Post-quantum zero-knowledge proofs with constant-depth quantum simulators

Gaspard Damoiseau-Malraux

Sorbonne Université

March 11, 2026

Research work under the supervision of Alex B. Grilo & Damien Vergnaud

Introduction

Zero-knowledge proof: interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. [GMR85]

Introduction

Zero-knowledge proof: interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. [GMR85]

1. **Completeness:** if the statement $x \in \mathbb{L}$, then $\mathcal{V}$ accepts;

Introduction

Zero-knowledge proof: interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. [GMR85]

1. **Completeness:** if the statement $x \in \mathbb{L}$, then $\mathcal{V}$ accepts;
2. **Soundness:** if the statement $x \notin \mathbb{L}$, then $\mathcal{V}$ does not accept;

Introduction

Zero-knowledge proof: interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. [GMR85]

1. **Completeness:** if the statement $x \in \mathbb{L}$, then $\mathcal{V}$ accepts;
2. **Soundness:** if the statement $x \notin \mathbb{L}$, then $\mathcal{V}$ does not accept;
3. **Zero-knowledge:** (informal) After executing of the protocol, the verifier did not learn anything that they could not already compute before.

Introduction

Zero-knowledge proof: interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. [GMR85]

1. **Completeness:** if the statement $x \in \mathbb{L}$, then $\mathcal{V}$ accepts;
2. **Soundness:** if the statement $x \notin \mathbb{L}$, then $\mathcal{V}$ does not accept;
3. **Zero-knowledge:** (informal) After executing of the protocol, the verifier did not learn anything that they could not already compute before.

$\rightarrow$ To define 3 , we must exhibit a simulator for the protocol.

Introduction

Zero-knowledge proof: interactive protocol between a prover $\mathcal{P}$ and verifier $\mathcal{V}$. [GMR85]

1. **Completeness:** if the statement $x \in \mathbb{L}$, then $\mathcal{V}$ accepts;
2. **Soundness:** if the statement $x \notin \mathbb{L}$, then $\mathcal{V}$ does not accept;
3. **Zero-knowledge:** (informal) After executing of the protocol, the verifier did not learn anything that they could not already compute before.

$\rightarrow$ To define 3 , we must exhibit a simulator for the protocol.

Definition

A **simulator** S for a protocol Π is a polytime probabilistic algorithm with black-box access to $\mathcal{V}$, and which output is indistinguishable from the transcript of a real execution of Π.

Introduction: goal

Classical adversary (=verifier)

It is possible to construct efficient simulators in terms of memory or time

Introduction: goal

Classical adversary (=verifier)

It is possible to construct efficient simulators in terms of memory or time

Adversary (=verifier) with access to a quantum computer

...it's more complicated

Introduction: goal

Classical adversary (=verifier)

It is possible to construct efficient simulators in terms of memory or time

Adversary (=verifier) with access to a quantum computer

...it's more complicated

- We know post-quantum ZK protocols, **but** with a complexity overhead on the simulator; $\rightarrow$ ongoing efforts to improve that.

Introduction: goal

Classical adversary (=verifier)

It is possible to construct efficient simulators in terms of memory or time

Adversary (=verifier) with access to a quantum computer

...it's more complicated

- We know post-quantum ZK protocols, **but** with a complexity overhead on the simulator; $\rightarrow$ ongoing efforts to improve that.

What kind of languages admit ZK protocols for which there exists a constant-depth simulator?

Introduction: goal

Classical adversary (=verifier)

It is possible to construct efficient simulators in terms of memory or time

Adversary (=verifier) with access to a quantum computer

...it's more complicated

- We know post-quantum ZK protocols, **but** with a complexity overhead on the simulator; $\rightarrow$ ongoing efforts to improve that.

What kind of languages admit ZK protocols for which there exists a constant-depth simulator?

Conjecture

Let $\mathbb{L}$ be a language that admits a ZK protocol with a constant-depth simulator; then, $\mathbb{L} \in \mathbf{BQP}$.[a]

[a] BQP = Bounded-error Quantum Polynomial time

Some tools needed

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])
  - S is a random subspace of $\mathbb{F}_2^n$

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])
  - S is a random subspace of $\mathbb{F}_2^n$
  - $|S\rangle = \sum_{x \in S} \frac{1}{\sqrt{2^{|S|}}} |x\rangle$ (uniform superposition of all states of S)

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])
  - S is a random subspace of $\mathbb{F}_2^n$
  - $|S\rangle = \sum_{x \in S} \frac{1}{\sqrt{2^{|S|}}} |x\rangle$ (uniform superposition of all states of S)
  - Easy to make if S is known;

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])
  - S is a random subspace of $\mathbb{F}_2^n$
  - $|S\rangle = \sum_{x \in S} \frac{1}{\sqrt{2^{|S|}}} |x\rangle$ (uniform superposition of all states of S)
  - Easy to make if S is known;
  - Otherwise exponentially hard to clone.

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])
  - S is a random subspace of $\mathbb{F}_2^n$
  - $|S\rangle = \sum_{x \in S} \frac{1}{\sqrt{2^{|S|}}} |x\rangle$ (uniform superposition of all states of S)
  - Easy to make if S is known;
  - Otherwise exponentially hard to clone.
  - Sometimes called 'quantum coin'

Some tools needed

- No-Cloning Theorem : there cannot exist a method $\mathcal{M}$ capable of cloning an arbitrary quantum state $|\Phi\rangle$.
  - $\rightarrow$ $\mathcal{M}(|\Phi\rangle \otimes |0\rangle) = |\Phi\rangle \otimes |\Phi\rangle$ **does not exist.**

- Subspace states $|S\rangle$ (Aaronson and Christiano [AC12])
  - S is a random subspace of $\mathbb{F}_2^n$
  - $|S\rangle = \sum_{x \in S} \frac{1}{\sqrt{2^{|S|}}} |x\rangle$ (uniform superposition of all states of S)
  - Easy to make if S is known;
  - Otherwise exponentially hard to clone.
  - Sometimes called 'quantum coin'

- Exact definition of black-box zero-knowledge:
  - $\rightarrow$ A ZK black-box protocol $\implies$ there exists a simulator *Sim* such that **for all verifiers**, Output$(Sim(\mathcal{V})) \approx$ Transcript$(\mathcal{P} \leftrightarrow \mathcal{V})$

Contributions: Lemma 1

→ Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim

Contributions: Lemma 1

→ Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim

→ $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)

Contributions: Lemma 1

- → Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim
- → $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)
- → Applying $U_i |\Phi\rangle$ corresponds to computing the i-th action of $\mathcal{V}$ on $|\Phi\rangle$

Contributions: Lemma 1

- → Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim
- → $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)
- → Applying $U_i |\Phi\rangle$ corresponds to computing the i-th action of $\mathcal{V}$ on $|\Phi\rangle$

Let's build a 2^{nd} verifier $\mathcal{V}' = (U'_1, ..., U'_M)$ that will execute the protocol.

Contributions: Lemma 1

- → Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim
- → $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)
- → Applying $U_i |\Phi\rangle$ corresponds to computing the i-th action of $\mathcal{V}$ on $|\Phi\rangle$

Let's build a 2nd verifier $\mathcal{V}' = (U'_1, ..., U'_M)$ that will execute the protocol.

$$U'_i = |S_{i+1}\rangle\langle S_i| \otimes U_i + |S_i\rangle\langle S_{i+1}| \otimes U_i^\dagger + \sum_{\substack{x \\ |x\rangle\perp|S_i\rangle \\ |x\rangle\perp|S_{i+1}\rangle}} |x\rangle\langle x| \otimes \mathbb{1}$$

Contributions: Lemma 1

- → Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim
- → $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)
- → Applying $U_i |\Phi\rangle$ corresponds to computing the i-th action of $\mathcal{V}$ on $|\Phi\rangle$

Let's build a 2nd verifier $\mathcal{V}' = (U'_1, ..., U'_M)$ that will execute the protocol.

$$U_i(|\Phi\rangle, |\Psi\rangle) = \left\{ \begin{array}{ll} |S_{i+1}\rangle, U_i |\Psi\rangle & \text{if } |\Phi\rangle = |S_i\rangle \\ |S_i\rangle, U_i^\dagger |\Psi\rangle & \text{if } |\Phi\rangle = |S_{i+1}\rangle \\ |\Phi\rangle, |\Psi\rangle & \text{otherwise} \end{array} \right\}$$

Contributions: Lemma 1

- $\rightarrow$ Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim
- $\rightarrow$ $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)
- $\rightarrow$ Applying $U_i |\Phi\rangle$ corresponds to computing the i-th action of $\mathcal{V}$ on $|\Phi\rangle$

Let's build a 2nd verifier $\mathcal{V}' = (U'_1, ..., U'_M)$ that will execute the protocol.

$$U_i(|\Phi\rangle, |\Psi\rangle) = \left\{ \begin{array}{ll} |S_{i+1}\rangle, U_i |\Psi\rangle & \text{if } |\Phi\rangle = |S_i\rangle \\ |S_i\rangle, U_i^\dagger |\Psi\rangle & \text{if } |\Phi\rangle = |S_{i+1}\rangle \\ |\Phi\rangle, |\Psi\rangle & \text{otherwise} \end{array} \right\}$$

By design

Sim is also a simulator for $\mathcal{V}'$ (remind the definition of black-box ZK).

Contributions: Lemma 1

- → Lets suppose that we have $(\mathcal{P}, \mathcal{V}, Sim)$, and let d be the constant depth of Sim
- → $\mathcal{V} = (U_1, ..., U_M)$ with U_i unitaries (i.e. $U_i U_i^\dagger = U_i^\dagger U_i = \mathbb{1}$)
- → Applying $U_i |\Phi\rangle$ corresponds to computing the i-th action of $\mathcal{V}$ on $|\Phi\rangle$

Let's build a 2nd verifier $\mathcal{V}' = (U'_1, ..., U'_M)$ that will execute the protocol.

$$U_i(|\Phi\rangle, |\Psi\rangle) = \left\{ \begin{array}{ll} |S_{i+1}\rangle, U_i |\Psi\rangle & \text{if } |\Phi\rangle = |S_i\rangle \\ |S_i\rangle, U_i^\dagger |\Psi\rangle & \text{if } |\Phi\rangle = |S_{i+1}\rangle \\ |\Phi\rangle, |\Psi\rangle & \text{otherwise} \end{array} \right\}$$

By design

Sim is also a simulator for $\mathcal{V}'$ (remind the definition of black-box ZK).

