} \vee \textbf{\textit{insample}} < \boldsymbol{y}$*, 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* $d = |\max(0, \mu_x\langle 1\rangle + \gamma_T^{(x)} - \mu_y\langle 1\rangle - \gamma_T^{(y)})| + (|\mu\langle 1\rangle - \mu\langle 2\rangle + \gamma_x|)d_x + (|-\textbf{\textit{in}}\langle 1\rangle + \textbf{\textit{in}}\langle 2\rangle - \gamma_q|)d_q + (|-\textbf{\textit{in}}\langle 1\rangle + \textbf{\textit{in}}\langle 2\rangle - \gamma_q'|)d_q'$.

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

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

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

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

**Case 1:**

Create the lifting $\zeta_x\langle 1\rangle + \gamma_{xy} (=)^{\#|\gamma_{xy}|d_{at}\varepsilon} \zeta_y\langle 1\rangle$ 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 $\zeta_x\langle 1 \rangle + \gamma_{xy}(=)^{\#|\gamma_{xy}|d_{at}\varepsilon}\zeta_y\langle 1 \rangle$ 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 $z^{(x)}\langle 2 \rangle (=)^{\#0} z^{(y)}\langle 2 \rangle$. Note that then $\texttt{insample}^{(x)}\langle 2 \rangle = \texttt{in}_i\langle 2 \rangle + z_i^{(x)}\langle 2 \rangle = \texttt{in}_i\langle 2 \rangle + z_i^{(y)}\langle 2 \rangle = \texttt{insample}^{(y)}\langle 2 \rangle$.

Then because $X\langle 2 \rangle \geq Y\langle 2 \rangle$, $\texttt{insample}^{(x)}\langle 2 \rangle < x\langle 2 \rangle \vee \texttt{insample}^{(y)}\langle 2 \rangle \geq y\langle 2 \rangle$ 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.

□

In essence, this allows us to construct liftings for certain transitions "for free" in a manner compatible with existing liftings. In particular, this construction applies to transitions whose guards intuitively correspond to checking if an input is within either the empty set or the entire real line, which are either always true or always false.

**Definition 5.7.** Let $\Sigma_T$ be a valid 2v-transition alphabet. A 2v-path over $\Sigma_T$ is a sequence of transitions $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$ such that for all $i$ there exists a location $q_i \in Q$ 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 $t_{init}^{(x)} t_{init}^{(y)} \cdot \rho$ for some $\rho \in \Sigma_T^*$.

The semantics of a 2v-path are again defined exactly analogously to the single variable case; we denote the probability of a path $\rho$"succeeding" as $\mathbb{P}[\mathtt{x}, \mathtt{y}, \rho, \mathtt{in}, \sigma]$ for initial values $\mathtt{x}, \mathtt{y} \in \mathbb{R}$, input sequence $\mathtt{in}$ and output sequence $\sigma$.

We allow couplings between different assignment transitions: from an $x$ assignment to a $y$ assignment.

**Lemma 5.8.** *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 $\sigma$ of $\rho$ of length $n$ and let $\sigma\langle 1\rangle$, $\sigma\langle 2\rangle$ be random variables representing possible outputs of $\rho$ 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:*
   - $\gamma_i^{(x)} \le \gamma_{at_x(i)}^{(x)}$
   - $c_i = \boldsymbol{insample} < \boldsymbol{x} \vee \boldsymbol{insample} \ge \boldsymbol{y}$ *and* $\gamma_{(at_x(i), at_y(i))} = -\min(\boldsymbol{in}_{at_y(i)}\langle 1\rangle - \boldsymbol{in}_{at_x(i)}\langle 1\rangle, 0)$
   - $c_i = \boldsymbol{insample} < \boldsymbol{x} \wedge \boldsymbol{insample} \ge \boldsymbol{y}$ *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 $c_i(x) = \boldsymbol{insample} \ge \boldsymbol{x}$, then at least one of the following is true:*
   - $\gamma_i^{(x)} \ge \gamma_{at_x(i)}^{(x)}$
   - $c_i = \boldsymbol{insample} \ge \boldsymbol{x} \vee \boldsymbol{insample} < \boldsymbol{y}$ *and* $\gamma_{(at_x(i), at_y(i))} = -\min(\boldsymbol{in}_{at_x(i)}\langle 1\rangle - \boldsymbol{in}_{at_y(i)}\langle 1\rangle, 0)$

