Oblivious Probabilistic Outcome Logic

Hanxi Chen, Noam Zilberstein, Andrew Myers, Alexandra Silva

PLDG, 09/09/2026

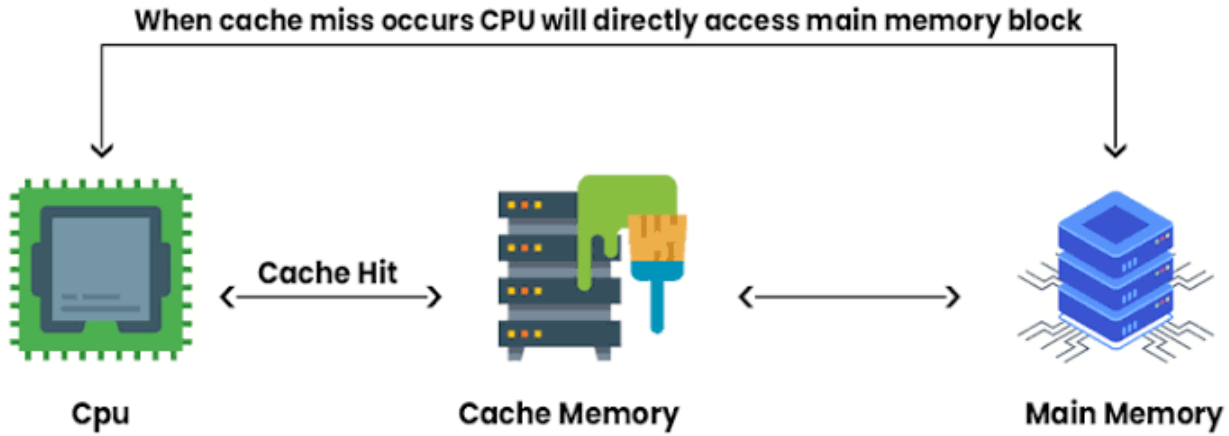
Probabilistic Programs with Adversarial Choice

Online algorithms and distributed systems often combine **probabilistic choices** with choices made by an **external environment**.

What the program can guarantee depends on what that environment can observe.

Probabilistic Programs with Adversarial Choice

Online algorithms and distributed systems often combine **probabilistic choices** with choices made by an **external environment.**



What the program can guarantee depends on what that environment can observe.

Example: Paging

```
# cache holds m of the n pages
while true do
    # adversary requests a page
    r ← [0, . . . , n] ⨾
    if r ∈ cache then skip else
        # evict a random page
        e :≈ Unif (cache) ⨾
        cache.replace(e, r)
```

- **Red** nondeterministic (adversarial) choice
- **Blue** probabilistic choice

The adversary models the request-generating environment.

Example: Paging – Demonic Adversary

```
# cache holds m of the n pages
while true do
    # adversary requests a page
    r ← [0, . . . , n] ;
    if r ∈ cache then skip else
        # evict a random page
        e :≈ Unif (cache) ;
        cache.replace(e, r)
```

So it requests e next — every request misses

Adversary observes the random eviction e

Example: Paging - Oblivious Adversary

```
# cache holds m of the n pages
while true do
    # adversary requests a page
    r ← [0, . . . , n] ⨾
    if r ∈ cache then skip else
        # evict a random page
        e :≈ Unif (cache) ⨾
        cache.replace(e, r)
```

All requests are fixed in advance — they cannot track the cache

Eviction e stays private from the adversary

Example: Paging - Comparison

Demonic adversary	Oblivious adversary
Observes random evictions	Fixes requests in advance
Requests may depend on the cache	Requests cannot depend on the cache
Can force every request to miss	Randomization provides a guarantee

We need a program logic whose semantics matches the intended oblivious adversary.

Prior Logics Model the Demonic Adversary

Demonic adversary — *stronger than intended*

- Polaris [Tassarotti and Harper 2019]
- dOL [Zilberstein, Kozen, Silva, Tassarotti 2025]
- Coneris [Li, Aguirre, Gregersen, Haselwarter, Tassarotti, Birkedal 2025]
- pcOL [Zilberstein, Silva, Tassarotti 2026]

...

