Isolation Lemma for Directed Reachability and NL vs. L

Nutan Limaye (IIT Bombay)
Joint work with Vaibhav Krishan

Dagstuhl Seminar 16411 - Algebraic Methods in Computational Complexity
Oct 09 - Oct 14, 2016

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on n input variables,

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on n input variables,

Output: a new formula ψ on the same n variables such that

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on n input variables,

Output: a new formula ψ on the same n variables such that

- every satisfying assignment of ψ also satisfies ϕ

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on n input variables,

Output: a new formula ψ on the same n variables such that

- every satisfying assignment of ψ also satisfies ϕ
  This implies that if ϕ is not satisfiable then ψ is also not satisfiable.

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on n input variables,

Output: a new formula ψ on the same n variables such that

- every satisfying assignment of ψ also satisfies ϕ
  This implies that if ϕ is not satisfiable then ψ is also not satisfiable.
- if ϕ is satisfiable, then ψ has exactly one satisfying assignment.

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on *n* input variables,

Output: a new formula ψ on the same *n* variables such that

- every satisfying assignment of ψ also satisfies ϕ
  This implies that if ϕ is not satisfiable then ψ is also not satisfiable.
- if ϕ is satisfiable, then ψ has exactly one satisfying assignment.

Valiant Vazirani Isolation Lemma

Isolation lemma for NP

SAT

Given a Boolean formula ϕ determine whether ϕ has a satisfying assignment.

Isolation Procedure for SAT

Given: a Boolean forumla ϕ on n input variables,

Output: a new formula ψ on the same n variables such that

- every satisfying assignment of ψ also satisfies ϕ
  This implies that if ϕ is not satisfiable then ψ is also not satisfiable.
- if ϕ is satisfiable, then ψ has exactly one satisfying assignment.

Valiant Vazirani Isolation Lemma

There is a randomized polynomial time isolation procedure for SAT with success probability $\Omega(\frac{1}{n})$

Bumping up the success probability

Determinizing Isolation Procedure for SAT

Can one get rid of the randomness in the Isolation Procedure for SAT?

Bumping up the success probability

Determinizing Isolation Procedure for SAT

Can one get rid of the randomness in the Isolation Procedure for SAT? Open!

Success probability of the Isolation Procedure for SAT

Bumping up the success probability

Determinizing Isolation Procedure for SAT

Can one get rid of the randomness in the Isolation Procedure for SAT? Open!

Success probability of the Isolation Procedure for SAT

Can one make the success probability of the Isolation Procedure higher?

Bumping up the success probability

Determinizing Isolation Procedure for SAT

Can one get rid of the randomness in the Isolation Procedure for SAT? Open!

Success probability of the Isolation Procedure for SAT

Can one make the success probability of the Isolation Procedure higher?

The success probability of the Isolation Procedure for SAT can be made greater than $2/3$ if and only if $\mathrm{NP} \subseteq \mathrm{P/poly}$.

Bumping up the success probability

Determinizing Isolation Procedure for SAT

Can one get rid of the randomness in the Isolation Procedure for SAT? Open!

Success probability of the Isolation Procedure for SAT

Can one make the success probability of the Isolation Procedure higher?

The success probability of the Isolation Procedure for SAT can be made greater than $2/3$ if and only if NP $\subseteq$ P/poly.

[DKvMW] *Is Valiant Vazinai's isolation probability improvable? Dell, Kabanets, van Melkebeek, Watanabe, Computational Complexity, 2013.*

NL, L, Directed Reachability, L/poly

Complexity classes NL, L, L/poly

NL, L, Directed Reachability, L/poly

Complexity classes NL, L, L/poly

L: the class of decision problems decidable by dereministic Turing machines using $O(\log n)$ space, where n is the length of the input.

NL, L, Directed Reachability, L/poly

Complexity classes NL, L, L/poly

L: the class of decision problems decidable by dereministic Turing machines using $O(\log n)$ space, where n is the length of the input.

NL: the class of decision problems decidable by non-dereministic Turing machine using $O(\log n)$ space, where n is the length of the input.

NL, L, Directed Reachability, L/poly

Complexity classes NL, L, L/poly

L: the class of decision problems decidable by dereministic Turing machines using $O(\log n)$ space, where n is the length of the input.

NL: the class of decision problems decidable by non-dereministic Turing machine using $O(\log n)$ space, where n is the length of the input.

L/poly: the class of decision problems decidable by dereministic Turing machine using $O(\log n)$ space and poly(n) amount of advice, where n is the length of the input.

NL, L, Directed Reachability, L/poly

Complexity classes NL, L, L/poly

L: the class of decision problems decidable by dereministic Turing machines using $O(\log n)$ space, where n is the length of the input.