- $c_i = \textit{insample} \geq x \wedge \textit{insample} < y$ *and* $\gamma_{(at_x(i),at_y(i))} = -\max(0, \textit{in}_{at_x(i)}\langle 1\rangle + \gamma^{(x)}_{at_x(i)} - \textit{in}_{at_y(i)}\langle 1\rangle - \gamma^{(y)}_{at_y(i)})$

3. *If* $c_i(y) = \textit{insample} < y$*, then at least one of the following is true:*
   - $\gamma_i^{(y)} \leq \gamma^{(y)}_{at_y(i)}$
   - $c_i = \textit{insample} \geq x \vee \textit{insample} < y$ *and* $\gamma_{(at_x(i),at_y(i))} = -\min(\textit{in}_{at_x(i)}\langle 1\rangle - \textit{in}_{at_y(i)}\langle 1\rangle, 0)$
   - $c_i = \textit{insample} \geq x \wedge \textit{insample} < y$ *and* $\gamma_{(at_x(i),at_y(i))} = -\max(0, \textit{in}_{at_x(i)}\langle 1\rangle + \gamma^{(x)}_{at_x(i)} - \textit{in}_{at_y(i)}\langle 1\rangle - \gamma^{(y)}_{at_y(i)})$
4. *If* $c_i(y) = \textit{insample} \geq y$*, then at least one of the following is true:*
   - $\gamma_i^{(x)} \geq \gamma^{(x)}_{at_x(i)}$
   - $c_i = \textit{insample} < x \vee \textit{insample} \geq y$ *and* $\gamma_{(at_x(i),at_y(i))} = -\min(\textit{in}_{at_y(i)}\langle 1\rangle - \textit{in}_{at_x(i)}\langle 1\rangle, 0)$
   - $c_i = \textit{insample} < x \wedge \textit{insample} \geq y$ *and* $\gamma_{(at_x(i),at_y(i))} = -\max(0, \textit{in}_{at_y(i)}\langle 1\rangle + \gamma^{(y)}_{at_y(i)} - \textit{in}_{at_x(i)}\langle 1\rangle - \gamma^{(x)}_{at_x(i)})$
5. *If* $\sigma_i = \textit{insample}^{(x)}$*, then* $\gamma_i^{(x)} = 0$
6. *If* $\sigma_i = \textit{insample}^{(x)\prime}$*, then* $\gamma_i^{(x)\prime} = 0$
7. *If* $\sigma_i = \textit{insample}^{(y)}$*, then* $\gamma_i^{(y)} = 0$
8. *If* $\sigma_i = \textit{insample}^{(y)\prime}$*, 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* $d = \sum_{i=0}^{n-1}(|-\textit{in}_i\langle 1\rangle + \textit{in}_i\langle 2\rangle - \gamma_i|)d_i + (|-\textit{in}_i\langle 1\rangle + \textit{in}_i\langle 2\rangle - \gamma_i'|)d_i' + \sum_{i_x,i_y}|\gamma_{(i_x,i_y)}|d_{at}$*, and therefore* $t$ *is* $d\varepsilon$*-differentially private.*

*Proof.* Follows from lemma 5.6 exactly as lemma 4.19 follows from lemma 4.7. □

This leads to a natural definition of two variable coupling strategies for paths:

**Definition 5.9** (Two Variable Coupling Strategies)**.** A (2v)-coupling strategy for a 2v-path $\rho$ is a collection of shifts $(\gamma^{(x)}, \gamma^{(x)\prime}, \gamma^{(y)}, \gamma^{(y)\prime}, \gamma_{(i,j)})$ such that

