Notes on DIMACS Work

Aaron Jaggard, Catherine Meadows, Michael Mislove, Roberto Segala

July 2, 2008

Abstract

These are notes based on discussions at DIMACS in May 2008.

1 Dining Cryptographers

Figures 1, 2, and 3 describe the master process, the signature and state set of a cryptographer, and the transition relation of a cryptographer, respectively. In the definitions above, each action name is a task. Thus, one of the tasks is $\{pay_i(T), pay_i(F)\}$.

From the discussion about the correctness of Dining Cryptographers we ended up with the following informal formulation of correctness: for every scheduler σ and every observation o of player 0, $\mu_\sigma(\mathcal{M}_1\|\mathcal{C}_0\|\mathcal{C}_1\|\mathcal{C}_2)(o) = \mu_\sigma(\mathcal{M}_2\|\mathcal{C}_0\|\mathcal{C}_1\|\mathcal{C}_2)(o)$, where with the notation $\mu_\sigma(\mathcal{M}_1\|\mathcal{C}_0\|\mathcal{C}_1\|\mathcal{C}_2)(o)$ we mean the probability of o in the probabilistic execution of $\mathcal{M}_1\|\mathcal{C}_0\|\mathcal{C}_1\|\mathcal{C}_2$ induced by σ. In other words, for every scheduler σ cryptographer 0 has no way to distinguish the situation in which cryptographer 1 is paying from the situation in which cryptographer 2 is paying. The scheduler has to be restricted in power, and we reached the conclusion that according to the definition given in the CONCUR paper of Chatzikokolakis and Palamidessi [3] using task schedulers should be ok.

We take this definition for granted, given that we derived it from the CONCUR paper mentioned above. However, we will need to see whether there are alternative definitions (e.g., by exchanging quantifications) that are equivalent or more interesting.

An observation of a cryptographer should be seen as the actual execution performed by the cryptographer. Thus, in the framework of task PIOAs, we can reformulate the correctness condition as follows: let $\mathcal{P}_i$, $i \in \{0, 1, 2\}$, be $\mathcal{M}_i\|\mathcal{C}_0\|\mathcal{C}_1\|\mathcal{C}_2$. Then,

$\mathcal{M}$

Signature:

Input:

Output:
$pay_i(c), c \in \{T, F\}, i \in \{0, 1, 2\}$

Internal:

Tasks:

$choose\_culprit = \{choose\_culprit_0,$
$choose\_culprit_1, choose\_culprit_2\}$

Task Distributions:

$\sum_{i=0}^{2} r_i \, \delta_{choose\_culprit=i}$

State:
$tell\_crypto_i \in \{\perp, T, F\}$, $i \in \{0, 1, 2\} - \{n\}$, initially F
$tell\_crypto_n \in \{\perp, T, F\}$, initially T

Transitions:

Output $pay_i(c)$
Precondition:
$tell\_crypto_i = c$
Effect:
$tell\_crypto_i := \perp$

Figure 1: The Master automaton that chooses the culprit

$\mathcal{C}_i$

Signature:

Input:

$pay_i(c), c \in \{T, F\}$

$coin_{i-1}(c), c \in \{T, F\}$

$announce_j(c), j \in \{0, 1, 2\} - \{i\}, c \in \{T, F\}$

Output:

$coin_i(c), c \in \{T, F\}$

$announce_i(c), c \in \{T, F\}$

Internal:

$flip_i$

State:

$lcoin \in \{\bot, T, F\}$ initially $\bot$

$ncoin \in \{\bot, T, F\}$ initially $\bot$

$compare_j \in \{\bot, T, F\}, j \in \{0, 1, 2\}$ initially $\bot$

$payer \in \{\bot, T, F\}$ initially $\bot$

$coin\text{-}sent \in \{T, F\}$ initially F

$announced \in \{T, F\}$ initially F

Figure 2: The signature and state set of cryptographer i

$$\forall \rho \ \mu_{\mathcal{P}_1,\rho}\lceil\mathcal{C}_0 = \mu_{\mathcal{P}_2,\rho}\lceil\mathcal{C}_0. \tag{1}$$

Actually, to be more precise, we should do the same operation for each cryptographer, so we should write the same formula for $\mathcal{C}_1$ and $\mathcal{C}_2$. We omit this for the moment.

Now we would like to use simulation relations to prove the equality above. Thus, we want to find a simulation relation from $\mathcal{P}_1$ to $\mathcal{P}_2$ and vice versa that implies the equality above. In particular, given that the scheduler must be the same in both probabilistic executions. This seems to be achievable by requiring the map c from tasks of the implementation to sequences of tasks of the specification to be the identity.

Observe first that we can rewrite the correctness equation as

$$\forall \rho \ \mu_{\mathcal{P}'_1,\rho}\lceil\mathcal{C}_0 = \mu_{\mathcal{P}'_2,\rho}\lceil\mathcal{C}_0, \tag{2}$$

where $\mathcal{P}'_i$ is defined as $Hide_{A_0}(\mathcal{P}_i)$ and A_0 is the set of actions that are not external in $\mathcal{C}_0$.

We can prove a theorem of this style: Equation (2) is true if all actions of $\mathcal{C}_0$ are external and we can find a simulation relation from $\mathcal{P}'_1$ to $\mathcal{P}'_2$ and

Transitions:

Input $pay_i(c)$
  Effect:
    $payer = c$

Input $coin_{i-1}(c)$
  Effect:
    $ncoin = c$
    if $lcoin \neq \perp$ then $compare_i = lcoin \oplus c$

Input $announce_j(c)$
  Effect:
    $compare_j = c$

Internal $flip_i$
  Precondition:
    $lcoin = \perp$
  Effect:
    $lcoin :=_R \{T, F\}$
    if $ncoin \neq \perp$ then $compare_i = lcoin \oplus c$

Output $coin_i(c)$
  Precondition:
    $coin\text{-}sent = F$
    $lcoin = c$
  Effect:
    $coin\text{-}sent := T$

Output $announce_i(c)$
  Precondition:
    $announced = F$
    $payer \neq \perp$
    $c = compare_i \oplus payer$
  Effect:
    $announced := T$

Figure 3: The transition relation of cryptographer i

vice versa that is the identity on $\mathcal{C}_0$ and such that the task map c is the identity. In practice we need to externalize also actions flip_i, which is not a problem, though.

The simulation essentially should be the identity function with the exception that the values of payer_1 and payer_2 are reversed, when defined, the value of *lcoin* of $\mathcal{C}_1$ is reversed, when defined, and similarly the value of *ncoin* of $\mathcal{C}_2$ is reversed.

To prove that the simulation is correct we can try to be smarter than just applying the step condition. In particular we can check the step condition just from single states and single actions. Of course we need to prove a theorem stating that this is sound. This follows from the fact that we have defined the relation on states. The relation on measures is just the lifting.

Almost all actions are rather trivial to handle since they affect the states in the same way in the specification and the implementation. The actions that have an impact are flip_1 where essentially we have to exchange the outcomes and where this can be done because all coins are fair, coin_1, where the actual action is different but causes no problem since it is internal. In both cases we just need to make sure that variable $\mathit{compare}_?$ is set correctly. We will work out the details later.

From the discussion it emerged that it is possible to exhibit a simulation relation in the style above from any master that probabilistically chooses between $\mathcal{C}_1$ and $\mathcal{C}_2$ as payer. There should be also a general way to say that a randomized master is just a convex combination of deterministic masters, and thus, by convexity, the correctness under randomized masters follows from the correctness under deterministic masters.

We have discussed what happens if we consider randomized schedulers. We can define a randomized task scheduler just as a sequence of probability measures on tasks.

We have also discussed about what would happen if we leave the master nondeterministic. Correctness follows provided that we compare two task schedulers that differ only on the probabilities chosen within the master. A formal definition of this idea is not immediate. A simplification is to state that the choice of the master is resolved at the first step.

We should get the details of this proof worked out as well as the general theorems that are needed within the model. It should be useful also to investigate what we can do with respect to the more general cases studies by Kostas and Catuscia in the context of the dining cryptographers.

