Infinite-state Liveness Checking with RLive

Alessandro Cimatti, Alberto Griggio, Christopher Johannsen, Kristin Yvonne Rozier, and Stefano Tonetta

FONDAZIONE BRUNO KESSLER

IOWA STATE UNIVERSITY OF SCIENCE AND TECHNOLOGY

Liveness Checking

- “Something good eventually always happens”

Liveness Checking

- "Something good eventually always happens"
- Proving queries of the form: $S \models \Diamond \Box q$

Liveness Checking

- "Something good eventually always happens"
- Proving queries of the form: $S \models \Diamond \Box q$
- Can reduce any LTL formula to formulas of the form $\Diamond \Box \varphi$

Liveness Checking

- “Something good eventually always happens”
- Proving queries of the form: $S \models \Diamond \Box q$
- Can reduce any LTL formula to formulas of the form $\Diamond \Box \varphi$
- **Finite-state:**
  - Liveness-to-safety (L2S), K-Liveness, RLive

Liveness Checking

- “Something good eventually always happens”
- Proving queries of the form: $S \models \Diamond \Box q$
- Can reduce any LTL formula to formulas of the form $\Diamond \Box \varphi$
- **Finite-state:**
  - Liveness-to-safety (L2S), K-Liveness, RLive
- **Infinite-state:**
  - Abstract L2S, T2-CTL*, Ultimate-LTL

Liveness Checking

- “Something good eventually always happens”
- Proving queries of the form: $S \vDash \Diamond \Box q$
- Can reduce any LTL formula to formulas of the form $\Diamond \Box \varphi$
- **Finite-state:**
  - Liveness-to-safety (L2S), K-Liveness, **RLive**
- **Infinite-state:**
  - Abstract L2S, T2-CTL*, Ultimate-LTL
- **Extend RLive to infinite-state systems**

RLive-inf Setup

$$S = \langle X, I(X), T(X, X') \rangle$$
$$\varphi = \Diamond \Box q$$

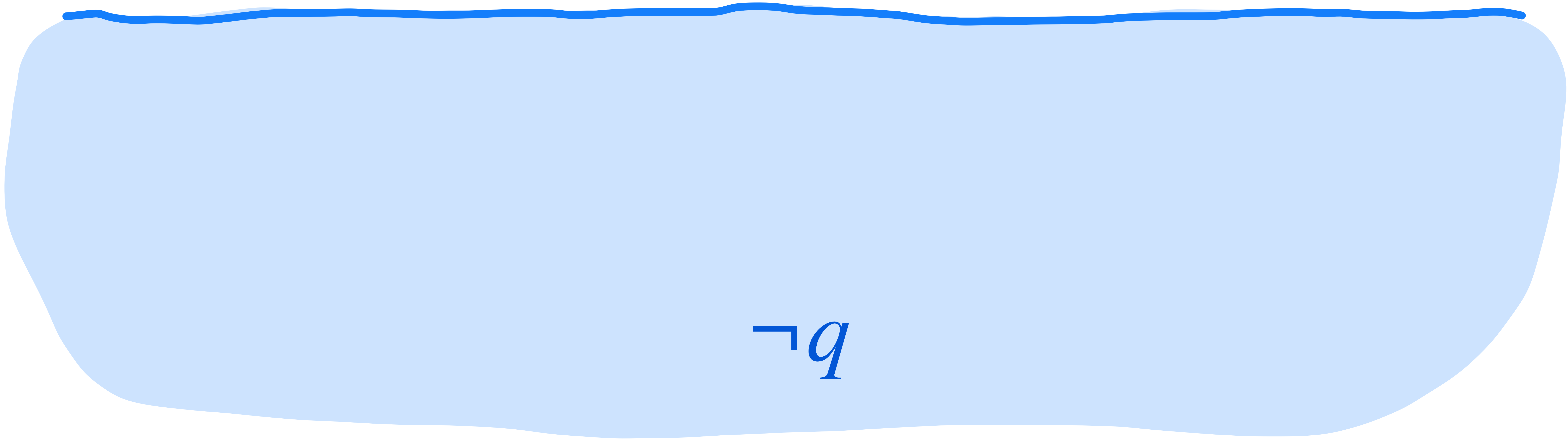
RLive–inf Setup

$$S = \langle X, I(X), T(X, X') \rangle$$
$$\varphi = \Diamond \Box q$$

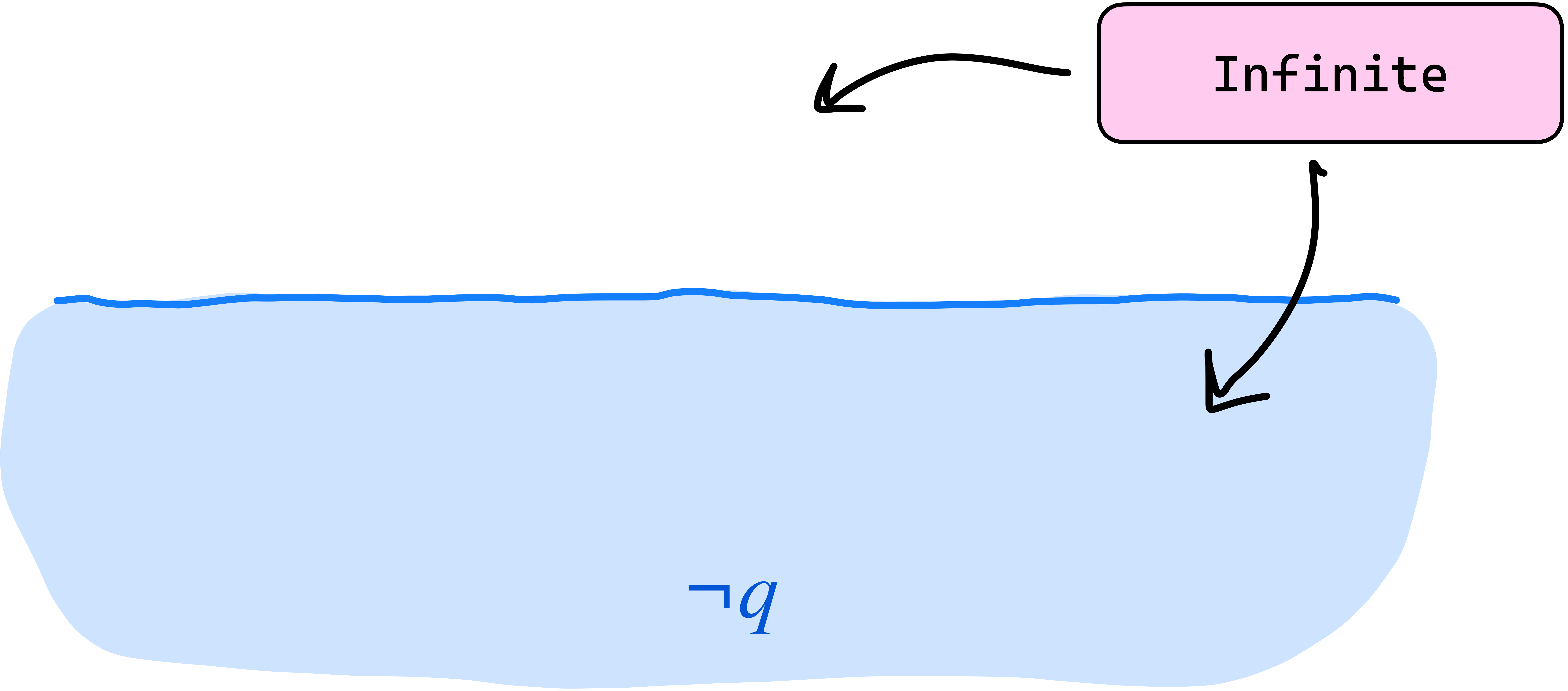
- **Falsify** φ by finding counterexample lasso $\pi = \pi_{prefix} \cdot \pi_{loop}^{\omega}$
- **Prove** φ by showing $\neg q$ is only finitely reachable