Lemma 1

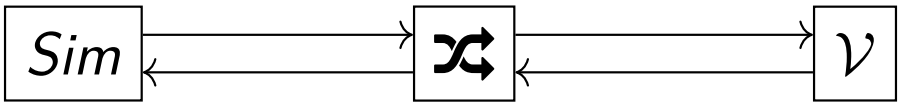
Transcript$(\mathcal{P} \leftrightarrow \mathcal{V}') \approx$ Transcript$(\mathcal{P} \leftrightarrow \mathcal{V})$ **and** Output$(Sim(\mathcal{V}')) \approx$ Output$(Sim(\mathcal{V}))$

Contributions: a second simulator

Let us now build a second simulator Sim_2 for $\mathcal{V}$, where Sim_2 takes care of doing the additional actions $\mathcal{V}'$.

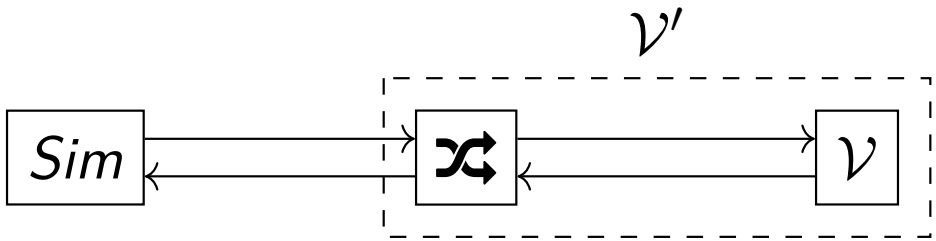
Contributions: a second simulator

Let us now build a second simulator Sim_2 for $\mathcal{V}$, where Sim_2 takes care of doing the additional actions $\mathcal{V}'$.



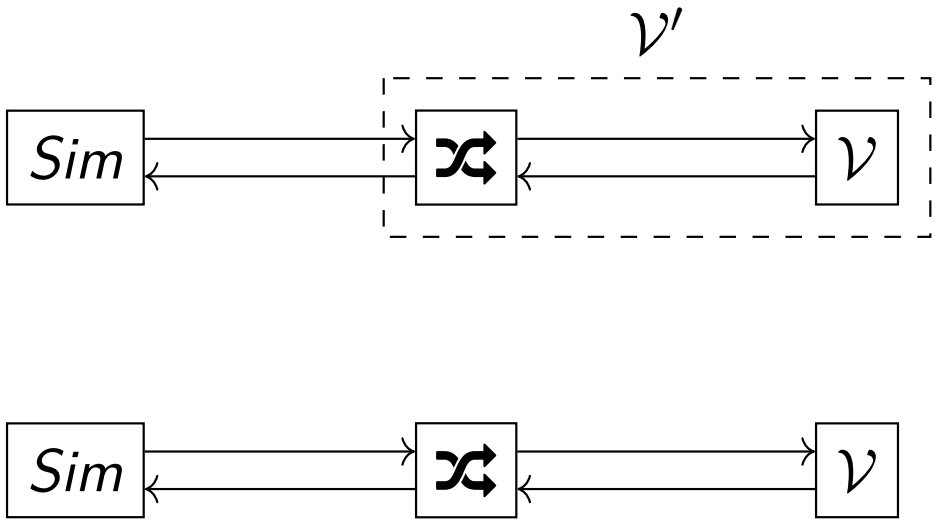
Contributions: a second simulator

Let us now build a second simulator Sim_2 for $\mathcal{V}$, where Sim_2 takes care of doing the additional actions $\mathcal{V}'$.



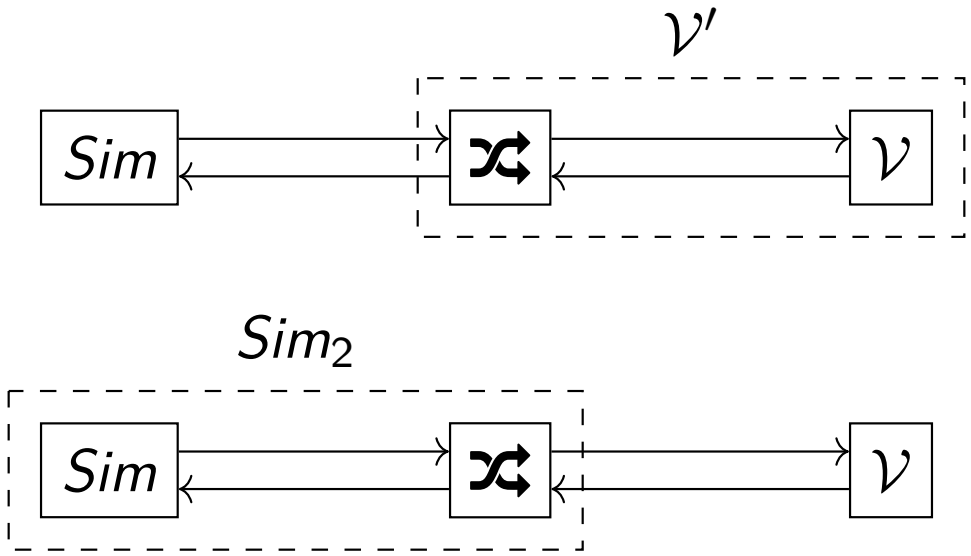
Contributions: a second simulator

Let us now build a second simulator Sim_2 for $\mathcal{V}$, where Sim_2 takes care of doing the additional actions $\mathcal{V}'$.



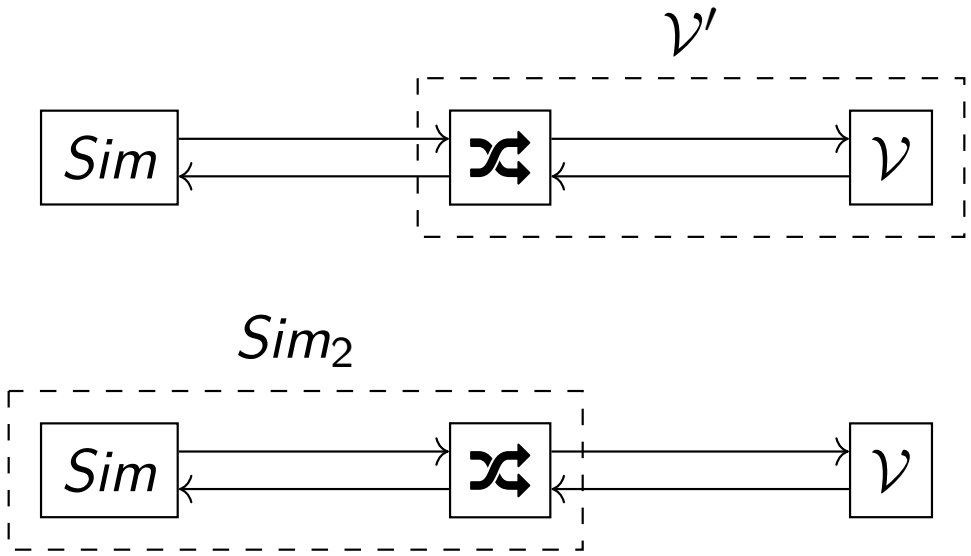
Contributions: a second simulator

Let us now build a second simulator Sim_2 for $\mathcal{V}$, where Sim_2 takes care of doing the additional actions $\mathcal{V}'$.



Contributions: a second simulator

Let us now build a second simulator Sim_2 for $\mathcal{V}$, where Sim_2 takes care of doing the additional actions $\mathcal{V}'$.



Consequence

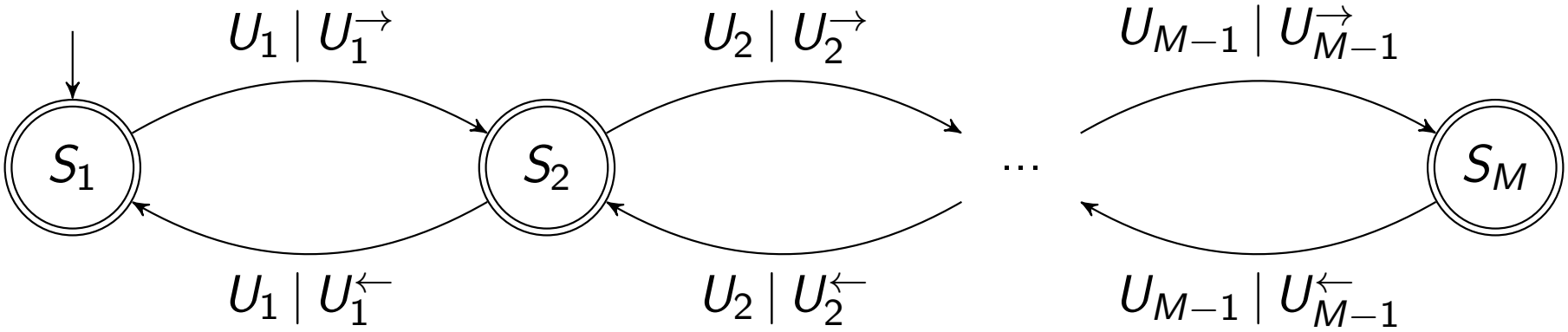
$\text{Output}(Sim_2(\mathcal{V})) = \text{Output}(Sim(\mathcal{V}'))$

Contributions: Lemma 2

Let us now build Sim_3, based on Sim_2 with an additional twist.