NL: the class of decision problems decidable by non-dereministic Turing machine using $O(\log n)$ space, where n is the length of the input.

L/poly: the class of decision problems decidable by dereministic Turing machine using $O(\log n)$ space and poly(n) amount of advice, where n is the length of the input.

Directed Reachability, **Reach**

Given: A directed graph $G = (V, E)$ and two designated vertices s, t

NL, L, Directed Reachability, L/poly

Complexity classes NL, L, L/poly

L: the class of decision problems decidable by dereministic Turing machines using $O(\log n)$ space, where n is the length of the input.

NL: the class of decision problems decidable by non-dereministic Turing machine using $O(\log n)$ space, where n is the length of the input.

L/poly: the class of decision problems decidable by dereministic Turing machine using $O(\log n)$ space and poly(n) amount of advice, where n is the length of the input.

Directed Reachability, **Reach**

Given: A directed graph $G = (V, E)$ and two designated vertices s, t

Output: yes if and only if there is a directed path from s to t in G.

Our result

Isolation Procedure for Directed Reachability

Our result

Isolation Procedure for Directed Reachability

Given: a graph G on n input vertices, and two designated vertices s, t

Our result

Isolation Procedure for Directed Reachability

Given: a graph G on n input vertices, and two designated vertices s, t

Output: a new graph H on the same n variables such that

Our result

Isolation Procedure for Directed Reachability

Given: a graph G on n input vertices, and two designated vertices s, t

Output: a new graph H on the same n variables such that

- every s to t path in H is also a path in G

Our result

Isolation Procedure for Directed Reachability

Given: a graph G on n input vertices, and two designated vertices s, t

Output: a new graph H on the same n variables such that

- every s to t path in H is also a path in G

  This implies that in G if t is not reachable from s then in H as well there is no s to t path.

Our result

Isolation Procedure for Directed Reachability

Given: a graph G on n input vertices, and two designated vertices s, t

Output: a new graph H on the same n variables such that

- every s to t path in H is also a path in G
  This implies that in G if t is not reachable from s then in H as well there is no s to t path.
- if G has an s to t path, then H has exactly one s to t path.

Our result

Isolation Procedure for Directed Reachability

Given: a graph G on n input vertices, and two designated vertices s, t

Output: a new graph H on the same n variables such that

- every s to t path in H is also a path in G
  This implies that in G if t is not reachable from s then in H as well there is no s to t path.
- if G has an s to t path, then H has exactly one s to t path.

Success probability of the Isolation Procedure for **Reach**

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than* $2/3$ *if and only if* $NL \subseteq L/poly$.

Isolation for other classes

Isolation for LogCFL

Isolation for other classes

Isolation for LogCFL

There exists a randomized isolation procedure for a hard problem in LogCFL that runs in L/poly with success probability greater than 2/3 if and only if LogCFL $\subseteq$ L/poly.

Isolation for other classes

Isolation for LogCFL

There exists a randomized isolation procedure for a hard problem in LogCFL that runs in L/poly with success probability greater than $2/3$ if and only if LogCFL $\subseteq$ L/poly.

Isolation for NP

There exists a randomized isolation procedure for SAT that runs in L/poly with success probability greater than $2/3$ if and only if NP $\subseteq$ L/poly.

Proof outline

The proofs follows a similar structure as in [DKvMW]

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Definition (Promise sets for a version of Reach)

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Definition (Promise sets for a version of Reach)

$\text{Yes}_{\textbf{Reach}} = \{(G, s, t) \mid \text{unique reachable path between } s \text{ and } t\}$,

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Definition (Promise sets for a version of Reach)

$$\begin{aligned} \mathrm{Yes}_{\mathbf{Reach}} &= \{(G, s, t) \mid \text{unique reachable path between } s \text{ and } t\}, \text{ and} \\ \mathrm{No}_{\mathbf{Reach}} &= \{(G, s, t) \mid \text{no path between } s \text{ and } t\} \end{aligned}$$

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Definition (Promise sets for a version of Reach)

$$\begin{aligned}
\mathsf{Yes}_{\textbf{Reach}} &= \{(G, s, t) \mid \text{unique reachable path between } s \text{ and } t\}, \text{ and} \\
\mathsf{No}_{\textbf{Reach}} &= \{(G, s, t) \mid \text{no path between } s \text{ and } t\}
\end{aligned}$$

Definition (PrUReach)

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Definition (Promise sets for a version of **Reach**)

$$\begin{aligned}\mathsf{Yes}_{\textbf{Reach}} &= \{(G, s, t) \mid \text{unique reachable path between } s \text{ and } t\}, \text{ and} \\ \mathsf{No}_{\textbf{Reach}} &= \{(G, s, t) \mid \text{no path between } s \text{ and } t\}\end{aligned}$$

Definition (**PrUReach**)

Given: $G \in \mathsf{Yes}_{\textbf{Reach}} \cup \mathsf{No}_{\textbf{Reach}}$

Proof outline

The proofs follows a similar structure as in [DKvMW]

We will start with some definitions.