$$S = \langle X, I(X), T(X, X') \rangle$$

$$\varphi = \Diamond \Box q$$

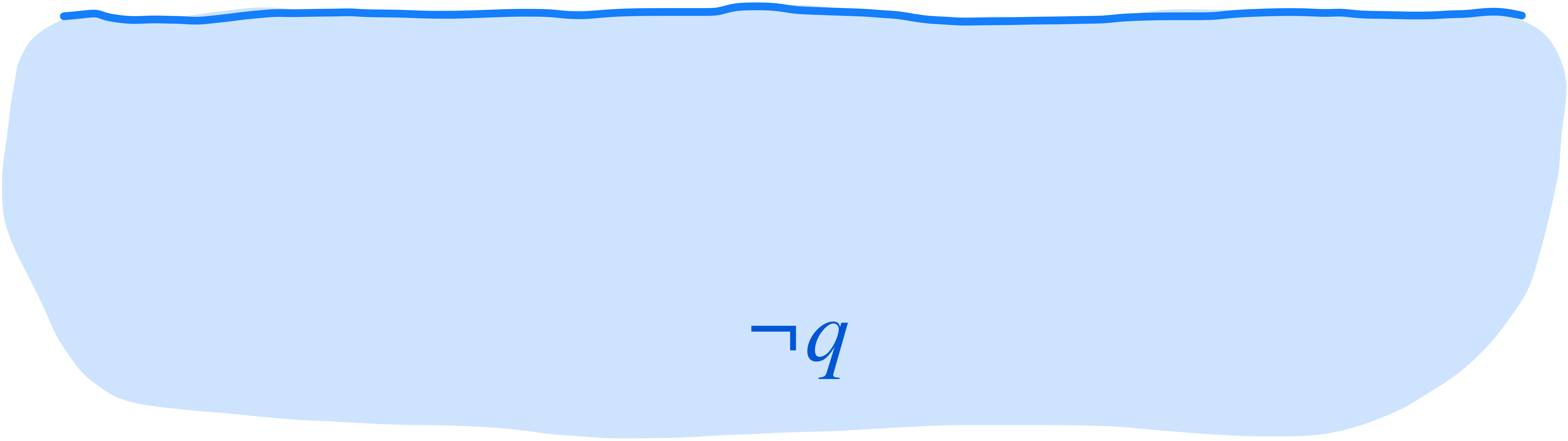


$$\varphi = \Diamond \Box q$$



$$S = \langle X, I(X), T(X, X') \rangle$$

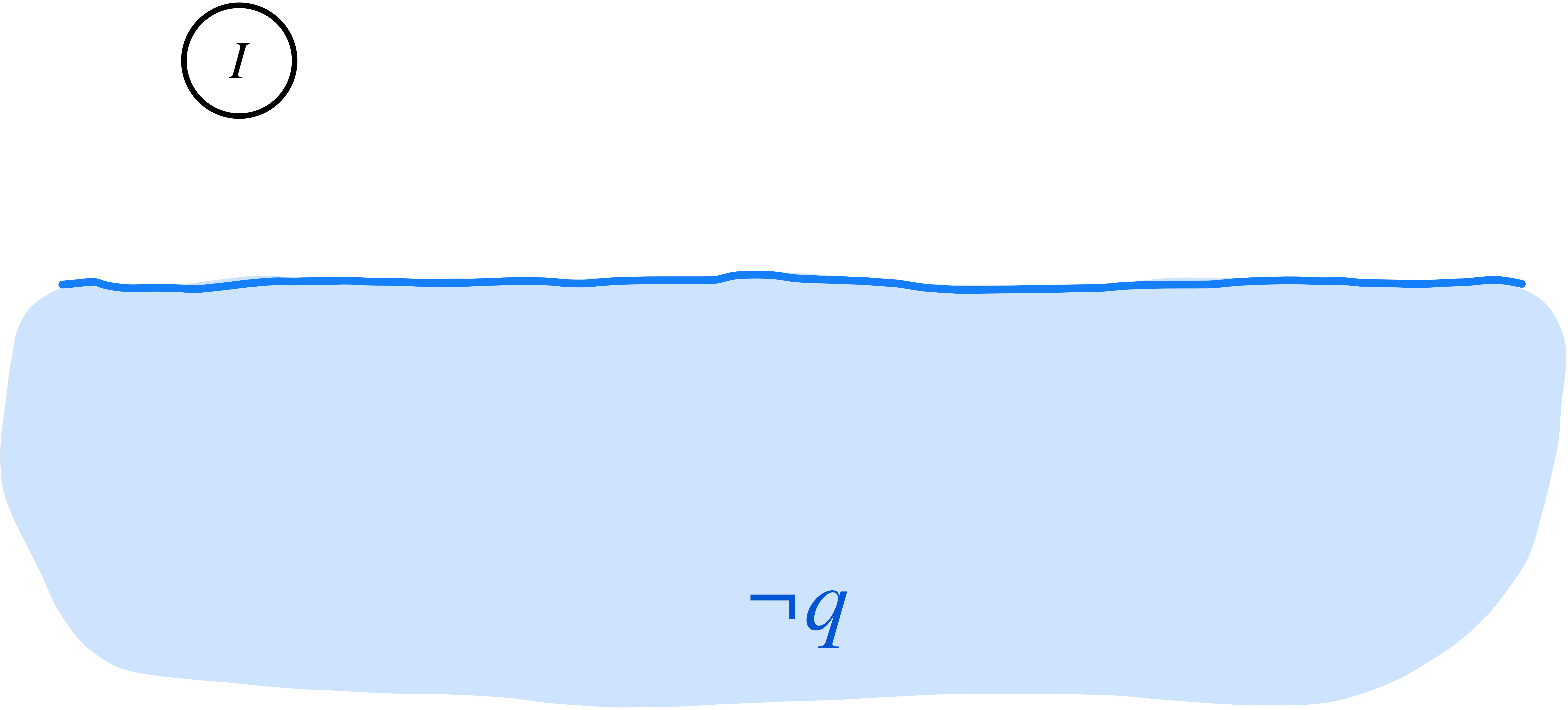
$$\varphi = \Diamond \Box q$$



