A Symbolic Treatment of Randomization

Ling Cheung

Nijmegen Institute for Computing and Information Sciences
([A-Z][a-z]+)? University of Nijmegen$, the Netherlands

VOSS, January 2005, Saarbrücken, Germany

Outline

Outline

Outline

Outline

Acknowledgment

This work is done in the context of two projects.

- Compositionality of trace distribution semantics: Ling Cheung, Nancy Lynch, Roberto Segala, Frits Vaandrager.
- Modeling cryptographic protocols in PIOA: Ran Canetti, Ling Cheung, Dilsun Kaynar, Moses Liskov, Nancy Lynch, Olivier Pereira, Roberto Segala, Frits Vaandrager.

I thank everyone involved for the very helpful discussions.

Trace Distribution Semantics: Recent Developments

From the last talk:

- semantic compositionality fails in Segala's Simple Probabilistic Automata Framework;

Trace Distribution Semantics: Recent Developments

From the last talk:

- semantic compositionality fails in Segala's Simple Probabilistic Automata Framework;
- distributed scheduling saves the day: *Switched PIOA*.

Trace Distribution Semantics: Recent Developments

From the last talk:

- semantic compositionality fails in Segala's Simple Probabilistic Automata Framework;
- distributed scheduling saves the day: *Switched PIOA*.

This talk: *Symbolic PA*, a new system type designed to weaken the adversary model of Segala's Simple PA.

Trace Distribution Semantics: Recent Developments

From the last talk:

- semantic compositionality fails in Segala's Simple Probabilistic Automata Framework;
- distributed scheduling saves the day: *Switched PIOA*.

This talk: *Symbolic PA*, a new system type designed to weaken the adversary model of Segala's Simple PA.

Disclaimer: This is ongoing work.

The Role of An Adversary

In a probabilistic setting, we wish to talk about probability measures over the collection of all system executions.

The Role of An Adversary

In a probabilistic setting, we wish to talk about probability measures over the collection of all system executions.

For this to make sense, all non-deterministic choices in the system must be resolved.

The Role of An Adversary

In a probabilistic setting, we wish to talk about probability measures over the collection of all system executions.

For this to make sense, all non-deterministic choices in the system must be resolved.

Typically, we do so by specifying an adversary: a function from finite histories to available next transitions.

A Few Remarks

- The notion of adversaries is purely conceptual.

A Few Remarks

- The notion of adversaries is purely conceptual.
- Not all adversaries are created equal: one may vary the power/knowledge of an adversary to obtain different adversary models.

A Few Remarks

- The notion of adversaries is purely conceptual.
- Not all adversaries are created equal: one may vary the power/knowledge of an adversary to obtain different adversary models.

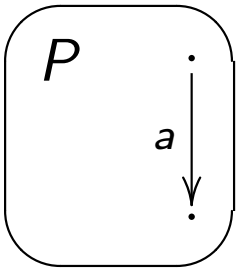
  Example: Segala's Simple PA has a strong adversary model, in that the adversaries have complete knowledge over execution history of all components, including their internal states.

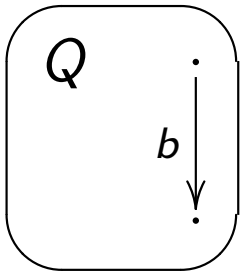
A Few Remarks

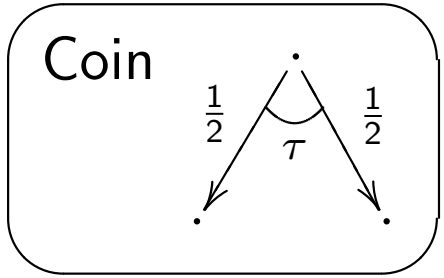
- The notion of adversaries is purely conceptual.
- Not all adversaries are created equal: one may vary the power/knowledge of an adversary to obtain different adversary models.

  Example: Segala's Simple PA has a strong adversary model, in that the adversaries have complete knowledge over execution history of all components, including their internal states.
- Different adversary models give rise to different notions of probabilistic executions.

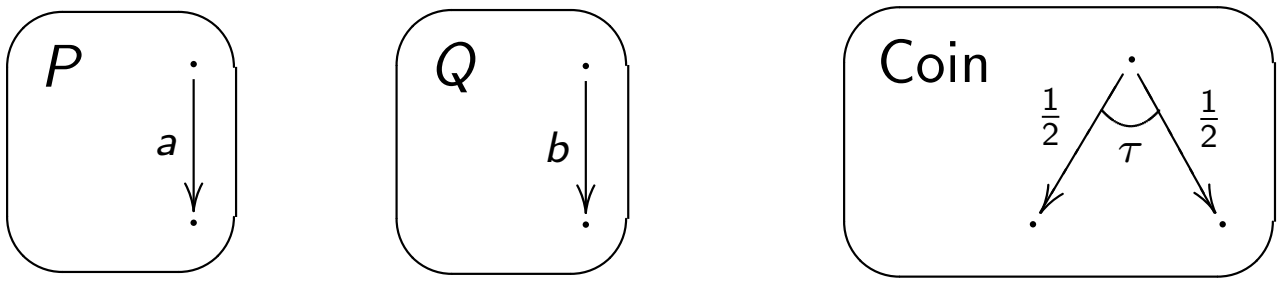
Silly Example I



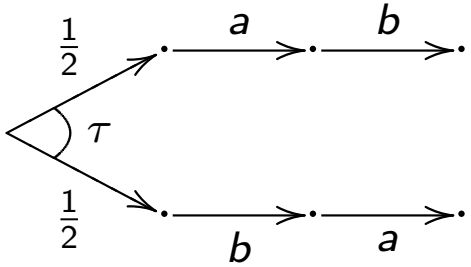




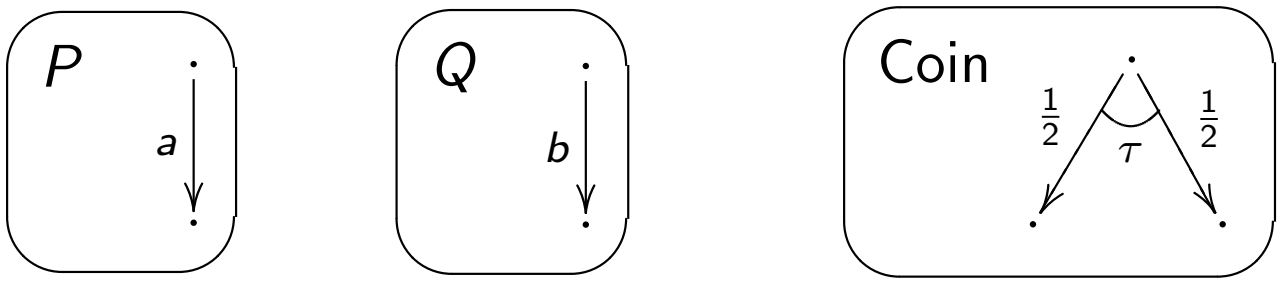
Silly Example I



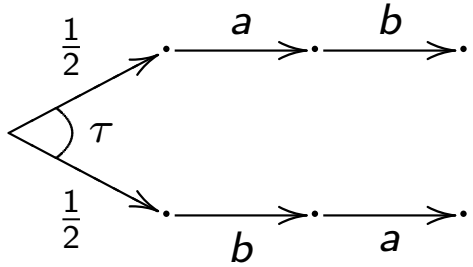
In Simple PA, the following is a "legal" probabilistic execution.



Silly Example I



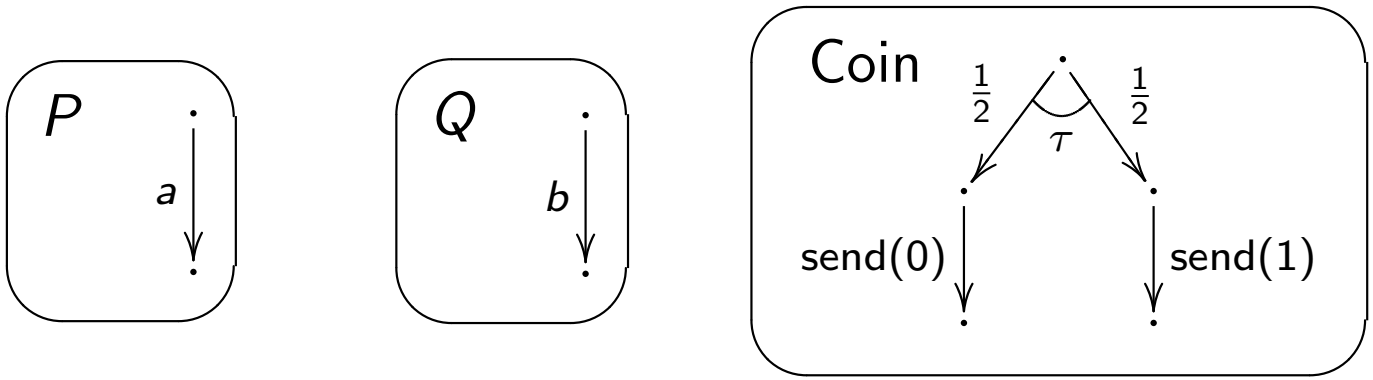
In Simple PA, the following is a "legal" probabilistic execution.



In other words, a hidden random choice in Coin can affect which of P and Q wins the race.