Definition (Promise sets for a version of Reach)

$$\begin{aligned}\mathrm{Yes}_{\mathbf{Reach}} &= \{(G,s,t) \mid \text{unique reachable path between } s \text{ and } t\}, \text{ and}\\ \mathrm{No}_{\mathbf{Reach}} &= \{(G,s,t) \mid \text{no path between } s \text{ and } t\}\end{aligned}$$

Definition (PrUReach)

Given: $G \in \mathrm{Yes}_{\mathbf{Reach}} \cup \mathrm{No}_{\mathbf{Reach}}$

Output: yes if and only if $G \in \mathrm{Yes}_{\mathbf{Reach}}$.

Proof outline

Recall the statement we wish to prove

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than* $2/3$ *if and only if* $NL \subseteq L/poly$.

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than 2/3 if and only if* $NL \subseteq L/poly$.

Step 1 Prove that $\mathsf{NL} \subseteq \mathsf{L}/\mathsf{poly}$ if and only if **PrUReach** $\in \mathsf{L}/\mathsf{poly}$.

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than 2/3 if and only if NL $\subseteq$ L/poly.*

Step 1 Prove that NL $\subseteq$ L/poly if and only if **PrUReach** $\in$ L/poly.

Step 2 Prove the following statement:

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than 2/3 if and only if NL $\subseteq$ L/poly.*

Step 1 Prove that NL $\subseteq$ L/poly if and only if **PrUReach** $\in$ L/poly.

Step 2 Prove the following statement:
There exists a randomized isolation procedure for **Reach** with success probability greater than 2/3 if and only if **PrUReach** $\in$ L/poly.

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than* $2/3$ *if and only if* $NL \subseteq L/poly$.

Step 1 Prove that $\mathsf{NL} \subseteq \mathsf{L/poly}$ if and only if **PrUReach** $\in \mathsf{L/poly}$.

Step 2 Prove the following statement:
There exists a randomized isolation procedure for **Reach** with success probability greater than $2/3$ if and only if **PrUReach** $\in \mathsf{L/poly}$.

Details of Step 1

$\mathsf{NL} \subseteq \mathsf{L/poly} \Leftrightarrow$ **PrUReach** $\in \mathsf{L/poly}$

Details of Step 1

$\mathsf{NL} \subseteq \mathsf{L/poly} \Leftrightarrow$ **PrUReach** $\in \mathsf{L/poly}$

$\Rightarrow$ This direction is trivial.

Details of Step 1

$\mathsf{NL} \subseteq \mathsf{L/poly} \Leftrightarrow$ **PrUReach** $\in \mathsf{L/poly}$

$\Rightarrow$ This direction is trivial.

$\Leftarrow$ Uses an algorithm developed in the following work.

Details of Step 1

$\mathsf{NL} \subseteq \mathsf{L/poly} \Leftrightarrow$ **PrUReach** $\in \mathsf{L/poly}$

$\Rightarrow$ This direction is trivial.

$\Leftarrow$ Uses an algorithm developed in the following work.

[RA] *Making Nondeterminism Unambiguous, Reinhardt, Allender, SIAM Journal of Computing, 2000.*

Details of Step 1

$\mathsf{NL} \subseteq \mathsf{L/poly} \Leftrightarrow$ **PrUReach** $\in \mathsf{L/poly}$

$\Rightarrow$ This direction is trivial.

$\Leftarrow$ Uses an algorithm developed in the following work.

[RA] *Making Nondeterminism Unambiguous, Reinhardt, Allender, SIAM Journal of Computing, 2000.*

We need one more definition.

Details of Step 1

$\mathsf{NL} \subseteq \mathsf{L/poly} \Leftrightarrow$ **PrUReach** $\in \mathsf{L/poly}$

$\Rightarrow$ This direction is trivial.

$\Leftarrow$ Uses an algorithm developed in the following work.

[RA] *Making Nondeterminism Unambiguous, Reinhardt, Allender, SIAM Journal of Computing, 2000.*

We need one more definition.

Definition (Min-unique graph [GW, RA])

Details of Step 1

NL $\subseteq$ L/poly $\Leftrightarrow$ **PrUReach** $\in$ L/poly

$\Rightarrow$ This direction is trivial.