$$S = \langle X, I(X), T(X, X') \rangle$$

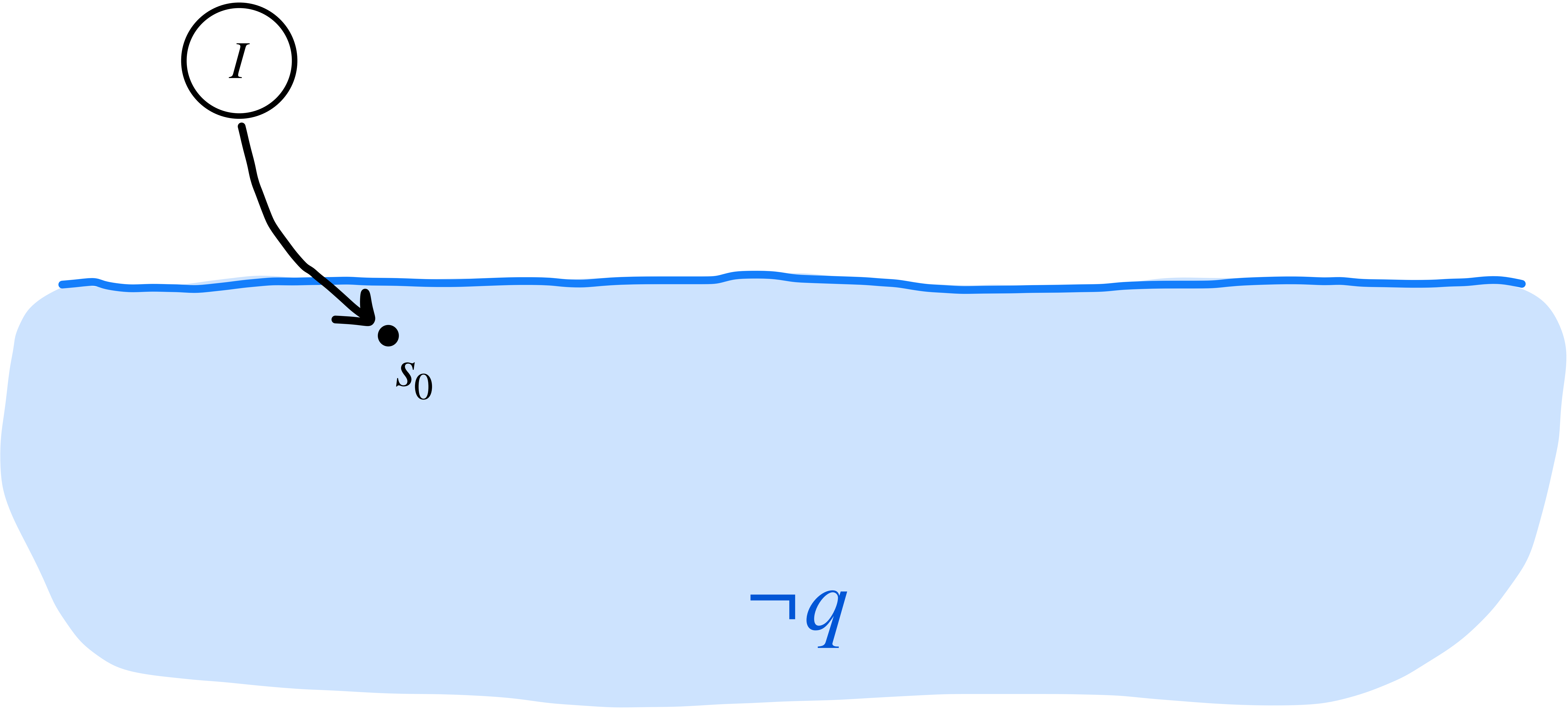
Search Stack:

$$\varphi = \Diamond \Box q$$



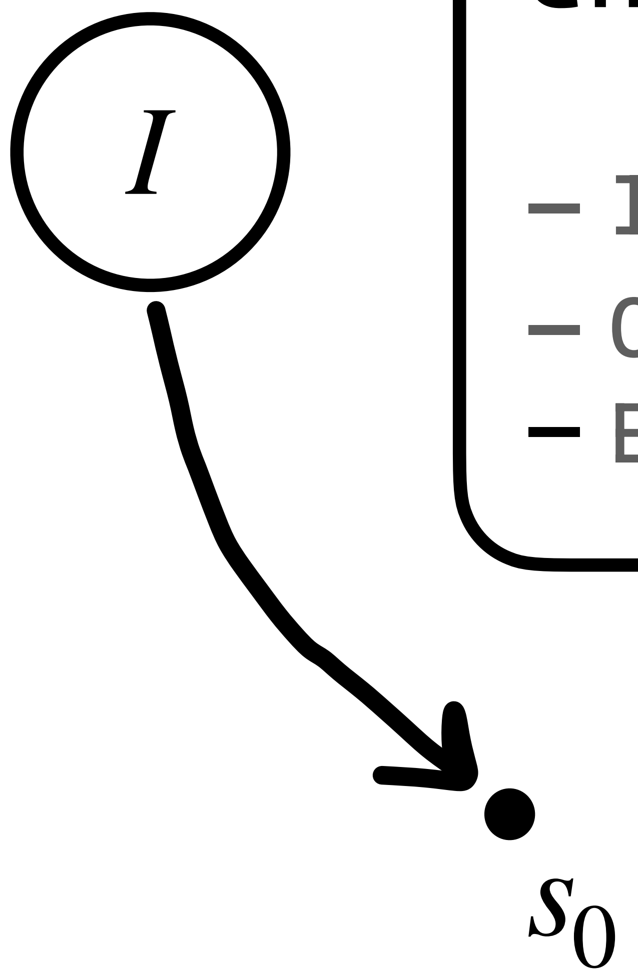
$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** s_0