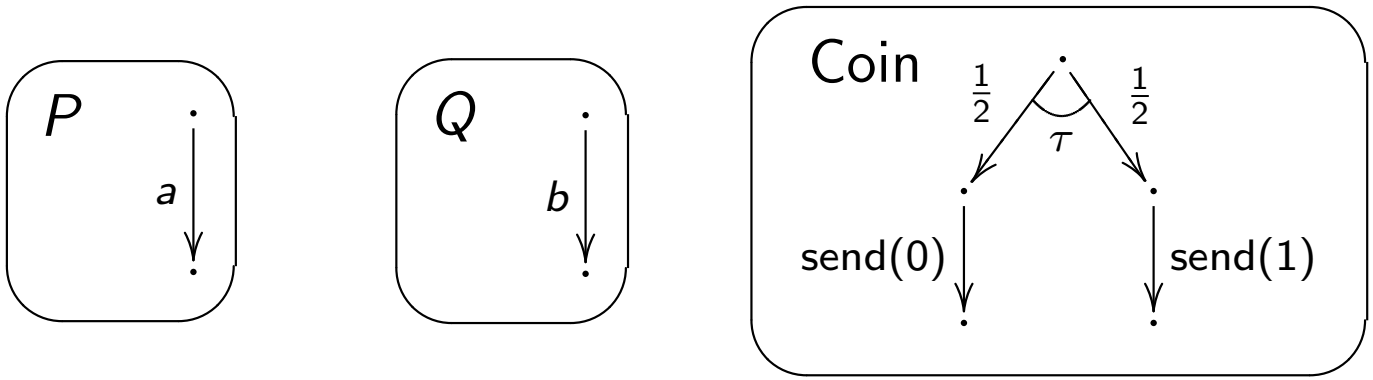
Silly Example II

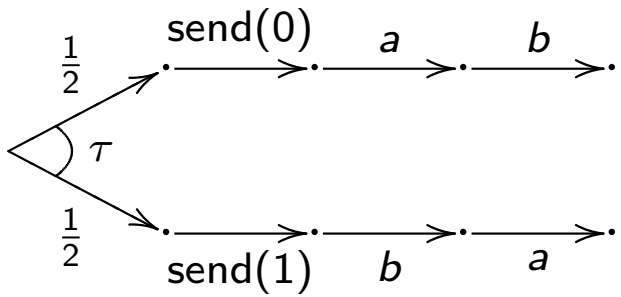
Suppose Coin announces the outcome of its random choice.



Silly Example II

Suppose Coin announces the outcome of its random choice.



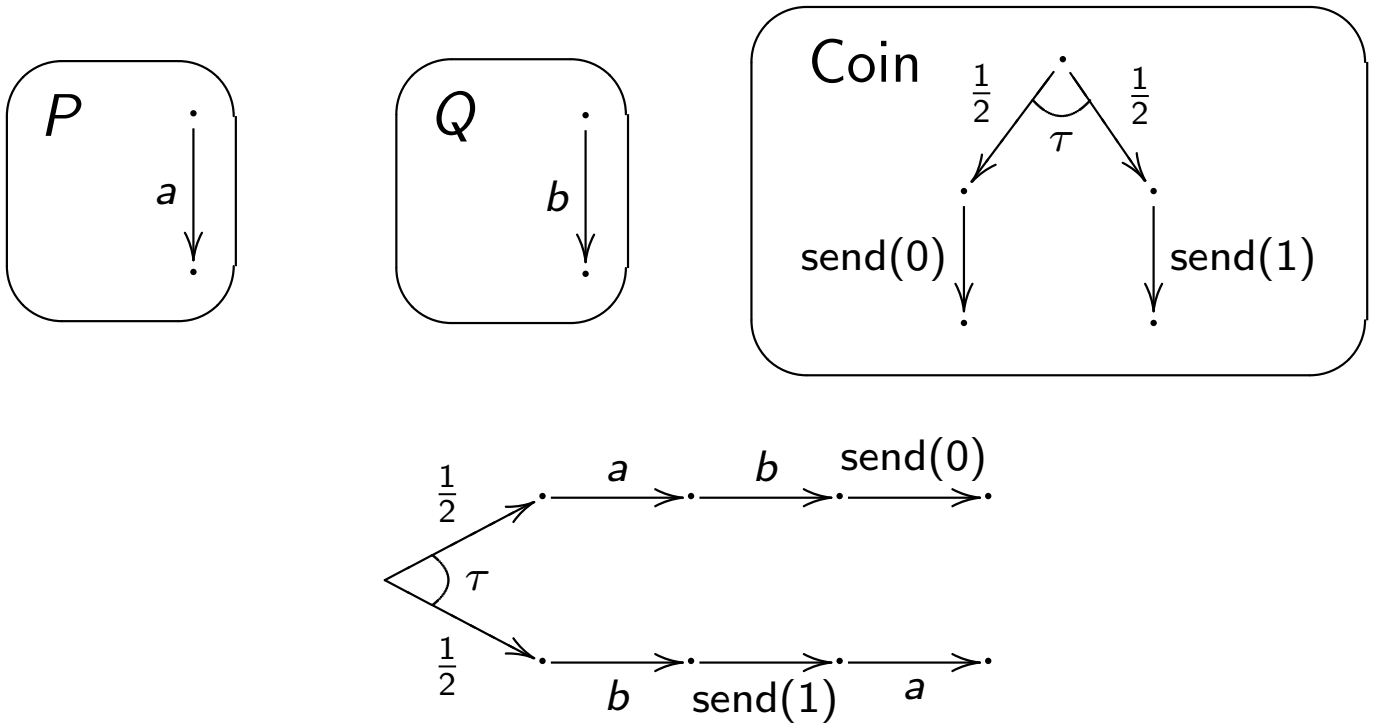


A bit more plausible now?

Perhaps the message from Coin somehow changed the world?

Silly Example II

Suppose Coin announces the outcome of its random choice.



What about this execution?

Our Goal

Claim: Lots of silly things can happen under a strong adversary.

Our Goal

Claim: Lots of silly things can happen under a strong adversary.

Our goal: to weaken the adversary model so that the resulting notion of probabilistic execution is a bit more reasonable.

Our Goal

Claim: Lots of silly things can happen under a strong adversary.

Our goal: to weaken the adversary model so that the resulting notion of probabilistic execution is a bit more reasonable.

In particular, the adversary's knowledge can increase dynamically, as more and more information becomes public.
E.g., the adversary should be able to use the random choice of Coin, but only after the $send(-)$ transition.

Examples of Adversary models

In increasing order of strength.

- Oblivious adversaries: only component names.

Examples of Adversary models

In increasing order of strength.

- Oblivious adversaries: only component names.
- Content-oblivious adversaries: ignore parameter values (e.g. cannot distinguish send(0) from send(1)).

Examples of Adversary models

In increasing order of strength.

- Oblivious adversaries: only component names.
- Content-oblivious adversaries: ignore parameter values (e.g. cannot distinguish send(0) from send(1)).
- Probabilistic polynomial time (PPT) adversaries: cannot perform expensive computation on parameter values (e.g. cannot compute x from $f(x)$ if f is one-way).

Examples of Adversary models

In increasing order of strength.

- Oblivious adversaries: only component names.
- Content-oblivious adversaries: ignore parameter values (e.g. cannot distinguish send(0) from send(1)).
- Probabilistic polynomial time (PPT) adversaries: cannot perform expensive computation on parameter values (e.g. cannot compute x from $f(x)$ if f is one-way).
- Symbolic PA: unbounded computational capabilities, but limited knowledge of outcomes of coin tosses (i.e., only those that are announced via visible actions).

Examples of Adversary models

In increasing order of strength.

- Oblivious adversaries: only component names.
- Content-oblivious adversaries: ignore parameter values (e.g. cannot distinguish send(0) from send(1)).
- Probabilistic polynomial time (PPT) adversaries: cannot perform expensive computation on parameter values (e.g. cannot compute x from $f(x)$ if f is one-way).
- Symbolic PA: unbounded computational capabilities, but limited knowledge of outcomes of coin tosses (i.e., only those that are announced via visible actions).
- Segala's Simple PA: everything.

Randomized Scheduling

All of our adversaries are deterministic. I.e., given any finite execution, the adversary chooses

- either to halt the execution,
- or to extend the execution with one transition,

Why?

Randomized Scheduling

Three main reasons to stay away from randomized scheduling.

Randomized Scheduling

Three main reasons to stay away from randomized scheduling.

- It lacks intuitive justification.
  "Randomized scheduling allows for very simple algorithms; unfortunately, it depends on assumptions about the behavior of the world that may not be justified in practice."

  — James Aspnes, "Randomized Protocols for Asynchronous Consensus"

Randomized Scheduling

Three main reasons to stay away from randomized scheduling.

- It lacks intuitive justification.
  "Randomized scheduling allows for very simple algorithms; unfortunately, it depends on assumptions about the behavior of the world that may not be justified in practice."

  — James Aspnes, "Randomized Protocols for Asynchronous Consensus"
- It introduces "noise", i.e., probabilities that are not in the problem statement.

Randomized Scheduling

Three main reasons to stay away from randomized scheduling.

- It lacks intuitive justification.
  "Randomized scheduling allows for very simple algorithms; unfortunately, it depends on assumptions about the behavior of the world that may not be justified in practice."

  — James Aspnes, "Randomized Protocols for Asynchronous Consensus"
- It introduces "noise", i.e., probabilities that are not in the problem statement.
- It is technically more complicated: e.g. extra proof obligation for convex combinations of deterministic schedules.

Randomized Scheduling

Three main reasons to stay away from randomized scheduling.

- It lacks intuitive justification.
  "Randomized scheduling allows for very simple algorithms; unfortunately, it depends on assumptions about the behavior of the world that may not be justified in practice."

  — James Aspnes, "Randomized Protocols for Asynchronous Consensus"
- It introduces "noise", i.e., probabilities that are not in the problem statement.
- It is technically more complicated: e.g. extra proof obligation for convex combinations of deterministic schedules.

Expressivity: what have we lost?