Contributions: Lemma 2

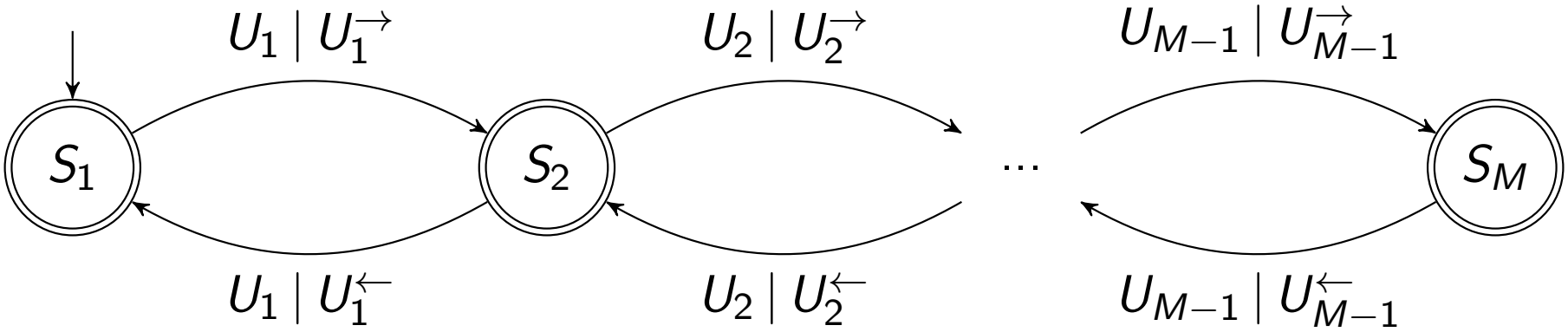
Let us now build Sim_3, based on Sim_2 with an additional twist.



Finite State Machine (FSM) used to decide which $U_i^{\leftrightarrows}$ is being used.

Contributions: Lemma 2

Let us now build Sim_3, based on Sim_2 with an additional twist.

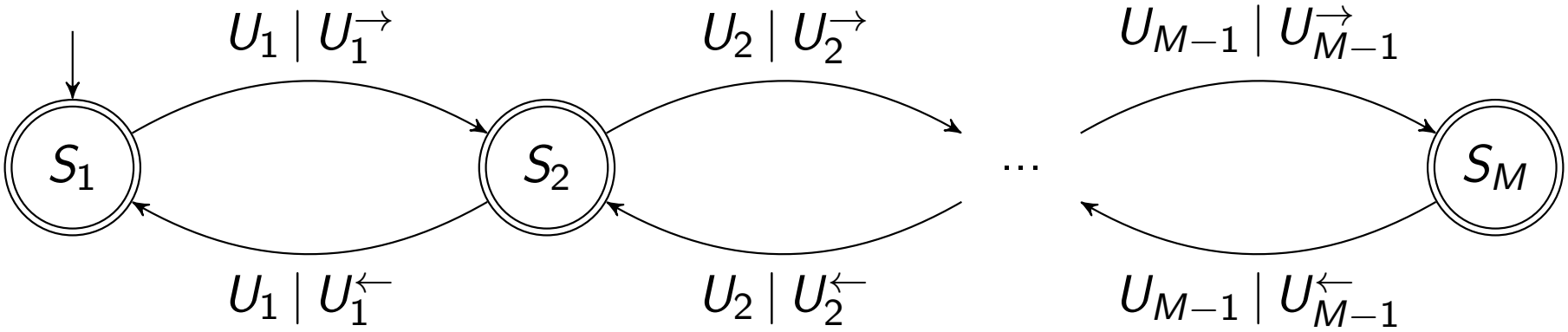


Finite State Machine (FSM) used to decide which $U_i^{\leftrightarrows}$ is being used.

- If Sim_3 applies $U_i^{\rightarrow}$, it projects the subspace state on $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$

Contributions: Lemma 2

Let us now build Sim_3, based on Sim_2 with an additional twist.

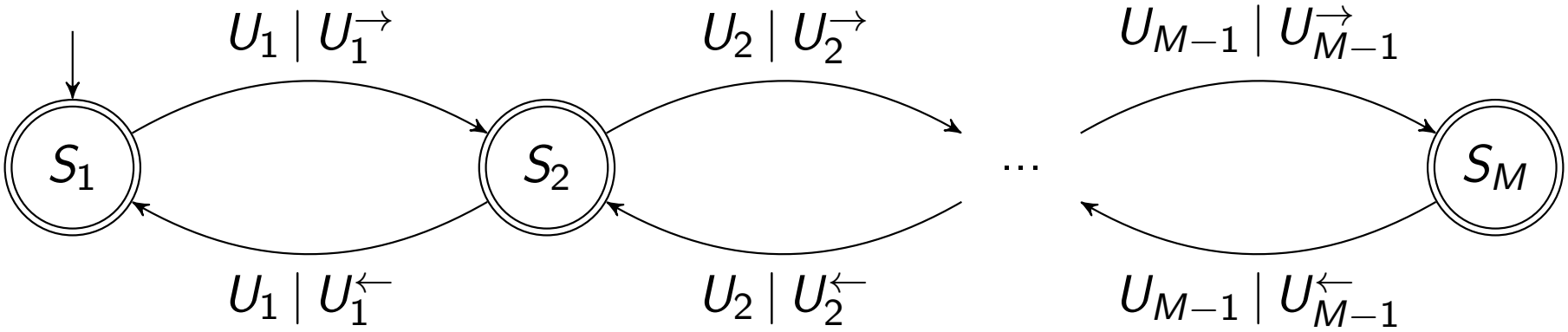


Finite State Machine (FSM) used to decide which $U_i^{\leftrightarrows}$ is being used.

- If Sim_3 applies $U_i^{\rightarrow}$, it projects the subspace state on $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$, it projects on $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$ instead.

Contributions: Lemma 2

Let us now build Sim_3, based on Sim_2 with an additional twist.



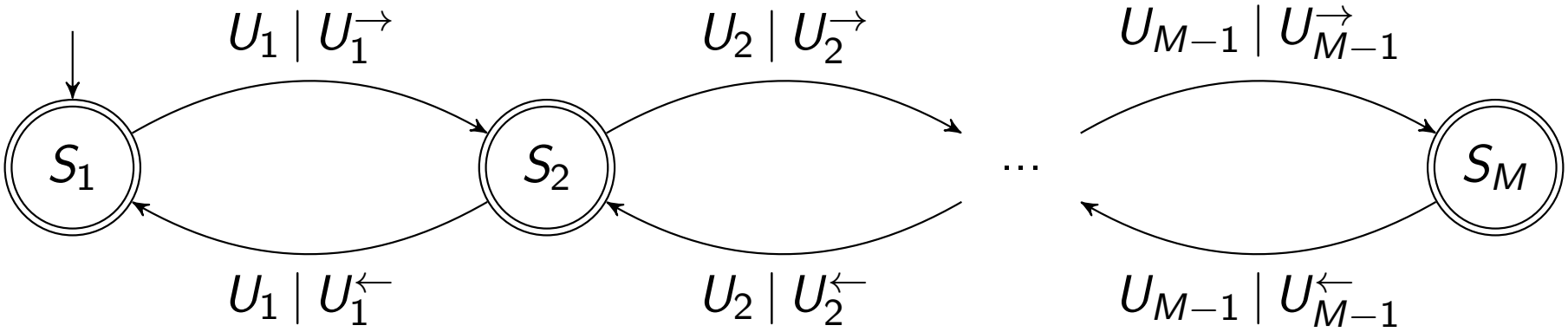
Finite State Machine (FSM) used to decide which $U_i^{\leftrightarrows}$ is being used.

- If Sim_3 applies $U_i^{\rightarrow}$, it projects the subspace state on $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$, it projects on $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$ instead.

$\rightarrow$ **Deterministic;**

Contributions: Lemma 2

Let us now build Sim_3, based on Sim_2 with an additional twist.



Finite State Machine (FSM) used to decide which $U_i^{\leftrightarrows}$ is being used.

- If Sim_3 applies $U_i^{\rightarrow}$, it projects the subspace state on $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$, it projects on $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$ instead.

$\rightarrow$ **Deterministic;**

$\rightarrow$ **Every $|S_i\rangle$ or $|S_{i+1}\rangle$ given to the adversary at instant t must be given back at instant $t+1$.**

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Let us prove it by contradiction:

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Let us prove it by contradiction:

1. The word π that corresponds to the sequence of gates executed is not in the language that the FSM accepts.

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Let us prove it by contradiction:

1. The word π that corresponds to the sequence of gates executed is not in the language that the FSM accepts.
   $\rightarrow$ Example: $U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_3^{\leftarrow} U_3^{\rightarrow} U_8^{\rightarrow}$

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Let us prove it by contradiction:

1. The word π that corresponds to the sequence of gates executed is not in the language that the FSM accepts.
   → Example: $U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_3^{\leftarrow} U_3^{\rightarrow} U_8^{\rightarrow}$
2. At least one projection associated to a $U_i^{\leftrightarrows}$ fails, with nonnegligible probability.

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Let us prove it by contradiction:

1. The word π that corresponds to the sequence of gates executed is not in the language that the FSM accepts.
   → Example: $U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_3^{\leftarrow} U_3^{\rightarrow} U_8^{\rightarrow}$
2. At least one projection associated to a $U_i^{\leftrightarrows}$ fails, with nonnegligible probability.
   → Example: $|S_3\rangle$ is provided to $U_3^{\leftarrow}$ ($U_3^{\leftarrow}$ only 'accepts' $|S_4\rangle$).

Contributions: proof of indistinguishability

Lemme 2

$Sim_3(\mathcal{V}) \approx Sim_2(\mathcal{V})$

Let us prove it by contradiction:

1. The word π that corresponds to the sequence of gates executed is not in the language that the FSM accepts.
   → Example: $U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_3^{\leftarrow} U_3^{\rightarrow} U_8^{\rightarrow}$
2. At least one projection associated to a $U_i^{\leftrightarrows}$ fails, with nonnegligible probability.
   → Example: $|S_3\rangle$ is provided to $U_3^{\leftarrow}$ ($U_3^{\leftarrow}$ only 'accepts' $|S_4\rangle$).

For both case, we can build $\mathcal{A}$, which takes a quantum coin $|\$\rangle$ as input, and returns $|\$\rangle \otimes |\$\rangle$ with nonnegligible probability.

Contributions: intuition of proof

Example, case 2: $\pi \in \mathcal{L}(FSM)$, one projection associated to a $U_i^{\leftrightarrows}$ failed.

Contributions: intuition of proof

Example, case 2: $\pi \in \mathcal{L}(FSM)$, one projection associated to a $U_i^{\leftrightarrows}$ failed.