$\varphi = \Diamond \Box q$



s_0

$\varphi = \Diamond \Box q$



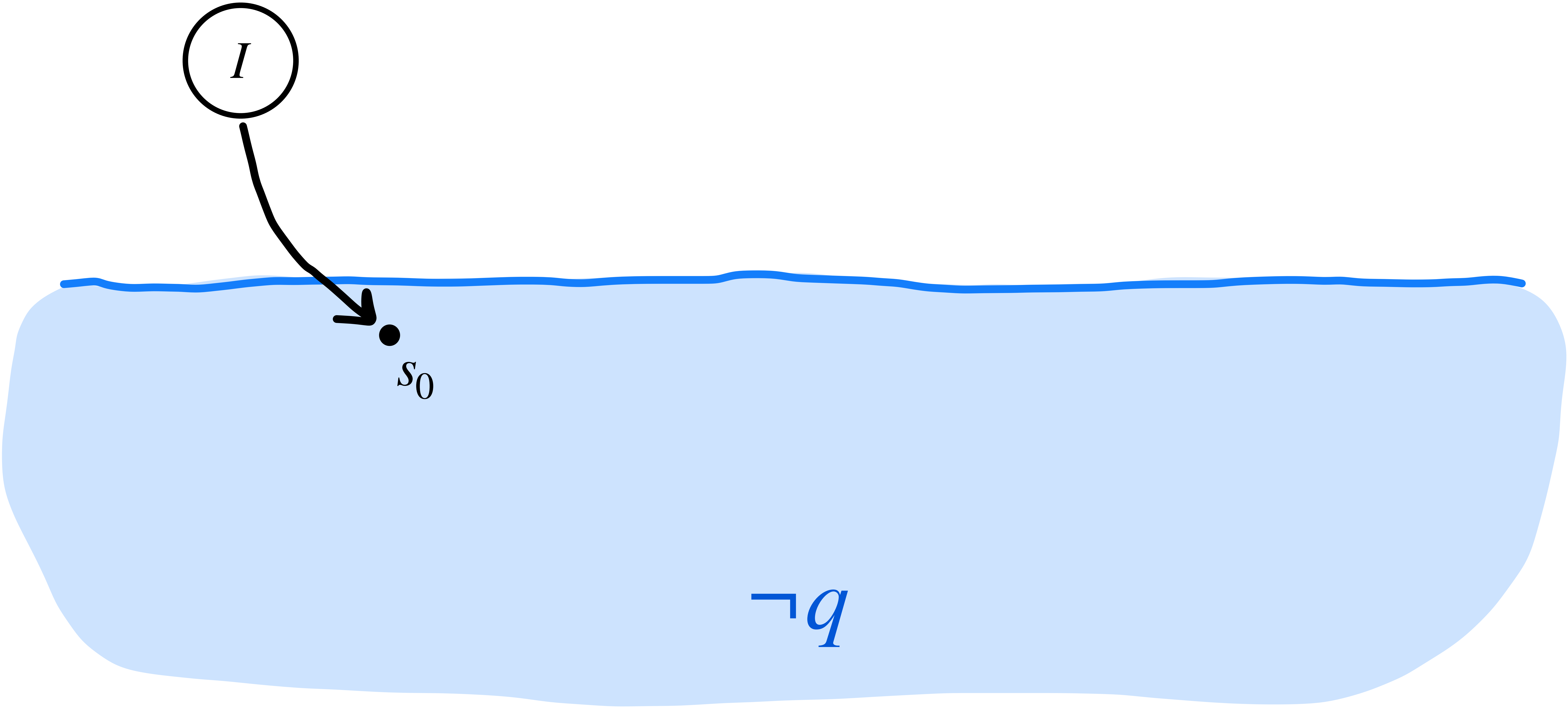
check-inv(I, T, q)

- If $\neg q$ is reachable, returns counterexample with final state s_0
- Otherwise, returns inductive invariant C
- Example implementations: ic3ia, pdkind, etc.

