1 Reworked Transition/Path semantics

1.1 (v1)

1.1.1 Transitions

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

More specifically, given some threshold value $\texttt{x}$, each transition $t = (q, q', c, \sigma, \tau)$ will first read a real number input $\texttt{in}$, sample two random variables $z \sim \texttt{Lap}(0, \frac{1}{d\varepsilon})$, $z' \sim \texttt{Lap}(0, \frac{1}{d'\varepsilon})$, where $P(q) = (d, d')$, and then assign two variables $\texttt{insample} = \texttt{in} + z$ and $\texttt{insample}' = \texttt{in} + z'$. If the guard c is satisfied by comparing $\texttt{insample}$ to $\texttt{x}$, then we transition to location q', outputting $\sigma \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$ 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 state and a real-valued input to a subsequent program state.

We define a program state as a tuple consisting of a program location, a (sub)distribution of possible values for the program value $\texttt{x}$, and a distribution of possible values for the current output σ.

Let $S = Q \times dist(\mathbb{R}) \times (\Gamma \cup dist(\mathbb{R}))^*$ be the set of all possible program states. As expected, every possible input is simply an element of $\mathbb{R}$.

Then the semantics of a transition t can be defined as a function $\Phi_t : dist_{\downarrow}(S) \times \mathbb{R} \to dist_{\downarrow}(S)$ that maps a subdistribution of initial program states and an input to a subdistribution of subsequent program states.

More precisely, if $t = (q, q', c, \sigma, \tau)$, let $P(q) = (d_q, d'_q)$, let $\texttt{insample}$ be the Laplace distribution $\texttt{Lap}(\texttt{in}, \frac{1}{d_q\varepsilon})$, and $\texttt{insample}'$ be the independent Laplace distribution $\texttt{Lap}(\texttt{in}, \frac{1}{d'_q\varepsilon})$. Let θ be a subdistribution of program states.

Then $\Phi_t(\theta, \texttt{in})$ is a subdistribution O such that for all states $(q, \texttt{x}, \sigma_0)$,

$$\mathbb{P}_O[(q', \texttt{x}, \sigma_0 \cdot \sigma)] = \begin{cases} \mathbb{P}_\theta[(q, \texttt{x}, \sigma_0)]\mathbb{P}[c \text{ is satisfied}] & \tau = \texttt{false} \\ 0 & \tau = \texttt{true} \end{cases}$$

$$\mathbb{P}_O[(q', \texttt{insample}, \sigma_0 \cdot \sigma)] = \begin{cases} \mathbb{P}_\theta[(q, \texttt{x}, \sigma_0)]\mathbb{P}[c \text{ is satisfied}] & \tau = \texttt{true} \\ 0 & \tau = \texttt{false} \end{cases}$$

Every other possible state is assigned probability 0 by O; with probability $1 - \sum_{s \in S} \mathbb{P}_O[s]$, we consider the transition to have halted.

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 o, where o is either a symbol from a finite alphabet Γ or a measurable event of $\mathbb{R}$.

We denote this probability as $\mathbb{P}[\mathtt{x}, t, \mathtt{in}, o]$, where $\mathtt{x} \in dist(\mathbb{R})$ is the initial distribution of $\mathtt{x}$, t is a transition, $\mathtt{in} \in \mathbb{R}$ is a real-valued input, and $o \in \Gamma \cup \mathcal{P}(\mathbb{R})$ is a possible output of t. Specifically, let $O = \Phi_t((q, \mathtt{x}, \lambda), \mathtt{in})$, where λ represents the empty string, be a subdistribution; we abuse notation here by using $(q, \mathtt{x}, \lambda)$ to represent the subdistribution that assigns probability 1 to the state $(q, \mathtt{x}, \lambda)$. Then $\mathbb{P}[\mathtt{x}, t, \mathtt{in}, o] = \int_{-\infty}^{\infty} \mathbb{P}_O[\mathtt{x} = x]\mathbb{P}_O[(q', x, o)]dx$. **Sky: ¡¡is $\mathbb{P}_O[\mathtt{x} = x]$ coherent?¿¿**

Note that this is an aggregated probability over all possible final values of $\mathtt{x}$ — it is not particularly important what the specific final value of $\mathtt{x}$ is.