$\Leftarrow$ Uses an algorithm developed in the following work.

[RA] *Making Nondeterminism Unambiguous, Reinhardt, Allender, SIAM Journal of Computing, 2000.*

We need one more definition.

Definition (Min-unique graph [GW, RA])

A weighted directed graph $G = (V, E)$ with weight function $w : E \rightarrow \mathbb{N}$ is said to be *min-unique* if between every pair of vertices the minimum weight path is unique.

Breaking down Step 1

Step 1.1: Generating min-unique graph using advice

Breaking down Step 1

Step 1.1: Generating min-unique graph using advice

Given a graph G on n vertices a procedure P_1 generates graphs $G_1, G_2, \ldots, G_{n^2}$

- For all $1 \leq i \leq n^2$, G_i is on the same set of vertices as G.

Breaking down Step 1

Step 1.1: Generating min-unique graph using advice

Given a graph G on n vertices a procedure P_1 generates graphs $G_1, G_2, \ldots, G_{n^2}$

- For all $1 \leq i \leq n^2$, G_i is on the same set of vertices as G.
- G has an s to t path iff $\forall i \in [n^2]$, G_i has an s to t path.

Breaking down Step 1

Step 1.1: Generating min-unique graph using advice

Given a graph G on n vertices a procedure P_1 generates graphs $G_1, G_2, \ldots, G_{n^2}$

- For all $1 \leq i \leq n^2$, G_i is on the same set of vertices as G.
- G has an s to t path iff $\forall i \in [n^2]$, G_i has an s to t path.
- If G has an s to t path then $\exists i \in [n^2]$: such that G_i is min-unique.

Breaking down Step 1

Step 1.1: Generating min-unique graph using advice

Given a graph G on n vertices a procedure P_1 generates graphs $G_1, G_2, \ldots, G_{n^2}$

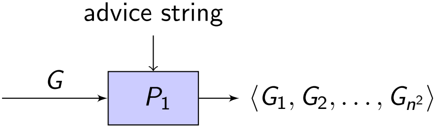
- For all $1 \leq i \leq n^2$, G_i is on the same set of vertices as G.
- G has an s to t path iff $\forall i \in [n^2]$, G_i has an s to t path.
- If G has an s to t path then $\exists i \in [n^2]$: such that G_i is min-unique.
- P_1 has an L/poly algorithm.

Breaking down Step 1

Step 1.1: Generating min-unique graph using advice

Given a graph G on n vertices a procedure P_1 generates graphs $G_1, G_2, \ldots, G_{n^2}$

- For all $1 \leq i \leq n^2$, G_i is on the same set of vertices as G.
- G has an s to t path iff $\forall i \in [n^2]$, G_i has an s to t path.
- If G has an s to t path then $\exists i \in [n^2]$: such that G_i is min-unique.
- P_1 has an L/poly algorithm.



Breaking down Step 1

Step 1.2: Generating $G' \in \mathsf{Yes}_{\mathsf{Reach}} \cup \mathsf{No}_{\mathsf{Reach}}$ given a min-unique G

Breaking down Step 1

Step 1.2: Generating $G' \in \mathsf{Yes}_{\mathsf{Reach}} \cup \mathsf{No}_{\mathsf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices a procedure P_2 generates a graph C_G

Breaking down Step 1

Step 1.2: Generating $G' \in \mathrm{Yes}_{\mathbf{Reach}} \cup \mathrm{No}_{\mathbf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices a procedure P_2 generates a graph C_G

- On input (G, s, t), (C_G, s', t') is produced.

Breaking down Step 1

Step 1.2: Generating $G' \in \text{Yes}_{\textbf{Reach}} \cup \text{No}_{\textbf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices a procedure P_2 generates a graph C_G

- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.

Breaking down Step 1

Step 1.2: Generating $G' \in \text{Yes}_{\text{Reach}} \cup \text{No}_{\text{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices a procedure P_2 generates a graph C_G

- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.
- If G is min-unique and has an s to t path then there is a unique path from s' to t'.

Breaking down Step 1

Step 1.2: Generating $G' \in \text{Yes}_{\text{Reach}} \cup \text{No}_{\text{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices a procedure P_2 generates a graph C_G

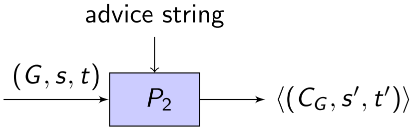
- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.
- If G is min-unique and has an s to t path then there is a unique path from s' to t'.
- P_2 has an L/poly algorithm.