$\neg q$

$S = \langle X, I(X), T(X, X') \rangle$ Search Stack: s_0

$\varphi = \Diamond \Box q$

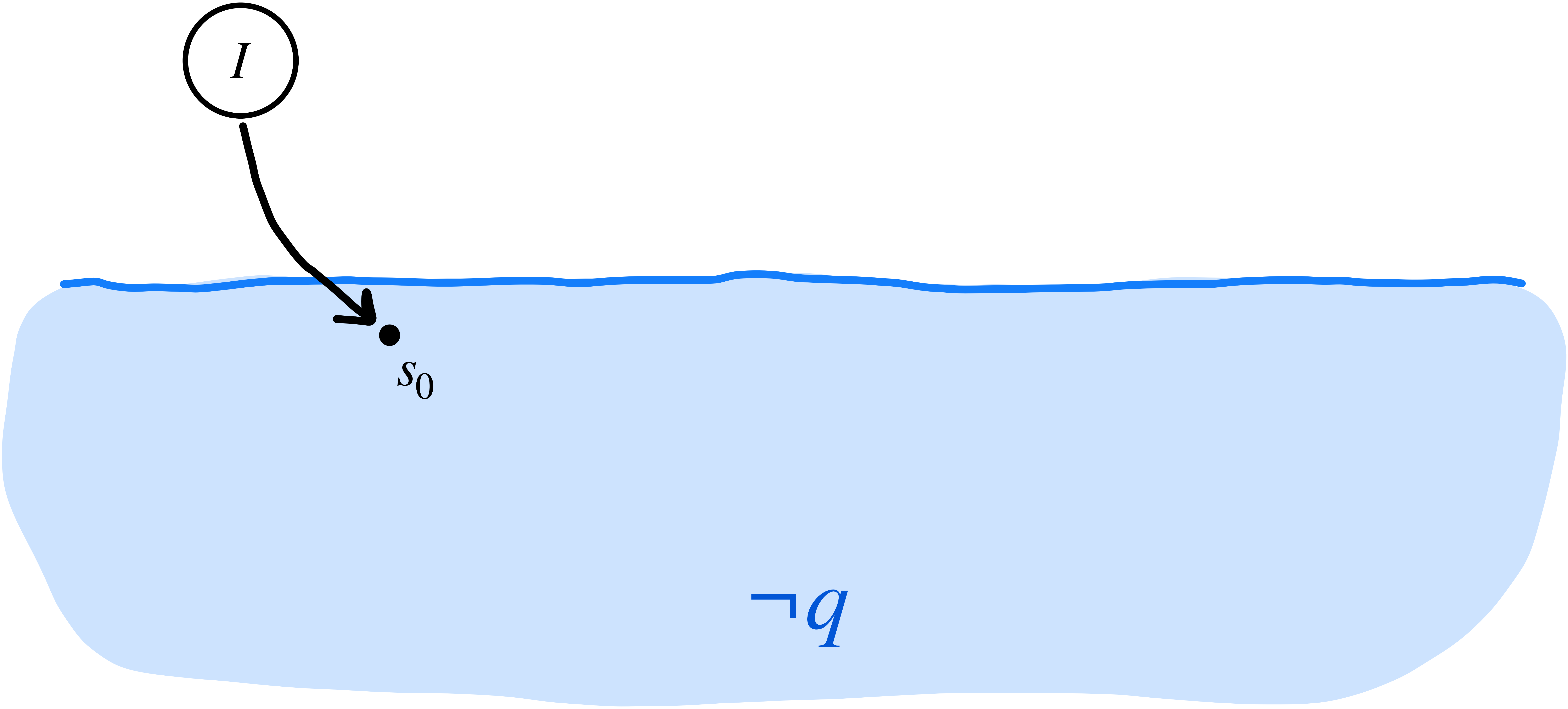


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: s_0

$\mathbb{P} = \{p_0, p_1, \ldots\}$

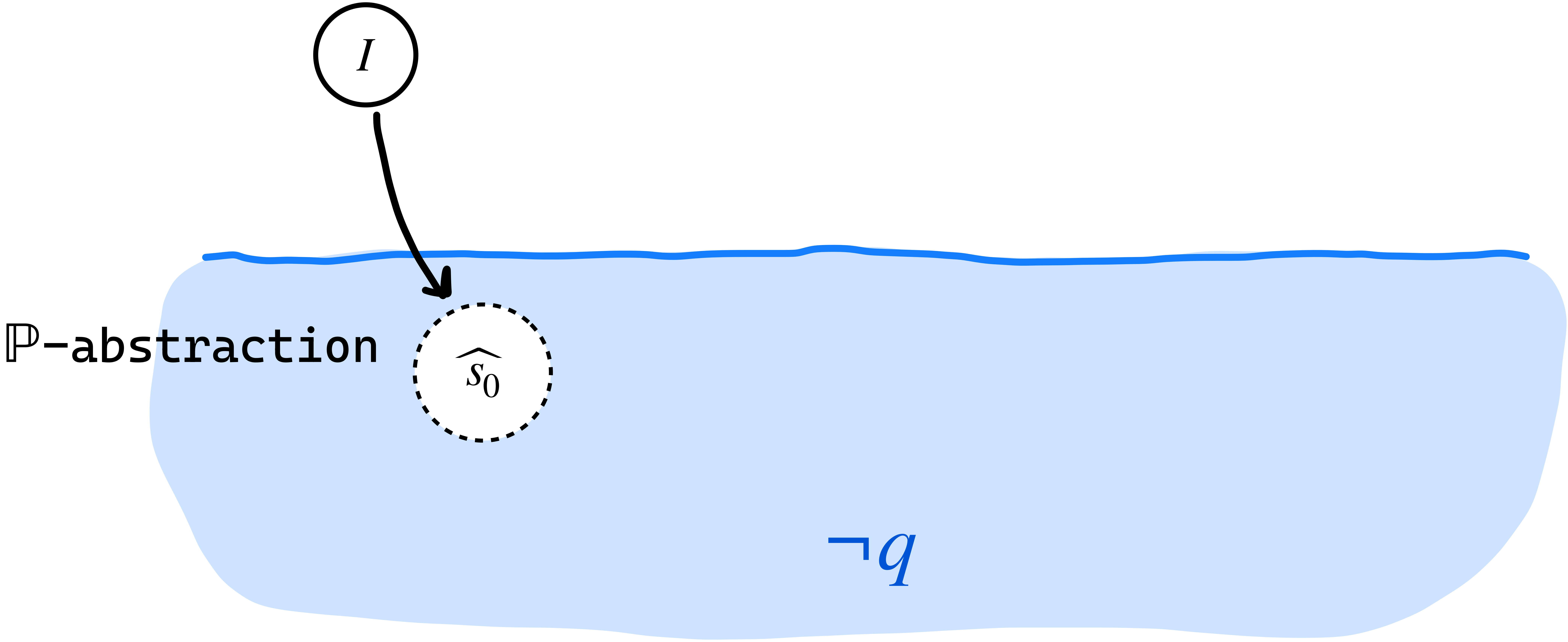


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

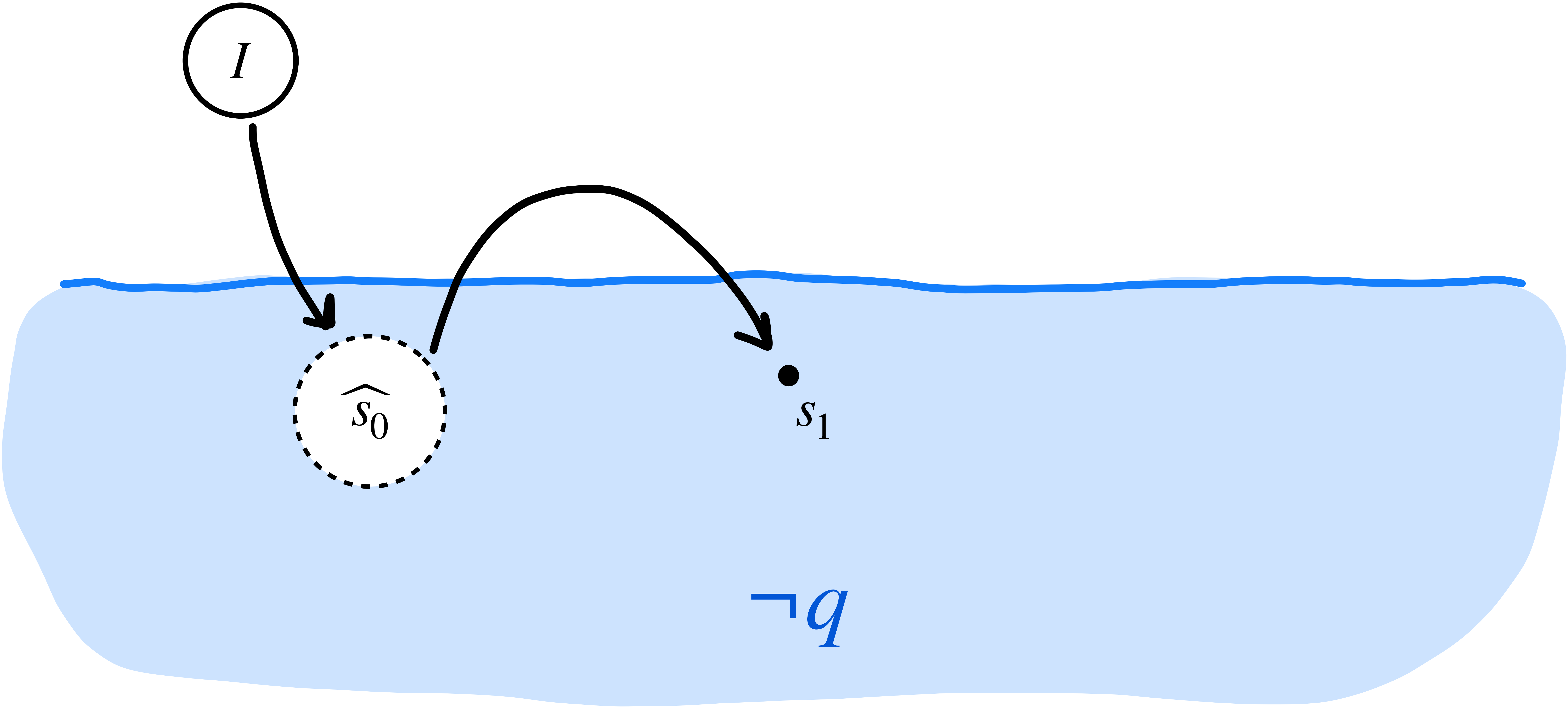


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

`Search Stack:` $\widehat{s_0}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