Basic Ingredients

We have X, a set of symbolic states, together with a transition function

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

Basic Ingredients

We have X, a set of symbolic states, together with a transition function

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

- Given a symbolic state $\mu \in X$,

Basic Ingredients

We have X, a set of symbolic states, together with a transition function

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

- Given a symbolic state $\mu \in X$,
- $\Delta(\mu)$ yields a set of transition bundles from μ, where

Basic Ingredients

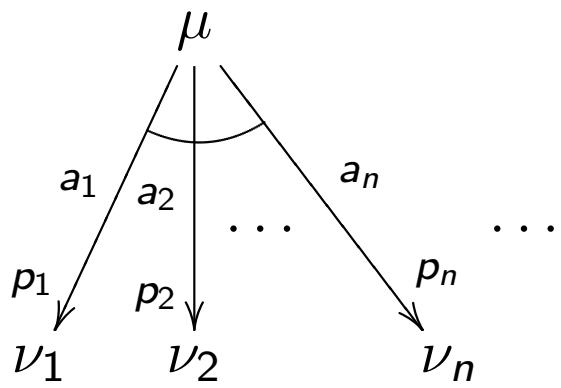
We have X, a set of symbolic states, together with a transition function

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

- Given a symbolic state $\mu \in X$,
- $\Delta(\mu)$ yields a set of transition bundles from μ, where
- each bundle is a function

$$f : Act \longrightarrow [0,1] \times X$$

such that $\pi_1(f)$ is a discrete sub-distribution on Act.



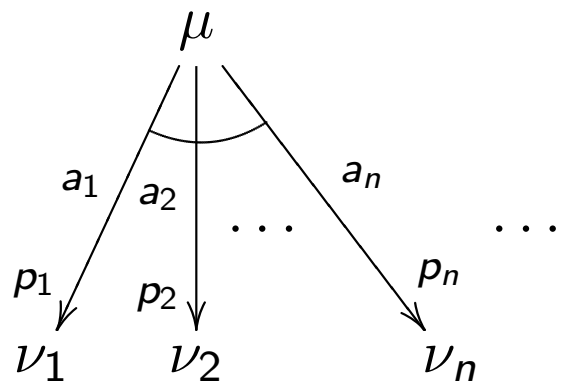
$$Act = \{a_i \mid i \in \mathbb{N}\}$$
$$f(a_i) = \langle p_i, \nu_i \rangle$$

Basic Ingredients

We have X, a set of symbolic states, together with a transition function

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

- $p_i = \pi_1(f(a_i))$ is the fiber probability of a_i in f.



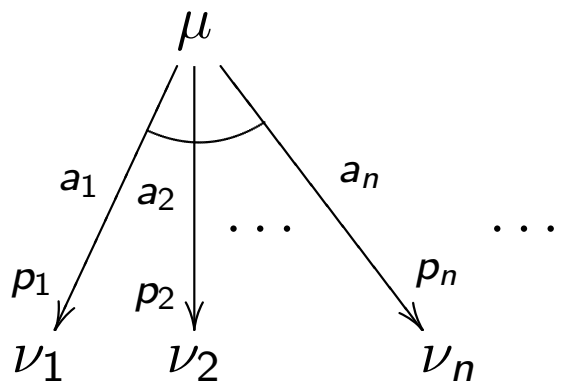
$$Act = \{a_i \mid i \in \mathbb{N}\}$$
$$f(a_i) = \langle p_i, \nu_i \rangle$$

Basic Ingredients

We have X, a set of symbolic states, together with a transition function

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

- $p_i = \pi_1(f(a_i))$ is the fiber probability of a_i in f.
- $\nu_i = \pi_2(f(a_i))$ is the end state of the a_i-fiber in f.



$$Act = \{a_i \mid i \in \mathbb{N}\}$$
$$f(a_i) = \langle p_i, \nu_i \rangle$$

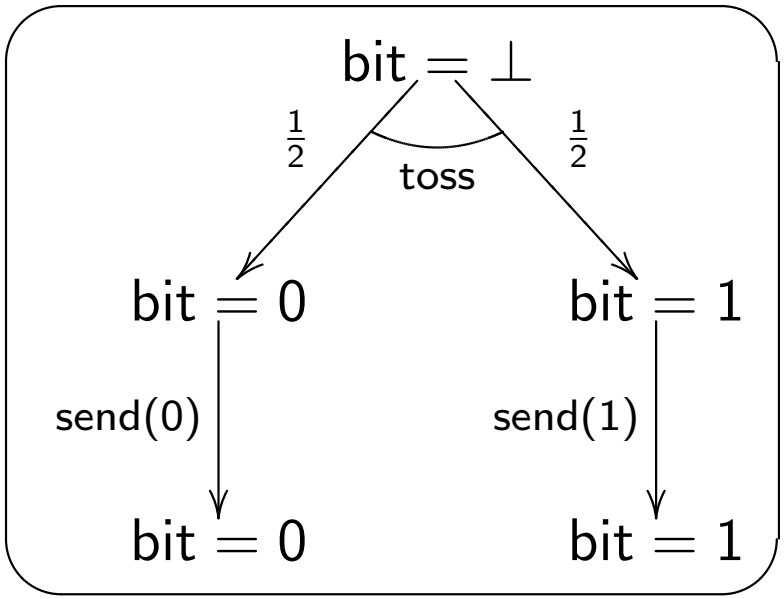
Why Is It Symbolic?

Think of a symbolic state $\mu \in X$ as a discrete distribution on concrete states.

Why Is It Symbolic?

Example: the automaton Coin.

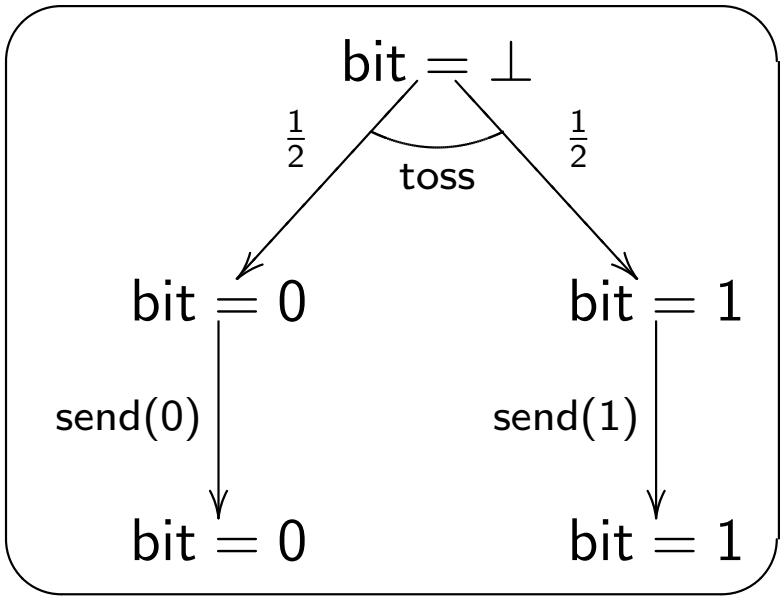
Before (Simple PA):



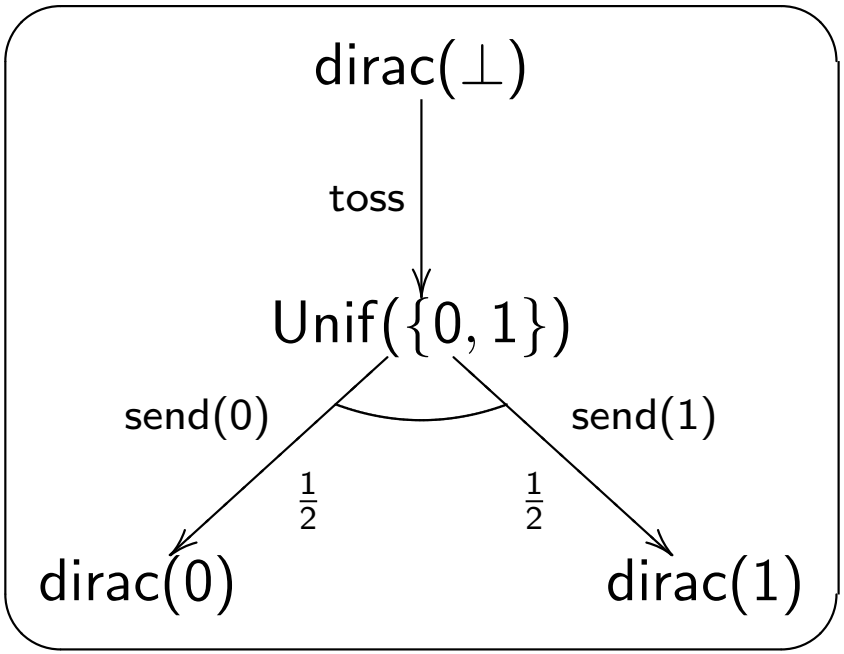
Why Is It Symbolic?

Example: the automaton Coin.