2 The GMW Framework for Computing a function

We have investigated how to use the Task-PIOAs framework to analyze the passive adversary case of the protocol. We want to compose oblivious transfer with implementation of and and or gates. Using compositionality we can replace the implementations needed for oblivious transfer with ideal specifications. We can prove correctness of the gates assuming ideal oblivious transfer, and again by compositionality get to a point where we have ideal gates only and from there derive correctness.

We need to check whether the compositionality result of Cheung et. al., CSF 2007, applies and see exactly how to use it. We have not thought of active adversaries yet, but should be interesting to consider them. The main point of the case study is to show that compositionality can simplify greatly the analysis.

3 Probabilistic Input/Output Automata

Our approach utilizes task-structured probabilistic input/output automata as a basic model of computation. In this section, we make this notion precise, and also make precise what we mean by a computation in this setting.

A *Probabilistic Input/Output Automaton* $\mathcal{A}$ is a 6-tuple (Q, q_0, I, O, H, D) where:

1. Q is a countable set of *states*;
2. $q_0 \in Q$ is the *start state*;
3. I, O and H are pairwise disjoint, countable sets of actions, referred to as *input, output and internal actions*, respectively. The set $\mathsf{Act} ::= I \cup O \cup H$ is called the set of *actions* of $\mathcal{A}$. If $I = \emptyset$, then $\mathcal{A}$ is *closed*. The set of *external* actions of $\mathcal{A}$ is $E ::= I \cup O$ and the set of *locally controlled* actions is $L ::= O \cup H$.
4. $D \subseteq Q \times A \times \mathsf{Prob}(Q)$ is a *transition relation*. An action a is *enabled* in a state q if $(q, a, \mu) \in D$ for some μ.

In addition, $\mathcal{A}$ satisfies:

- **Input enabling:** For every $q \in Q$ and $a \in I$, a is enabled in q.

- **Transition determinism:** For every $q \in Q$ and $a \in A$, there is at most one $\mu \in \mathsf{Prob}(Q)$ such that $(q, a, \mu) \in D$.

We denote probabilistic input-output automata as PIOAs. We have retained the traditional assumptions from, e.g., [8], that the state space and set of actions of a probabilistic I/O automaton are countable in order to utilize the fact that all measures are discrete. A probabilistic input/output automaton can be realized as a labelled transition system where:

1. Given a state $q \in Q$ and an action $a \in \mathsf{Act}$, there is a unique probability distribution $\mu_{q,a} \in \mathsf{Prob}(Q)$ which can be executed in state q "by action a", if there is such a distribution at all. This means the transition relation is a partial function $\Delta \colon Q \times \mathsf{Act} \rightharpoonup \mathsf{Prob}(Q)$ with $\Delta(q, a) = \mu_{q,a}$, when $\Delta(q, a)$ is defined.

2. For any state $q \in Q$, every $a \in I$ is enabled in q, so the restriction $\Delta|_{Q \times I} \colon Q \times I \to \mathsf{Prob}(Q)$ is a total function.

3. The obvious domain equation for PIOAs is

$$E \simeq E \times \mathsf{Act} \rightharpoonup \mathsf{Prob}(E),$$

   where $X \rightharpoonup Y$ is the set of *partial* functions from X to Y.

Since we have no final states in Q, there is a question of what a computation of a probabilistic automaton is. As we describe in more detail later, our interest is in the trace distributions of observable events of the automation $\mathcal{A}$. Toward that end, we develop the following definition.

If $a \in \mathsf{Act}$, then we define

$$A_a = \{\alpha \in \mathsf{Exec}(\mathcal{A}) \mid a \text{ enabled in } \mathsf{lstate}(\alpha)\}.$$

Then, for $\mathbf{a} = a_0 a_1 \cdots \in \mathsf{Act}^\infty$, is a sequence of actions of the PIOA $\mathcal{A}$, we define a sequence of probability distributions on $\mathsf{Exec}(\mathcal{A})$ by

$$\begin{aligned} \mu_0 &= \delta_{q_0} \\ \mu_n &= \sum_{\alpha \notin A_{a_n}} \mu_{n-1}(\alpha)\delta_\alpha + \sum_{\alpha \in A_{a_n}} \mu_{n-1}(\alpha) \left(\sum_{q \in Q} \mu_{\mathsf{lstate}(\alpha), a_n}(q) \delta_{\alpha a_n q} \right) \end{aligned}$$

where $\mu_{\mathsf{lstate}(\alpha), a_n}$ is the unique probability distribution for which $(\mathsf{lstate}(\alpha), a_n, \mu_{\mathsf{lstate}(\alpha), a_n}) \in D$.

1. $\mu_n \in \mathsf{Prob}(\mathsf{Exec}(\mathcal{A}))$ for each $n \in \mathbb{N}$.
2. $\mu_n \leq \mu_{n+1}$ for each $n \in \mathbb{N}$.
3. If $\mathbf{a} \in \mathsf{Act}^\omega$ is an infinite sequence, then $\mu_{\mathbf{a}} = \sup_n \mu_n \in \mathsf{Prob}(\mathsf{Exec}(\mathcal{A}))$ is well-defined.

Proof. For (1), we proceed by induction. Clearly $\mu_0 = \delta_{q_0} \in \mathsf{Prob}(\mathsf{Exec}(\mathcal{A}))$. Suppose $\mu_{n-1} \in \mathsf{Prob}(\mathsf{Exec}(\mathcal{A}))$, and consider μ_n. If $\alpha \in A_{a_n}$, then $\mu_{\mathsf{lstate}(\alpha),a_n} \in \mathsf{Prob}(Q)$, and so $||\mu_{n-1}(\alpha)(\sum_{q\in Q} \mu_{\mathsf{lstate}(\alpha),a_n}(q)\delta_{\alpha a_n q})|| = \mu_{n-1}(\alpha)$. It follows that

$$||\mu_n|| = ||\sum_{\alpha \notin A_{a_n}} \mu_{n-1}(\alpha) + \sum_{\alpha \in A_{a_n}} \mu_{n-1}(\alpha)|| = ||\mu_{n-1}|| = 1$$

For (2), let $n \in \mathbb{N}$. Given $\alpha \in \mathsf{supp}\,\mu_n \cap A_{a_n}$, we define $t_{\alpha,\alpha a_n q} = \mu_{n-1}(\alpha)\mu_{\mathsf{lstate}(\alpha),a_n}(q)$. Then as in (1),

- $\mu_{n-1}(\alpha) = \mu_{n-1}(\alpha)\sum_q \mu_{\mathsf{lstate}(\alpha),a_n}(q) = \sum_q t_{\alpha,\alpha a_n q}$, and
- $t_{\alpha,\alpha a_n q} = \mu_{n-1}\mu_{\mathsf{lstate}(\alpha),a_n}(q)$

from which it follows that $\mu_{n-1} \leq \mu_n$.

Finally, (3) now follows from (1) and (2). □

Let $\mathcal{A}$ be a probabilistic input/output automaton. Given a sequence $\mathbf{a} = a_0 a_1 \cdots \in \mathsf{Act}^\infty$ of actions of $\mathcal{A}$, the *computation* of $\mathcal{A}$ over $a_0 a_1 \cdots$ is the probability measure $\mu_{\mathbf{a}} = \sup\{\mu_n \mid \mu_n \text{ is defined}\}$.

If $\mathcal{A}$ is a PIOA and $\mathbf{a} = a_0 \cdots \in \mathsf{Act}^\infty$ is a sequence of actions, then $\mathsf{supp}\,\mu_{\mathbf{a}}$ consists of incomparable execution fragments.