Breaking down Step 1

Step 1.2: Generating $G' \in \mathsf{Yes}_{\mathsf{Reach}} \cup \mathsf{No}_{\mathsf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices a procedure P_2 generates a graph C_G

- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.
- If G is min-unique and has an s to t path then there is a unique path from s' to t'.
- P_2 has an L/poly algorithm.



Breaking down Step 1

Step 1.3: Using P_1, P_2 to solve **Reach**.

Breaking down Step 1

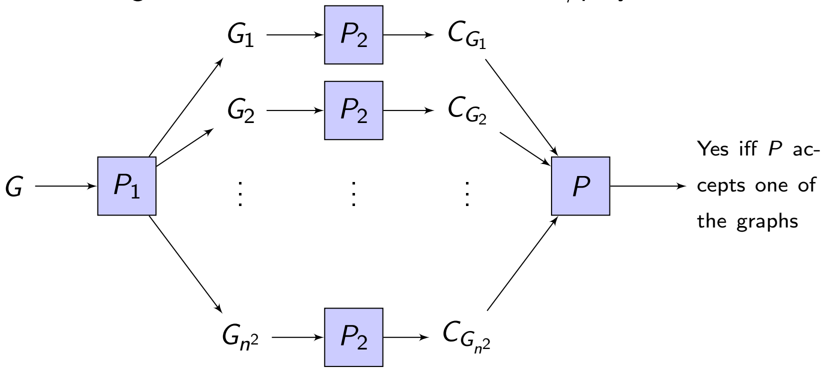
Step 1.3: Using P_1, P_2 to solve **Reach**.

P be the algorithm that solves **PrUReach** in L/poly.

Breaking down Step 1

Step 1.3: Using P_1, P_2 to solve **Reach**.

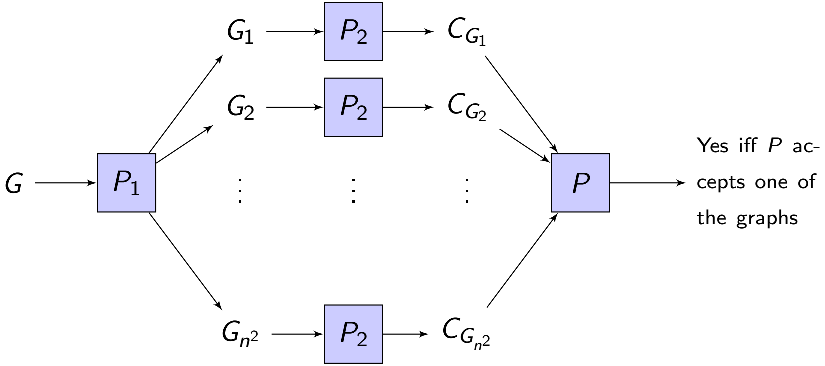
P be the algorithm that solves **PrUReach** in L/poly.



Breaking down Step 1

Step 1.3: Using P_1, P_2 to solve **Reach**.

P be the algorithm that solves **PrUReach** in L/poly.

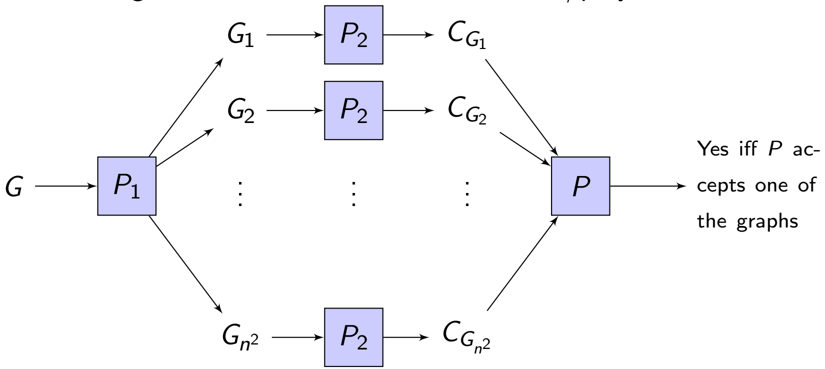


If G does not have an s to t path, we reject.

Breaking down Step 1

Step 1.3: Using P_1, P_2 to solve **Reach**.

P be the algorithm that solves **PrUReach** in L/poly.