Before (Simple PA):



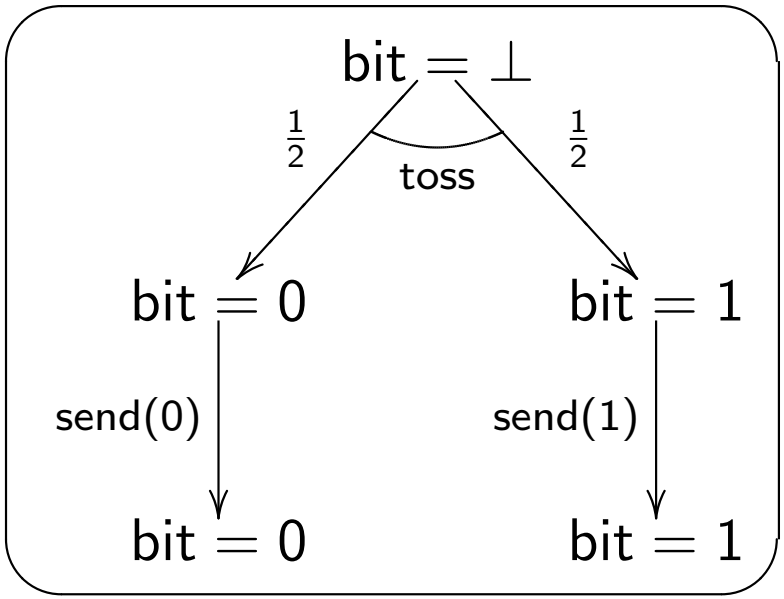
After (Symbolic PA):



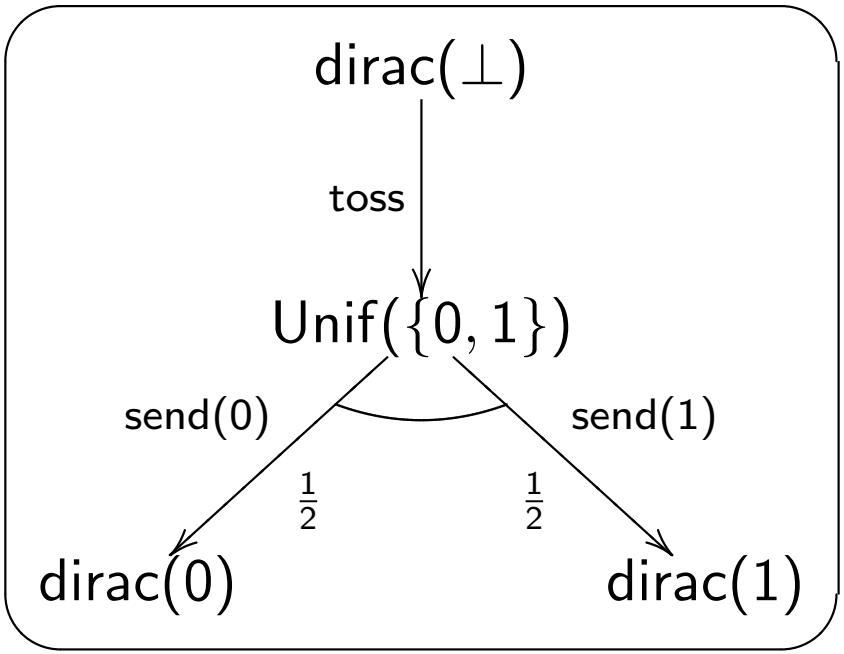
Why Is It Symbolic?

Example: the automaton Coin.

Before (Simple PA):



After (Symbolic PA):



Notice, the adversary learns the value of bit only after observing a send(−) action.

Transition Structure

Symbolic PA:

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

Transition Structure

Symbolic PA:

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

Segala's *General PA*:

$$\Delta : X \longrightarrow \mathcal{P}(\mathrm{Disc}(Act \times X)).$$

Transition Structure

Symbolic PA:

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

Segala's *General PA*:

$$\Delta : X \longrightarrow \mathcal{P}(\mathrm{Disc}(Act \times X)).$$

What's the difference?

Transition Structure

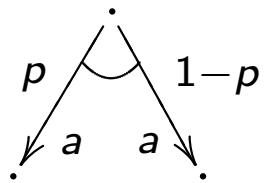
Symbolic PA:

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

Segala's *General PA*:

$$\Delta : X \longrightarrow \mathcal{P}(\mathrm{Disc}(Act \times X)).$$

What's the difference?
We do not allow bundles of this form.



Transition Structure

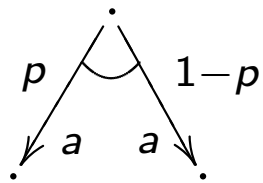
Symbolic PA:

$$\Delta : X \longrightarrow \mathcal{P}(Act \rightarrow ([0,1] \times X)).$$

Segala's *General PA*:

$$\Delta : X \longrightarrow \mathcal{P}(\mathrm{Disc}(Act \times X)).$$

What's the difference?
We do not allow bundles of this form.



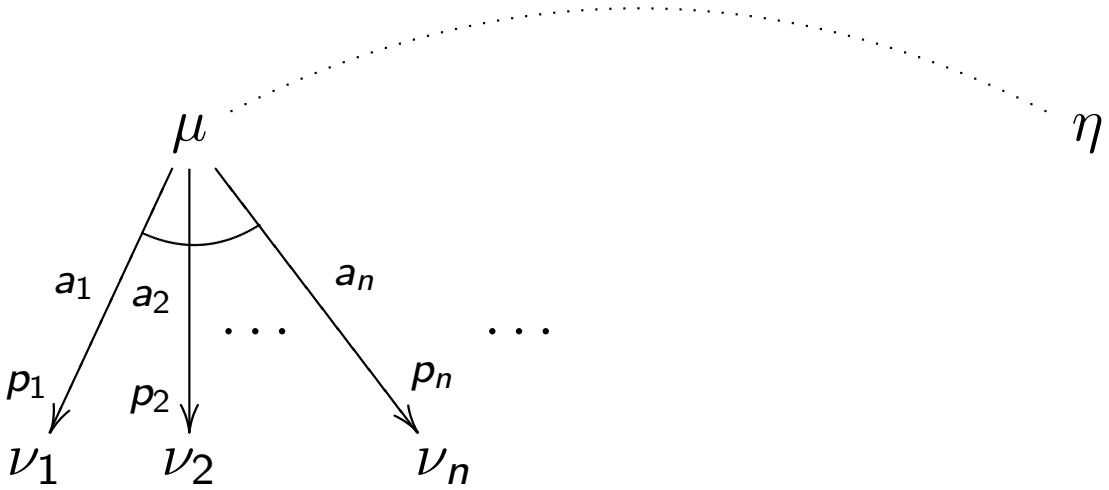
From the Coin example: this restriction is precisely what we need to weaken the adversary model.

Simulation Relation

A relation $R \subseteq X \times X'$ is called a simulation just in case,

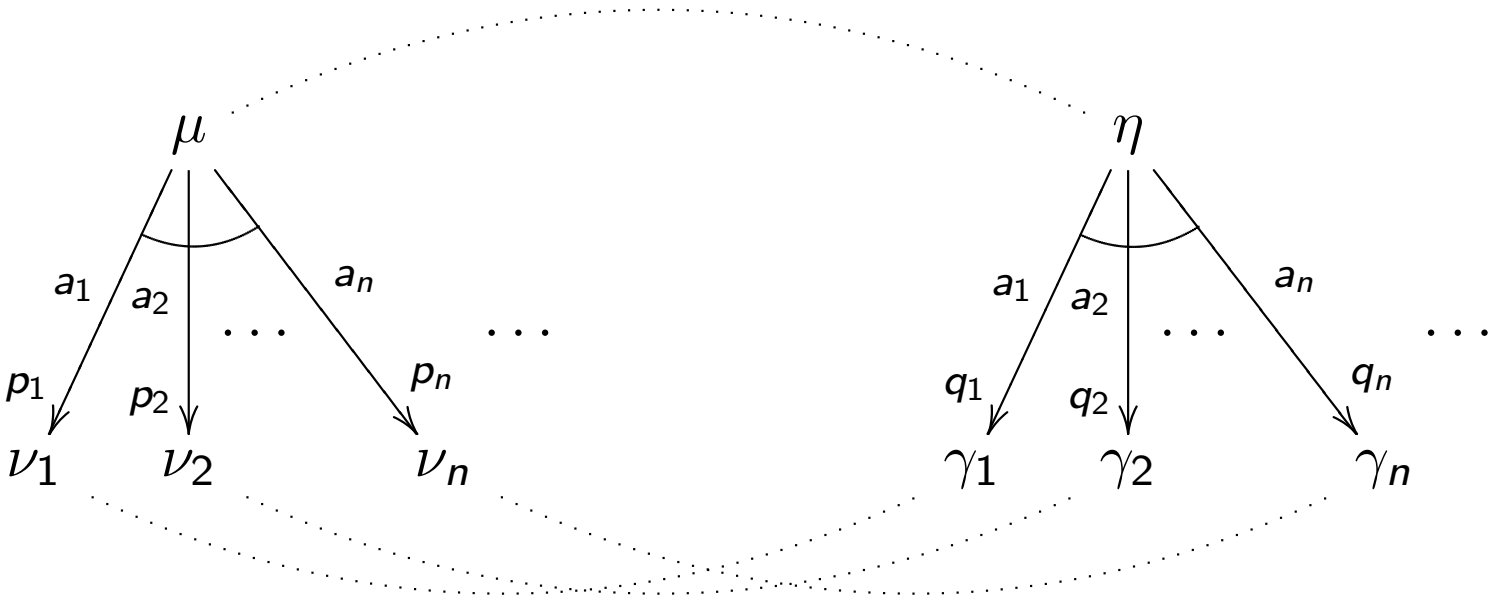
Simulation Relation

A relation $R \subseteq X \times X'$ is called a simulation just in case, for all $\langle \mu, \eta \rangle \in R$ and transition bundles f from μ,



Simulation Relation

A relation $R \subseteq X \times X'$ is called a simulation just in case, for all $\langle \mu, \eta \rangle \in R$ and transition bundles f from μ,



there exists a transition bundle g from η such that: for all i,

$$p_i = q_i \text{ and } \langle \nu_i, \gamma_i \rangle \in R.$$

Soundness

Theorem: For all symbolic PAs P and Q, $P \leq_{\mathsf{sim}} Q$ implies $P \leq_{\mathsf{td}} Q$.

Soundness

Theorem: For all symbolic PAs P and Q, $P \leq_{\mathsf{sim}} Q$ implies $P \leq_{\mathsf{td}} Q$.

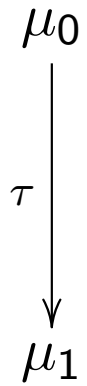
Proof: Induction on depth of probabilistic executions.

Weak Transition Bundles

Add a special symbol τ to the alphabet Act.