Proof. By definition, $\mu_{\mathbf{a}} = \sup_n\{\mu_n \mid \mu_n \text{ is defined}\}$. Moreover, δ_{q_0} consists of execution fragments that are incomparable, since its support consists of just q_0. Suppose that μ_{n-1} is defined and has support consisting of incomparable fragments. Then

$$\mu_n = \sum_{\alpha \notin A_{a_n}} \mu_{n-1}(\alpha)\delta_\alpha + \sum_{\alpha \in A_{a_n}} \mu_{n-1}(\alpha)\left(\sum_{q\in Q} \mu_{\mathsf{lstate}(\alpha),a_n}(q)\delta_{\alpha a_n q}\right).$$

By assumption, all $\alpha \notin A_{a_n}$ are incomparable with one another and with all $\alpha \in A_{a_n}$, and vice versa. It follows that all $\alpha \notin A_{a_n}$ are incomparable with all fragments in $\{\alpha a_n q \mid \alpha \in A_{a_n} \;\&\; q \text{ a state}\}$. It also is clear that all fragments in $\{\alpha a_n q \mid \alpha \in A_{a_n} \;\&\; q \text{ a state}\}$ are incomparable since they

are distinct extensions of incomparable the fragments in A_{a_n}. Thus the support of μ_n consists of incomparable fragments. Thus, $\mathbf{supp}\,\mu_{\mathbf{a}}$ consists of incomparable fragments if $\mathbf{a}$ is finite.

Finally, suppose $\mathbf{a}$ is infinite. If β' is a finite fragment in the support of $\mu_{\mathbf{a}}$, then β' arises in μ_n for some n, and then no extension of β' is in the support of $\mu_{\mathbf{a}}$ by the previous argument. Hence β and β' are incomparable. Last, any two infinite fragments are incomparable, so β cannot compare to any other infinite fragment in the support. □

3.1 Tasks

A *task probabilistic Input/Output automaton* is a pair $\mathcal{T} = (\mathcal{A}, \mathcal{R})$ where

- $\mathcal{A} = (Q, q_0, I, O, H, D)$ is a probabilistic I/O automaton, and
- $\mathcal{R} \subseteq (O \times O) \cup (H \times H)$ is an equivalence relation on the locally controlled actions. The equivalence classes of $\mathcal{R}$ are called *tasks*.

A task-PIOA $\mathcal{T}$ is said to be *action deterministic* if it satisfies:

- **Action determinism:** For every state $q \in A$ and every task $T \in \mathcal{R}$, there is at most one action $a \in \mathsf{Act}$ that is enabled in q.

As opposed to [2], we will consider more general task-PIOAs than those that are action-deterministic. Because of this, we must impose an additional requirement on such processes:

- A task PIOA $\mathcal{T} = (\mathcal{A}, \mathcal{R})$ is also equipped with a function $\sigma\colon \mathcal{R} \times \mathsf{Exec}^*(\mathcal{A}) \to \mathsf{Prob}(\mathsf{Act})$ satisfying the property that

$$\sigma(T, \alpha)(a) > 0 \quad \Rightarrow \quad a \in T \;\&\; a \text{ enabled in } \mathsf{lstate}(\alpha).^1$$

A *task schedule* for a task-PIOA $\mathcal{T}$ is a finite or infinite sequence $\rho = T_1 T_2 \cdots$ of tasks in $\mathcal{R}$.

4 Task Schedules and Their Schedulers

We now define how to apply a task to a probability distribution on the execution sequences of a task-PIOA. This definition of Apply is first given

[1] Note that if $\{a \mid a \in T \;\&\; a \text{ enabled in } \mathsf{lstate}(\alpha)\}$ consists of a single action, a_0, then $\sigma(T, \alpha)(a) > 0$ implies $a = a_0$, so $\sigma(T, \alpha)(a_0) = \delta_{a_0}$, which means this definition agrees with the one in [2] in case T is action-deterministic.

for a single task. Our definition here generalizes the one in [2] in two ways: first, we consider tasks that are not action-deterministic, and second, even in the case of action -deterministic tasks, our definition is not the same one as in [2]. However, for action-deterministic tasks, they do amount to the same thing.

For a task T, let A_T be the set of finite execution fragments in whose last state the task T is enabled:

$$A_T = \{\alpha \in \mathsf{Exec}^*(\mathcal{A}) \mid (\exists a \in T)\ a \text{ enabled in } \mathsf{lstate}(\alpha)\}.$$

For each finite execution fragment $\alpha \in A_T$ and action $a \in \mathsf{Act}$ for which $\sigma(T,\alpha)(a) > 0$, let $\nu(\alpha, a)$ be the target measure of the unique transition $(\mathsf{lstate}(\alpha), a, \nu(\alpha, a)) \in D$. Then we define

$$\mathsf{Apply}(\mu, T) = \sum_{\alpha \notin A_T} \mu(\alpha)\delta_\alpha + \sum_{\alpha \in A_T} \mu(\alpha) \left[\sum_{a \in T} \sigma(T,\alpha)(a) \left(\sum_s \nu(\alpha, a)(s)\delta_{\alpha a s}\right)\right] \tag{3}$$

For a finite task schedule $\rho = T_1 \cdots T_n$, we define

$$\mathsf{Apply}(\mu, \rho) = \mathsf{Apply}(\mathsf{Apply}(\mu, T_1 \cdots T_{n-1}), T_n)$$

and if ρ is infinite, we define

$$\mathsf{Apply}(\mu, \rho) = \sup_n\ \mathsf{Apply}(\mu, T_1 \cdots T_n).$$

Of course, we must show that this last is well-defined.

Let $\mu = \sum_\alpha \mu(\alpha)\delta_\alpha \in \mathsf{Prob}(\mathsf{Frags}^*(\mathcal{A}))$, then

$$\mu \leq \mathsf{Apply}(\sum_\alpha \mu(\alpha)\delta_\alpha, \rho) = \sum_\alpha \mu(\alpha)\mathsf{Apply}(\delta_\alpha, \rho).$$

In particular, $\mathsf{Apply}(\mu, \rho)$ is well-defined for any task sequence $\rho = T_1 \cdots$.

Proof. If $\rho = \lambda$ then $\mathsf{Apply}(\rho, \mu) = \mu$, so the result is trivial, while if $\rho = T$ is a single task, then

$$\begin{aligned}
\mathsf{Apply}(\mu, T) &= \sum_{\alpha \notin A_T} \mu(\alpha)\delta_\alpha + \\
&\quad \sum_{\alpha \in A_T} \mu(\alpha) \left[\sum_{a \in T} \sigma(T,\alpha)(a) \left(\sum_s \nu(\alpha, a)(s)\delta_{\alpha a s}\right)\right] \qquad (4)\\
&= \sum_{\alpha \notin A_T} \mu_i(\alpha)\mathsf{Apply}(\delta_\alpha, T) + \sum_{\alpha \in A_T} \mu(\alpha)\mathsf{Apply}(\delta_\alpha, T)\\
&= \sum_\alpha \mu(\alpha)\mathsf{Apply}(\delta_\alpha, T).
\end{aligned}$$

The fact that $\mu \leq \mathsf{Apply}(\mu, T)$ is obvious from Equation 4.

Next, if $\rho = \rho' T$ is finite, then

$$\begin{aligned}
\mathsf{Apply}(\mu, \rho) &= \mathsf{Apply}(\mathsf{Apply}(\mu, \rho'), T) \\
&= \mathsf{Apply}(\sum_{\alpha} \mu(\alpha)\mathsf{Apply}(\delta_\alpha, \rho'), T) \\
&= \sum_{\alpha} \mu(\alpha)\mathsf{Apply}(\mathsf{Apply}(\delta_\alpha, \rho'), T)
\end{aligned}$$

Assuming by induction that $\mu \leq \mathsf{Apply}(\mu, \rho')$, then $\mu \leq \mathsf{Apply}(\mu, \rho)$ follows from the basis step.

Finally, if $\rho = T_1 \cdots$ is infinite, then we have just shown that $\{\mathsf{Apply}(\mu, T_1 \cdots T_n)\}_n$ is an increasing sequence in $\mathsf{Prob}(\mathsf{Frags}^*(\mathcal{A}))$ whose supremum, $\mathsf{Apply}(\mu_i, \rho)$ is well-defined. Clearly $\mu \leq \mathsf{Apply}(\mu, \rho)$ in this case. □

If $\mu = \sum_i p_i \mu_i$, then

$$\mu \leq \mathsf{Apply}(\sum_i p_i \mu_i, \rho) = \sum_i p_i \mathsf{Apply}(\mu_i, \rho).$$

Proof. By the Proposition, we know that $\mu_i \leq \mathsf{Apply}(\mu_i, \rho)$ for each index i, so

$$\mu = \sum_i p_i \mu_i \leq \sum_i p_i \mathsf{Apply}(\mu_i, \rho) = \mathsf{Apply}(\sum_i p_i \mu_i, \rho),$$

the last equality following by expanding each μ_i as a convex sum of point masses and then applying the Proposition once again. □

We now prove a useful result about Apply and measures μ whose support consists of incomparable fragments.