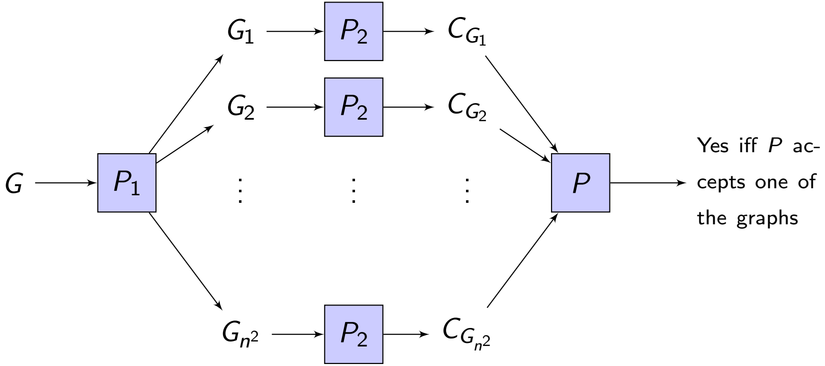
If G does not have an s to t path, we reject.

If G has an s to t path,

Breaking down Step 1

Step 1.3: Using P_1, P_2 to solve **Reach**.

P be the algorithm that solves **PrUReach** in L/poly.



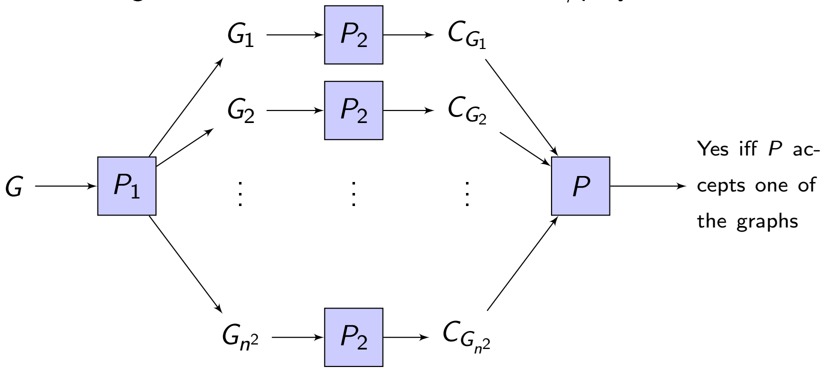
If G does not have an s to t path, we reject.

If G has an s to t path,
at least one of the G_i is min-unique.

Breaking down Step 1

Step 1.3: Using P_1, P_2 to solve **Reach**.

P be the algorithm that solves **PrUReach** in L/poly.



If G does not have an s to t path, we reject.

If G has an s to t path,
at least one of the G_i is min-unique.
The corresponding $C_{G_i} \in \text{Yes}_{\textbf{Reach}}$.

Details about Step 1.2

Step 1.2: Generating $G' \in \mathsf{Yes}_{\mathsf{Reach}} \cup \mathsf{No}_{\mathsf{Reach}}$ given a min-unique G

Details about Step 1.2

Step 1.2: Generating $G' \in \text{Yes}_{\textbf{Reach}} \cup \text{No}_{\textbf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices

- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.

Details about Step 1.2

Step 1.2: Generating $G' \in \text{Yes}_{\textbf{Reach}} \cup \text{No}_{\textbf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices

- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.
- If G is min-unique then there is a unique path from s' to t'.

Details about Step 1.2

Step 1.2: Generating $G' \in \mathsf{Yes}_{\mathsf{Reach}} \cup \mathsf{No}_{\mathsf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices

- On input (G, s, t), (C_G, s', t') is produced.
- G has an s to t path if and only if C_G has an s' to t' path.
- If G is min-unique then there is a unique path from s' to t'.

[RA] If G is min-unqiue then there is a UL algorithm that decides the reachability in G.

Details about Step 1.2

Step 1.2: Generating $G' \in \mathsf{Yes}_{\mathsf{Reach}} \cup \mathsf{No}_{\mathsf{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices

On input (G, s, t), (C_G, s', t') is produced.

G has an s to t path if and only if C_G has an s' to t' path.

If G is min-unique then there is a unique path from s' to t'.

[RA] If G is min-unqiue then there is a UL algorithm that decides the reachability in G.

C_G is the configuration graph of the UL algorithm on input G.

Details about Step 1.2

Step 1.2: Generating $G' \in \text{Yes}_{\text{Reach}} \cup \text{No}_{\text{Reach}}$ given a min-unique G

Given a min-unique graph G on n vertices

On input (G, s, t), (C_G, s', t') is produced.
G has an s to t path if and only if C_G has an s' to t' path.
If G is min-unique then there is a unique path from s' to t'.

[RA] If G is min-unqiue then there is a UL algorithm that decides the reachability in G.

C_G is the configuration graph of the UL algorithm on input G.

C_G can be computed in L.