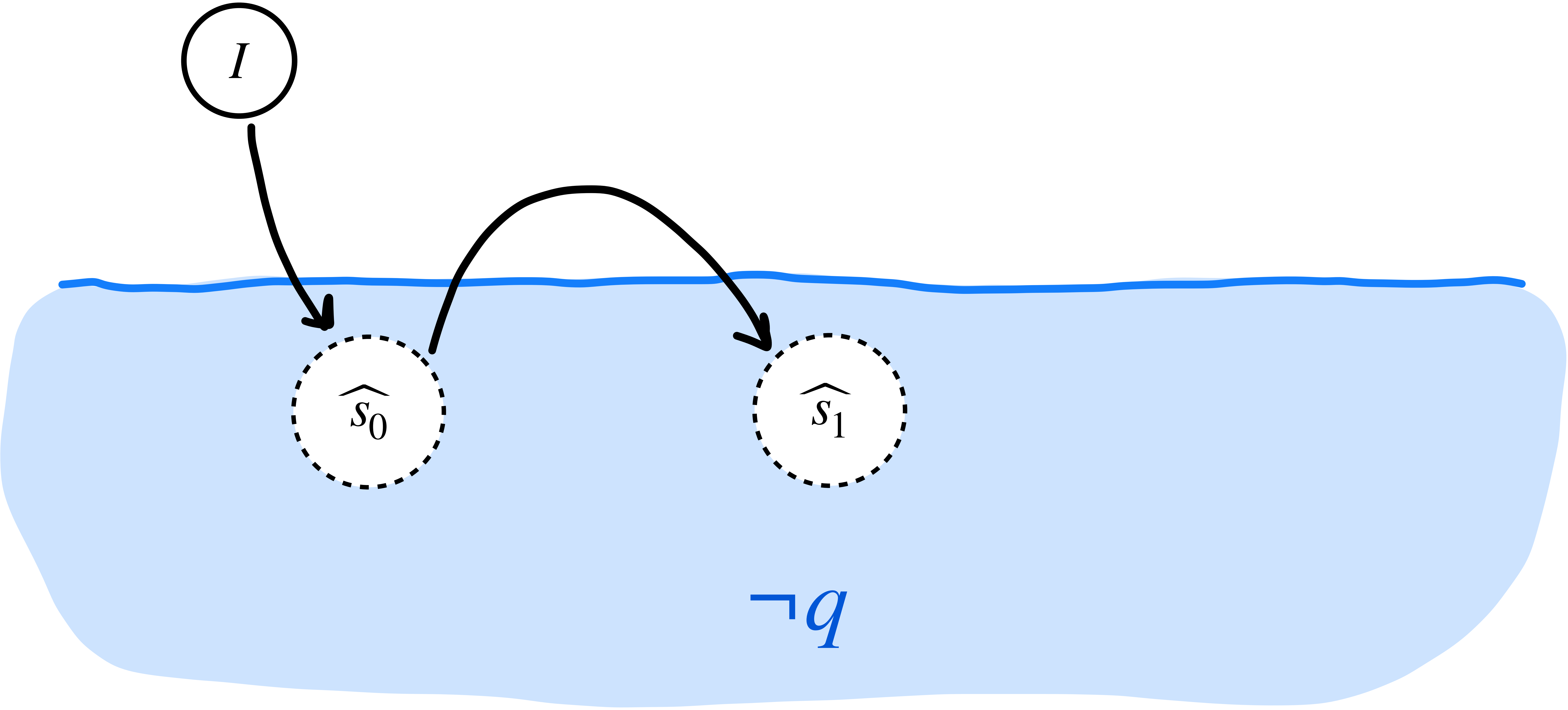


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

`Search Stack:` $\widehat{s_0}\ \widehat{s_1}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