Oblivious Adversary - Fan, Liang, Feng, Jiang 2025

- ! No separation
- ! Requires manual program annotation (lazy coins)
- ! Partial correctness
- ! No mechanization

Goal

Develop a program logic for reasoning about programs with

- 🎲 Randomization
- 🙈 Oblivious Adversary
- 🔁 Unbounded Loops

While keeping the reasoning

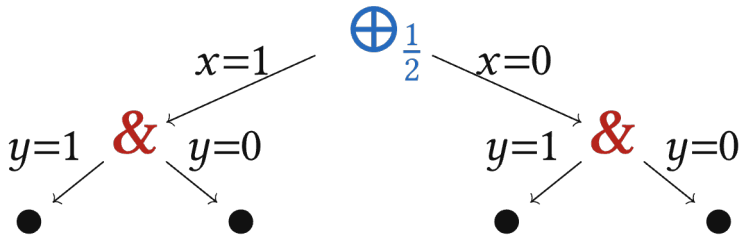
- 🧩 Compositional
- ✍️ Expressive
- 🙂 User-friendly
- ✅ Machine-checked

Example: Swapping Sampling and Choice

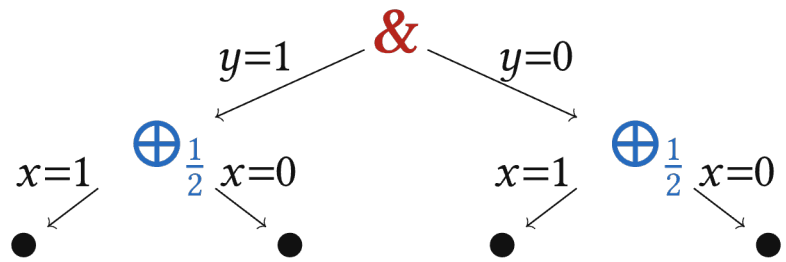
$$x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \mathbin{⨟} y \leftarrow [0,1] \qquad\qquad y \leftarrow [0,1] \mathbin{⨟} x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right)$$

Example: Swapping Sampling and Choice

$$x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \mathbin{⨟} y \leftarrow [0,1]$$

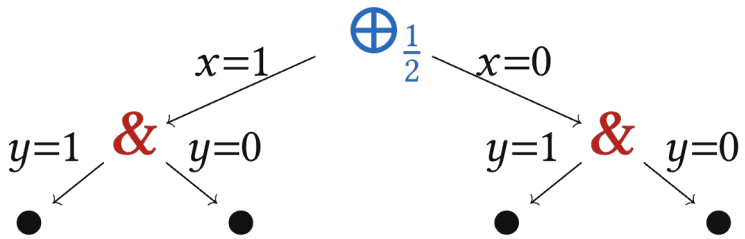


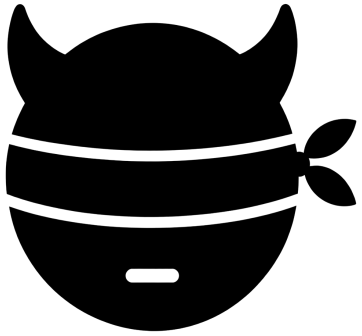
$$y \leftarrow [0,1] \mathbin{⨟} x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right)$$