1.1.2 Paths

The semantics of a path can be defined as a function mapping a subdistribution of program states and a real-valued input sequence to a subdistribution of final program states.

More specifically, the semantics of a path $\rho = t_0 t_1 \cdots t_{n-1}$ are a function $\Phi_\rho : dist_\downarrow(S) \times \mathbb{R}^n \to dist_\downarrow(S)$. Φ_ρ can be defined by composing transition semantics:

$$\Phi_\rho(s, \mathtt{in}) = \begin{cases} \Phi_{t_0}(s, \mathtt{in}_0) & |\rho| = 1 \\ \Phi_{t_{n-1}}(\Phi_{\rho_{0:n-1}}(s, \mathtt{in}_{0:n-1}), \mathtt{in}_{n-1}) & |\rho| > 1 \end{cases}$$

Sky: ¡¡Is there a nicer way to write this that I'm forgetting¿¿

Like with transitions, we denote the probability that a path $\rho = q_0 \to \ldots \to q_n$ "succeeds" given an initial $\mathtt{x} \in dist(\mathbb{R})$, input sequence $\mathtt{in} \in \mathbb{R}^n$, and possible output sequence $o \in (\Gamma \cup \Sigma)^n$as $\mathbb{P}[\mathtt{x}, \rho, \mathtt{in}, o] = \int_{-\infty}^{\infty} \mathbb{P}_O[\mathtt{x} = x]\mathbb{P}_O[(q_n, x, o)]dx$, where O is the subdistribution produced by $\Phi_\rho((q, \mathtt{x}, \lambda), \mathtt{in})$. **Sky: ¡¡Same issue with $\mathbb{P}_O[\mathtt{x} = x]$¿¿**

2 Without program locations

2.1 Transitions

Definition 2.1 (Transitions)**.** A transition is a tuple $t = (c, \sigma, \tau)$, where

- $c \in \{\mathtt{true}, \mathtt{insample} < \mathtt{x}, \mathtt{insample} \geq \mathtt{x}\}$ is a guard for the transition.
- $\sigma \in \Gamma \cup \{\mathtt{insample}, \mathtt{insample}'\}$ is the output of t.
- $\tau \in \{\mathtt{true}, \mathtt{false}\}$ is a boolean value indicating whether or not the stored value of $\mathtt{x}$ will be updated.

We additionally associate two real-valued noise parameters $P(t) = (d, d')$ with each transition.

2.1.1 Semantics

[...]

Let $S = dist(\mathbb{R}) \times (\Gamma \cup dist(\mathbb{R}))^*$ be the set of all possible program states. As expected, every possible input is an element of $\mathbb{R}$.

Then the semantics of a transition t can be defined as a function $\Phi_t : dist_\downarrow(S) \times \mathbb{R} \to dist_\downarrow(S)$ that maps a subdistribution of initial program states and an input to a subdistribution of subsequent program states.

More precisely, if $t = (c, \sigma, \tau)$, let $P(t) = (d_t, d'_t)$, let `insample` be the Laplace distribution $\texttt{Lap}(\texttt{in}, \frac{1}{d_t \varepsilon})$, and `insample`$'$ be the independent Laplace distribution $\texttt{Lap}(\texttt{in}, \frac{1}{d'_t \varepsilon})$. Let θ be a subdistribution of program states.

Then $\Phi_t(\theta, \texttt{in})$ is a subdistribution O such that for all states $(\texttt{x}, \sigma_0)$,

$$\mathbb{P}_O[(\texttt{x}, \sigma_0 \cdot \sigma)] = \begin{cases} \mathbb{P}_\theta[(\texttt{x}, \sigma_0)]\mathbb{P}[c \text{ is satisfied}] & \tau = \texttt{false} \\ 0 & \tau = \texttt{true} \end{cases}$$

$$\mathbb{P}_O[(\texttt{insample}, \sigma_0 \cdot \sigma)] = \begin{cases} \mathbb{P}_\theta[(\texttt{x}, \sigma_0)]\mathbb{P}[c \text{ is satisfied}] & \tau = \texttt{true} \\ 0 & \tau = \texttt{false} \end{cases}$$

Every other possible state is assigned probability 0 by O; with probability $1 - \sum_{s \in S} \mathbb{P}_O[s]$, we consider the transition to have halted.

[...]