Proof outline

Recall the statement we wish to prove

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than* $2/3$ *if and only if* $NL \subseteq L/poly$*.*

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than* $2/3$ *if and only if* $NL \subseteq L/poly$.

Step 1 Prove that $\mathsf{NL} \subseteq \mathsf{L/poly}$ if and only if **PrUReach** $\in \mathsf{L/poly}$.

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than 2/3 if and only if* $NL \subseteq L/poly$.

Step 1 Prove that $\mathsf{NL} \subseteq \mathsf{L}/\mathsf{poly}$ if and only if **PrUReach** $\in \mathsf{L}/\mathsf{poly}$.

Step 2 Prove the following statement:

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than 2/3 if and only if NL $\subseteq$ L/poly.*

Step 1 Prove that NL $\subseteq$ L/poly if and only if **PrUReach** $\in$ L/poly.

Step 2 Prove the following statement:
There exists a randomized isolation procedure for **Reach** with success probability greater than 2/3 if and only if **PrUReach** $\in$ L/poly.

Proof outline

Recall the statement we wish to prove

Theorem (Main result)

There exists a randomized isolation procedure for **Reach** *with success probability greater than 2/3 if and only if* $NL \subseteq L/poly$.

Step 1 Prove that $\mathsf{NL} \subseteq \mathsf{L}/\mathsf{poly}$ if and only if **PrUReach** $\in \mathsf{L}/\mathsf{poly}$.

Step 2 Prove the following statement:
There exists a randomized isolation procedure for **Reach** with success probability greater than $2/3$ if and only if **PrUReach** $\in \mathsf{L}/\mathsf{poly}$.

Details about Step 2

From the hypothesis we can show that there is an L/poly procedure, say B s.t.

Details about Step 2

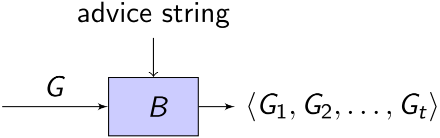
From the hypothesis we can show that there is an L/poly procedure, say B s.t.

Given a graph G as input, it outputs $\langle G_1, G_2, \ldots, G_t \rangle$ such that $> 2/3$ fraction of the G_is have unique s to t paths.

Details about Step 2

From the hypothesis we can show that there is an L/poly procedure, say B s.t.

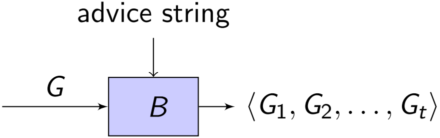
Given a graph G as input, it outputs $\langle G_1, G_2, \ldots, G_t \rangle$ such that $> 2/3$ fraction of the G_is have unique s to t paths.



Details about Step 2

From the hypothesis we can show that there is an L/poly procedure, say B s.t.

Given a graph G as input, it outputs $\langle G_1, G_2, \ldots, G_t \rangle$ such that $> 2/3$ fraction of the G_is have unique s to t paths.

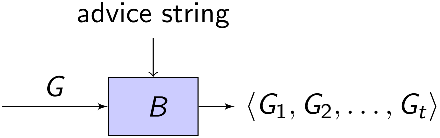


H be a graph with π as its s to t path.

Details about Step 2

From the hypothesis we can show that there is an L/poly procedure, say B s.t.

Given a graph G as input, it outputs $\langle G_1, G_2, \ldots, G_t \rangle$ such that $> 2/3$ fraction of the G_is have unique s to t paths.



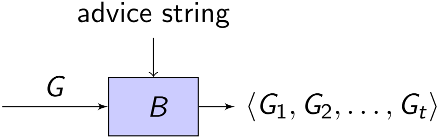
H be a graph with π as its s to t path.

For two graph G, H we say that (G, H) is good if π is a reachable path in $< 2/3$ fraction of graphs in $B(G + H)$.

Details about Step 2

From the hypothesis we can show that there is an L/poly procedure, say B s.t.

Given a graph G as input, it outputs $\langle G_1, G_2, \ldots, G_t \rangle$ such that $> 2/3$ fraction of the G_is have unique s to t paths.



H be a graph with π as its s to t path.

For two graph G, H we say that (G, H) is good if π is a reachable path in $< 2/3$ fraction of graphs in $B(G + H)$.

Let $B(G + H) = \langle G_1, G_2, \ldots, G_t \rangle$.

Details about Step 2

Properties of a good pair (G, H)

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \mathsf{Yes}_{\mathbf{Reach}}$.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \mathsf{Yes}_{\mathsf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \mathsf{Yes}_{\mathbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \mathsf{Yes}_{\mathbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \mathrm{Yes}_{\mathbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \mathrm{Yes}_{\mathbf{Reach}}$.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \text{Yes}_{\textbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \text{Yes}_{\textbf{Reach}}$.