- $\gamma^{(x)}, \gamma^{(x)\prime}, \gamma^{(y)}$, and $\gamma^{(y)\prime}$ are functions of a transition $t_i$ in $\rho$ as well as two adjacent inputs $\texttt{in}_i\langle 1\rangle$ and $\texttt{in}_i\langle 2\rangle$ that output a shift in the range [-1, 1].
- $\gamma_{(i,j)}$ is a function of an `x`-assignment transition $t_i$ and a `y`-assignment transition $t_j$ as well as the inputs $\texttt{in}_i\langle 1\rangle$ and $\texttt{in}_j\langle 1\rangle$ that outputs a real-valued shift.
- In particular, we require that $\gamma_{(i,j)}(\texttt{in}_i\langle 1\rangle, \texttt{in}_j\langle 1\rangle) \in \{-\min(\texttt{in}_j\langle 1\rangle - \texttt{in}_i\langle 1\rangle, 0), -\max(0, \texttt{in}_j\langle 1\rangle + \gamma_j^{(y)} - \texttt{in}_i\langle 1\rangle - \gamma_i^{(x)}), -\min(\texttt{in}_i\langle 1\rangle - \texttt{in}_j\langle 1\rangle, 0), -\max(0, \texttt{in}_i\langle 1\rangle + \gamma_i^{(x)} - \texttt{in}_j\langle 1\rangle - \gamma_j^{(y)})\}$

The extension of 2v-paths to looping branches and programs is extremely natural and follows exactly the same structure as single variable programs.

**Definition 5.10** (Two Variable Looping Branches)**.** A 2v-looping branch is simply a looping branch (i.e. a language of complete paths that can be represented by a union-free regular expression) over a valid 2v-transition alphabet $\Sigma_T$.

Just as with the single variable, a single constraint system can fully capture privacy for two variable programs.

**Definition 5.11.** Let $L$ be a 2v-looping branch over a valid transition alphabet $\Sigma_T$. If, for a candidate coupling strategy $C_L = (\gamma, \gamma', \gamma_{(i,j)})$ for $L$, the following constraints are satisfied for all possible inputs $\texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$ and all $i$:

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

(c) $c_i = \texttt{insample} < \texttt{x} \wedge \texttt{insample} \geq \texttt{y}$ and $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_y(i)}\langle 1\rangle + \gamma^{(y)}_{at_y(i)} - \texttt{in}_{at_x(i)}\langle 1\rangle - \gamma^{(x)}_{at_x(i)})$

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 $\gamma_i^{(y)\prime} = 0$
9. If $t_i$ is in a cycle, then $\gamma_i = -\texttt{in}\langle 1\rangle_i + \texttt{in}\langle 2\rangle_i$
10. If $t_i$ is in a cycle, then $\gamma_i' = -\texttt{in}\langle 1\rangle_i + \texttt{in}\langle 2\rangle_i$
11. If $t_i$ is in a cycle, then for all $j$, $\gamma_{(i,j)} = 0$ and $\gamma_{(j,i)} = 0$
12. If $\gamma_{(at_x(i),at_y(i))} = -\min(\texttt{in}_{at_y(i)}\langle 1\rangle - \texttt{in}_{at_x(i)}\langle 1\rangle, 0)$, then $\texttt{in}_{at_y(i)}\langle 1\rangle \geq \texttt{in}_{at_x(i)}\langle 1\rangle$
13. If $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_y(i)}\langle 1\rangle + \gamma^{(y)}_{at_y(i)} - \texttt{in}_{at_x(i)}\langle 1\rangle - \gamma^{(x)}_{at_x(i)})$, then $\texttt{in}_{at_y(i)}\langle 1\rangle \leq \texttt{in}_{at_x(i)}\langle 1\rangle$
14. If $\gamma_{(at_x(i),at_y(i))} = -\min(\texttt{in}_{at_x(i)}\langle 1\rangle - \texttt{in}_{at_y(i)}\langle 1\rangle, 0)$, then $\texttt{in}_{at_x(i)}\langle 1\rangle \geq \texttt{in}_{at_y(i)}\langle 1\rangle$
15. If $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_x(i)}\langle 1\rangle + \gamma^{(x)}_{at_x(i)} - \texttt{in}_{at_y(i)}\langle 1\rangle - \gamma^{(y)}_{at_y(i)})$, then $\texttt{in}_{at_x(i)}\langle 1\rangle \leq \texttt{in}_{at_y(i)}\langle 1\rangle$

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

**Lemma 5.12.** *If a coupling strategy $C$ satisfies the privacy constraint system for a 2v-looping branch $L$, then $cost(C) < \infty$ and $L$ is $cost(C)\varepsilon$-differentially private.*

*Proof.* Because $C$ satisfies the privacy constraint system, we know that $C$ is a valid coupling strategy. It remains to show that $C$ has finite cost. We can separate $cost(C)$ into cost contributed from $\gamma^{(x)}$ and $\gamma^{(x)\prime}$, $\gamma^{(y)}$ and $\gamma^{(y)\prime}$, and cross-coupling cost. We will denote these costs $c_x, c_{x\prime}, c_y, c_{y\prime}$, and $c_{xy}$ such that $cost(C) = c_x + c_{x\prime} + c_y + c_{y\prime} + c_{xy}$.