2.2 Paths

Definition 2.2 (Paths)**.** A path is a string composed of transitions. For a path $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$, if t_0 is of the form $t_0 = (\texttt{true}, \sigma, \texttt{true})$ for some σ, then we call ρ a **complete** path.

2.2.1 Semantics

[identical to above]

2.3 Branching

In a vacuum, each path in a branching program describes a completely isolated program. However, by associating transitions with **program locations**, we can combine paths together to model program branching.

Suppose we have some finite set Q of program locations.

Definition 2.3 (Fixed Transition)**.** A fixed transition is a transition that transitions between two specific program locations. More precisely, a fixed transition is a transition $t = (c, \sigma, \tau)$ along with a start program location $q \in Q$ and a destination program location $q' \in Q$. We will often use the notation $t = (q, q', c, \sigma, \tau)$ to denote the fixed transition from q to q'.

Definition 2.4 (Fixed Path)**.** A fixed path is a sequence of fixed transitions $\rho = t_0 \cdots t_{n-1}$ such that for all i, t_i is of the form $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ for program locations $q_0, \ldots, q_n \in$

Q. A fixed path is complete if, for some $q_0, q_1 \in Q$, the fixed transition t_0 has the form $t_0 = (q_0, q_1, \texttt{true}, \sigma_0, \texttt{true})$. A fixed path can alternatively be modeled as a string of the form $(Q\Sigma_T)^*Q$, where Σ_T is a finite alphabet of transitions.

We can now define branching programs.

Definition 2.5 (Branching Programs)**.** A branching program is a finite set of complete fixed paths over a finite transition alphabet Σ_T.

Let $\Sigma(B)$ be the set of all fixed transitions in a branching program B.

To ensure that a branching program models a coherent program, we must impose some constraints:

Definition 2.6 (Coherence)**.** A branching program B is coherent if it satisfies all of the following properties:

- **Initialization:** There exists some $q_0, q_1 \in Q$ such that for all paths $\rho = t_0 \cdots t_{n-1} \in B$, $t_0 = (q_0, q_1, \texttt{true}, \sigma_0, \texttt{true})$ for some σ_0.
- **Determinism**: If any fixed transition $t \in \Sigma(B)$ is of the form $t = (q, q', c, \sigma, \tau)$, then no other fixed transitions of the form $(q, q^*, c, \sigma', \tau')$ for $q, q', q^* \in Q$ exist in $\Sigma(B)$. Additionally, if there exists a transition $t = (q, q', \texttt{true}, \sigma, \tau)$ such that $t \in \Sigma(B)$, 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 $\Sigma(B)$.
- **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 $c = \texttt{true}$. **Sky: ¡¡necessary for *equivalence* with DiPA, not necessary for counterexample results¿¿**

3 Original

We now begin the process of building up a program model through the lens of regular languages. We will model programs as simply regular languages comprised of possible program paths (or executions) where each path corresponds to a word in the language. This approach simultaneously allows us to 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 our alphabet.

Sky: ¡¡todo: add a note about treating ε as a parameter to the program¿¿

3.1 Individual Transitions

We will define transitions over an underlying finite set of program **locations**; a transition defines how a program moves from one location to the next.

Definition 3.1 (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** using the function $P : Q \to \mathbb{R}^{\geq 0} \times \mathbb{R}^{\geq 0}$.

Transitions act as guarded statements whose guard is dependent on a persistent real-valued variable `x` and a real-valued input to which random noise is added; we can thus formally define individual transitions as follows:

Definition 3.2 (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.
- $c \in \{\texttt{true}, \texttt{insample} < \texttt{x}, \texttt{insample} \geq \texttt{x}\}$ is a guard for the transition.
- $\sigma \in \Gamma \cup \{\texttt{insample}, \texttt{insample}'\}$ 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 $q \to q'$.