Reminder

- If Sim_3 applies $U_i^{\rightarrow}$: it projects onto $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$: it projects onto $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$

Contributions: intuition of proof

Example, case 2: $\pi \in \mathcal{L}(FSM)$, one projection associated to a $U_i^{\leftrightarrows}$ failed.

Reminder

- If Sim_3 applies $U_i^{\rightarrow}$: it projects onto $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$: it projects onto $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$

Two observations if $U_i^{\leftrightarrows}$ **failed**:

Contributions: intuition of proof

Example, case 2: $\pi \in \mathcal{L}(FSM)$, one projection associated to a $U_i^{\leftrightarrows}$ failed.

Reminder

- If Sim_3 applies $U_i^{\rightarrow}$: it projects onto $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$: it projects onto $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$

Two observations if $U_i^{\leftrightarrows}$ **failed**:

$\rightarrow$ If $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$, then the input subspace state $|\Psi\rangle$ can only be $|S_{i+1}\rangle$

Contributions: intuition of proof

Example, case 2: $\pi \in \mathcal{L}(FSM)$, one projection associated to a $U_i^{\leftrightarrows}$ failed.

Reminder

- If Sim_3 applies $U_i^{\rightarrow}$: it projects onto $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$: it projects onto $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$

Two observations if $U_i^{\leftrightarrows}$ **failed**:

$\rightarrow$ If $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$, then the input subspace state $|\Psi\rangle$ can only be $|S_{i+1}\rangle$

$\rightarrow$ If $U_i^{\leftrightarrows}$ is $U_i^{\leftarrow}$, then the input subspace state $|\Psi\rangle$ can only be $|S_i\rangle$

Contributions: intuition of proof

Example, case 2: $\pi \in \mathcal{L}(FSM)$, one projection associated to a $U_i^{\leftrightarrows}$ failed.

Reminder

- If Sim_3 applies $U_i^{\rightarrow}$: it projects onto $\{|S_i\rangle\langle S_i|, \mathbb{1} - |S_i\rangle\langle S_i|\}$
- If Sim_3 applies $U_i^{\leftarrow}$: it projects onto $\{|S_{i+1}\rangle\langle S_{i+1}|, \mathbb{1} - |S_{i+1}\rangle\langle S_{i+1}|\}$

Two observations if $U_i^{\leftrightarrows}$ **failed**:

$\rightarrow$ If $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$, then the input subspace state $|\Psi\rangle$ can only be $|S_{i+1}\rangle$

$\rightarrow$ If $U_i^{\leftrightarrows}$ is $U_i^{\leftarrow}$, then the input subspace state $|\Psi\rangle$ can only be $|S_i\rangle$

In both subcases, $\mathcal{A}$ can clone any arbitrary quantum coin:
$\mathcal{A}(|\$\rangle \otimes |0\rangle) = |\$\rangle \otimes |\$\rangle$.

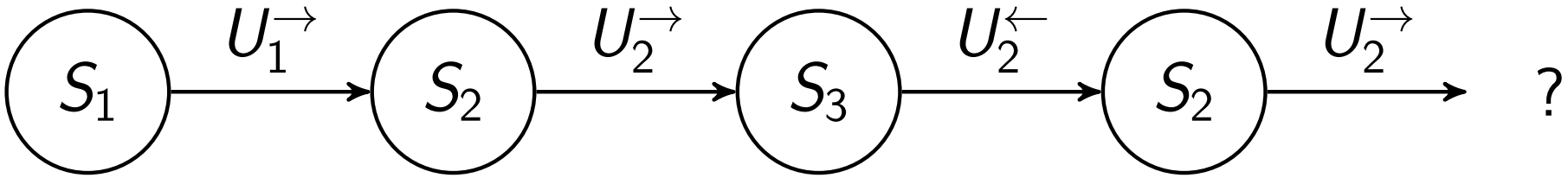
Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

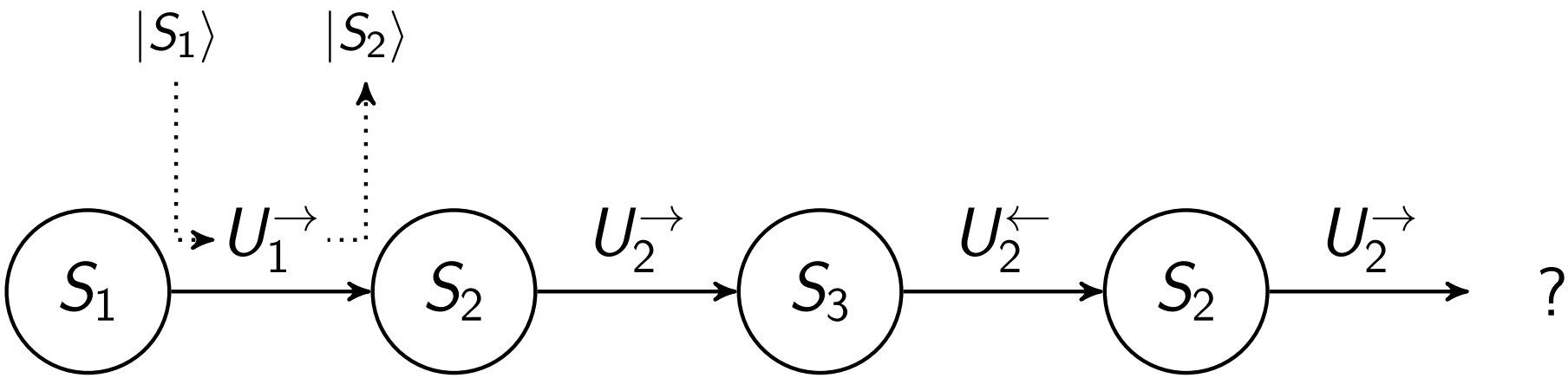
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$



Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

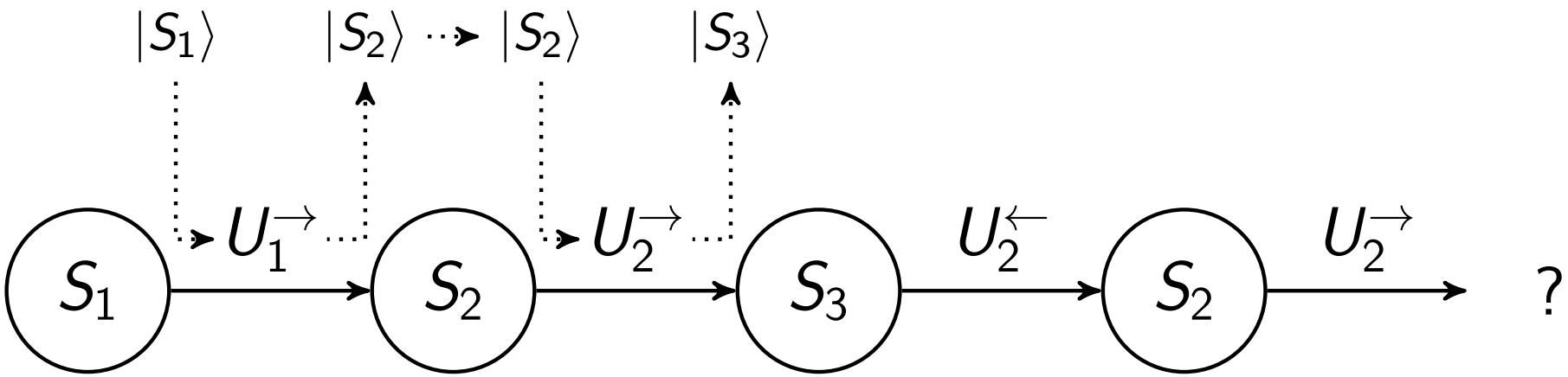
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$



Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

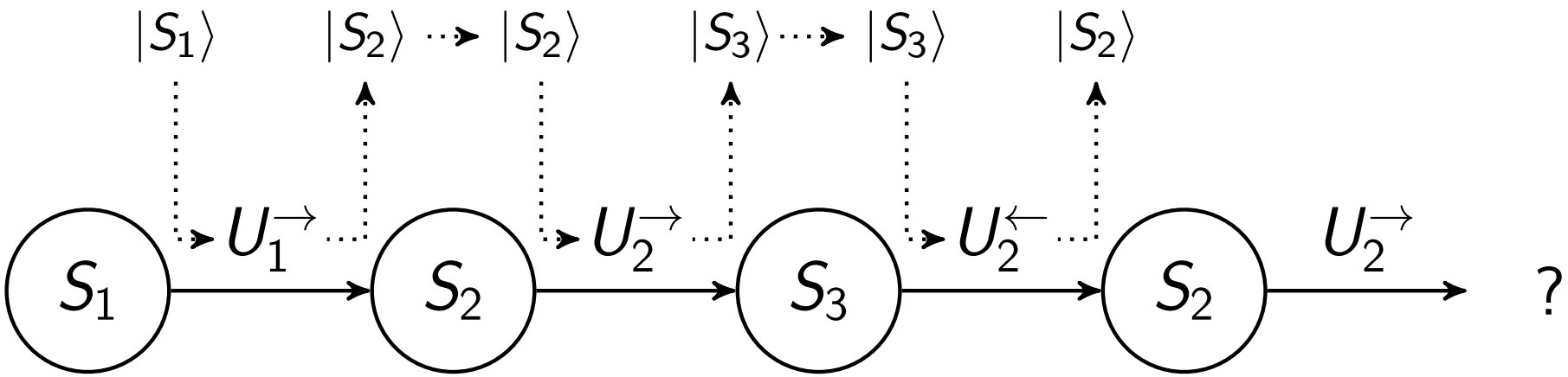
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$



Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

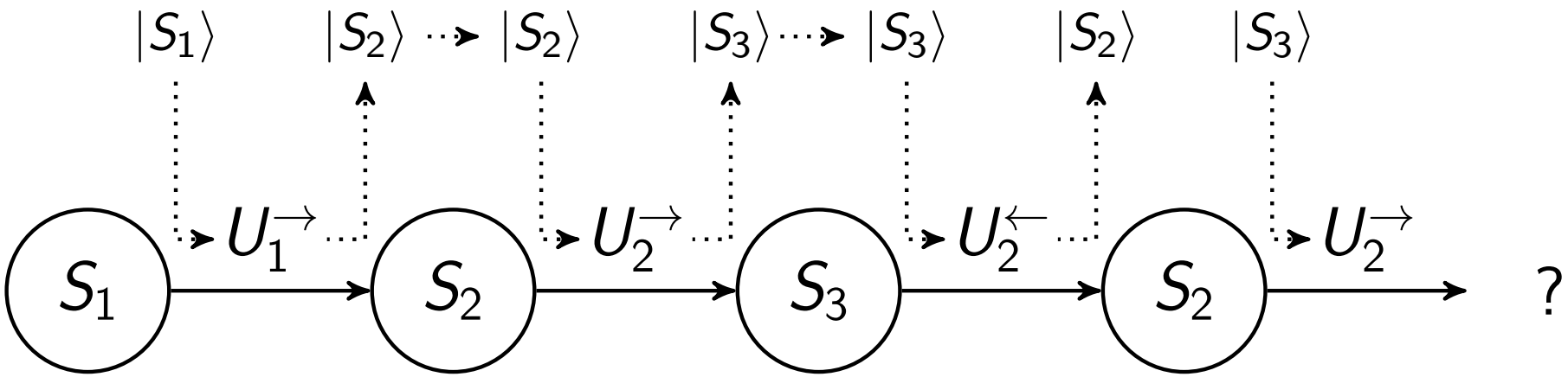
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$



Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

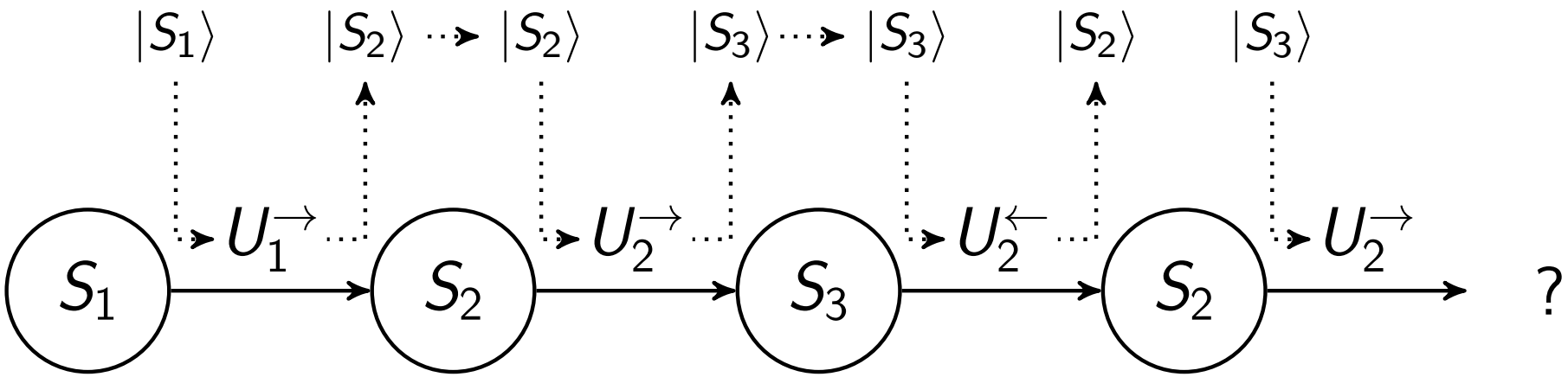
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$



Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$

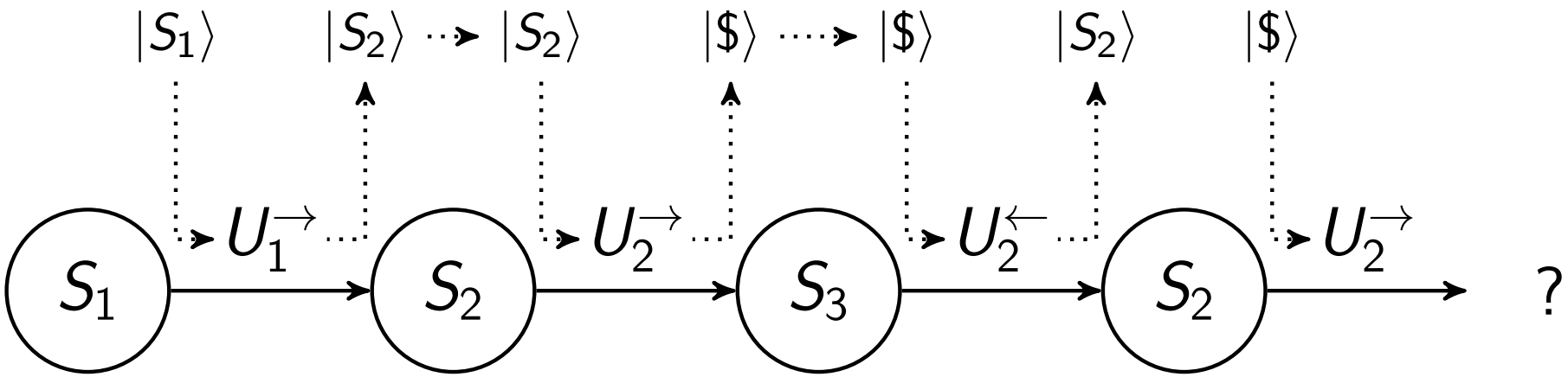


We replace $|S_3\rangle$ with the quantum coin $|\$\rangle$.

Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$

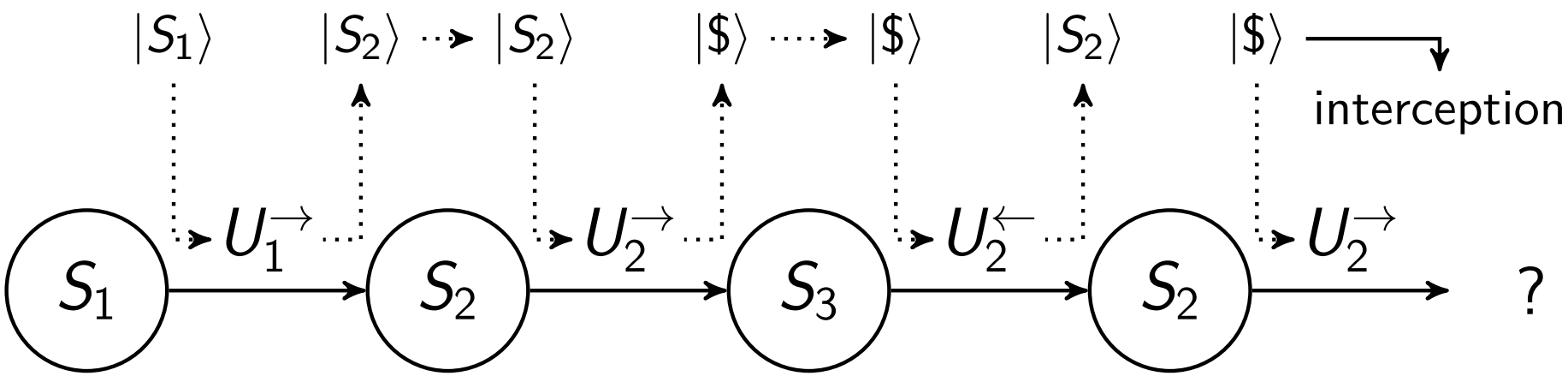


We replace $|S_3\rangle$ with the quantum coin $|\$\rangle$.

Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$

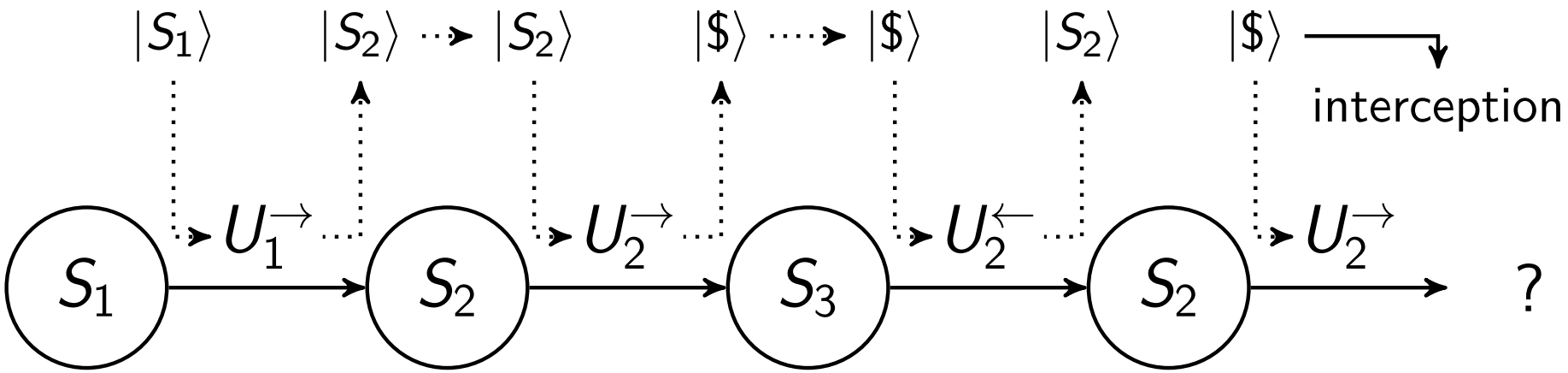


We replace $|S_3\rangle$ with the quantum coin $|\$\rangle$.

Contributions: intuition of the proof (cont'd)

Subcase 1: $U_i^{\leftrightarrows}$ is $U_i^{\rightarrow}$ and failed; therefore, $|\Psi\rangle = |S_{i+1}\rangle$ necessarily.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_2^{\leftarrow} U_2^{\rightarrow}$



We replace $|S_3\rangle$ with the quantum coin $|\$\rangle$.

$\rightarrow$ What $\mathcal{A}$ returns is $|\$\rangle \otimes |\$\rangle$ with probability $1/\text{poly}$.