1. If μ is a measure whose support consists of incomparable fragments and ρ is a task sequence, then all fragments in the support of $\mathsf{Apply}(\mu, \rho)$ are incomparable.

2. If ρ is a task sequence, then all fragments in the support of $\mathsf{Apply}(\delta_\alpha, \rho)$ are incomparable.

Proof. For (1), suppose the support of μ consists of incomparable elements. Then $\mu = \sum_\alpha \mu(\alpha)\delta_\alpha$ and $\alpha \neq \alpha'$ implies α and α' are incomparable. If T is a task, then

$$\mathsf{Apply}(\mu, T) = \sum_{\alpha \notin A_T} \delta_\alpha + \sum_{\alpha \in A_T} \mu(\alpha) \left[\sum_{a \in T} \sigma(T, \alpha)(a) \left(\sum_s \nu(\alpha, a)(s) \delta_{\alpha a s} \right) \right],$$

where $a \in T$ is enabled in $\mathsf{lstate}(\alpha)$. Thus, the support of $\mathsf{Apply}(\mu, T)$ consists of $\alpha \notin A_T$ together with $\{\alpha a s \mid \alpha \in A_T, a \in T \;\&\; s \text{ a state}\}$. By assumption, all $\alpha \notin A_T$ are incomparable with one another and with all $\alpha \in A_T$, and vice versa. It follows that all $\alpha \notin A_T$ are incomparable with all fragments in $\{\alpha a s \mid \alpha \in A_T, a \in T \;\&\; s \text{ a state}\}$. It also is clear that all fragments in $\{\alpha a s \mid \alpha \in A_T, a \in T \;\&\; s \text{ a state}\}$ are incomparable since they are distinct extensions of incomparable the fragments in A_T.

If ρ is finite, then the result follows by induction on the length of ρ. If $\rho = T_1 \cdots$ is infinite, then $\mathsf{Apply}(\mu, \rho) = \sup_n \mathsf{Apply}(\mu, T_1 \cdots T_n)$. If β is in the support of $\mathsf{Apply}(\mu, \rho)$ and β is finite, then $\beta \in \mathsf{Apply}(\mu, T_1 \cdots T_n)$ for some n. The result for finite task schedules implies β is incomparable with all elements in the support of $\mathsf{Apply}(\mu, T_1 \cdots T_n)$, and because β is in the support of $\mathsf{Apply}(\mu, \rho)$, it follows that no subsequent task T_m, $m > n$ is enabled in $\mathsf{lstate}(\beta)$. Hence, no extension of β is in the support of $\mathsf{Apply}(\mu, \rho)$. Since no prefix of β is in the support either, β is incomparable with all other fragments in the support.

Finally, suppose β is infinite. If β' is a finite fragment in the support of $\mathsf{Apply}(\mu, \rho)$, then β' arises in $\mathsf{Apply}(\mu, T_1 \cdots T_n)$ for some n, and then no extension of β' is in the support of $\mathsf{Apply}(\mu, \rho)$ by the previous argument. Hence β and β' are incomparable. Last, any two infinite fragments are incomparable, so β cannot compare to any other infinite fragment in the support.

Part (2) follows since point masses have only one fragment in their support. □

We now consider the analog to applying task schedules to distributions, viz. realizing them via task schedulers. We first make this notion precise. **([2])** If $\mathcal{A}$ is a Task PIOA, then a *task scheduler* for $\mathcal{A}$ is a mapping $\sigma \colon \mathsf{Exec}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$ satisfying $\sigma(\alpha)(a) > 0 \;\Rightarrow\; a$ is enabled in $\mathsf{lstate}(\alpha)$, where $\mathbb{V}(\mathsf{Act})$ is the family of subprobability measures over Act. Given such

a mapping, and an $\alpha \in \mathsf{Exec}^*(\mathcal{A})$, we define $\epsilon_{\sigma,\alpha} \in \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}))$ by:

$$\epsilon_{\sigma,\alpha}(\uparrow \alpha') = \begin{cases} 0 & \text{if } \alpha' \not\leq \alpha \not\leq \alpha' \\ 1 & \text{if } \alpha' \leq \alpha \\ \epsilon_{\sigma,\alpha}(\uparrow \alpha'')\sigma(\alpha'')(a)\nu(\alpha'',a)(s) & \text{if } \alpha \leq \alpha' = \alpha'' as, \end{cases}$$

where $\nu(\alpha'', a)(s)$ is the probability of landing in state s starting from $\mathsf{lstate}(\alpha'')$ after executing action a.

Further, if $\mu \in \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}))$, then we define $\epsilon_{\sigma,\mu} = \sum_\alpha \mu(\alpha)\epsilon_{\sigma,\alpha}$.

Let $\mathcal{A}$ be a task PIOA and let $\sigma \colon \mathsf{Exec}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$ be a task scheduler. If $\alpha \in \mathsf{Exec}^*(\mathcal{A})$, we define

$$\epsilon'_{\sigma,\alpha} = (1 - ||\sigma(\alpha)||)\delta_\alpha + \sum_{a \in \mathsf{Act}} \sigma(\alpha)(a) \left(\sum_s \nu(\alpha, a)(s)\epsilon'_{\sigma,\alpha as} \right)$$

Then, $\epsilon'_{\sigma,\alpha} = \epsilon_{\sigma,\alpha}$ for all schedulers σ and $\alpha \in \mathsf{Exec}^*(\mathcal{A})$.

Proof. Before proving the result, we first define a sequence of approximants to $\epsilon'_{\sigma,\alpha}$ as follows:

$$\begin{aligned} \epsilon'_{\sigma,\alpha,0} &= \delta_\alpha \\ \epsilon'_{\sigma,\alpha,n+1} &= (1 - ||\sigma(\alpha)||)\delta_\alpha + \sum_{a \in \mathsf{Act}} \sigma(\alpha)(a) \left(\sum_s \nu(\alpha, a)(s)\epsilon'_{\sigma,\alpha as,n} \right) \end{aligned}$$

Claim 1: $||\epsilon'_{\sigma,\alpha,n}|| = 1$ and $\epsilon'_{\sigma,\alpha,n} \leq \epsilon'_{\sigma,\alpha,n+1}$ for each $\alpha \in \mathsf{Exec}^*(\mathcal{A})$ and each $n \in \mathbb{N}$.

Proof: We proceed by induction on n. For $n = 0$ let $\alpha \in \mathsf{Frags}^*(\mathcal{A})$. The clearly $||\epsilon'_{\sigma,\alpha,0}|| = 1$ Also we have $\epsilon'_{\sigma,\alpha,0} = \delta_\alpha$ and

$$\begin{aligned} \epsilon'_{\sigma,\alpha,1} &= (1 - ||\sigma(\alpha)||)\delta_\alpha + \sum_{a \in \mathsf{Act}} \sigma(\alpha)(a) \left(\sum_s \nu(\alpha, a)(s)\epsilon'_{\sigma,\alpha as,0} \right) \\ &= (1 - ||\sigma(\alpha)||)\delta_\alpha + \sum_{a \in \mathsf{Act}} \sigma(\alpha)(a) \left(\sum_s \nu(\alpha, a)(s)\delta_{\alpha as} \right) \end{aligned}$$

But, $\nu(\alpha, a)$ is a probability distribution, so $||\sum_s \nu(\alpha, a)(s)\delta_{\alpha as}|| = 1$ for each $a \in \mathsf{Act}$. It follows that $||\epsilon'_{\sigma,\alpha,1}|| = 1$. Moreover, we can define "transport numbers" $t_{0,1} = (1 - ||\sigma(\alpha)||)$ and $t_{0,1}(a, s) = \sigma(\alpha)(a)\nu(\alpha, a)(s)$ so that $1 = t_{0,1}(\alpha) + \sum_{a,s} t_{0,1}(a, s)$ and

$$\epsilon'_{\sigma,\alpha,1} = t_{0,1}(\alpha)\delta_\alpha + \sum_{a,s} t_{0,1}(a, s)\delta_{\alpha as}$$

This shows $\epsilon'_{\sigma,\alpha,0} \leq \epsilon'_{\sigma,\alpha,1}$.