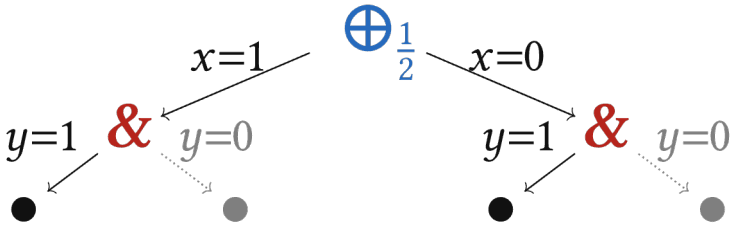


Example: Swapping Sampling and Choice

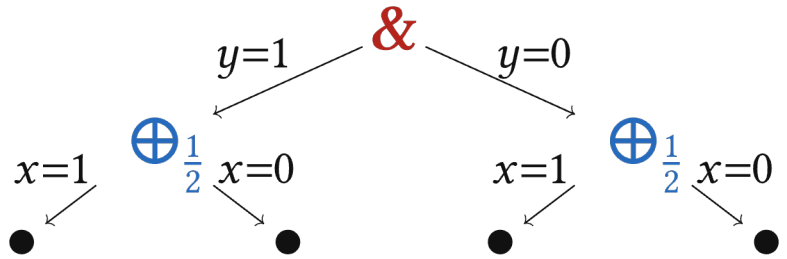
$x :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \mathbin{⨟} y \leftarrow [0,1]$

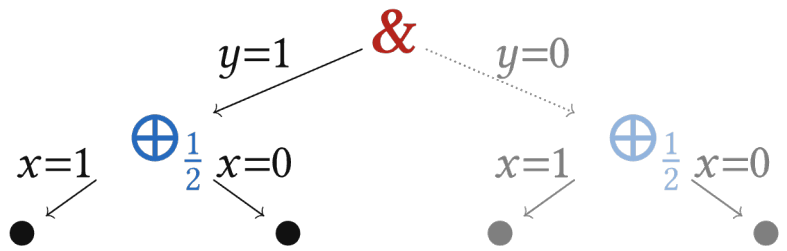






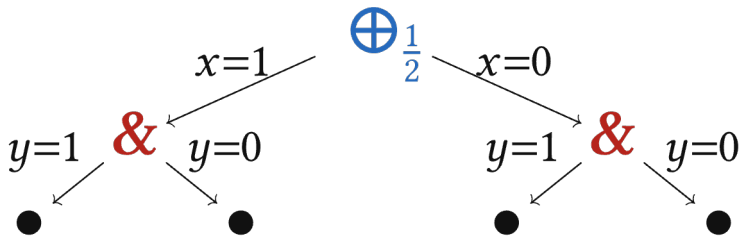
$y \leftarrow [0,1] \mathbin{⨟} x :\approx \mathbf{Ber}\left(\frac{1}{2}\right)$