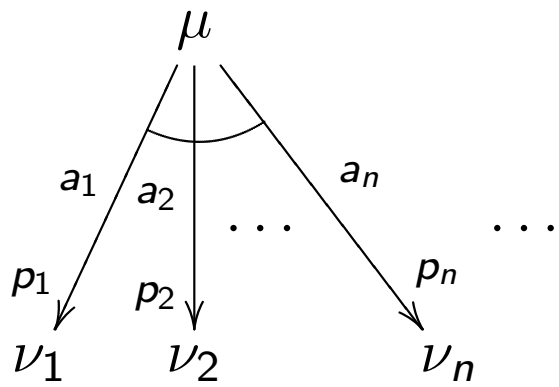
Weak Transition Bundles

- A hidden transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) = \{\tau\}$.



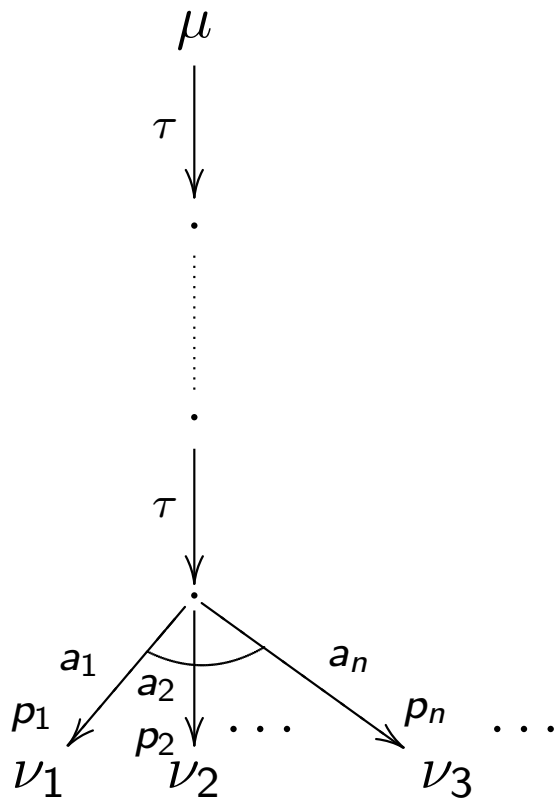
Weak Transition Bundles

- A hidden transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) = \{\tau\}$.
- A visible transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) \subseteq Act \setminus \{\tau\}$.



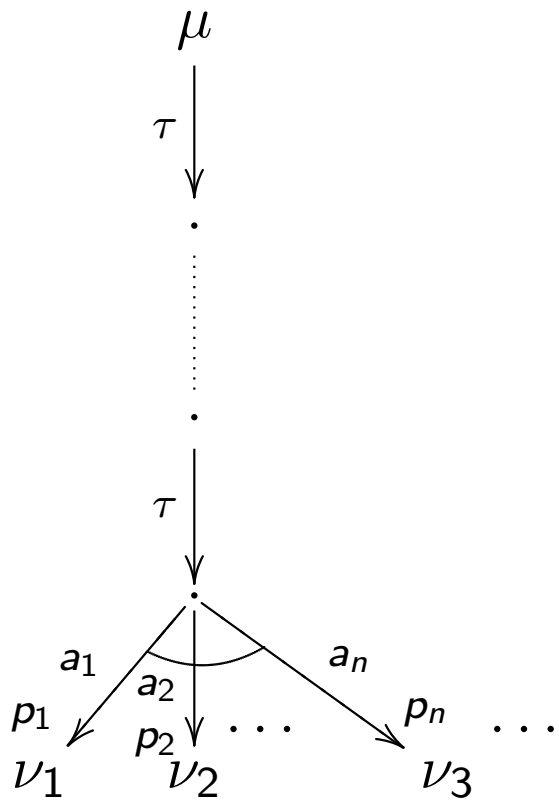
Weak Transition Bundles

- A hidden transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) = \{\tau\}$.
- A visible transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) \subseteq Act \setminus \{\tau\}$.
- A weak transition bundle is a (possibly empty) sequence of hidden bundles followed by a visible bundle.



Weak Transition Bundles

- A hidden transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) = \{\tau\}$.
- A visible transition bundle is a transition bundle f satisfying $\mathrm{Supp}(\pi_1(f)) \subseteq Act \setminus \{\tau\}$.
- A weak transition bundle is a (possibly empty) sequence of hidden bundles followed by a visible bundle.
- Why not partially hidden bundles? See appendix . . .

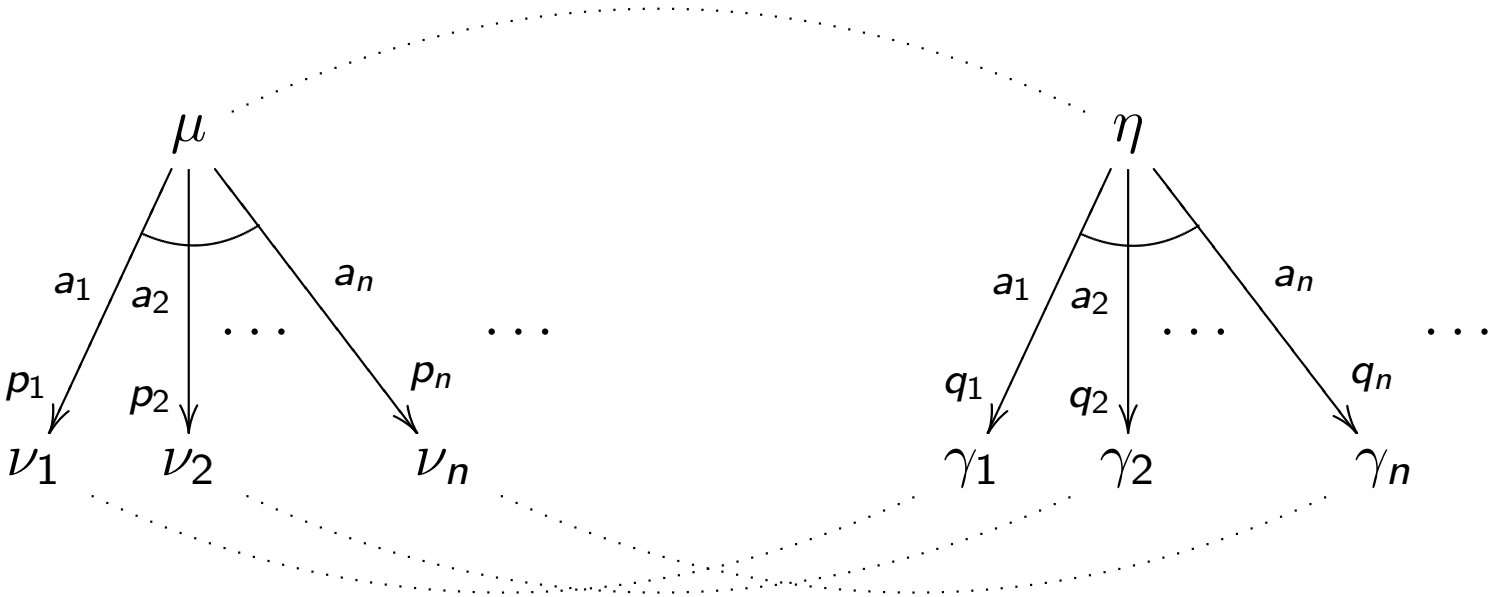


Weak Simulation

A relation $R \subseteq X \times X'$ is called a weak simulation just in case,

Weak Simulation

A relation $R \subseteq X \times X'$ is called a weak simulation just in case, for all $\langle \mu, \eta \rangle \in R$ and weak transition bundles f from μ,



there exists a weak transition bundle g from η such that: for all i,

$$p_i = q_i \text{ and } \langle \nu_i, \gamma_i \rangle \in R.$$

Soundness of Weak Simulation

Theorem: For all symbolic PAs P and Q, $P \leq_{\mathsf{wsim}} Q$ implies $P \leq_{\mathsf{td}} Q$.

Soundness of Weak Simulation

Theorem: For all symbolic PAs P and Q, $P \leq_{\mathsf{wsim}} Q$ implies $P \leq_{\mathsf{td}} Q$.

Proof Outline:

- given P, define $\overline{P}$ based on weak bundles;

Soundness of Weak Simulation

Theorem: For all symbolic PAs P and Q, $P \leq_{\mathsf{wsim}} Q$ implies $P \leq_{\mathsf{td}} Q$.

Proof Outline:

- given P, define $\overline{P}$ based on weak bundles;
- show that $\mathsf{td}(P) = \mathsf{td}(\overline{P})$;

Soundness of Weak Simulation

Theorem: For all symbolic PAs P and Q, $P \leq_{\text{wsim}} Q$ implies $P \leq_{\text{td}} Q$.

Proof Outline:

- given P, define $\overline{P}$ based on weak bundles;
- show that $\text{td}(P) = \text{td}(\overline{P})$;
- show that $P \leq_{\text{wsim}} Q$ implies $\overline{P} \leq_{\text{sim}} \overline{Q}$;

Soundness of Weak Simulation

Theorem: For all symbolic PAs P and Q, $P \leq_{\text{wsim}} Q$ implies $P \leq_{\text{td}} Q$.

Proof Outline:

- given P, define $\overline{P}$ based on weak bundles;
- show that $\text{td}(P) = \text{td}(\overline{P})$;
- show that $P \leq_{\text{wsim}} Q$ implies $\overline{P} \leq_{\text{sim}} \overline{Q}$;
- apply soundness of strong simulation.

I/O Distinction

We have *three* categories of transition bundles.

I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.

I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.
- Output: $\mathrm{Supp}(\pi_1(f)) \subseteq O$.

I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.
- Output: $\mathrm{Supp}(\pi_1(f)) \subseteq O$.
- Hidden: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in H$.

I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.
- Output: $\mathrm{Supp}(\pi_1(f)) \subseteq O$.
- Hidden: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in H$.