Contributions: properties of Sim_3

Consequence

If $Sim_3(\mathcal{V})$ is distinguishable from $Sim_2(\mathcal{V})$, we can easily clone a quantum coin. **Absurd by [AC12], therefore Lemma 2 is true.**

Contributions: properties of Sim_3

Consequence

If $Sim_3(\mathcal{V})$ is distinguishable from $Sim_2(\mathcal{V})$, we can easily clone a quantum coin. **Absurd by [AC12], therefore Lemma 2 is true.**

Proposition

$\text{Output}(Sim_3(\mathcal{V})) \approx \text{Transcript}(\mathcal{V} \leftrightarrow \mathcal{P})$.

Contributions: properties of Sim_3

Consequence

If $Sim_3(\mathcal{V})$ is distinguishable from $Sim_2(\mathcal{V})$, we can easily clone a quantum coin. **Absurd by [AC12], therefore Lemma 2 is true.**

$$\Downarrow$$

Proposition

$\text{Output}(Sim_3(\mathcal{V})) \approx \text{Transcript}(\mathcal{V} \leftrightarrow \mathcal{P})$.

$\rightarrow$ All sequential simulators respect the constraints imposed by FSM + Sim_3.

Contributions: properties of Sim_3

Consequence

If $Sim_3(\mathcal{V})$ is distinguishable from $Sim_2(\mathcal{V})$, we can easily clone a quantum coin. **Absurd by [AC12], therefore Lemma 2 is true.**

$$\Downarrow$$

Proposition

$\text{Output}(Sim_3(\mathcal{V})) \approx \text{Transcript}(\mathcal{V} \leftrightarrow \mathcal{P})$.

$\rightarrow$ All sequential simulators respect the constraints imposed by FSM + Sim_3.

$\rightarrow$ What about parallel ones?

Contributions: parallel simulator

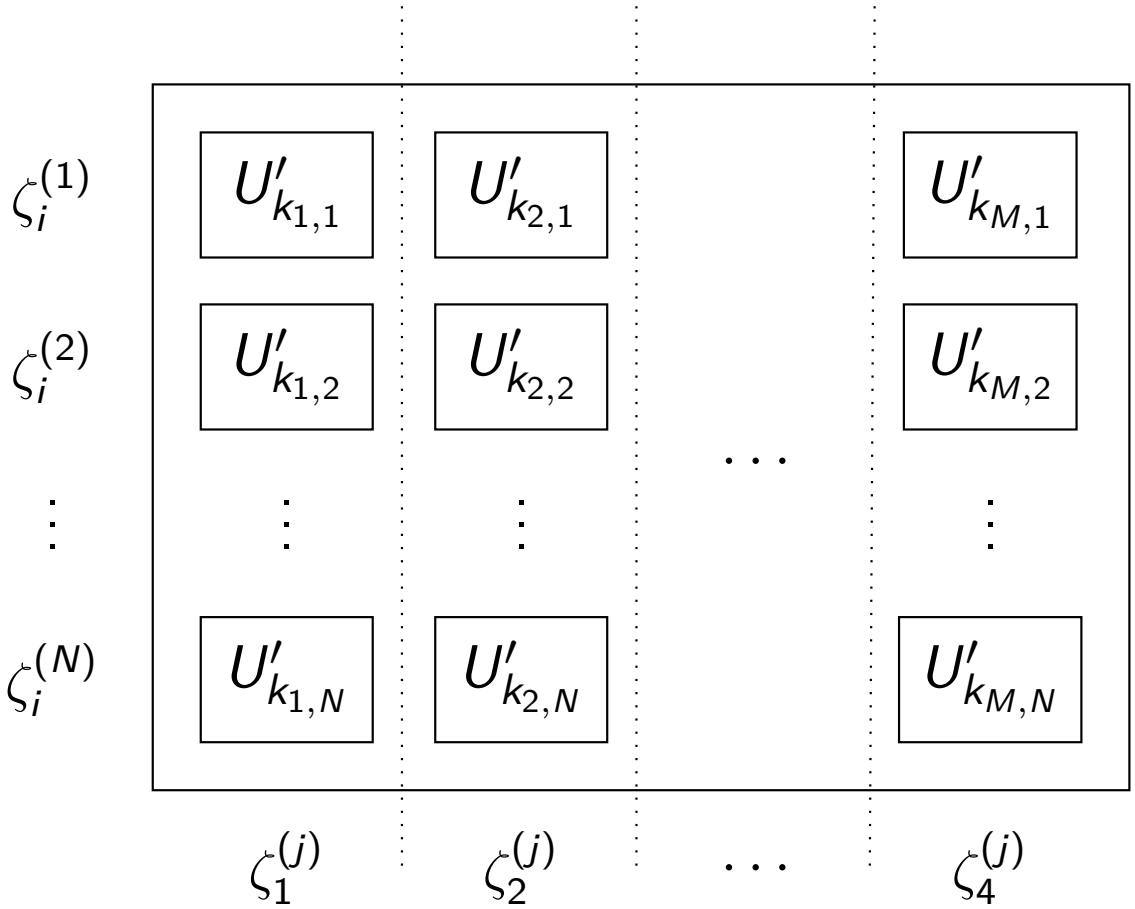


Figure: Parallel simulator

Contributions: Lemma 3

Considering that:

1. Sim_3 has a constant depth d (initial hypothesis at slide 5).

Contributions: Lemma 3

Considering that:

1. Sim_3 has a constant depth d (initial hypothesis at slide 5).
2. Sim_3 **cannot successfully** execute two U_i, U_j in parallel.

Contributions: Lemma 3

Considering that:

1. Sim_3 has a constant depth d (initial hypothesis at slide 5).
2. Sim_3 **cannot successfully** execute two U_i, U_j in parallel.

$$\Downarrow$$

Lemme 3

Sim_3 make a constant number of effective queries to $\mathcal{V}$.

Contributions: Lemma 3

Considering that:

1. Sim_3 has a constant depth d (initial hypothesis at slide 5).
2. Sim_3 **cannot successfully** execute two U_i, U_j in parallel.

Lemme 3

Sim_3 make a constant number of effective queries to $\mathcal{V}$.

..."effective"?

Contributions: Lemma 3

Considering that:

1. Sim_3 has a constant depth d (initial hypothesis at slide 5).
2. Sim_3 **cannot successfully** execute two U_i, U_j in parallel.

$$\Downarrow$$

Lemme 3

Sim_3 make a constant number of effective queries to $\mathcal{V}$.

..."effective"?

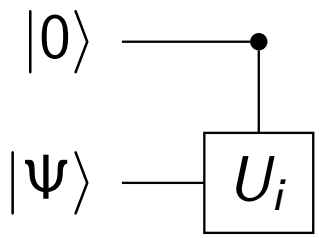
$\rightarrow$ Effective query to $\mathcal{V}$ = request that actually happened **at runtime**.

Discussion, next steps

$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1}$$

Discussion, next steps

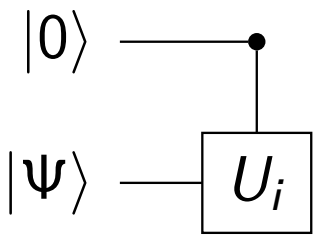
$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1}$$



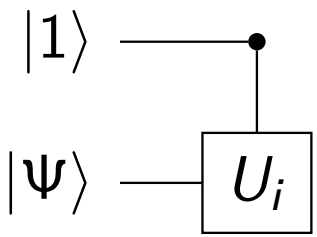
(a)

Discussion, next steps

$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1}$$



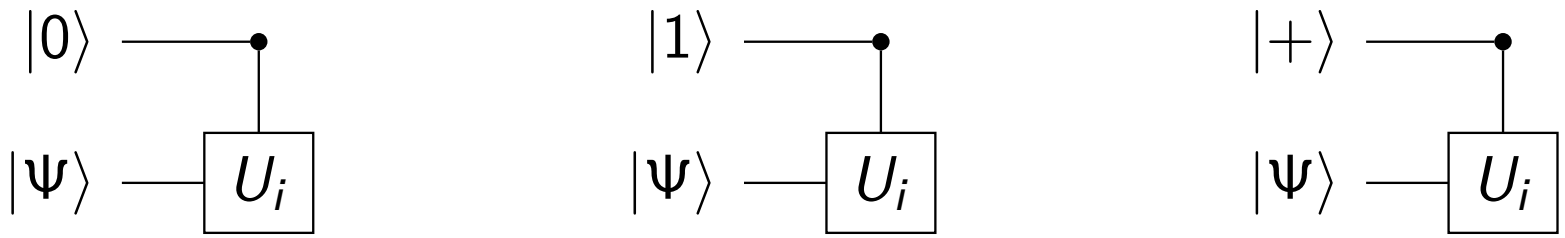
(a)



(b)

Discussion, next steps

$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1}$$



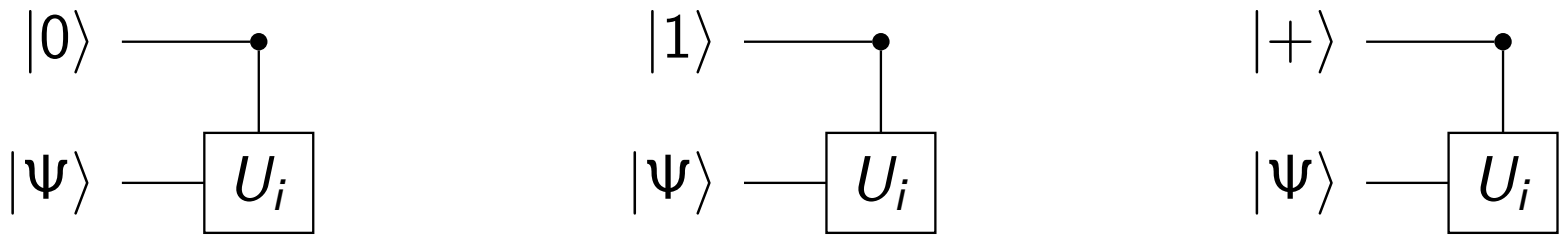
(a) (b) (c)

Figure: Some variations of the control qubit in W

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Discussion, next steps

$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1}$$



(a) (b) (c)

Figure: Some variations of the control qubit in W

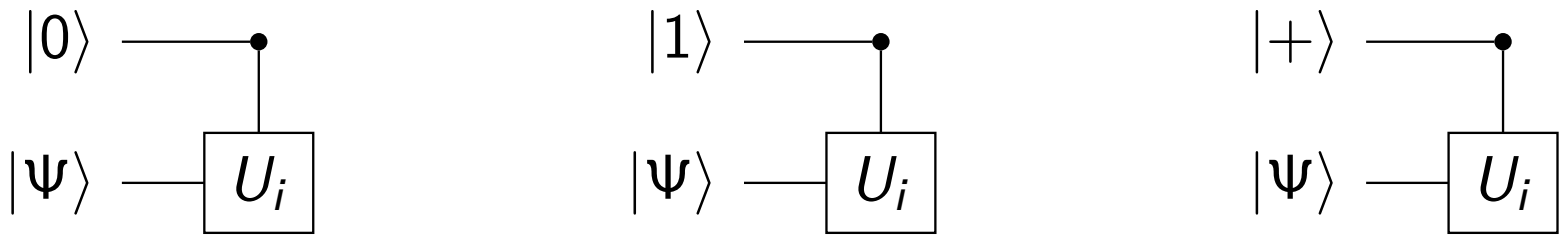
$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Next steps :

1. Prove that Lemma 3 implies the existence of Sim_4, which makes a constant number of queries to $\mathcal{V}$ (without 'effective').