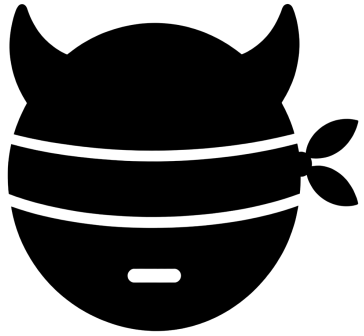


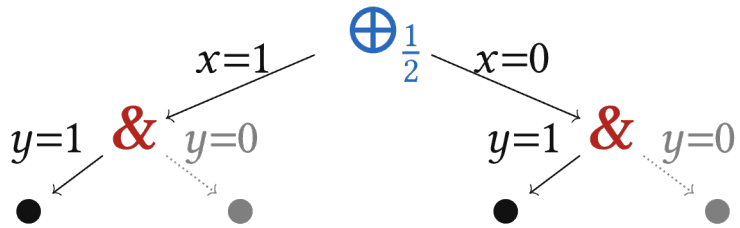
Example: Swapping Sampling and Choice

$x :\approx \mathbf{Ber}\left(\frac{1}{2}\right) \mathbin{⨟} y \leftarrow [0,1]$

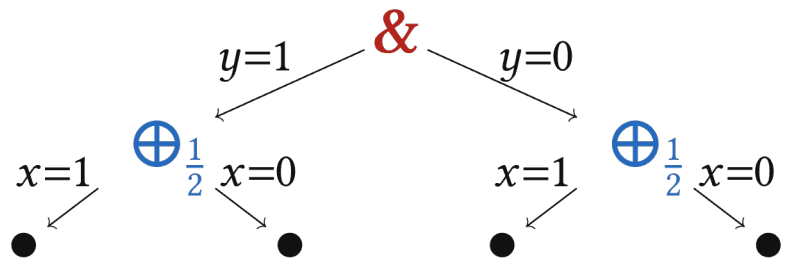


UI

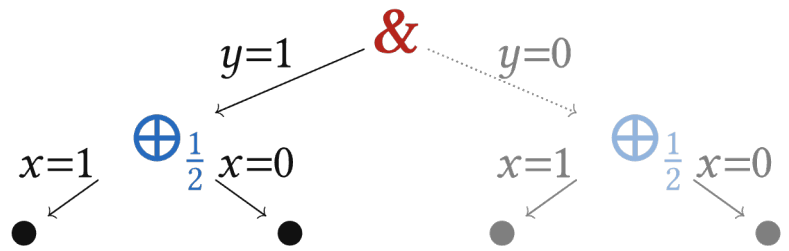




$y \leftarrow [0,1] \mathbin{⨟} x :\approx \mathbf{Ber}\left(\frac{1}{2}\right)$



UI



Semantics for Oblivious Adversary

$$\mathsf{Sched} \triangleq \mathbb{N} \to \mathbb{N}$$

Stream of Natural Number

$$\mathcal{O}(X) = \mathsf{Sched} \to \mathbb{N} \to \mathcal{D}_{\perp}(X \times \mathbb{N})$$

Distribution Monad

State Monad

Reader Monad

Oblivious Semantics  $\subseteq$ Demonic Semantics 

We are done!

Thank you for showing up!!

We are not done!? 🧐

Problem: Expressivity

$$\frac{\langle \lceil P \rceil \rangle\ C_1\ \langle \psi_1 \rangle \qquad \langle \lceil P \rceil \rangle\ C_2\ \langle \psi_2 \rangle}{\langle \lceil P \rceil \rangle\ C_1 \,\&\, C_2\ \langle \psi_1 \,\&\, \psi_2 \rangle}\ \textsc{Nondet}$$

P holds with probability 1 **∃ p ∈ [0, 1], $\psi_1 \oplus_p \psi_2$**

$$\frac{\langle \varphi_1 \rangle\ C\ \langle \psi_1 \rangle \qquad \langle \varphi_2 \rangle\ C\ \langle \psi_2 \rangle}{\langle \varphi_1 \oplus_p \varphi_2 \rangle\ C\ \langle \psi_1 \oplus_p \psi_2 \rangle}\ \textsc{Prob Split}$$

φ_1 holds with probability p, φ_2 holds with probability (1 – p)

Problem: Expressivity

$$\frac{\langle \lceil P \rceil \rangle\ C_1\ \langle \psi_1 \rangle \qquad \langle \lceil P \rceil \rangle\ C_2\ \langle \psi_2 \rangle}{\langle \lceil P \rceil \rangle\ C_1 \,\&\, C_2\ \langle \psi_1 \,\&\, \psi_2 \rangle}\ \textsc{Nondet}$$

P holds with probability 1 ∃ p ∈ [0, 1], $\psi_1 \oplus_p \psi_2$

$$\frac{\langle \varphi_1 \rangle\ C\ \langle \psi_1 \rangle \qquad \langle \varphi_2 \rangle\ C\ \langle \psi_2 \rangle}{\langle \varphi_1 \oplus_p \varphi_2 \rangle\ C\ \langle \psi_1 \oplus_p \psi_2 \rangle}\ \textsc{Prob Split}$$

φ_1 holds with probability p, φ_2 holds with probability (1 – p)