3.1.1 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 set 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 3.3 (Valid Transition Alphabets)**.** Let Σ_T be a finite alphabet of transitions. We call Σ_T **valid** if it satisfies the following conditions:

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

3.1.2 Transition Semantics

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

More specifically, given some threshold value $\texttt{x}$, each transition $t = (q, q', c, \sigma, \tau)$ will first read a real number input $\texttt{in}$, sample two random variables $z \sim \texttt{Lap}(0, \frac{1}{d\varepsilon})$, $z' \sim \texttt{Lap}(0, \frac{1}{d'\varepsilon})$, where $P(q) = (d, d')$, and then assign two variables $\texttt{insample} = \texttt{in} + z$ and $\texttt{insample}' = \texttt{in} + z'$. If the guard c is satisfied by comparing $\texttt{insample}$ to $\texttt{x}$, then we transition to location q', outputting σ 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, an output value, and a possibly new value for $\texttt{x}$.

To be precise, we define a program state as a tuple consisting of a program location and a value for the program variable $\texttt{x}$. Let $S = Q \times \mathbb{R}$ be the set of all possible program states. As expected, every possible input is simply an element of $\mathbb{R}$. An output can either be a symbol from some finite alphabet Γ or a real number; thus, the set of all possible output events is $\Gamma \cup \Sigma$, where Σ is some σ-algebra over $\mathbb{R}$. In particular, we will take Σ to be the standard σ-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 $q \in Q$ and an input $\texttt{in} \in \mathbb{R}$ to a distribution of subsequent program locations and an output event following the expected semantics; λ here denotes the empty string (i.e. no output).

[If we need space, 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 $\texttt{insample}$ 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 $\texttt{x}$ 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. Equivalently, we could say that the program simply terminates and fails to transition to any new state.

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, $\mathbb{P}[\texttt{x}, t, \texttt{in}, o] = \int_{-\infty}^{\infty} \mathbb{P}[\texttt{x} \leftarrow \texttt{x}'] \Phi_t((q, \texttt{x}), \texttt{in})((q', \texttt{x}'), o) d\texttt{x}'$, where $\mathbb{P}[\texttt{x} \leftarrow \texttt{x}']$ is the probability that $\texttt{x}$ is updated to have the value $\texttt{x}'$.

Note that this is an aggregated probability over all possible final values of $\texttt{x}$—it is not particularly important what the specific final value of $\texttt{x}$ is.

3.1.3 Couplings

We will now construct couplings for transitions with the aim of using them as building blocks for proofs of privacy.

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 3.4 (Adjacency)**.** Two inputs $\texttt{in} \sim_\Delta \texttt{in}'$ are Δ-adjacent if $|\texttt{in} - \texttt{in}'| \leq \Delta$. If Δ 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 3.5 (Valid inputs)**.** Let $t = (q, q', c, \sigma, \tau)$ be a transition over Q. A valid input to t is a real number $\texttt{in}$ such that if $q \in Q_{non}$, then $\texttt{in} = 0$. In general, we will assume that all inputs are valid.

To construct approximate liftings, we will analyze the behaviour of two different **runs** of a transition $t = (q, q', c, \sigma, \tau)$, 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 $\texttt{x}$ at location q in the run of t with input $\texttt{in}\langle 1 \rangle$ and $X\langle 2 \rangle$ to be the value of $\texttt{x}$ in the run of t with input $\texttt{in}\langle 2 \rangle$.

We thus want to create the lifting $o\langle 1 \rangle \{(a, b) : a = \sigma \implies b = \sigma\} o\langle 2 \rangle$, where $o\langle 1 \rangle$ and $o\langle 2 \rangle$ are random variables representing the possible outputs of $t\langle 1 \rangle$ and $t\langle 2 \rangle$, respectively.

We must guarantee two things: that if the first transition is taken, then the second is also taken and that both runs output the same value σ when taking the transition. Note that if $c = \texttt{true}$, the first condition is trivially satisfied and when $\sigma \in \Gamma$, the second condition is trivially satisfied.

This gives us our major coupling lemma, which defines a family of couplings for privacy proofs.

Lemma 3.6. *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 some valid transition alphabet Σ_T. Let $P(q) = (d_q, d'_q)$.*

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

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

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

the lifting $o\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon} o\langle 2\rangle$ is valid for $d = (|\mu\langle 1\rangle - \mu\langle 2\rangle + \gamma_x|)d_x + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma_q|)d_q + (|-\boldsymbol{in}\langle 1\rangle + \boldsymbol{in}\langle 2\rangle - \gamma'_q|)d'_q$.

Definition 3.7 (Valid Coupling Strategies)**.** A **valid coupling strategy** for a transition $t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i)$ is a tuple $C_i = (\gamma_x^{(i)}, \gamma_i, \gamma'_i) \in [-1, 1]^3$ such that 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.