Remarks:

- Each input bundle consists of a unique fiber. Idea: to avoid deadlock, inputs must be received with probability 1.

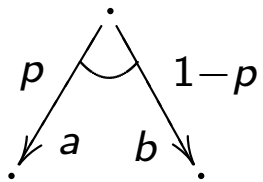
I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.
- Output: $\mathrm{Supp}(\pi_1(f)) \subseteq O$.
- Hidden: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in H$.

Remarks:

- Each input bundle consists of a unique fiber. Idea: to avoid deadlock, inputs must be received with probability 1.
- Same for hidden bundles. Otherwise the adversary can learn private information.



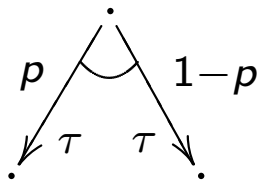
I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.
- Output: $\mathrm{Supp}(\pi_1(f)) \subseteq O$.
- Hidden: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in H$.

Remarks:

- Each input bundle consists of a unique fiber. Idea: to avoid deadlock, inputs must be received with probability 1.
- Same for hidden bundles. Otherwise the adversary can learn private information.



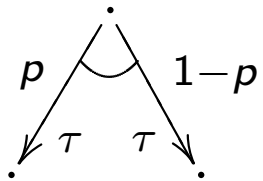
I/O Distinction

We have *three* categories of transition bundles.

- Input: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in I$.
- Output: $\mathrm{Supp}(\pi_1(f)) \subseteq O$.
- Hidden: $\mathrm{Supp}(\pi_1(f)) = \{a\}$ for some $a \in H$.

Remarks:

- Each input bundle consists of a unique fiber. Idea: to avoid deadlock, inputs must be received with probability 1.
- Same for hidden bundles. Otherwise the adversary can learn private information. (Alternative: special system type for hidden actions.)

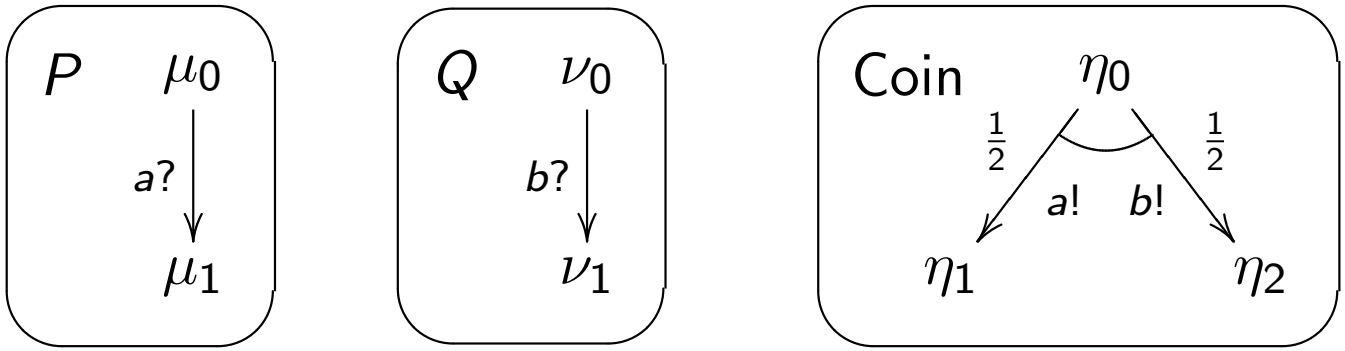


Parallel Composition

Definition by example . . .

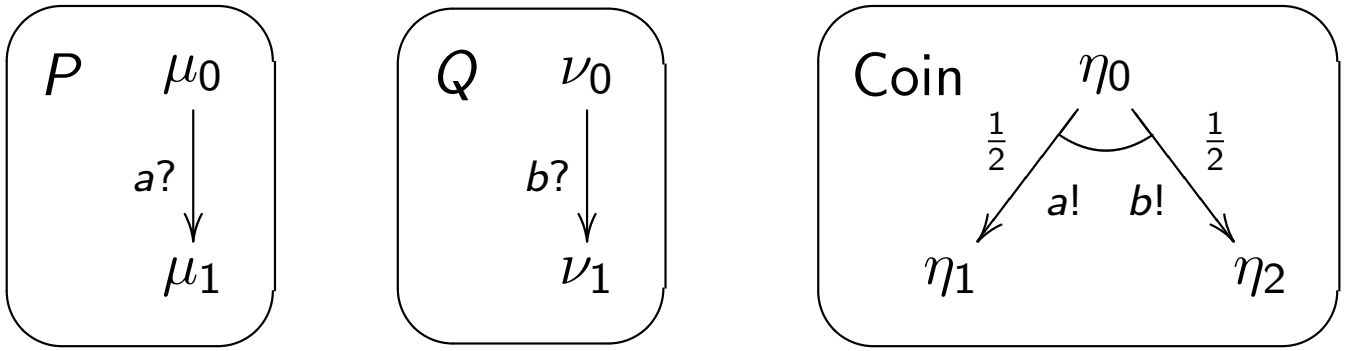
Parallel Composition

Definition by example . . .

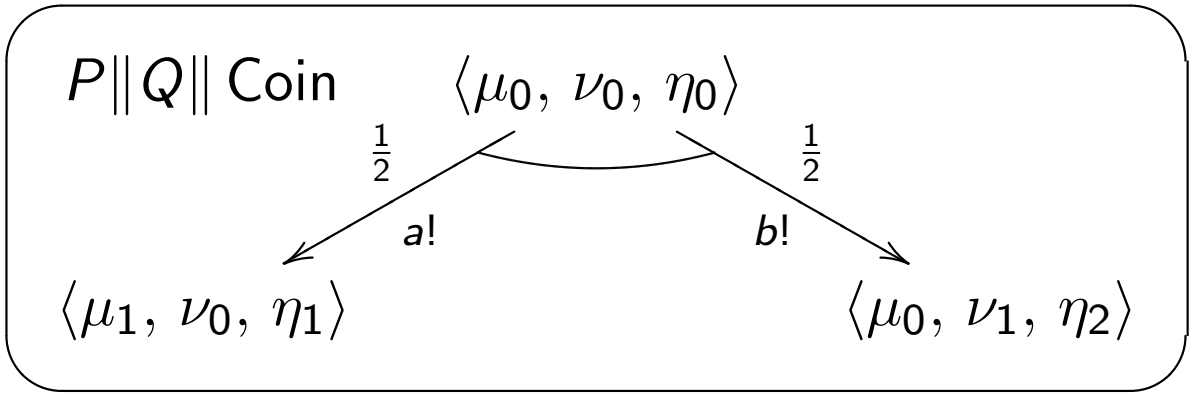


Parallel Composition

Definition by example . . .



Their composite:



What Have We Achieved?

- A system type for which adversaries:

What Have We Achieved?

- A system type for which adversaries:
  - can observe visible actions and symbolic states;

What Have We Achieved?

- A system type for which adversaries:
  - can observe visible actions and symbolic states;
  - cannot observe internal actions and random choices . . .

What Have We Achieved?

- A system type for which adversaries:
  - can observe visible actions and symbolic states;
  - cannot observe internal actions and random choices . . . until the outcomes are publicized via visible actions.

What Have We Achieved?

- A system type for which adversaries:
  - can observe visible actions and symbolic states;
  - cannot observe internal actions and random choices . . . until the outcomes are publicized via visible actions.
- Strong simulation and soundness w.r.t. trace distribution inclusion.

What Have We Achieved?

- A system type for which adversaries:
  - can observe visible actions and symbolic states;
  - cannot observe internal actions and random choices . . . until the outcomes are publicized via visible actions.
- Strong simulation and soundness w.r.t. trace distribution inclusion.
- Weak transition structure.

What Have We Achieved?

- A system type for which adversaries:
  - can observe visible actions and symbolic states;
  - cannot observe internal actions and random choices . . . until the outcomes are publicized via visible actions.
- Strong simulation and soundness w.r.t. trace distribution inclusion.
- Weak transition structure.
- Weak simulation and soundness w.r.t. trace distribution inclusion (proof sketch).

What To Do Next?

- Obviously, compositionality theorems.

What To Do Next?

- Obviously, compositionality theorems.
- Do we really have a good model?

What To Do Next?

- Obviously, compositionality theorems.
- Do we really have a good model?
  - Are there interesting examples that cannot be expressed in Symbolic PA?

What To Do Next?

- Obviously, compositionality theorems.
- Do we really have a good model?
  - Are there interesting examples that cannot be expressed in Symbolic PA?
  - Is it difficult to write down system definitions?
    Case study: randomized consensus.

What To Do Next?

- Obviously, compositionality theorems.
- Do we really have a good model?
  - Are there interesting examples that cannot be expressed in Symbolic PA?
  - Is it difficult to write down system definitions?
    Case study: randomized consensus.
  - Is it difficult to carry out proofs?

What To Do Next?

- Obviously, compositionality theorems.
- Do we really have a good model?
  - Are there interesting examples that cannot be expressed in Symbolic PA?
  - Is it difficult to write down system definitions?
    Case study: randomized consensus.
  - Is it difficult to carry out proofs?

The end . . .
Questions? Or move on to Oblivious Transfer?

Oblivious Transfer: The Problem Statement

There are two protocol participants:

Oblivious Transfer: The Problem Statement

There are two protocol participants:

- the *Transmitter* has as its input two bits x_0 and x_1;

Oblivious Transfer: The Problem Statement

There are two protocol participants:

- the *Transmitter* has as its input two bits x_0 and x_1;
- the *Receiver* has as its input a selector bit i.

Oblivious Transfer: The Problem Statement

There are two protocol participants:

- the *Transmitter* has as its input two bits x_0 and x_1;
- the *Receiver* has as its input a selector bit i.

The goal: by the end of the protocol,

- "transfer": the Receiver learns x_i;

Oblivious Transfer: The Problem Statement

There are two protocol participants:

- the *Transmitter* has as its input two bits x_0 and x_1;
- the *Receiver* has as its input a selector bit i.