From lemma 4.34, we know that $c_x + c_{x\prime} + c_y + c_{y\prime}$ is finite because of constraints (9) and (10).

Similarly, observe that by constraints (12) through (15), $|\gamma_{(i,j)}| \leq 2$ because $\gamma^{(x)}_{at_x(i)}, \gamma^{(y)}_{at_y(i)} \in [-1, 1]$. Because of constraint (11), this means that the $c_{xy} \leq 2m$, where $m$ is the total number of distinct assignment transitions in $L$, which is finite. This completes the proof. □

### 5.1.4 Couplings for GDiPA

We introduce a two-variable analogue to DiPA, which allows us to reason about counterexamples more easily.

**Definition 5.13.** A GDiPA $A$ is an 8-tuple $(Q, \Sigma, \mathcal{C}, \Gamma, q_{init}, X, P, \delta)$ where

- $Q$ is a finite set of locations partitioned into input locations $Q_{in}$ and non-input locations $Q_{non}$.
- $\Sigma = \mathbb{R}$ is the input alphabet
- $\mathcal{C} = \mathcal{C}^{(2)}$ is a set of guard conditions
- $\Gamma$ is a finite output alphabet
- $q_{init} \in Q$ is the initial location
- $X = \{\texttt{x}, \texttt{y}, \texttt{insample}^{(x)}, \texttt{insample}^{(x)\prime}, \texttt{insample}^{(y)}, \texttt{insample}^{(y)\prime}\}$ 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 location
- $\delta : (Q \times \mathcal{C}) \to (Q \times (\Gamma \cup \{\texttt{insample}^{(x)}, \texttt{insample}^{(x)\prime}, \texttt{insample}^{(y)}, \texttt{insample}^{(y)\prime}\}) \times \{0, 1, 2\})$ is a partial transition function.

In addition, $\delta$ must satisfy some additional conditions:

- **Determinism:** For any location $q \in Q$, if $\delta(q, c)$ and $\delta(q, c')$ are defined for distinct guards $c, c'$, then $c$ and $c'$ must be logically disjoint events.
- **Output Distinction:** For any location $q \in Q$, if $\delta(q, c) = (q_1, o_1, b_1)$ and $\delta(q, c') = (q_2, o_2, b_2)$, for distinct guards $c, c'$, 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 location $q_0$ has only one outgoing transition of the form $\delta(q_0, \texttt{true}) = (q_1, o, 1)$ and $q_1$ has only one outgoing transition of the form $\delta(q_1, \texttt{true}) = (q, o', 2)$.
- **Non-input transition:** From any $q \in Q_{non}$, if $\delta(q, c)$ is defined, then $c = \texttt{true}$.

Just as with DiPAs and Programs, we can equivalently define a GDiPA as regular language that can be described as a finite union of 2-variable looping branches over a valid 2v-transition alphabet $\Sigma_T$.

**Lemma 5.14.** *If no coupling strategy for a 2v-looping branch $L$ satisfies the privacy constraint system, then there exists a leaking cycle, non-cancelling leaking pair, disclosing cycle, or privacy violating path in a variable in $L$.*

*Proof.* Suppose that we have some maximally satisfied coupling strategy $C$ for $L$. There must be some constraint that is violated by $C$. Note that if constraint (11) is violated, then there must be a leaking cycle in $L$ since we only allow cross-couplings between assignment transitions.

Thus, we can assume that constraint (11) is not violated.

By lemma 4.40, there must then be either a leaking cycle, leaking pair, disclosing cycle, or privacy violating path with respect to a single variable.

We will show that, if there only exist leaking pairs in $L$, then at least one leaking pair must be a non-cancelling leaking pair.

For the sake of contradiction, suppose that every leaking pair $\kappa, \kappa'$ in $L$ is a cancelling leaking pair.

By definition, this means that for every two transitions $t_i, t_j$ in $\kappa, \kappa'$, $at_x(i) = at_x(j)$ and $at_y(i) = at_y(j)$.