If G, H are both in $\text{Yes}_{\textbf{Reach}}$ then either (G, H) or (H, G) is good.

Let π, ρ be unique s to t paths in H, G, respectively.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \text{Yes}_{\textbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \text{Yes}_{\textbf{Reach}}$.

If G, H are both in $\text{Yes}_{\textbf{Reach}}$ then either (G, H) or (H, G) is good.

Let π, ρ be unique s to t paths in H, G, respectively. $(\pi \neq \rho)$.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \text{Yes}_{\textbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \text{Yes}_{\textbf{Reach}}$.

If G, H are both in $\text{Yes}_{\textbf{Reach}}$ then either (G, H) or (H, G) is good.

Let π, ρ be unique s to t paths in H, G, respectively. $(\pi \neq \rho)$.

If neither good, then each π and ρ are reachable paths in $2/3$ of the G_is.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \text{Yes}_{\textbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \text{Yes}_{\textbf{Reach}}$.

If G, H are both in $\text{Yes}_{\textbf{Reach}}$ then either (G, H) or (H, G) is good.

Let π, ρ be unique s to t paths in H, G, respectively. $(\pi \neq \rho)$.

If neither good, then each π and ρ are reachable paths in $2/3$ of the G_is.

This means $> 1/3$ of G_is have two distinct s to t paths.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \text{Yes}_{\textbf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \text{Yes}_{\textbf{Reach}}$.

If G, H are both in $\text{Yes}_{\textbf{Reach}}$ then either (G, H) or (H, G) is good.

Let π, ρ be unique s to t paths in H, G, respectively. $(\pi \neq \rho)$.

If neither good, then each π and ρ are reachable paths in $2/3$ of the G_is.

This means $> 1/3$ of G_is have two distinct s to t paths.
This contradicts the hypothesis of B.

Details about Step 2

Properties of a good pair (G, H)

If (G, H) is good then $G \in \mathsf{Yes}_{\mathsf{Reach}}$.

If (G, H) good then there is a ρ such that $\rho \neq \pi$ and ρ is a unique reachable path in some G_i, that ρ is a reachable path in G. Therefore $G \in \mathsf{Yes}_{\mathsf{Reach}}$.

If G, H are both in $\mathsf{Yes}_{\mathsf{Reach}}$ then either (G, H) or (H, G) is good.

Let π, ρ be unique s to t paths in H, G, respectively. $(\pi \neq \rho)$.

If neither good, then each π and ρ are reachable paths in $2/3$ of the G_is.

This means $> 1/3$ of G_is have two distinct s to t paths.
This contradicts the hypothesis of B.

Given H, π as advice and G as input, whether (G, H) is good or not can be decided in L/poly.

Wrap-up

Design the advice strings

As advice we need $(H_1, \pi_1), (H_2, \pi_2), \ldots, (H_\ell, \pi_\ell)$.

Wrap-up

Design the advice strings

As advice we need $(H_1, \pi_1), (H_2, \pi_2), \ldots, (H_\ell, \pi_\ell)$.

The advice ensures that

Wrap-up

Design the advice strings

As advice we need $(H_1, \pi_1), (H_2, \pi_2), \ldots, (H_\ell, \pi_\ell)$.

The advice ensures thatif $G \in \mathsf{Yes}_{\mathbf{Reach}}$ then there is an H_i such that $H_i \in \mathsf{Yes}_{\mathbf{Reach}}$ and π_i is corresponding path.

Wrap-up

Design the advice strings

As advice we need $(H_1, \pi_1), (H_2, \pi_2), \ldots, (H_\ell, \pi_\ell)$.

The advice ensures thatif $G \in \text{Yes}_{\textbf{Reach}}$ then there is an H_i such that $H_i \in \text{Yes}_{\textbf{Reach}}$ and π_i is corresponding path.

If $G \in \text{No}_{\textbf{Reach}}$, then each $H_i \in \text{No}_{\textbf{Reach}}$.

Wrap-up

Design the advice strings

As advice we need $(H_1, \pi_1), (H_2, \pi_2), \ldots, (H_\ell, \pi_\ell)$.

The advice ensures thatif $G \in \mathrm{Yes}_{\mathbf{Reach}}$ then there is an H_i such that $H_i \in \mathrm{Yes}_{\mathbf{Reach}}$ and π_i is corresponding path.

If $G \in \mathrm{No}_{\mathbf{Reach}}$, then each $H_i \in \mathrm{No}_{\mathbf{Reach}}$.

Putting it together

Overall, this gives a L/poly algorithm for **PrUReach**.

Thank You!