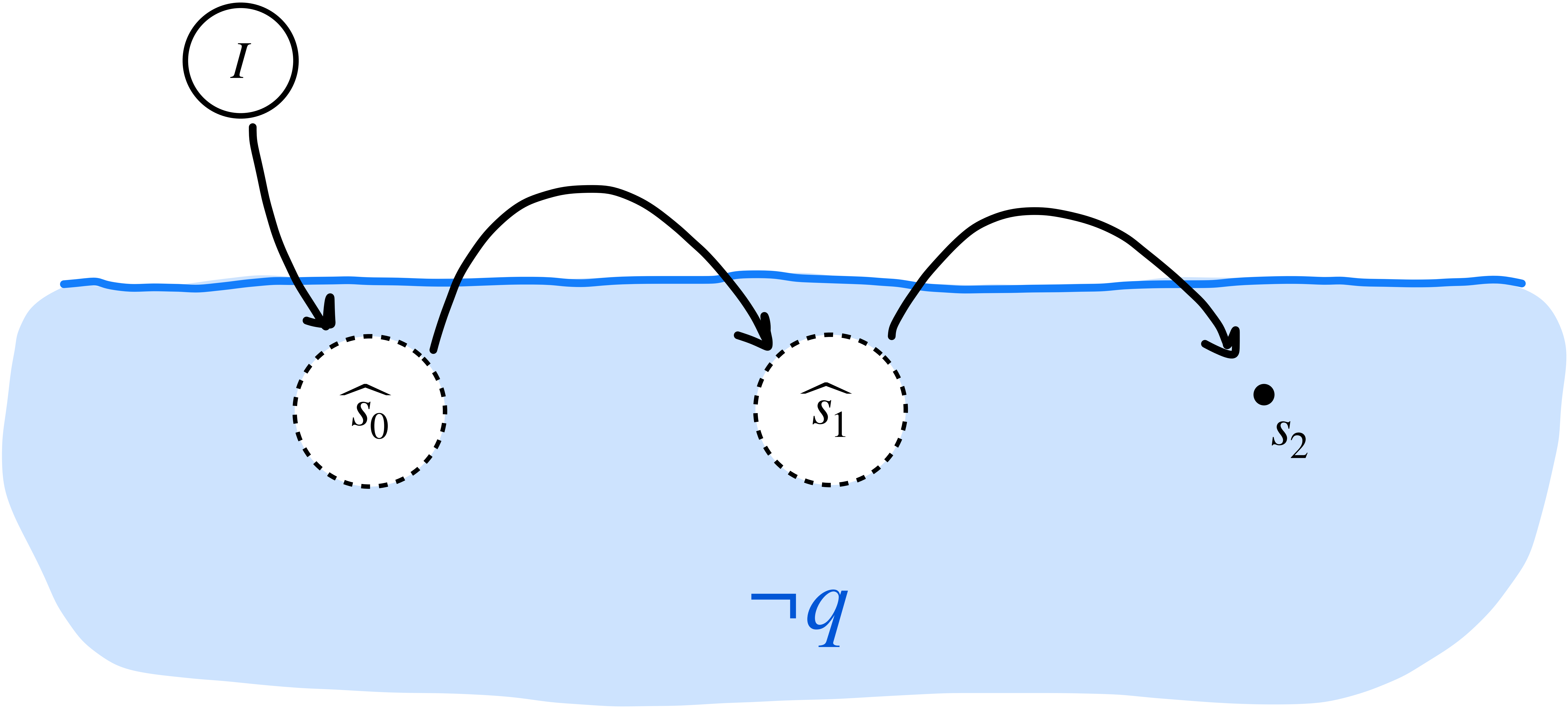


$$S = \langle X, I(X), T(X, X') \rangle$$

$$\varphi = \Diamond \Box q$$

Search Stack: $\widehat{s_0}\ \widehat{s_1}$

$$\mathbb{P} = \{p_0, p_1, \ldots\}$$

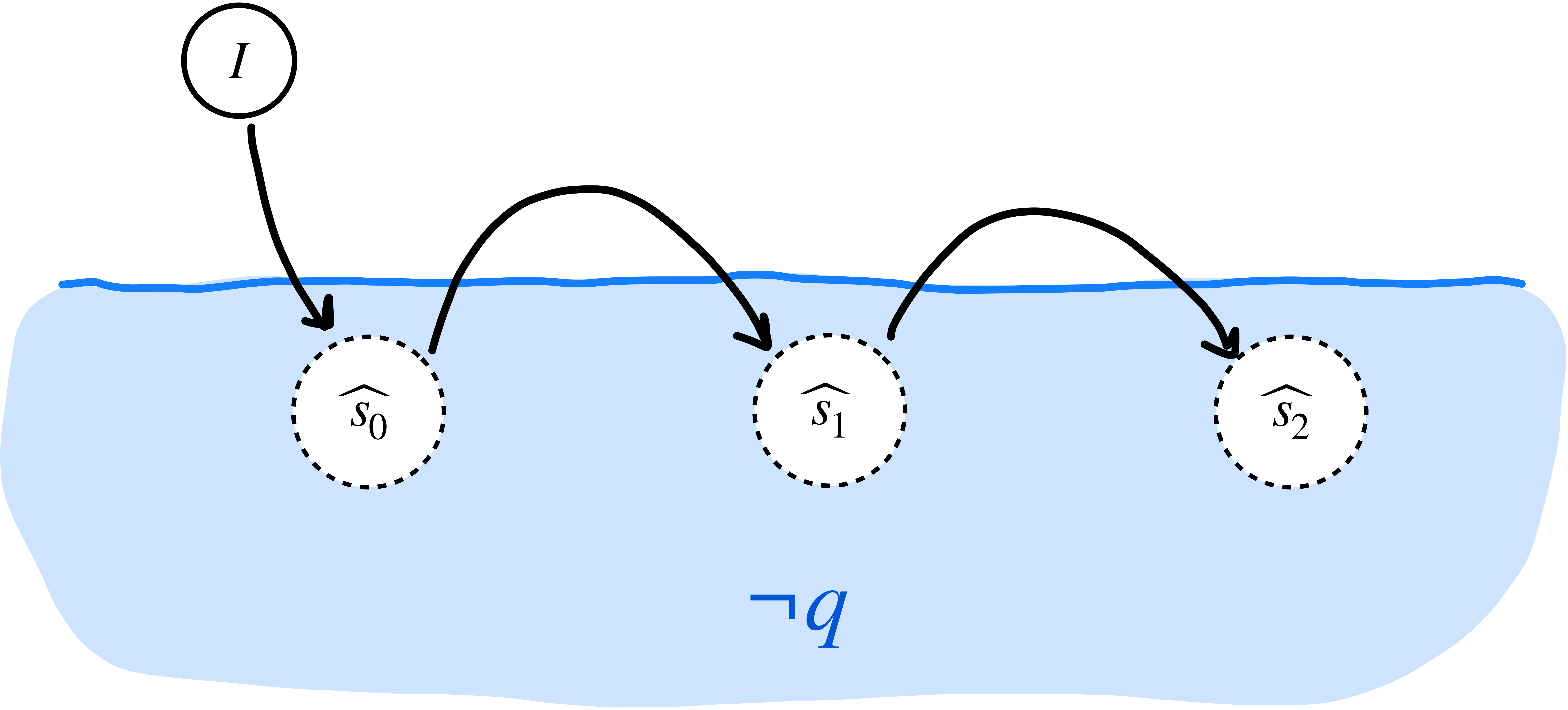


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

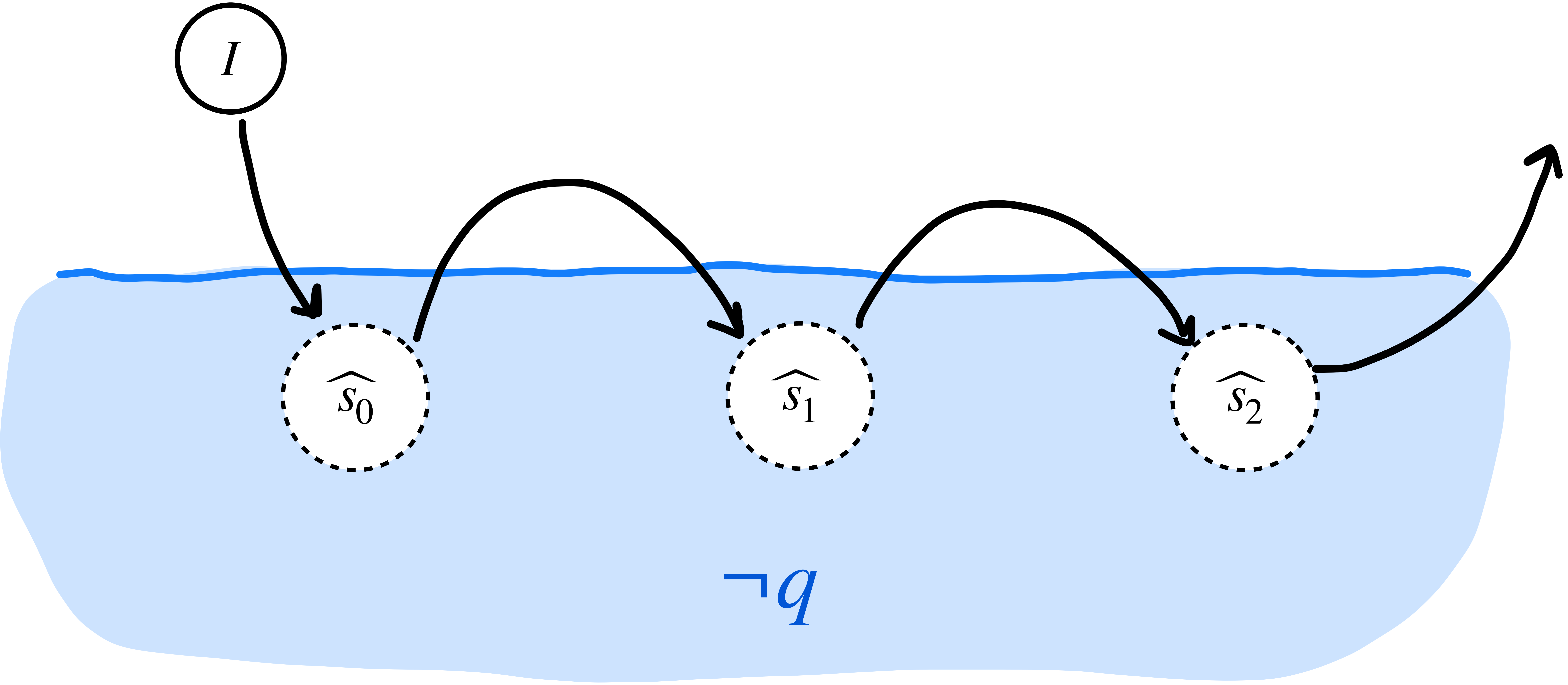


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

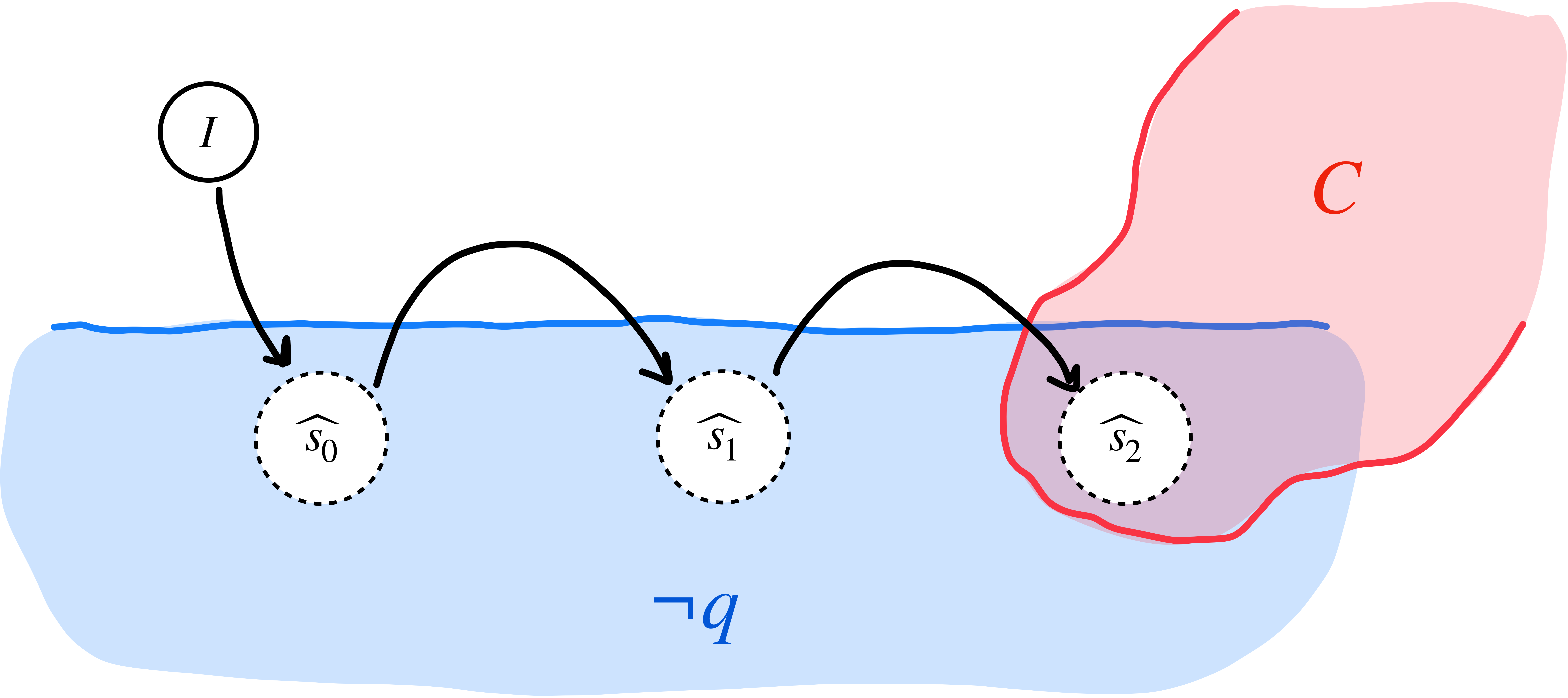