Without loss of generality, we will assume that every non-`true` transition in $\kappa$ must either have guard $\texttt{insample} < \texttt{x} \wedge \texttt{insample} \geq \texttt{y}$ or $\texttt{insample} < \texttt{x} \vee \texttt{insample} \geq \texttt{y}$. Thus, every non-`true` transition in $\kappa'$ must either have guard $\texttt{insample} \geq \texttt{x} \wedge \texttt{insample} < \texttt{y}$ or $\texttt{insample} \geq \texttt{x} \vee \texttt{insample} < \texttt{y}$, respectively.

Let $t_i$ be an arbitrary non-`true` transition in $\kappa$ and $t_j$ be an arbitrary non-`true` transition in $\kappa'$.

Consider the case where $c_i = \texttt{insample} < \texttt{x} \wedge \texttt{insample} \geq \texttt{y}$ and $c_j = \texttt{insample} \geq \texttt{x} \wedge \texttt{insample} < \texttt{y}$; the other case is symmetric.

Observe that at least one of $\texttt{in}_{at_x(i)}\langle 1\rangle \leq \texttt{in}_{at_y(i)}\langle 1\rangle$ or $\texttt{in}_{at_y(i)}\langle 1\rangle \geq \texttt{in}_{at_y(i)}\langle 1\rangle$ must be true. Suppose that $\texttt{in}_{at_x(i)}\langle 1\rangle \leq \texttt{in}_{at_y(i)}\langle 1\rangle$. The case where $\texttt{in}_{at_y(i)}\langle 1\rangle \geq \texttt{in}_{at_y(i)}\langle 1\rangle$ is symmetric.

Then we can set $\gamma_{at_x(i)} = -1$, $\gamma_{at_y(i)} = 1$, and $\gamma_{(at_x(i),at_y(i))} = -\max(0, \texttt{in}_{at_x(i)}\langle 1\rangle + \gamma^{(x)}_{at_x(i)} - \texttt{in}_{at_y(i)}\langle 1\rangle - \gamma^{(y)}_{at_y(i)})$.

Because there do not exist disclosing cycles, leaking cycles, or privacy violating paths in either variable, this cannot violate any further constraints. Thus, $C$ is not maximal, which is our contradiction, so $L$ must contain either a leaking cycle, disclosing cycle, privacy violating path, or non-cancelling leaking pair in at least one variable. □

**Lemma 5.15.** *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 dvarepsilon-differentially private for any $d > 0$.*

**Theorem 5.16.** *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 $A$.*

## 5.2 Beyond Two Variables

It is straightforward to extend the definitions of 2v-transitions, paths, looping branches, etc. to more than two variables; it is additionally straightforward to extend results like lemma 5.4 to apply more than two variables. However, it is not immediately clear how cross-couplings would be extended to additional variables; one obvious possibility is to allow for cross-couplings between *all* possible pairs, but the completeness result likely would not generalize in the same fashion.

# 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] Benedek Nagy. Union-free regular languages and 1-cycle-free-path automata. *Publ. Math. Debrecen*, 68(1-2):183–197, 2006.

[12] 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

*Proof of lemma 4.34.* ( $\Longleftarrow$ )

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 $C_\rho$ be the coupling strategy for $\rho$ induced by $C$.

Let $D_\rho$ 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

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

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

$(\implies)$

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 $\rho$ 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 $k \in \mathbb{N}$,

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

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

**Lemma 8.1.** *If a coupling strategy $C = (\gamma, \gamma')$ for a looping branch $L$ is valid and has finite cost, then the following must hold for all $i$:*

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

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

Consider some $t_i$ in a cycle where $c_i = \mathtt{insample} < \mathtt{x}$. Because $C$ is has finite cost, we know from lemma 4.34 that $\gamma_i = -\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle$ for all $\mathtt{in}_i\langle 1\rangle \sim \mathtt{in}_i\langle 2\rangle$. In particular, when $-\mathtt{in}_i\langle 1\rangle + \mathtt{in}_i\langle 2\rangle = 1$, then $\gamma_i = 1$.