Now, assume that, for all $\alpha \in \mathsf{Exec}^*(\mathcal{A})$ and all $k \leq n$, $||\epsilon'_{\sigma,\alpha,k}|| = 1$ and $\epsilon'_{\sigma,\alpha,k-1} \leq \epsilon'_{\sigma,\alpha,k}$. Then

$$\epsilon'_{\sigma,\alpha,n} = (1 - ||\sigma(\alpha)||)\delta_\alpha + \sum_{a \in \mathsf{Act}} \sigma(\alpha)(a) \left(\sum_s \nu(\alpha, a)(s) \epsilon'_{\sigma,\alpha a s,n-1} \right)$$

and

$$\epsilon'_{\sigma,\alpha,n+1} = (1 - ||\sigma(\alpha)||)\delta_\alpha + \sum_{a \in \mathsf{Act}} \sigma(\alpha)(a) \left(\sum_s \nu(\alpha, a)(s) \epsilon'_{\sigma,\alpha a s,n} \right)$$

By the inductive hypothesis, for each $a \in \mathsf{Act}$ and each state s, we have

$$||\epsilon'_{\sigma,\alpha a s,n-1}|| = ||\epsilon'_{\sigma,\alpha a s,n}|| = 1 \quad \text{and} \quad \epsilon'_{\sigma,\alpha a s,n-1} \leq \epsilon'_{\sigma,\alpha a s,n}.$$

It then is clear that $||\epsilon'_{\sigma,\alpha a s,n+1}|| = 1$, and the fact that $\epsilon'_{\sigma,\alpha a s,n-1} \leq \epsilon'_{\sigma,\alpha a s,n}$ means there are transport numbers demonstrating this fact; these can then be augmented by numbers analogous to those given in the base case to show $\epsilon'_{\sigma,\alpha a s,n} \leq \epsilon'_{\sigma,\alpha a s,n+1}$. □(Claim 1)

Since $\{\epsilon'_{\sigma,\alpha a s,n}\}_n \subseteq \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}))$ is an increasing chain, it has a supremum $\epsilon''_{\sigma,\alpha} = \sup_n \epsilon'_{\sigma,\alpha a s,n}$.

Claim 2: $\epsilon''_{\sigma,\alpha} = \epsilon'_{\sigma,\alpha}$

Proof: We show $\epsilon''_{\sigma,\alpha}(\alpha') = \epsilon'_{\sigma,\alpha}(\alpha')$ for all α'. First, if $\alpha' \not\leq \alpha \not\leq \alpha'$, it is clear that both are 0, so they are equal.

Next, for $\alpha' < \alpha$, $\epsilon''_{\sigma,\alpha}(\alpha') = 0$ since $\epsilon'_{\sigma\alpha,n}(\alpha') = 0$ for all n. Likewise, since $\epsilon'_{\sigma,\alpha}$ is concentrated on $\uparrow \alpha$, we have $\epsilon'_{\sigma,\alpha}(\alpha') = 0$.

Further, $\epsilon''_{\sigma,\alpha}(\alpha) = (1 - ||\sigma(\alpha)||)$ since this holds for $\epsilon'_{\sigma,\alpha,n}$ for each $n \geq 1$. Likewise, $\epsilon'_{\sigma,\alpha}(\alpha) = (1 - ||\sigma(\alpha)||)$. Since α is arbitrary, we can use this result to prove our claim. Namely, if $\alpha \leq \alpha'$ then there are $a_1, \ldots, a_n \in \mathsf{Act}$ and states $s_1, \ldots, s_n$ so that $\alpha' = \alpha a_1 s_1 \ldots a_n s_n$. Then

$$\epsilon''_{\sigma,\alpha}(\alpha a_1 s_1 \ldots a_n s_n) = \sup_m \epsilon'_{\sigma,\alpha,m}(\alpha a_1 s_1 \ldots a_n s_n).$$

For $m \geq n + 1$,

$$\begin{aligned} \epsilon'_{\sigma,\alpha,m}(\alpha a_1 s_1 \ldots a_n s_n) &= (1 - ||\sigma(\alpha a_1 s_1 \ldots a_n s_n||) \\ &= \epsilon'_{\sigma,\alpha a_1 s_1 \ldots a_n s_n}(\alpha a_1 s_1 \ldots a_n s_n), \end{aligned}$$

the second equality following from the first step. An easy induction on the definitions of $\epsilon'_{\sigma,\alpha}$ shows that

$$\begin{aligned}\epsilon'_{\sigma,\alpha}(\alpha a_1 s_1 \dots a_n s_n) &= \sigma(\alpha)(a_1)\nu(\alpha,a_1)(s_1)\cdots\sigma(\alpha\dots a_{n-1}s_{n-1})(a_n)\\ &\quad\cdot\nu(\alpha\dots a_{n-1}s_{n-1},a_n)(s_n)\epsilon'_{\sigma,\alpha a_1 s_1\dots a_n s_n}(\alpha a_1 s_1\dots a_n s_n)\\ = \quad &\sigma(\alpha)(a_1)\nu(\alpha,a_1)(s_1)\cdots\nu(\alpha\dots a_{n-1}s_{n-1},a_n)(s_n)(1-||\sigma(\alpha\dots a_n s_n)||),\end{aligned}$$

and that an analogous result holds for $\epsilon''_{\sigma,\alpha,m}$ if $m \geq n+1$. It follows that

$$\epsilon'_{\sigma,\alpha}(\alpha a_1 s_1 \dots a_n s_n) = \epsilon''_{\sigma,\alpha}(\alpha a_1 s_1 \dots a_n s_n)$$

Since $\epsilon''_{\sigma,\alpha}$ and $\epsilon_{\sigma,\alpha}$ agree at all fragments $\alpha' \geq \alpha$, they are the same, since both are concentrated on $\uparrow\alpha$. Moreover, since $\epsilon'_{\sigma,\alpha,m}$ is a probability measure for all m, the same is true of $\epsilon''_{\sigma,\alpha}$, and hence also of $\epsilon'_{\sigma,\alpha}$. □(Claim 2)

We are now ready to prove the proposition. We show $\epsilon'_{\sigma,\alpha}(\uparrow\alpha') = \epsilon_{\sigma,\alpha}(\uparrow\alpha')$ for all σ, α and α'. First, it is clear that $\epsilon'_{\sigma,\alpha}(\uparrow\alpha') = 0$ if $\alpha' \not\leq \alpha \not\leq \alpha'$. Next, assume that $\alpha' \leq \alpha$. The $\epsilon'_{\sigma,\alpha}(\uparrow\alpha') = 1$ since $\epsilon'_{\sigma,\alpha}$ is a probability measure concentrated on $\uparrow\alpha$. Thus the value of $\epsilon'_{\sigma,\alpha}$ agrees with that of $\epsilon_{\sigma,\alpha}$ for those $\alpha' \leq \alpha$ and those α' incomparable with α.

Finally, we consider α' with $\alpha \leq \alpha'$. If $\alpha = \alpha'$, we already have the result, so we can assume $\alpha < \alpha'$, in which case there are $a_1, \dots, a_n \in \mathsf{Act}$ and states $s_1, \dots, s_n$ with $\alpha' = \alpha a_1 s_1 \dots a_n s_n$.We induct on n, and assume the result holds for all $k < n$. Moreover, since $\epsilon_{\sigma,\alpha'}$ is a probability measure, $\epsilon'_{\sigma,\alpha'}(\uparrow\alpha') = 1$. So, if $\alpha'' = \alpha a_1 s_1 \dots a_{n-1} s_{n-1}$, then

$$\begin{aligned}\epsilon'_{\sigma,\alpha}(\uparrow\alpha') &= \epsilon_{\sigma,\alpha}(\uparrow\alpha'')\sigma(\alpha'')\nu(\alpha'',a_n)(s_n)\\ &= \epsilon'_{\sigma,\alpha}(\uparrow\alpha'')\sigma(\alpha'')\nu(\alpha'',a_n)(s_n)\\ &= \epsilon'_{\sigma,\alpha}(\uparrow\alpha'')\sigma(\alpha'')\nu(\alpha'',a_n)(s_n)\epsilon'_{\sigma,\alpha'}(\uparrow\alpha')\\ &= \epsilon'_{\sigma,\alpha}(\uparrow\alpha'),\end{aligned}$$

the next-to-last equality following from the fact that $\epsilon'_{\sigma,\alpha'}(\uparrow\alpha') = 1$. □

Finally, we are now ready to turn our attention to the main result of this section: that task schedules can be realized by "oblivious" schedulers. Our result achieves this in a limited case, but one that suffices for the application of Task-PIOAs to modeling crypto-protocols.