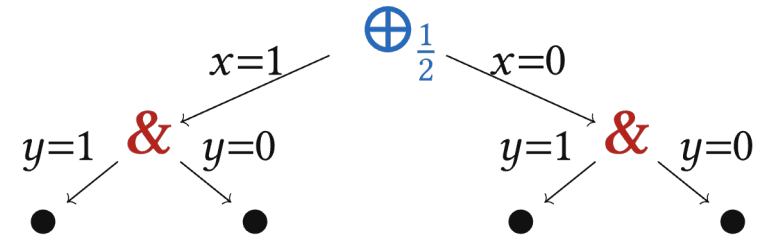
Problem: Expressivity

$$\dfrac{\dfrac{}{\langle \lceil x=0 \rceil \rangle\ y \leftarrow [0,1]\ \langle \lceil x=0 \rceil * (\lceil y=0 \rceil \,\&\, \lceil y=1 \rceil) \rangle}\ \textsc{NonDet} \quad \dfrac{}{\langle \lceil x=1 \rceil \rangle\ y \leftarrow [0,1]\ \langle \cdots \rangle}\ \textsc{NonDet}}{\langle \lceil x=0 \rceil \oplus_{\frac{1}{2}} \lceil x=1 \rceil \rangle\ y \leftarrow [0,1]\ \langle (\lceil x=0 \rceil * (\lceil y=0 \rceil \,\&\, \lceil y=1 \rceil)) \oplus_{\frac{1}{2}} (\lceil x=1 \rceil * (\lceil y=0 \rceil \,\&\, \lceil y=1 \rceil)) \rangle}\ \textsc{Prob-Split}$$

$$\not\Rightarrow$$

$$(\lceil y=0 \rceil \,\&\, \lceil y=1 \rceil) * (\lceil x=0 \rceil \oplus_{\frac{1}{2}} \lceil x=1 \rceil)$$

Precondition of NAssign requires probability 1 event!



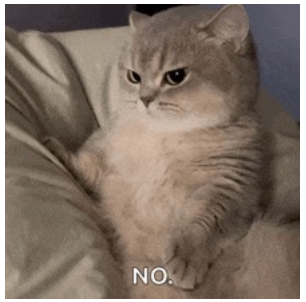
Execution tree from before

Problem: Expressivity

$$\frac{\langle \lceil P \rceil \rangle\ C_1\ \langle \psi_1 \rangle \qquad \langle \lceil P \rceil \rangle\ C_2\ \langle \psi_2 \rangle}{\langle \lceil P \rceil \rangle\ C_1 \,\&\, C_2\ \langle \psi_1 \,\&\, \psi_2 \rangle}\ \textsc{Nondet}$$

P holds with probability 1

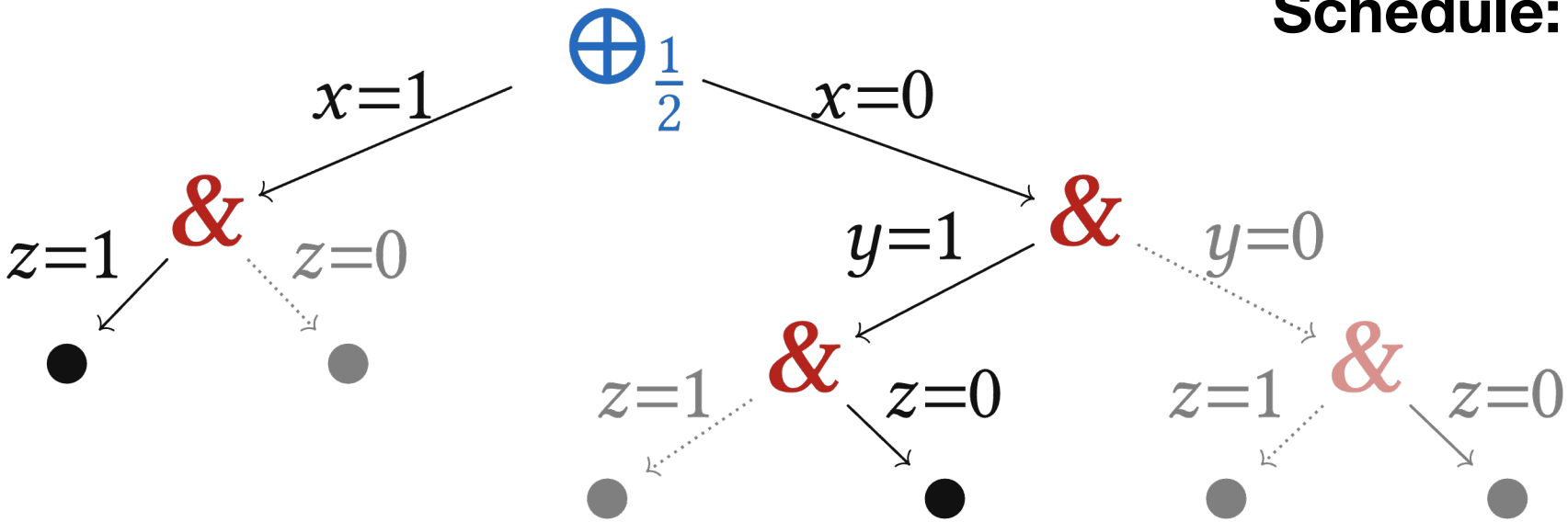
Can we make the precondition to be any assertion φ?