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

**Lemma 8.2.** *If a valid finite cost coupling strategy $C = (\gamma, \gamma')$ exists for a lasso $L_\rho$, then there exists a valid finite cost coupling strategy $C^* = (\gamma^*, \gamma^{*\prime})$ such that for all $i \in AT(L_\rho)$, $\gamma_i^* \in \{-1, 1\}$.*

*Proof.* Since $L_\rho$ is differentially private, the cost $opt(L_\rho)$ of its optimal coupling strategy is finite. By Proposition 4.60, we see also that $approx(L)$ is finite. Thus, there exists $\beta \in [-1,1]^n$ which satisfies the approximate privacy constraints on $L_\rho$.

Define

$$\gamma_i = \lfloor \beta_i \rfloor$$

and notice that $\gamma_i, \gamma'$ also satisfy the approximate privacy constraints on $L_\rho$:

$$\begin{aligned}
\beta_{at(i)} \le \beta_i &\implies \lfloor \beta_{at(i)} \rfloor \le \lfloor \beta_i \rfloor \implies \gamma_{at(i)} \le \gamma_i \\
\beta_{at(i)} \ge \beta_i &\implies \lfloor \beta_{at(i)} \rfloor \ge \lfloor \beta_i \rfloor \implies \gamma_{at(i)} \ge \gamma_i \\
\beta_i = 0 &\implies \lfloor \beta_i \rfloor = 0 \implies \gamma_i = 0 \\
\beta_i = 1 &\implies \lfloor \beta_i \rfloor = 1 \implies \gamma_i = 1 \\
\beta_i = -1 &\implies \lfloor \beta_i \rfloor = -1 \implies \gamma_i = -1
\end{aligned}$$

Since $L_\rho$ is private, none of the transitons $t_i$ for $i \in AT(L_\rho)$ are in cycles. **Vishnu: ¡¡Can I say this?¿¿**

By Proposition 4.58, we see that there is a valid coupling strategy $C^* = (\gamma^*, \gamma^{*\prime})$ for $\gamma_i^* = \gamma_i \in \{\pm 1\}$ for all $i \in AT(L_\rho)$. □

*Proof of proposition 4.33.* 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 $\rho$ and a coupling strategy $C_\rho$, recall that we effectively assign each transition $t_i$ in $\rho$ 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 $\rho_n$ be the path in $[\rho]$ with every cycle in $\rho_n$ 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 $\rho_n$ 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 `insample` < `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 $\rho_n$ as $n \to \infty$.

Because we know that all coupling strategies $C_\rho$ 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 $C_\rho$, 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 $\gamma$ 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 $\gamma^*$, and costs attributed to all transitions in $T_{un}$ by $\gamma^*$.

First, note that the coupling cost attributed to $\gamma^{*\prime}$ in $C^*$ must be at most the maximimum coupling cost attributed to $\gamma'$ over all path-specific coupling strategies. From before, we additionally know that the cost attributed to all transitions $t \notin T_{un}$ by $\gamma^*$ 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 $\rho_n$ 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 4.40.* 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 $\rho$ in $A$ such that at least one constraint is not satisfied by $C$ as applied to $\rho$.

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 4.47, 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 $\gamma_i$**

In this case, $c_i = \texttt{insample} < \texttt{x}$ 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 $\rho$ that $t_{at(k)}$ is not in a cycle, since otherwise there would be a leaking cycle in $A$.

**Case 1.1: $t_i$ is in a cycle**

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

Noe that the cycle containing $t_i$ is also an `L`-cycle by definition.

Then attempting to resolve (1) for $\gamma_i$ 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 $\gamma_j$ 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 $c_{at(l)} = \texttt{insample} < \texttt{x}$. 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 $\gamma_j$ 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 $\gamma_j$.

If constraint (3) is violated, then $\gamma_j$ 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 $\gamma_j$ 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 $\gamma_i$ 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 $\gamma_i$ 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 $\gamma_j$ such that $at(j) = at(i)$; note that setting $\gamma_{at(i)} = 0$ cannot *newly* violate constraint (1) for some $\gamma_j$ such that $at(j) = at(i)$. Alternatively, setting $\gamma_i = 0$ could potentially newly violate either constraint (1) or constraint (2) for some $\gamma_j$ 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 $\gamma_j$ 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 $\gamma_j$; 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 $\gamma_j$ such that $at(j) = i$**

If $\gamma_j$ is not an assignment transition, then setting $\gamma_j = \gamma_i = 0$ must violate constraint (5) for $\gamma_j$, 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 $\gamma_j$ 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 $\gamma_j$ 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 $\gamma_i$ 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 $\gamma_j$ 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 $\gamma_j$ such that $at(j) = i$. If $t_j$ is not an assignment transition, then setting $\gamma_j = 0$ can only violate constraint (5) for $\gamma_j$, 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 $\gamma_l$ 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 $\gamma_l$.