3.2 Program Paths

We now demonstrate how to concatenate transitions and their corresponding coupling strategies together into program *paths*.

Definition 3.8 (Program paths)**.** Let Σ_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, σ_i, τ_i. We will often notate an path ρ as $\rho = q_0 \to q_1 \to \ldots \to q_n$.

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

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

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

Given a path (or sequence) $\rho = t_0 \cdot t_1 \cdot \ldots \cdot t_{n-1}$, the **tail** of ρ is notated by $tail(\rho) = t_1 \cdot \ldots \cdot t_{n-1}$. We may additionally use the notation $\rho_{i:j}$ to represent the subpath (or subsequence) $t_i t_{i+1} \cdots t_{j-1}$ of ρ. 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 3.9. For a path ρ of length n, an input sequence $\texttt{in} \in \mathbb{R}^n$ is valid if, for all q_i in ρ such that $q_i \in Q_{non}$, $\texttt{in}_i = 0$.

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

3.2.1 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 $\texttt{x}$. 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.

Φ_ρ 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 $\texttt{insample}$ and $\texttt{insample}'$.

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 $\mathtt{x}$, ρ 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 ρ. As before, $\mathbb{P}[\mathtt{x}_0, \rho, \mathtt{in}, \sigma] = \int_{-\infty}^{\infty} \mathbb{P}[\mathtt{x} \leftarrow \mathtt{x}']\Phi_\rho((q_0,\mathtt{x}_0),\mathtt{in})((q_n,\mathtt{x}'),\sigma)d\mathtt{x}'$, where $t_0 = (q_0, q_1, c_0, \sigma_0, \tau_0)$ is the first character of ρ and $t_{n-1} = (q_{n-1}, q_n, c_{n-1}, \sigma_{n-1}, \tau_{n-1})$ is the final character of ρ.

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

3.2.2 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 3.11 (Adjacency for a sequence of inputs)**.** Two input sequences $\{\mathtt{in}_i\langle 1\rangle\}_{i=1}^n, \{\mathtt{in}_i\langle 2\rangle\}_{i=1}^n$ of length n are Δ-adjacent (notated $\mathtt{in}\langle 1\rangle \sim_\Delta \mathtt{in}\langle 2\rangle$) if, for all $i \in [1 \ldots n]$, $|\mathtt{in}_i\langle 1\rangle - \mathtt{in}_i\langle 2\rangle| \leq \Delta$.

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

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

Definition 3.12 ($d\varepsilon$-differential privacy for a path)**.** A complete path ρ of length n is $d\varepsilon$-differentially private for some $d > 0$ if $\forall \varepsilon > 0$, for all valid adjacent input sequences $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$ of length n and all possible output sequences σ of length n, $\mathbb{P}[\rho, \mathtt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[\rho, \mathtt{in}\langle 2\rangle, \sigma]$.

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 3.13. 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 $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$, $\mathbb{P}[\rho, \mathtt{in}\langle 1\rangle, \sigma] \leq e^{d\varepsilon}\mathbb{P}[\rho, \mathtt{in}\langle 2\rangle, \sigma]$.

We observe that because of the path-output correspondence, we can equivalently look at each path of a set in isolation:

Proposition 3.14. *Let S be a set of complete paths; S is $d\varepsilon$-differentially private for some $d > 0$ if and only if, for all paths $\rho \in S$, ρ is $d\varepsilon$-differentially private.*

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

3.2.3 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 $o\langle 1\rangle, o\langle 2\rangle$ are random variables representing the output of ρ given input sequences $\mathtt{in}\langle 1\rangle$ and $\mathtt{in}\langle 2\rangle$, respectively, then in order to show that a program path ρ is differentially private we want to create the coupling $o\langle 1\rangle\{(a, b) : a = \sigma \implies b = \sigma\}^{\#d\varepsilon}o\langle 2\rangle$ for some $d > 0$ for all adjacent inputs $\mathtt{in}\langle 1\rangle \sim \mathtt{in}\langle 2\rangle$ and all possible outputs σ.

As it turns out, directly composing together the couplings from lemma 3.6 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 ρ (i.e. the most recent transition t_j such that $\tau_j = \mathtt{true}$). The coupling shifts for *non-assignment transitions* can thus never impact each other.