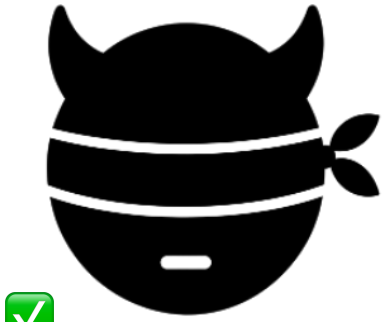
Implicit Leakage

$$x :\approx \mathbf{Ber}\left(\tfrac{1}{2}\right) \,;$$
$$(\texttt{if}\ x = 1\ \texttt{then skip else}\ y \leftarrow [0,1]) \,;$$
$$z \leftarrow [0,1]$$

Schedule: 1, 0, …

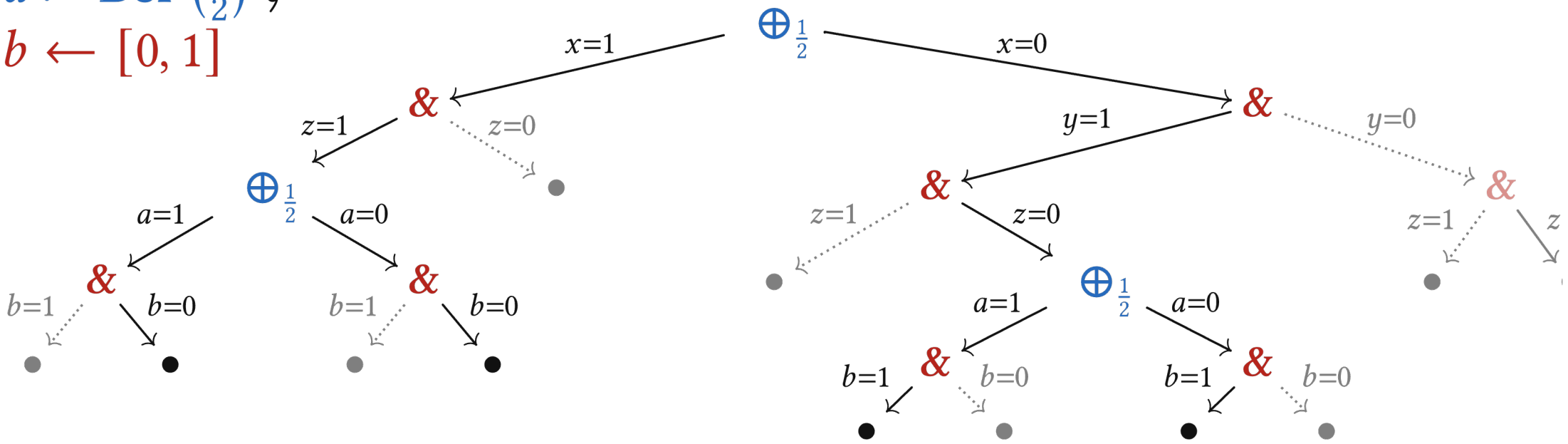


Implicit Leakage?