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 $\gamma_j$ 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 $\gamma_j$. 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 $\gamma_j$ where $at(j) = i$. If $t_j$ is not an assignment transition, then setting $\gamma_j = -1$ can only newly violate constraint (3) for $\gamma_j$, 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 $\gamma_j$ where $at(j) = i$. If $t_j$ is not an assignment transition, then setting $\gamma_j = -1$ can only newly violate constraint (5) for $\gamma_j$, 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 $\gamma_l$ 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 $\gamma_l$. Thus, $\gamma_l$ 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 $\gamma_l$. 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 $\gamma_j$ such that $at(j) = at(i)$**

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

If $\gamma_i = 0$, then setting $\gamma_j = \gamma_{at(i)} = 0$ can only newly violate constraint (5) for $\gamma_j$. Thus, $\gamma_j$ 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 $\gamma_l$. Further, since $\gamma_i = 1$, there exists some $t_k$ such that there is an AL-path from $t_i$ to $t_k$ and $t_k$ is in an L-cycle. Thus, there is an AL path from $t_j$ to $t_k$.

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

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

**Case 2: (2) is unsatisfied for $\gamma_i$**

This case is exactly symmetric to case (1).

**Case 3: (3) is unsatisfied for $\gamma_i$**

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 $\gamma_i$ or (1) for some $\gamma_l$ such that $at(l) = i$.

**Case 3.1: Satisfying (3) for $\gamma_i$ would violate (1) for $\gamma_i$**

This means that $\gamma_{at(i)} < 0 \implies \gamma_{at(i)} = -1$. Further, $c_i =$ `insample` $<$ `x`. Then changing $\gamma_{at(i)} = 0$ can newly violate constraints (1) or (5) for $\gamma_{at(i)}$ or constraint (2) for some $\gamma_j$ 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 $c_i =$ `insample` $<$ `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 $c_{at(i)} =$ `insample` $<$ `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 $t_{at(i)}$, $c_{at(k)} =$ `insample` $<$ `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 $\gamma_j$ 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 $\gamma_l$ where $at(l) = j$ or (5) for $\gamma_j$.

If setting $\gamma_j = 0$ would violate constraint (2) for some $\gamma_l$ 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 $\gamma_j$, 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 $\gamma_i$ would violate (2) for $\gamma_i$**

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

**Case 3.3: Satisfying (3) for $\gamma_i$ would violate (1) for some $\gamma_l$ 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 $\gamma_l$. 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 $\gamma_l$, then $\gamma_l$ 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 $\gamma_i$ would violate (2) for some $\gamma_l$ where $at(l) = i$**

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

**Case 4: (4) is unsatisfied for $\gamma'_i$**

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 $\gamma_i$.

**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 $\gamma_i$, we must have that $\gamma_{at(i)} < 1$ due to other constraints.

Consider all assignment transitions in $\rho$ 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.

So assume that all assignment transitions in $\rho$ 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 $\gamma_l$ 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 $\gamma_l$ 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 $\gamma_l$ 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 $\gamma_l$ such that $at(l) = at(k)$, then as before, $\gamma_l$ 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 $\gamma_i'$. In particular, constraint (4) must be violated, since no other constraint involves $\gamma_i'$. 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 $\rho$ is an assignment transition and the guard of the transition $q_j \to q_{j+1}$ is not `true`. If $\rho$ is also a cycle, then we call it a leaking cycle.

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

Additionally, a path $\rho$ of a DiPA $A$ is an `AL`-path (respectively, `AG`-path) if all assignment transitions in $\rho$ 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 location in $C$ to a location in $C'$.

2. $C$ is a G-cycle, $C'$ is an L-cycle and there is an AL-path from a location in $C$ to a location 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`.
- $\rho$ 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 `insample` $<$ `x` (resp., `insample` $\geq$ `x`) and outputs `insample`
- $\rho$ 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 `insample` $\geq$ `x` (resp., `insample` $<$ `x`) and outputs `insample`