Definition 3.15 (Assignment transitions)**.** Let $A_\rho = \{t_i = (q_i, q_{i+1}, c_i, \sigma_i, \tau_i) : \tau_i = \mathtt{true}\}$ be the set of **assignment transitions** in a path ρ. Additionally, for every transition t_i in ρ, let $t_{at(i)}$ be the most recent assignment transition in ρ; i.e., $at(i) = \max\{j < i : t_j \in A_\rho\}$. If such a j does not exist, we set $at(i) = -1$.

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

Thus, for an individual transition t_i of ρ, 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 3.16. *Let $\rho = q_0 \to \ldots \to q_n$ be a complete path of length n. Let $\boldsymbol{in}\langle 1\rangle \sim \boldsymbol{in}\langle 2\rangle$ be arbitrary adjacent input sequences of length n. Additionally, fix some potential output σ of ρ of length n and let $\sigma\langle 1\rangle$, $\sigma\langle 2\rangle$ be random variables representing possible outputs of ρ given inputs $\boldsymbol{in}\langle 1\rangle$ and $\boldsymbol{in}\langle 2\rangle$, respectively. Additionally, for all q_i, let $P(q_i) = (d_i, d_i')$.*

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

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

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

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

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

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

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

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

for all i.

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

In particular, note that we have created at most one pair of couplings (for `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 3.16 implicitly defines an entire family of possible coupling proofs that demonstrate the privacy of a path.

Definition 3.17. For a complete path ρ 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 ρ 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 3.18. For a complete path ρ of length n, a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is **valid** if $\forall \texttt{in}\langle 1\rangle \sim \texttt{in}\langle 2\rangle$, $\boldsymbol{\gamma}(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle)$ and $\boldsymbol{\gamma}'(\texttt{in}\langle 1\rangle, \texttt{in}\langle 2\rangle)$ satisfy the constraints

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

Thus, if we have a **valid** coupling strategy C for a path ρ, then immediately by lemma 3.16, we have a proof that ρ is $d\varepsilon$-differentially private.

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

Definition 3.19. For a complete path ρ of length n, the **cost** of a coupling strategy $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is

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

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

Corollary 3.20. *If $C_\rho = (\boldsymbol{\gamma}, \boldsymbol{\gamma}')$ is valid, then ρ is $cost(C_\rho)\varepsilon$-differentially private.*

3.3 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 3.21 (Branching program)**.** A branching program B is a finite set of complete paths over a valid transition alphabet Σ_T.

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

3.3.1 Privacy

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

Definition 3.22 (Coupling strategies)**.** A coupling strategy C for a branching program B is a collection of (path) coupling strategies where each complete path $\rho \in B$ is assigned a coupling strategy C_ρ.

Definition 3.23. A coupling strategy C for a branching program B is valid if, for every constituent path coupling strategy C_ρ, C_ρ is valid.

Definition 3.24. 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.

Sky: ¡¡note: rephrase this proposition slightly¿¿

Proposition 3.25. *Optimal cost is dependent on path. There exists a valid transition alphabet Σ_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 ρ.*

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_{ρ_1} be a coupling strategy for ρ_1. We can bound its cost as follows:

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

  and by a similar argument, $cost(C_{\rho_2}) \geq 2$ for any coupling strategy C_{ρ_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 ρ_1 and ρ_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_{ρ_1}, then $cost(C_{\rho_1}) > 2$ in the same method as the above, a contradiction.

  Thus, $\gamma_0(0,1) = -1$ in C_{ρ_1}, which by hypothesis, means that $\gamma_0(0,1) = -1$ in C_{ρ_2}. We have the privacy constraint $\gamma_i \leq \gamma_0$ on ρ_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 ρ_1 and ρ_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.

Proposition 3.26. *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 $\texttt{insample} \geq x$.
- $t^{(i)}_{leq}$ are non-assignment transitions with guard $\texttt{insample} < x$.
- 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 $\texttt{insample} < x$.

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 3.25, 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 ρ, which has $n\cdot(n+2)$ transitions, we get

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

whereas there exists another coupling strategy $C^*_\rho = (\gamma^*, \gamma'^*)$ for ρ 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. □