Let $\mu \in \mathsf{Prob}(\mathsf{Exec}(\mathcal{A}))$ have support consisting of incomparable fragments, and let ρ be a task schedule. Then there is a scheduler $\sigma\colon \mathsf{Exec}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$ such that $\sigma(\alpha) \neq 0$ implies $\alpha \in \mathsf{supp}\,\mu$ and $\mathsf{Apply}(\mu,\rho) = \epsilon_{\sigma,\mu}$. Moreover, if $\mathcal{T} = (\mathcal{A},\mathcal{R})$ is action-deterministic, then σ is deterministic.

Proof. First, for ρ finite we proceed by induction on the length of ρ. If $\rho = T$ is a single task, then we define $\sigma_\rho \colon \mathsf{Exec}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$ by

$$\sigma_\rho(\alpha) = \begin{cases} \sigma(T,\alpha) & \text{if } \alpha \in A_T \cap \mathsf{supp}\,\mu, \\ 0 & \text{otherwise.} \end{cases}$$

Then,

$$\begin{aligned} \epsilon_{\sigma_\rho,\alpha} &= \begin{cases} (1 - ||\sigma_\rho(\alpha)||)\delta_\alpha + \sum_{a\in\mathsf{Act}} \sigma_\rho(\alpha)(a) \left(\sum_s \nu(\alpha,a)(s)\epsilon_{\sigma_\rho,\alpha as}\right) & \\ \qquad\qquad \text{if } \alpha \in A_T \cap \mathsf{supp}\,\mu, \\ \delta_\alpha \qquad\qquad \text{otherwise.} \end{cases} \\ &= \begin{cases} \sum_a \sigma(T,\alpha)(a) \left(\sum_s \nu(\alpha,a)(s)\delta_{\alpha as}\right) & \text{if } \alpha \in A_T \cap \mathsf{supp}\,\mu, \\ \delta_\alpha & \text{otherwise,} \end{cases} \end{aligned}$$

the second equality following from the definition of σ_ρ and the fact that $\mathsf{supp}\,\mu$ consists of incomparable fragments.

Hence

$$\begin{aligned} \epsilon_{\sigma_\rho,\mu} &= \sum_\alpha \mu(\alpha)\epsilon_{\sigma_\rho,\alpha} \\ &= \sum_{\alpha\in A_T\cap\mathsf{supp},\mu} \mu(\alpha) \left(\sum_a \sigma(T,\alpha)(a) \left(\sum_s \nu(\alpha,a)(s)\delta_{\alpha as}\right)\right) \\ &\quad + \sum_{\alpha\notin(A_T\cap\mathsf{supp}\,\mu)} \mu(\alpha)\delta_\alpha \\ &= \mathsf{Apply}(\mu,T), \end{aligned}$$

the last equality following from the fact that the only terms α for which $\mu(\alpha) \neq 0$ are those in the support of μ.

Suppose the result holds for any finite task sequence of length n, and let $\rho = T\rho'$ be a task sequence of length $n+1$. Then there is a scheduler $\sigma_{\rho'} \colon \mathsf{Exec}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$ satisfying $\sigma_{\rho'}(\alpha) \neq 0$ implies $\alpha \in \mathsf{supp}\,\mathsf{Apply}(\mu,T)$ and $\epsilon_{\sigma_{\rho'},\mathsf{Apply}(\mu,T)} = \mathsf{Apply}(\mathsf{Apply}(\mu,T),\rho')$. Note that $\mathsf{supp}\,\mathsf{Apply}(\mu,T)$ consists of incomparable fragments by Lemma 1, and this support includes those fragments in $\mathsf{supp}\,\mu$ that are not enabled under T.

We define $\sigma_\rho \colon \mathsf{Exec}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$ by

$$\sigma_\rho(\alpha) = \begin{cases} \sigma_{\rho'}(\alpha) & \text{if } \sigma_{\rho'}(\alpha) \neq 0, \\ \sigma(T,\alpha) & \text{if } \alpha \in A_T \cap \mathsf{supp}\,\mu, \\ 0 & \text{otherwise.} \end{cases}$$

$\sigma_\rho = \sigma_T \cup \sigma_{\rho'}$ (where σ_T is as defined in the base step) is well-defined because μ has incomparable fragments for its support, and $\sigma_{\rho'}(\alpha) \neq 0$ implies $\alpha \in \mathsf{supp}\,\mathsf{Apply}(\mu, T)$; further, $\sigma_\rho(\alpha) \leq 1$ since the same is true for $\sigma_{\rho'}(\alpha)$. Now

$$\begin{aligned}
\epsilon_{\sigma_\rho,\mu} &= \sum_\alpha \mu(\alpha)\epsilon_{\sigma_\rho,\alpha} = \sum_{\alpha \in \mathsf{supp}\,\mu} \mu(\alpha)\epsilon_{\sigma_\rho,\alpha} \\
&= \sum_{\alpha \in A_T \cap \mathsf{supp}\,\mu} \mu(\alpha)\epsilon_{\sigma_\rho,\alpha} + \sum_{\alpha \in \mathsf{supp}\,\mu - A_T} \mu(\alpha)\epsilon_{\sigma_\rho,\alpha} \\
&= \sum_{\alpha \in A_T \cap \mathsf{supp}\,\mu} \mu(\alpha) \left(\sum_a \sigma(T,\alpha)(a) \left(\sum_s \nu(\alpha,a)\epsilon_{\sigma_\rho,\alpha a s} \right) \right) \\
&\quad + \sum_{\alpha \in \mathsf{supp}\,\mu - A_T} \mu(\alpha)\epsilon_{\sigma_\rho,\alpha} \\
&= \sum_{\alpha \in A_T \cap \mathsf{supp}\,\mu} \mu(\alpha) \left(\sum_a \sigma(T,\alpha)(a) \left(\sum_s \nu(\alpha,a)\epsilon_{\sigma_{\rho'},\alpha a s} \right) \right) \\
&\quad + \sum_{\alpha \in \mathsf{supp}\,\mu - A_T} \mu(\alpha)\epsilon_{\sigma_{\rho'},\alpha} \\
&= \epsilon_{\sigma_{\rho'},\mathsf{Apply}(\mu,T)} = \mathsf{Apply}(\mathsf{Apply}(\mu,T),\rho') = \mathsf{Apply}(\mu,\rho).
\end{aligned}$$

If $\rho = T_1 \cdots T_n \cdots$ is infinite, then the function σ_ρ can be expressed as the increasing union $\sigma_\rho = \cup_n \sigma_{\rho_n}$, where σ_{ρ_n} is the scheduler for $\rho_n = T_1 \cdots T_n$. Likewise, the family $\epsilon_{\sigma_{\rho_n},\mu}$ forms an increasing sequence satisfying $\epsilon_{\sigma_{\rho_n},\mu} = \mathsf{Apply}(\mu, \rho_n)$ for each n, so that

$$\epsilon_{\sigma_\rho,\mu} = \sup_n \epsilon_{\sigma_{\rho_n},\mu} = \sup_n \mathsf{Apply}(\mu,\rho_n) = \mathsf{Apply}(\mu,\rho).$$

The claim that σ is deterministic if $\mathcal{T}$ is action-deterministic follows from the fact that $\sigma(T,\alpha) = \delta_\alpha$ in this case. □

If $\mathcal{A} = (S, s_0, \mathsf{Act}, D)$ is a task PIOA, ρ is a task sequence for $\mathcal{A}$ and $\epsilon = \mathsf{Apply}(\delta_{s_0}, \rho)$, then there exists a scheduler σ such that $\epsilon = \epsilon_{\sigma,\delta_{s_0}}$. That scheduler is deterministic if $\mathcal{T}$ is action-deterministic.