Can x correlate with b? ✅
Can a correlate with z or b? ❌

$x :\approx \mathrm{Ber}\left(\frac{1}{2}\right)$ ⨾
$(\texttt{if}\ x = 1\ \texttt{then skip else}\ y \leftarrow [0,1])$ ⨾
$z \leftarrow [0,1]$ ⨾
$a :\approx \mathrm{Ber}\left(\frac{1}{2}\right)$ ⨾
$b \leftarrow [0,1]$



Scheduling Consumption is Relevant

- A coin stays private when every branch induces the same distribution over adversary counts
- The tape pointer may move.
  - Privacy requires that it move the same way in every random branch

 random branch $\perp\!\!\!\perp$ adversary count

- Extending the resource of the assertion language to also track adversary consumption

$$\mathcal{P}(\Sigma) \quad \rightsquigarrow \quad \mathcal{P}(\Sigma \times \mathbb{N})$$ **State × Index**

Naïve Solution: Time Credit

- Prior work have invented time credits to account for amortized time complexity
- Idea: Create a resource algebra for annotating the number of bits consumed in each branch

$$\frac{\text{all branches of } \varphi \text{ have the same credit} \qquad \langle \varphi * \$1 \rangle\ C_1\ \langle \psi_1 \rangle \qquad \langle \varphi * \$1 \rangle\ C_2\ \langle \psi_2 \rangle}{\langle \varphi \rangle\ C_1 \ \&\ C_2\ \langle \psi_1 \ \&\ \psi_2 \rangle}\ \textsc{NonDet}$$

$$\varphi * \$0 \iff \varphi$$

$$\$n * \$m \iff \$(n+m)$$

$$\$n \oplus_p \$n \iff \$n$$

Our Solution: Privacy Assertion

$$\varphi \oplus_{p}^{\text{priv}} \psi$$

random branch must have the same adversary-count distribution.

Our Solution: Privacy Assertion

$$\varphi \oplus_p^{\text{priv}} \psi$$

random branch must have the same adversary-count distribution.

The probability space can be split into an independent product of a count space and a state space

$$\frac{\text{Private}(\varphi) \qquad \langle\varphi\rangle\ C_1\ \langle\psi_1\rangle \qquad \langle\varphi\rangle\ C_2\ \langle\psi_2\rangle}{\langle\varphi\rangle\ C_1\ \&\ C_2\ \langle\psi_1\ \&\ \psi_2\rangle}\ \text{ND}$$

Our Solution: Privacy Assertion

$$\frac{\mathsf{Private}(\lceil x = 0\rceil \oplus^{\mathsf{priv}}_{\frac{1}{2}} \lceil x = 1\rceil)}{\langle \lceil x = 0\rceil \oplus^{\mathsf{priv}}_{\frac{1}{2}} \lceil x = 1\rceil \rangle\ y \leftarrow [0,1]\ \langle \&_{Y\in\{0,1\}} \left((\lceil x = 0\rceil \oplus^{\mathsf{priv}}_{\frac{1}{2}} \lceil x = 1\rceil) * \lceil y \mapsto Y\rceil\right)\rangle}\ \mathsf{ND}$$

$$\Longrightarrow$$

$$(\lceil y = 0\rceil \,\&\, \lceil y = 1\rceil) * \left(\lceil x = 0\rceil \oplus^{\mathsf{priv}}_{\frac{1}{2}} \lceil x = 1\rceil\right)$$

$$\frac{\mathsf{Private}(\varphi) \qquad \langle\varphi\rangle\ C_1\ \langle\psi_1\rangle \qquad \langle\varphi\rangle\ C_2\ \langle\psi_2\rangle}{\langle\varphi\rangle\ C_1 \,\&\, C_2\ \langle\psi_1 \,\&\, \psi_2\rangle}\ \mathsf{ND}$$

Separation

- Frame rule is unsound around a nondeterministic program.