The goal: by the end of the protocol,

- "transfer": the Receiver learns x_i;
- "oblivious":
  - the Transmitter does not learn the selector bit i;

Oblivious Transfer: The Problem Statement

There are two protocol participants:

- the *Transmitter* has as its input two bits x_0 and x_1;
- the *Receiver* has as its input a selector bit i.

The goal: by the end of the protocol,

- "transfer": the Receiver learns x_i;
- "oblivious":
  - the Transmitter does not learn the selector bit i;
  - the Receiver does not learn the unselected bit x_{1-i}; and

Oblivious Transfer: The Problem Statement

There are two protocol participants:

- the *Transmitter* has as its input two bits x_0 and x_1;
- the *Receiver* has as its input a selector bit i.

The goal: by the end of the protocol,

- "transfer": the Receiver learns x_i;
- "oblivious":
  - the Transmitter does not learn the selector bit i;
  - the Receiver does not learn the unselected bit x_{1-i}; and
  - nobody else learns anything.

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.
- "hard to invert": the success probability of any PPT algorithm attempting to invert f is at most $\frac{1}{2}$ plus some negligible amount.

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.
- "hard to invert": the success probability of any PPT algorithm attempting to invert f is at most $\frac{1}{2}$ plus some negligible amount.

A trapdoor permutation is a one-way permutation f together with a trapdoor t_k, for each key length k, such that:

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.
- "hard to invert": the success probability of any PPT algorithm attempting to invert f is at most $\frac{1}{2}$ plus some negligible amount.

A trapdoor permutation is a one-way permutation f together with a trapdoor t_k, for each key length k, such that:

on inputs of length k, f is easy to invert given t_k.

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.
- "hard to invert": the success probability of any PPT algorithm attempting to invert f is at most $\frac{1}{2}$ plus some negligible amount.

The hardcore predicate of f is a predicate B such that the bit $B(f^{-1}(x))$ is indistinguishable from a random bit, provided:

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.
- "hard to invert": the success probability of any PPT algorithm attempting to invert f is at most $\frac{1}{2}$ plus some negligible amount.

The hardcore predicate of f is a predicate B such that the bit $B(f^{-1}(x))$ is indistinguishable from a random bit, provided:

- x is chosen randomly, and

Trapdoor Permutations

A one-way permutation is a length-preserving function f on the set of finite bit strings satisfying:

- "easy to compute": there is a poly-time Turing machine that returns $f(x)$ on any input x.
- "hard to invert": the success probability of any PPT algorithm attempting to invert f is at most $\frac{1}{2}$ plus some negligible amount.

The hardcore predicate of f is a predicate B such that the bit $B(f^{-1}(x))$ is indistinguishable from a random bit, provided:

- x is chosen randomly, and
- the observer is PPT and does not know f^{-1}.

Encryption and Decryption

Example: private key encryption.

Encryption and Decryption

Example: private key encryption.

- The string x is the plain text.

Encryption and Decryption

Example: private key encryption.

- The string x is the plain text.
- The string $f(x)$ is the cipher text.

Encryption and Decryption

Example: private key encryption.

- The string x is the plain text.
- The string $f(x)$ is the cipher text.
- The permutation f is the public key (for encryption).

Encryption and Decryption

Example: private key encryption.

- The string x is the plain text.
- The string $f(x)$ is the cipher text.
- The permutation f is the public key (for encryption).
- The trapdoor information t_k is the private key (for decryption).

A Computationally Secure Implementation of OT

Let D be the domain of trapdoor permutations.

A Computationally Secure Implementation of OT

Let D be the domain of trapdoor permutations.

On inputs $\langle x_0, x_1 \rangle$ for Transmitter T and i for Receiver R.

A Computationally Secure Implementation of OT

Let D be the domain of trapdoor permutations.

On inputs $\langle x_0, x_1 \rangle$ for Transmitter T and i for Receiver R.

- T selects a random trapdoor permutation f.
  $T \longrightarrow R$: f

A Computationally Secure Implementation of OT

Let D be the domain of trapdoor permutations.

On inputs $\langle x_0, x_1 \rangle$ for Transmitter T and i for Receiver R.

- T selects a random trapdoor permutation f.
  $T \rightarrow R$: f
- R selects two random elements $y_0, y_1 \in D$ and sets $\langle z_0, z_1 \rangle := \langle f^{1-i}(y_0), f^i(y_1) \rangle$.
  $R \rightarrow T$: $\langle z_0, z_1 \rangle$

A Computationally Secure Implementation of OT

Let D be the domain of trapdoor permutations.

On inputs $\langle x_0, x_1 \rangle$ for Transmitter T and i for Receiver R.

- T selects a random trapdoor permutation f.
  $T \rightarrow R$: f
- R selects two random elements $y_0, y_1 \in D$ and sets $\langle z_0, z_1 \rangle := \langle f^{1-i}(y_0), f^i(y_1) \rangle$.
  $R \rightarrow T$: $\langle z_0, z_1 \rangle$
- For each $j \in \{0, 1\}$, T sets $b_j := B(f^{-1}(z_j)) \oplus x_j$, where B is a hardcore predicate for f.
  $T \rightarrow R$: $\langle b_0, b_1 \rangle$

A Computationally Secure Implementation of OT

Let D be the domain of trapdoor permutations.

On inputs $\langle x_0, x_1 \rangle$ for Transmitter T and i for Receiver R.

- T selects a random trapdoor permutation f.
  $T \rightarrow R$: f
- R selects two random elements $y_0, y_1 \in D$ and sets $\langle z_0, z_1 \rangle := \langle f^{1-i}(y_0), f^i(y_1) \rangle$.
  $R \rightarrow T$: $\langle z_0, z_1 \rangle$
- For each $j \in \{0, 1\}$, T sets $b_j := B(f^{-1}(z_j)) \oplus x_j$, where B is a hardcore predicate for f.
  $T \rightarrow R$: $\langle b_0, b_1 \rangle$
- R outputs $B(y_i) \oplus b_i$.

Universally Composable Security

Modeling framework due to Ran Canetti:

Universally Composable Security

Modeling framework due to Ran Canetti:

- Components are modeled as probabilistic polynomial time ITMs (Interactive Turing Machines).

Universally Composable Security

Modeling framework due to Ran Canetti:

- Components are modeled as probabilistic polynomial time ITMs (Interactive Turing Machines).
- Ideal functionality for each protocol/task (e.g. trusted third-party or trusted channel).

Universally Composable Security

Modeling framework due to Ran Canetti:

- Components are modeled as probabilistic polynomial time ITMs (Interactive Turing Machines).
- Ideal functionality for each protocol/task (e.g. trusted third-party or trusted channel).
- Real system: protocol parties and adversary.

Universally Composable Security

Modeling framework due to Ran Canetti:

- Components are modeled as probabilistic polynomial time ITMs (Interactive Turing Machines).
- Ideal functionality for each protocol/task (e.g. trusted third-party or trusted channel).
- Real system: protocol parties and adversary.
- Ideal system: ideal functionality and simulator.

Universally Composable Security

Modeling framework due to Ran Canetti:

- Components are modeled as probabilistic polynomial time ITMs (Interactive Turing Machines).
- Ideal functionality for each protocol/task (e.g. trusted third-party or trusted channel).
- Real system: protocol parties and adversary.
- Ideal system: ideal functionality and simulator.
- Security is defined to be indistinguishability between the real system and the ideal system.

Universally Composable Security

Modeling framework due to Ran Canetti:

- Components are modeled as probabilistic polynomial time ITMs (Interactive Turing Machines).
- Ideal functionality for each protocol/task (e.g. trusted third-party or trusted channel).
- Real system: protocol parties and adversary.
- Ideal system: ideal functionality and simulator.
- Security is defined to be indistinguishability between the real system and the ideal system.

Idea: Ideal system is "obviously" secure.

Computational Security in PIOA

We model everything in PIOA, including: real system, ideal system and the environment/distinguisher.

Computational Security in PIOA

We model everything in PIOA, including: real system, ideal system and the environment/distinguisher.

Conjecture: given any probabilistic execution in real system, there exists a computationally indistinguishable execution in the ideal system.

Computational Security in PIOA

We model everything in PIOA, including: real system, ideal system and the environment/distinguisher.

Conjecture: given any probabilistic execution in real system, there exists a computationally indistinguishable execution in the ideal system.

Where we are stuck: adversarial scheduling.

Computational Security in PIOA

We model everything in PIOA, including: real system, ideal system and the environment/distinguisher.

Conjecture: given any probabilistic execution in real system, there exists a computationally indistinguishable execution in the ideal system.

Where we are stuck: adversarial scheduling.

- *Easy* solution: content-oblivious scheduling.

Computational Security in PIOA

We model everything in PIOA, including: real system, ideal system and the environment/distinguisher.

Conjecture: given any probabilistic execution in real system, there exists a computationally indistinguishable execution in the ideal system.

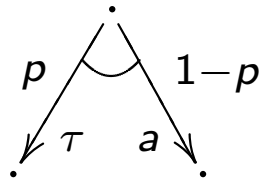
Where we are stuck: adversarial scheduling.

- *Easy* solution: content-oblivious scheduling.
- *Hard* solution: PPT scheduling.

The end . . . really.

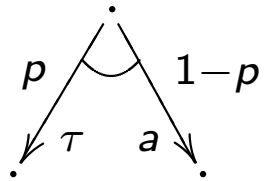
Partially Hidden Bundles

Question: Should we allow partially hidden bundles?



Partially Hidden Bundles

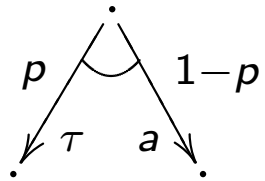
Question: Should we allow partially hidden bundles?



Answer: Not unless we have to.