Discussion, next steps

$$W = |1\rangle\langle 1| \otimes U_i + |0\rangle\langle 0| \otimes \mathbb{1}$$



(a) (b) (c)

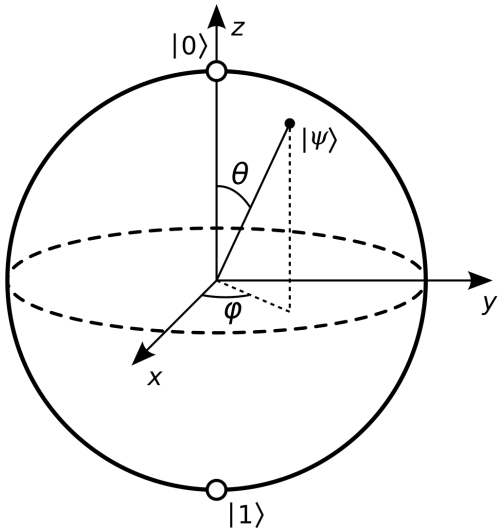
Figure: Some variations of the control qubit in W

$$|+\rangle = \frac{|0\rangle + |1\rangle}{\sqrt{2}}$$

Next steps :

1. Prove that Lemma 3 implies the existence of Sim_4, which makes a constant number of queries to $\mathcal{V}$ (without 'effective').
2. Prove that Sim_4 implies $\mathbb{L} \in \mathbf{BQP}$ (see [Chi+21])

Thank you for your attention



Refs

[AC12] Scott Aaronson and Paul Christiano. "Quantum money from hidden subspaces". In: *Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing*. STOC '12. New York, New York, USA: Association for Computing Machinery, 2012, pp. 41–60. ISBN: 9781450312455. DOI: 10.1145/2213977.2213983. URL: https://doi.org/10.1145/2213977.2213983.

[Chi+21] Nai-Hui Chia et al. *On the Impossibility of Post-Quantum Black-Box Zero-Knowledge in Constant Rounds*. 2021. arXiv: 2103.11244 [cs.CR]. URL: https://arxiv.org/abs/2103.11244.

[GMR85] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. "The knowledge complexity of interactive proof-systems". In: *Proceedings of the seventeenth annual ACM symposium on Theory of computing*. STOC '85. New York, NY, USA: Association for Computing Machinery, Dec. 1985, pp. 291–304. ISBN: 978-0-89791-151-1. DOI: 10.1145/22145.22178. URL: https://dl.acm.org/doi/10.1145/22145.22178.

Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

Appendix: proof of case 1

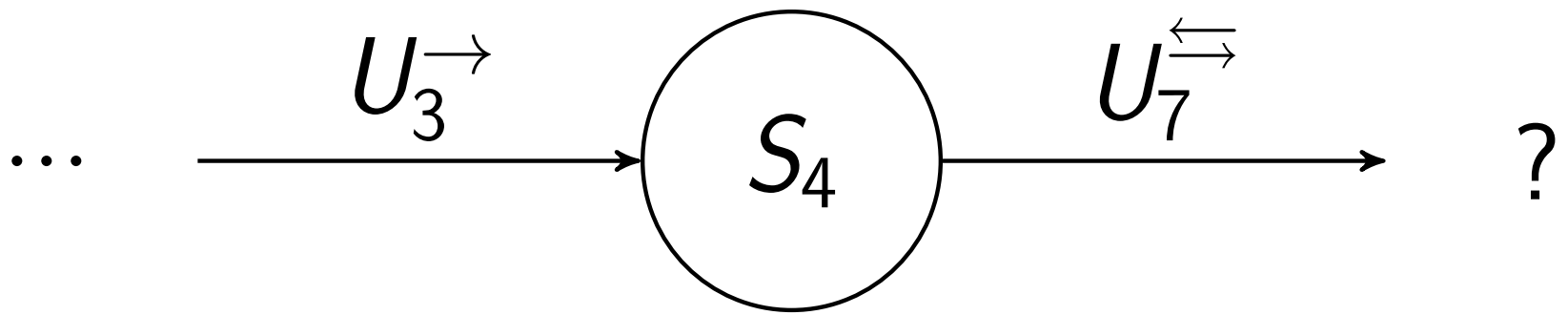
Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$

Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

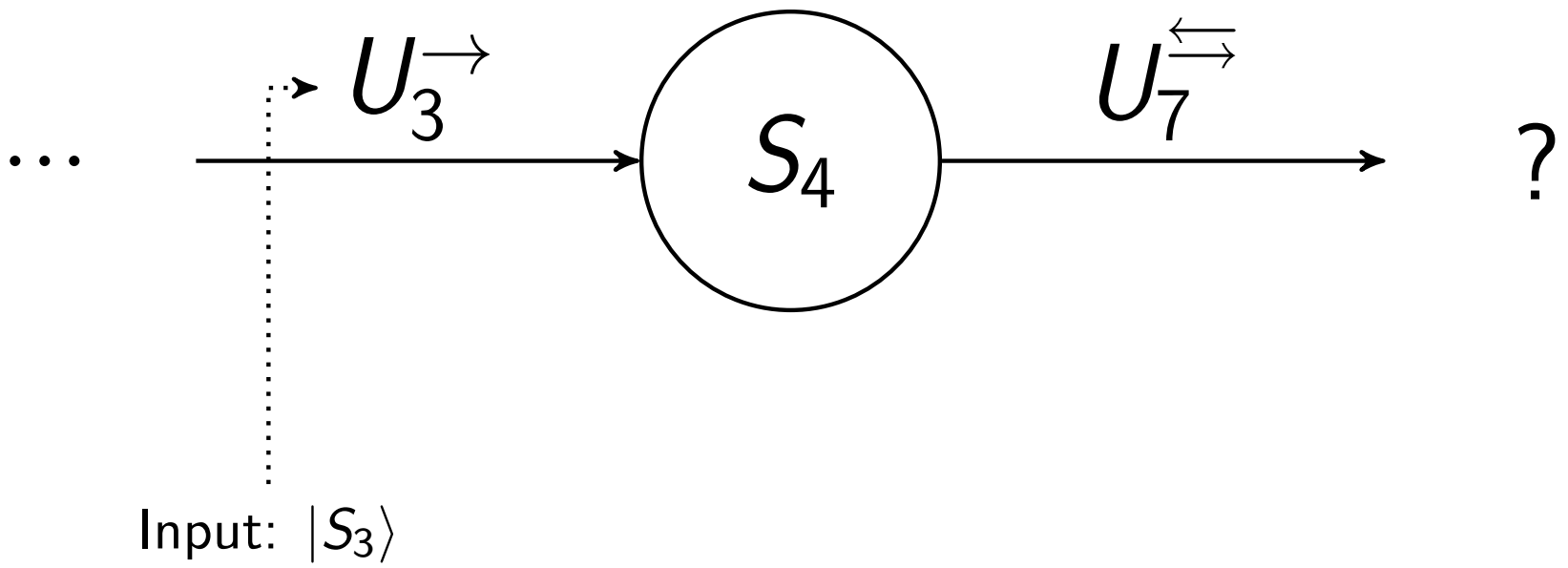
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$



Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

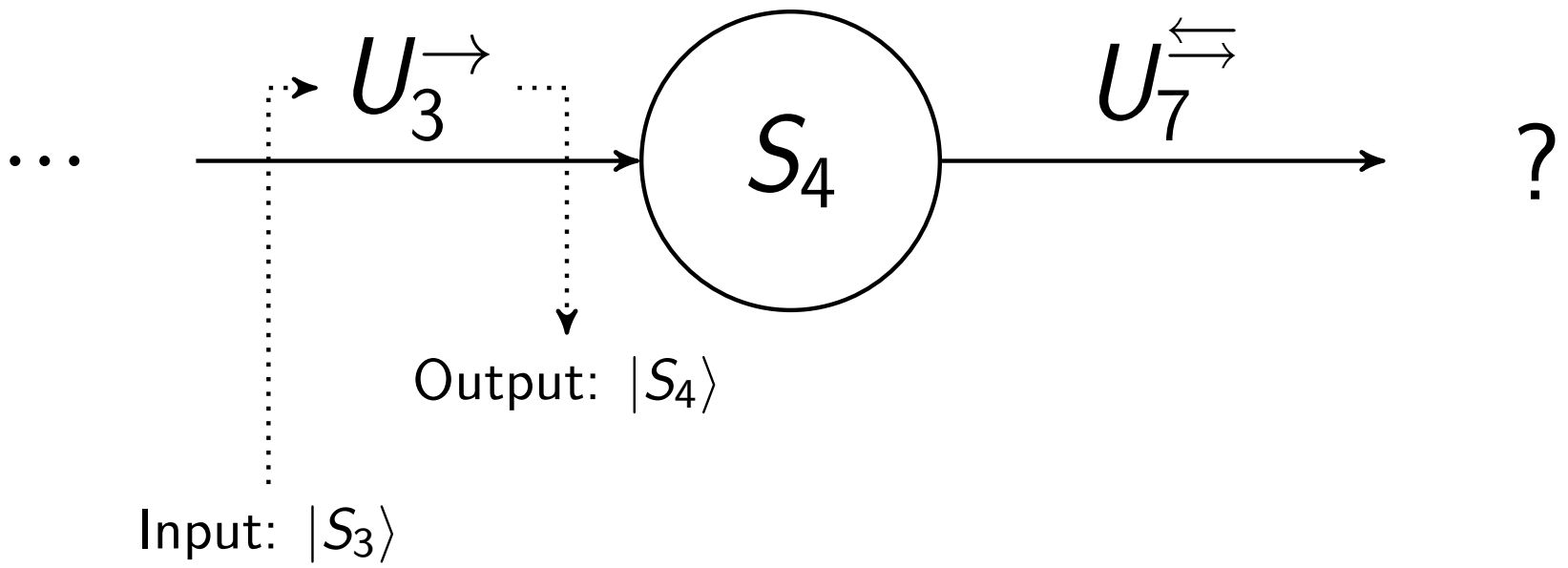
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$



Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

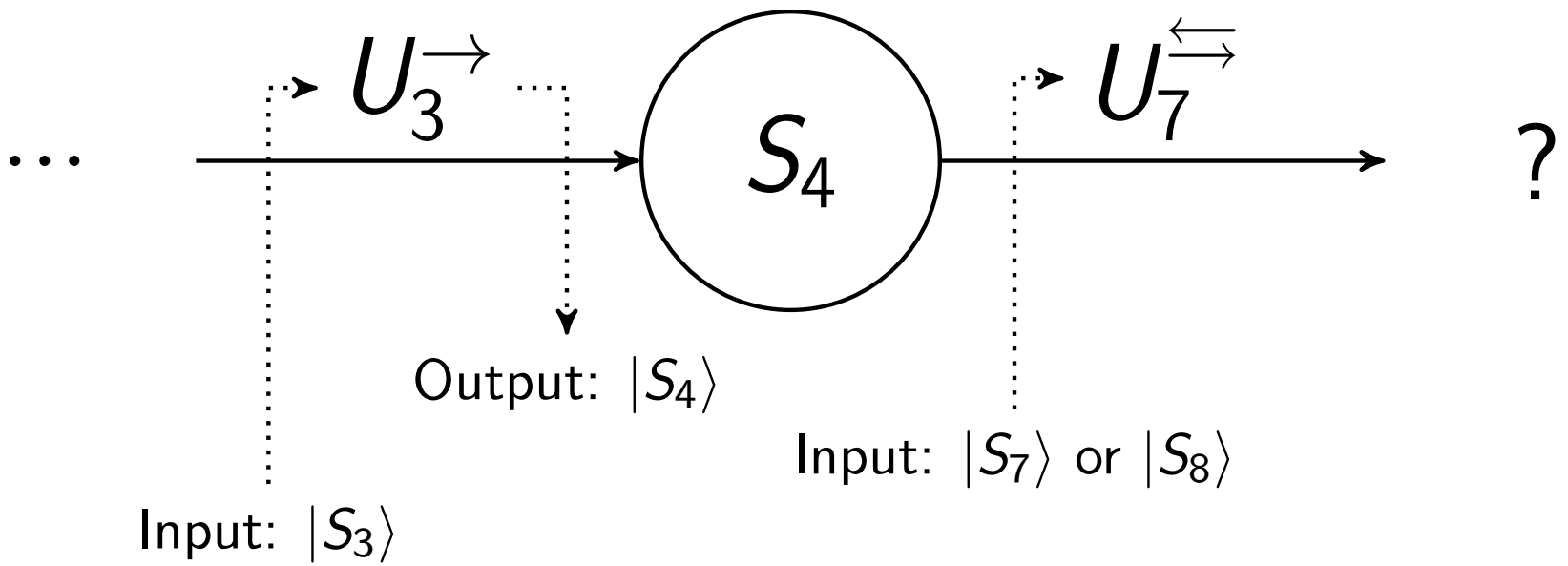
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$



Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

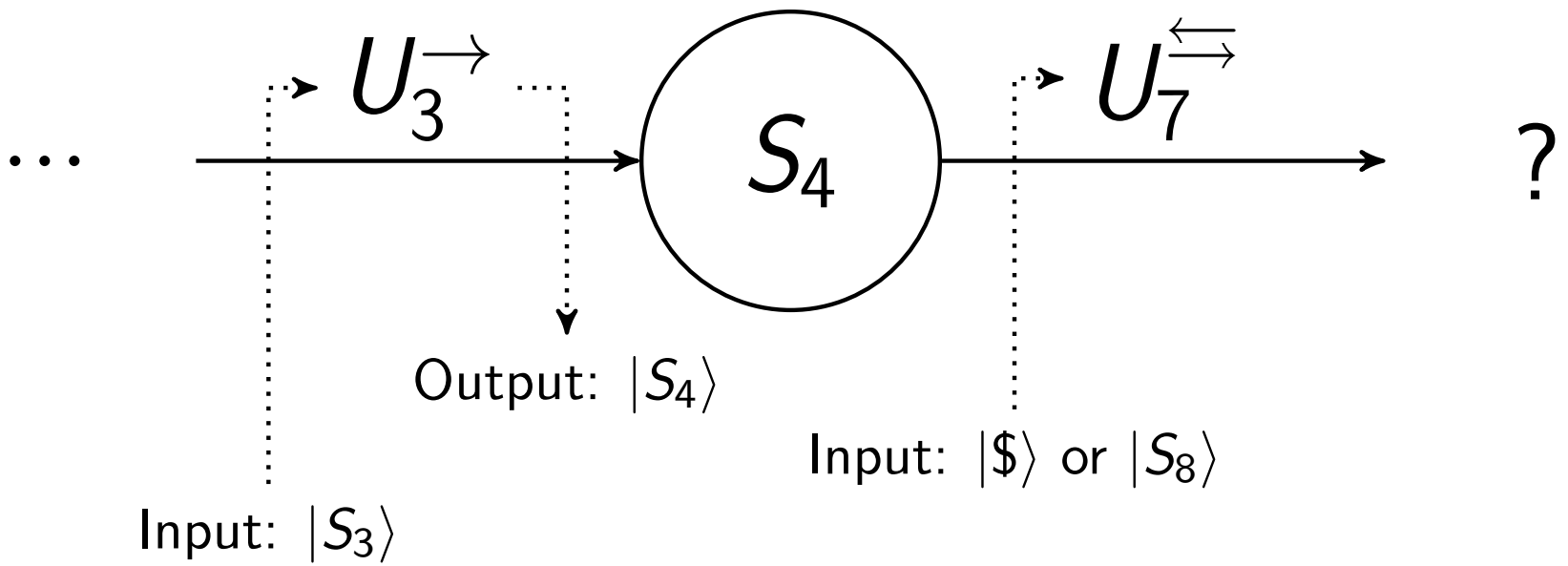
Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$



Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$

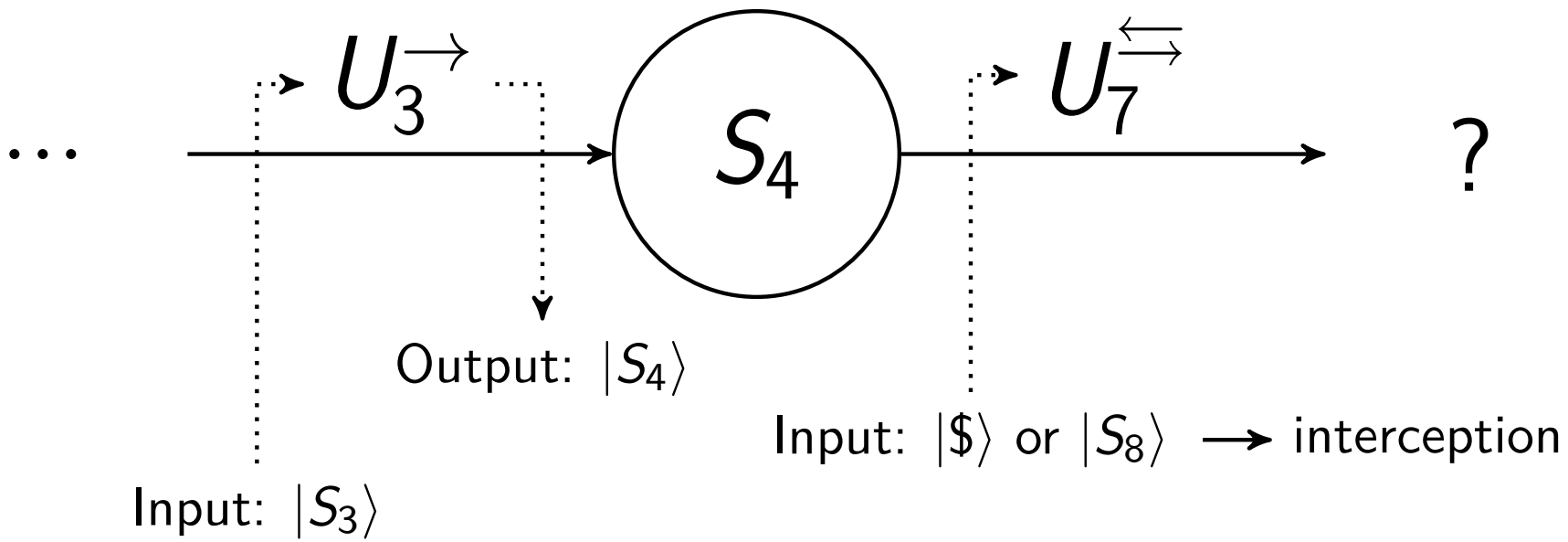


Let us replace $|S_7\rangle$ with $|\$\rangle$

Appendix: proof of case 1

Case 1 : the word π that corresponds to the sequence of gates executed **is not** in the language accepted by the FSM.

Example: $\pi = U_1^{\rightarrow} U_2^{\rightarrow} U_3^{\rightarrow} U_7^{\leftrightarrows}$



Let us replace $|S_7\rangle$ with $|\$\rangle$