Remark.

- As the Corollary states, given a task PIOA and a task sequence for $\mathcal{A}$, then there is a task scheduler that realizes the application of the task sequence to $\mathcal{A}$ starting at its start state. This is the basic situation in which crypto-protocols are modeled using task PIOAs.

- We do not see how to extend the proof given above to the case of an arbitrary distribution μ on $\mathsf{Prob}(\mathsf{Frags}^*(\mathcal{A}))$. The problem is the assumption that μ has incomparable fragments in its support. Its use is most apparent in the inductive step of the proof where we define σ_ρ in terms of $\sigma_{\rho'}$ – our definition would not make sense if $\mathsf{dom}\,\sigma_{\rho'} \cap \mathsf{supp}\,\mu \neq \emptyset$. It's at this point that randomized schedulers enter the proof (a close examination of the proof we have shows the schedulers we use can be regarded as functions with inputs that are the distributions $\sigma(T, \alpha)$ for the input PIOA $\mathcal{T}$), and finding a way to deal with this in our approach would be very useful.

We close this section with a result that anticipates the need for reasoning about randomized schedulers, as in the description of how we plan to attack the dining cryptographers example. First, recall that $\mathbb{V}(\mathsf{Act})$, the family of subprobability measures over Act is closed under convex sums – i. e., if $\mu_1, \mu_2 \in \mathbb{V}(\mathsf{Act})$ and $0 \leq r \leq 1$, then $r\mu_1 + (1-r)\mu_2 \in \mathbb{V}(\mathsf{Act})$. It follows that given $\sigma_1, \sigma_2 \colon \mathsf{Frags}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$, the function $r\sigma_1 + (1-r)\sigma_2$ defined point wise is also well-defined.

Let $\sigma_1, \sigma_2 \colon \mathsf{Frags}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$, $\alpha \in \mathsf{Frags}^*(\mathcal{A})$ and $0 \leq r \leq 1$. Then

$$||(r\sigma_1 + (1-r)\sigma_2)(\alpha)|| = r||\sigma_1(\alpha)|| + (1-r)||\sigma_2(\alpha)||.$$

Hence the family of schedulers is closed under convex sums.

Proof. Let $\sigma_1, \sigma_2 \colon \mathsf{Frags}^*(\mathcal{A}) \to \mathbb{V}(\mathsf{Act})$, $\alpha \in \mathsf{Frags}^*(\mathcal{A})$ and let $0 \leq r \leq 1$. Then $\sigma_i(\alpha) = \sum_{a \in \mathsf{Act}} \sigma_i(\alpha)(a)\delta_a$, so $||\sigma_i(\alpha)|| = \sum_s \sigma_i(\alpha)(s)$. Hence $(r\sigma_1 + (1-r)\sigma_2)(\alpha) = r\sigma_1(\alpha) + (1-r)\sigma_2(\alpha)$. Then

$$\begin{aligned} ||r\sigma_1(\alpha) + (1-r)\sigma_2(\alpha)|| &= \sum_s (r\sigma_1(\alpha) + (1-r)\sigma_2(\alpha))(s) \\ &= \sum_s r\sigma_1(\alpha)(s) + \sum_s (1-r)\sigma_2(\alpha)(s) \\ &= r||\sigma_1(\alpha)|| + (1-r)||\sigma_2(\alpha)|| \end{aligned}$$

The concluding remark is obvious from this result. □

5 Simulation Relations

A key notion for task PIOAs is that of a *simulation relation*, which are used to show that the observable behaviors of one task PIOA are a subset of another tack PIOA. This is used in [2] to simplify the process one has

to reason about, and then deductions about the behavior of the simpler process can be used to prove properties about the more complicated one. In our setting, we'll exploit this idea to show that the Dining Cryptographers Protocol fulfills what is claimed for it: if a given cryptographer is not paying, then the behaviors of the other two cryptographers looks the same.

To begin, we have to generalize the definitions from [2] to accommodate the more general setting we are in where task PIOAs are not necessarily action-deterministic. This actually turns out to be straightforward: the definitions don't need to be altered, since they make no specific reference to action determinism. We recall the key concepts, however:

- If $\mathcal{T}_1 = (\mathcal{A}, \mathcal{R})$ is a task PIOA, and ρ if a task sequence for $\mathcal{T}$, then a probability measure $\mu \in \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}))$ is *consistent with* ρ if $\mathsf{supp}\,\mu \subseteq \mathsf{supp}\,\mathsf{Apply}(\delta_{s_0}, \rho)$. I. e., any execution sequence in the support of μ is a possibly execution sequence obtained by applying the task sequence ρ starting in the start state s_0.

- Let $\mathcal{T}_1 = (\mathcal{A}_1, \mathcal{R}_1)$ and $\mathcal{T}_2 = (\mathcal{A}_2, \mathcal{R}_2)$ be task PIOAs, and let $c\colon \mathcal{R}_1^* \times \mathcal{R}_1 \to \mathcal{R}_2^*$ be a function. We define $\mathsf{full}(c)\colon \mathcal{R}_1^* \to \mathcal{R}_2^*$ by $\mathsf{full}(c)(\lambda) = \lambda$, and $\mathsf{full}(c)(\rho T) = \mathsf{full}(c)(\rho)^\frown c(\rho, T)$.

- If $\mathcal{T} = (\mathcal{A}, \mathcal{R})$ is a task PIOA, then $\mathsf{trace}\colon \mathsf{Exec}^*(\mathcal{A}) \to (I \cup O)^*$ is the function that extracts the sequence of observable actions (i. e., the input or output actions) from an execution sequence of $\mathcal{A}$. Since we are using only discrete measures, all sets are measurable, and so trace is a measurable map. Then $\mu \circ \mathsf{trace}^{-1}$ is a measure on $(I \cup O)^*$, so we define $\mathsf{tdist}(\mu)$ to be this measure. We also call $\mathsf{tdists}(\mathcal{A}) = \{\mu \circ \mathsf{trace}^{-1} \mid \mu \in \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}))\}$ the *trace distributions* of $\mathcal{A}$.

- The task PIOAs $\mathcal{T}_1$ and $\mathcal{T}_2$ are *comparable* if they have the same set of input actions and the same set of output actions. They are *closed* if they both have empty input action sets.

  A relation $R \subseteq \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}_1)) \times \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}_2))$ is a *simulation* if:

  1. $\mu_1\ R\ \mu_2 \ \Rightarrow\ \mathsf{tdist}(\mu_1) = \mathsf{tdist}(\mu_2)$;
  2. $\delta_{s_0^1}\ R\ \delta_{s_0^2}$ – i. e., the Dirac measures of the start states are related; and
  3. There is a function $c\colon \mathcal{R}_1^* \times \mathcal{R}_1 \to \mathcal{R}_2^*$ satisfying:
     If $\mu_1\ R\ \mu_2$ and $\rho \in \mathcal{R}_1^*$ satisfy μ_1 is consistent with ρ, μ_2 is consistent with $\mathsf{full}(c)(\rho)$ and $T \in \mathcal{R}_1$, then $\mathsf{Apply}(\mu_1, T)\ \mathcal{E}(R)\ \mathsf{Apply}(\mu_2, c(\rho, T))$.

To understand the last condition, we need to recall some notions about probability distributions. If X is a set, then $\mathsf{Prob}(X)$ denotes the set of discrete probability measures on X. Thus, we may form $\mathsf{Prob}(\mathsf{Prob}(X))$, the discrete probability measures of discrete probability measures on X.

If $R \subseteq \mathsf{Prob}(X) \times \mathsf{Prob}(Y)$, then we can "lift" R to $\mathcal{L}(R) \subseteq \mathsf{Prob}(\mathsf{Prob}(X)) \times \mathsf{Prob}(\mathsf{Prob}(Y))$ by $\sum_i r_i \delta_{\mu_i}\ \mathcal{L}(R)\ \sum_i s_i \delta_{\nu_i}$ iff there is a *weighting function* $w\colon \mathbb{N} \times \mathbb{N} \to [0,1]$ so that

- For each i, $\sum_j w(i,j) = r_i$;
- For each j, $\sum_i w(i,j) = s_j$;
- $w(i,j) > 0 \ \Rightarrow\ (\mu_i, \nu_j) \in R$.