$$\frac{\langle \lceil y \mapsto - \rceil \rangle\ y \leftarrow [0,1]\ \langle \&_{Y \in \{0,1\}} \lceil y \mapsto Y \rceil \rangle}{\langle x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) * \lceil y \mapsto - \rceil \rangle\ y \leftarrow [0,1]\ \langle x \sim \mathbf{Ber}\left(\tfrac{1}{2}\right) * \left(\&_{Y \in \{0,1\}} \lceil y \mapsto Y \rceil\right) \rangle}\ \textsc{Frame}\ \times$$

Adversary's guess is not independent if the coin is fixed first

Separation

- Frame rule is sound using the privacy assertion.

$$\frac{\mathsf{Private}(x \sim^{\mathsf{priv}} \mathbf{Ber}\left(\frac{1}{2}\right)) \quad \langle \lceil y \mapsto - \rceil \rangle\ y \leftarrow [0,1]\ \langle \&_{Y \in \{0,1\}} \lceil y \mapsto Y \rceil \rangle}{\langle x \sim^{\mathsf{priv}} \mathbf{Ber}\left(\frac{1}{2}\right) * \lceil y \mapsto - \rceil \rangle\ y \leftarrow [0,1]\ \langle x \sim^{\mathsf{priv}} \mathbf{Ber}\left(\frac{1}{2}\right) * (\&_{Y \in \{0,1\}} \lceil y \mapsto Y \rceil) \rangle}\ \textsc{Frame}$$

✅

Example: Sequential Leader Election

$$
\begin{array}{l}
a \coloneqq 0 \mathbin{⨟} b \coloneqq 0 \mathbin{⨟} \\
\textbf{while } |a - b| \leq 1 \textbf{ do} \\
\quad (a :\approx \mathbf{Unif}\,(a, a+1)) \;\&\; (b :\approx \mathbf{Unif}\,(b, b+1))
\end{array}
$$

Does the above program terminates?

Example: Sequential Leader Election

$$\langle \lceil a \mapsto - \rceil * \lceil b \mapsto - \rceil \rangle$$

$a \coloneqq 0 \,;\, b \coloneqq 0 \,;$

while $|a - b| \leq 1$ **do**

$(a :\approx \mathbf{Unif}\,(a, a+1)) \;\&\; (b :\approx \mathbf{Unif}\,(b, b+1))$

$$\langle \lceil a - b \mapsto 2 \rceil \;\&\; \lceil b - a \mapsto 2 \rceil \rangle$$

Does the above program terminates?

Prove Termination ⇔ Prove opOL triple holds

Termination Proof Sketch

- Intuition: For any iteration, the probability that the protocol terminates within 3 rounds is $\geq \frac{1}{8}$, so probability for non-termination decay exponentially (under a fixed schedule)

Termination Proof Sketch

- Say we start with a = b
  - In any oblivious schedule, either Alice or Bob runs at least twice, e.g., AAB
  - In AAB.., Alice advances twice with probability ¼ and Bob does not advance with probability ½
  - After 3 rounds, Alice wins with probability 1/8

Termination Proof Sketch

- Say we start with a = b
  - In any oblivious schedule, either Alice or Bob runs at least twice, e.g., AAB
  - In AAB.., Alice advances twice with probability ¼ and Bob does not advance with probability ½
  - After 3 rounds, Alice wins with probability 1/8
- The case when a ≠ b is similar

✅ **Proof is verified in Lean via opOL**

Mechanization

- This paper is fully mechanized in Lean
  - Semantics + Refinement Theorem
  - Probability Space – An improvement from Mathlib
  - Assertion Language
  - Proof Rules
  - Case Studies:
    - The Monty Hall Problem (generalized from the demonic outcome logic)
    - Leader Election
    - Paging
- ~ 30K LOC (ongoing optimization)
- Mechanized with AI's help in ≤ 1 months. After many months of pen-and-paper proof.

Ongoing Work

- Concurrency
  - Adversary choices arise at every interleaving step.
  - Can adversary consumption support a sound parallel-composition rule?
- Usable Lean Repository
  - Improve proof automation for routine obligations (e.g., arithmetic side condition) that currently require tedious helper lemmas.
  - Explore Iris-Lean integration for proof-mode support.
- More than one adversaries? Adversaries that get partial information?