Partially Hidden Bundles

Question: Should we allow partially hidden bundles?

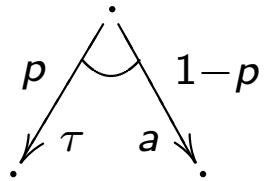


Answer: Not unless we have to.

Pro: More expressive.

Partially Hidden Bundles

Question: Should we allow partially hidden bundles?



Answer: Not unless we have to.

Pro: More expressive.
Con: No straightforward definition of weak simulation.

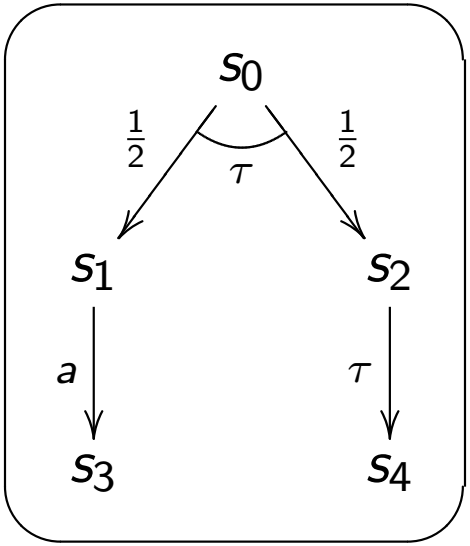
Doing Without Partially Hidden Bundles

Claim: Partially hidden bundles are not necessary under a certain finiteness assumption.

Doing Without Partially Hidden Bundles

Example 1:

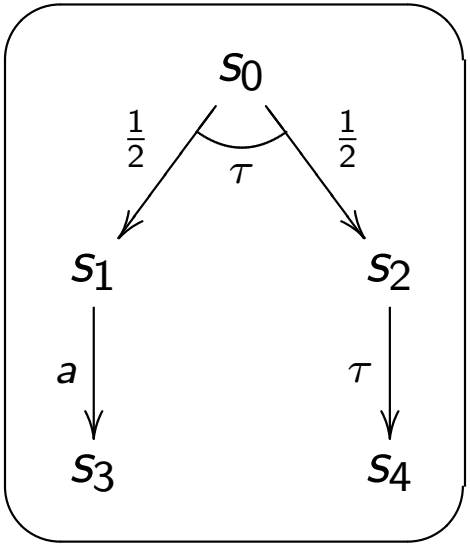
Before (Simple PA):



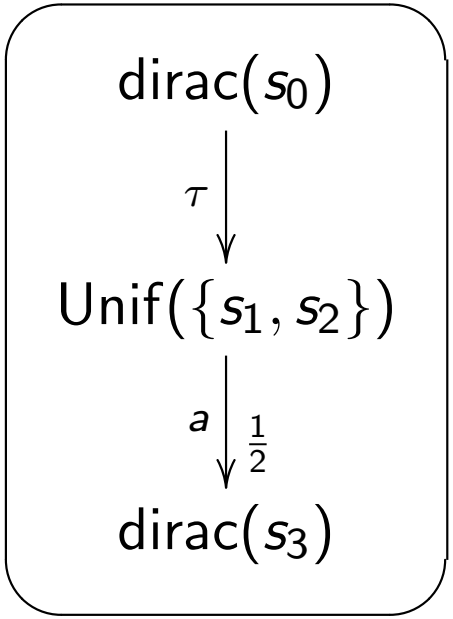
Doing Without Partially Hidden Bundles

Example 1:

Before (Simple PA):



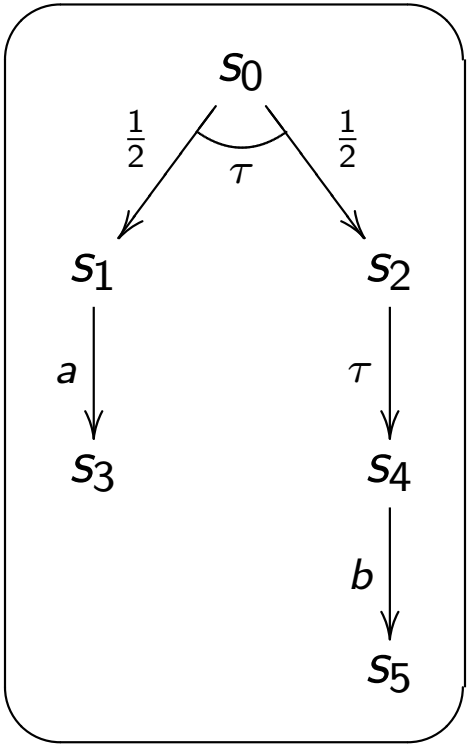
After (Symbolic PA):



Doing Without Partially Hidden Bundles

Example 2:

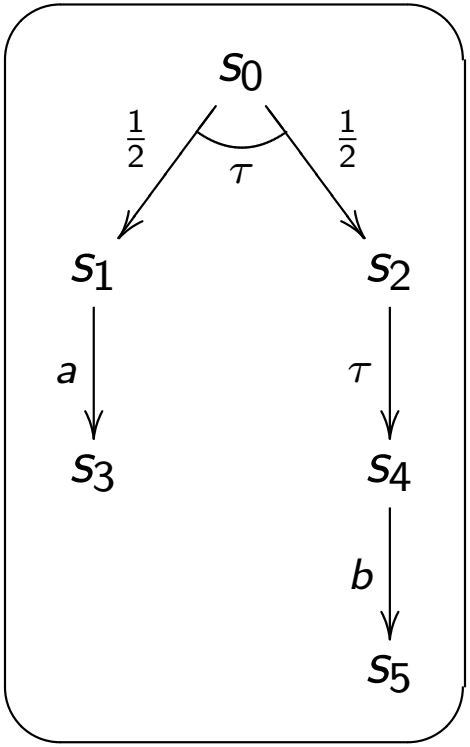
Before (Simple PA):



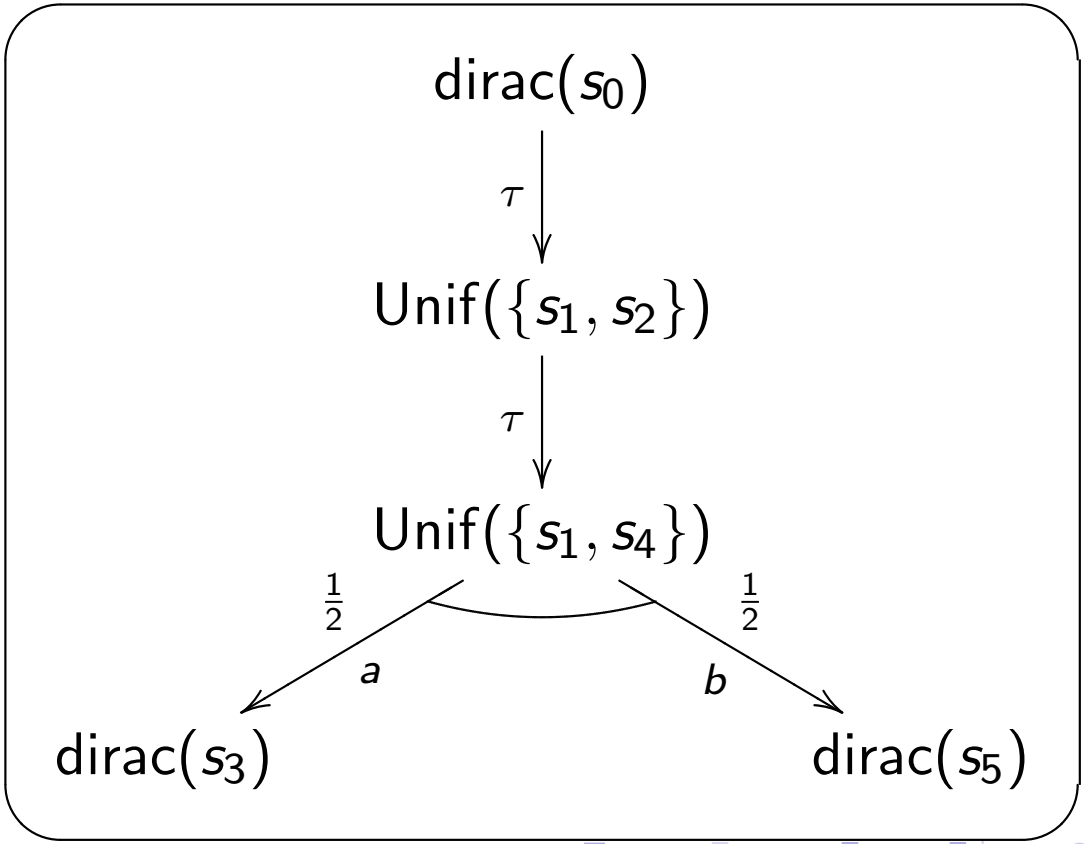
Doing Without Partially Hidden Bundles

Example 2:

Before (Simple PA):



After (Symbolic PA):



Doing Without Partially Hidden Bundles

This method works as long as the number of τ-steps before a visible action is bounded.

Doing Without Partially Hidden Bundles

This method works as long as the number of τ-steps before a visible action is bounded.
E.g., the following does not occur.

$$\frac{1}{2}(a_1) + \frac{1}{4}(\tau.a_2) + \frac{1}{8}(\tau.\tau.a_3) + \frac{1}{16}(\tau.\tau.\tau.a_4) + \ldots$$

Doing Without Partially Hidden Bundles

This method works as long as the number of τ-steps before a visible action is bounded.
E.g., the following does not occur.

$$\frac{1}{2}(a_1) + \frac{1}{4}(\tau.a_2) + \frac{1}{8}(\tau.\tau.a_3) + \frac{1}{16}(\tau.\tau.\tau.a_4) + \ldots$$

Notice, this restriction applies only to probabilistic branching, and not to non-deterministic branching.