There is a mapping $\mathsf{flatten}\colon \mathsf{Prob}(\mathsf{Prob}(X)) \to \mathsf{Prob}(X)$ defined by $\mathsf{flatten}(\sum_i r_i \delta_{\mu_i}) = \sum_{i,j} r_i s_{i,j} \delta_{x_{i,j}}$, where $\mu_i = \sum_j s_{i,j} \delta_{x_{i,j}}$. It is routine to show that this is well-defined.

Given a relation $R \subseteq \mathsf{Prob}(X) \times \mathsf{Prob}(Y)$, we can define a relation $\mathcal{E}(R) \subseteq \mathsf{Prob}(X) \times \mathsf{Prob}(Y)$ by $(\mu_1, \mu_2) \in \mathcal{E}(R)$ iff there are measures $\nu_1 \in \mathsf{Prob}(\mathsf{Prob}(X))$ and $\nu_2 \in \mathsf{Prob}(\mathsf{Prob}(Y))$ satisfying $\nu_1\ \mathcal{E}(R)\ \nu_2$ and $\mathsf{flatten}(\nu_i) = \mu_i$.

Here are some key results from [2] concerning these concepts:

Lemma 2.6: If $R \subseteq \mathsf{Prob}(X) \times \mathsf{Prob}(Y)$, then $\mu_1\ \mathcal{E}(R)\ \mu_2$ iff there is a family $p_i \in [0,1]$ with $\sum_i p_i = 1$ and there are measures $\mu_{1,j} \in \mathsf{Prob}(X)$, $\mu_{2,j} \in \mathsf{Prob}(Y)$ for $j \in \mathbb{N}$ satisfying:

- $\mu_1 = \sum_j p_j \mu_{1,j}$;
- $\mu_2 = \sum_j p_j \mu_{2,j}$;
- $\mu_{1,i}\ R\ \mu_{2,i}$ for each $i \in \mathbb{N}$.

Lemma 4.4: Let $\mathcal{T}_1, \mathcal{T}_2$ be two comparable, closed action-deterministic PIOAs and let $R \subseteq \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}_1)) \times \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}_2))$ satisfy $\mu\ R\ \nu \ \Rightarrow\ \mathsf{tdist}(\mu) = \mathsf{tdist}(\nu)$. Let $c\colon \mathcal{R}_1^* \times \mathcal{R}_1 \to \mathcal{R}_2^*$ and assume:

- $\delta_{s_0^1}\ R\ \delta_{s_0^2}$;
- If $\mu_1\ R\ \mu_2$, $\rho \in \mathcal{R}_1^*$ with μ_1 consistent with ρ and μ_2 consistent with $\mathsf{full}(c)(\rho)$ and $T \in \mathcal{R}_1$, then there is a family $\{p_i\}_i \subseteq [0,1]$ with $\sum_i p_i$ and measures $\mu_{i,j} \in \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}_i))$ for $i = 1, 2$ satisfying
  - $\mu_{1,j}\ R\ \mu_{2,j}$ for each $j \in \mathbb{N}$;

- $\sum_j p_j \mu_{1,j} = \mathsf{Apply}(\mu_1, T)$;
- $\sum_j p_j \mu_{2,j} = \mathsf{Apply}(\mu_2, c(\rho, T))$.

Then R is a simulation from $\mathcal{T}_1$ to $\mathcal{T}_2$.

Lemma 4.5: If $\mathcal{T}_1$ and $\mathcal{T}_2$ are comparable closed action-deterministic PIOAs, and R is a simulation relation from $\mathcal{T}_1$ to $\mathcal{T}_2$, then given $\mu_i \in \mathsf{Prob}(\mathsf{Exec}^*(\mathcal{A}_i))$, $i = 1, 2$ for which $\mu_1 \; \mathcal{E}(R) \; \mu_2$, then $\mathsf{tdist}(\mu_1) = \mathsf{tdist}(\mu_2)$.

Lemma 4.6: Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be comparable closed action-deterministic PIOAs, and R is a simulation relation from $\mathcal{T}_1$ to $\mathcal{T}_2$ with associated function c: $\mathcal{R}_1^* \times \mathcal{R}_1 \rightarrow \mathcal{R}_2^*$. Let $\rho \in \mathcal{R}_1^*$ and suppose $\mathsf{Apply}(\delta_{s_0^1}, \rho) \; \mathcal{E}(R) \; \mathsf{Apply}(\delta_{s_0^2}, \mathsf{full}(c)(\rho))$. Then $\mathsf{Apply}(\delta_{s_0^1}, \rho T) \; \mathcal{E}(R) \; \mathsf{Apply}(\delta_{s_0^2}, \mathsf{full}(c)(\rho)c(\rho, T))$ for each task $T \in \mathcal{R}_1$.

Theorem 4.7: If $\mathcal{T}_1$ and $\mathcal{T}_2$ are comparable closed action-deterministic task PIOAs for which there is a simulation relation R from $\mathcal{T}_1$ to $\mathcal{T}_2$, then $\mathsf{tdist}(\mathcal{T}_1) \subseteq \mathsf{tdist}(\mathcal{T}_2)$.

Note: I have read through the proofs of these results, and there appears to be no problems extending them to our case.

Roberto's approach to applying these results to the Dining Cryptographers was to posit the existence of simulations between $\mathcal{M}_1$ and $\mathcal{M}_2$ – back and forth – which would show that the trace distributions from cryptographer 0's viewpoint were the same for both, implying cryptographer 0 couldn't tell which of the other two were paying. We need a similar idea to show an analogous result for our more general setting of a nondeterministic master.

6 Domain Equations

We have discussed a paper of Mike's where a domain equation of the form $D \simeq Pow(\coprod_a D)$ is solved. Yet, the equation does not account for the fact that at any state there is a collection of transitions available. A scheduler chooses a measure over transition which, by integration, leads indeed to a measure over pairs action,state. A more natural equation seems to be $D \simeq Pow(\coprod_a \mathbb{V}(D))$.

Is it possible to solve this equation? What do we get in terms of logical and equational characterizations? Can we get the characterizations directly from the power-domain operators?

The two domain equations should be closely related, though. Assuming that the transition relation is compact, from the set of possible measures on action,state pairs we should be able to retrieve the set of enabled transitions. What is a nice statement based on domain theory that captures this property? Can we define mappings from one to the other whose composition is the identity up to bisimulation?

We have observed, though, that we do not distinguish systems whose transitions are not closed from systems that are completed with the extremal points. Is there a different domain equation that permits to distinguish? And if we keep the current equations, what are the operational limitations that we impose on non-finitary systems so that full abstraction holds?

References

[1] Abramsky, S. and A. Jung, Domain theory, in: S. Abramsky, D. M. Gabbay, and T. S. E. Maibaum, editors, Handbook of Logic in Computer Science, volume 3, pages 1–168. Clarendon Press, 1994.

[2] Canetti, R., N. Lynch, et al, Using Probabilistic I/O Automata to Analyze an Oblivious Transfer Protocol, preprint MIT-LCS-TR-1001.

[3] Chatzikokalakis, K. and C. Palamidessi, Making Random Choices Invisible to the Scheduler, Lecture Notes in Computer Science **4703**, Springer-Verlag, 2007, pp. 42–58.

[4] Gierz, G., K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove and D. Scott, Continuous Lattices and Domains, Encyclopedia of Mathematics and its Applications **93**, Cambridge University Press, 2003.

[5] Jones, C., Probabilistic non-determinism, University of Edinburgh, Edinburgh, Scotland, 1992

[6] Martin, K., A Foundation for Computation, Tulane University PhD Thesis, New Orleans, LA, 2000.

[7] Martin, K., M. Mislove, and J. Worrell. Measuring the probabilistic powerdomain. Theoretical Computer Science **312** (2004), pp. 99–119.

[8] Segala, R. and N. Lynch, Probabilistic simulations for probabilistic processes, Nordic Journal of Computing **2** (1995), 250–273.