$$S = \langle X, I(X), T(X, X') \rangle$$

$$\varphi = \Diamond \Box q$$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$$\mathbb{P} = \{p_0, p_1, \ldots\}$$

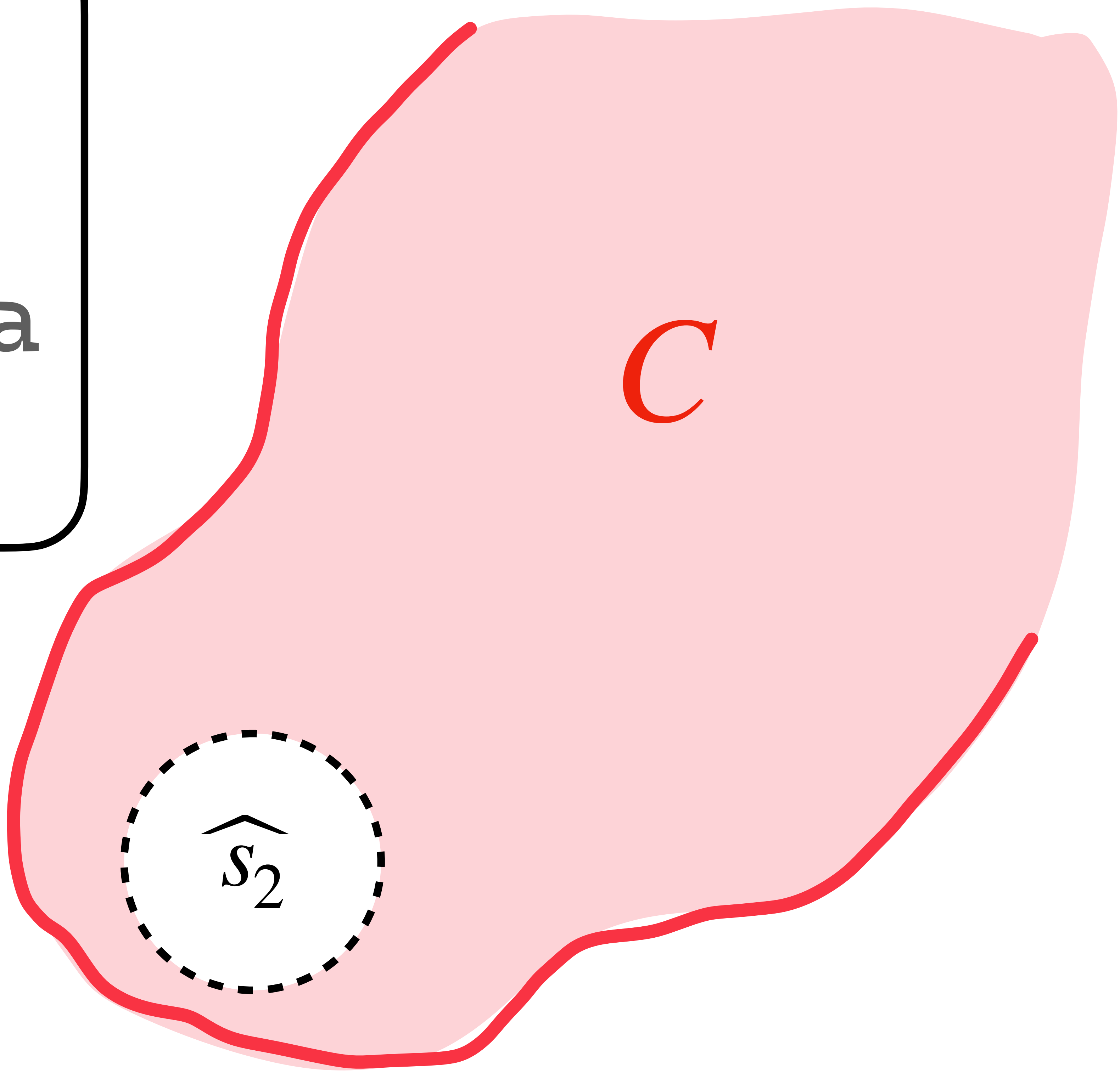


$\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\varphi = \Diamond\Box q$ $\quad$ $\mathbb{P} = \{p_0, p_1, \ldots\}$

(Shoals) $\quad C =$ `get-invariant()`

- Set of states that can reach $\neg q$ only a **finite** number of times



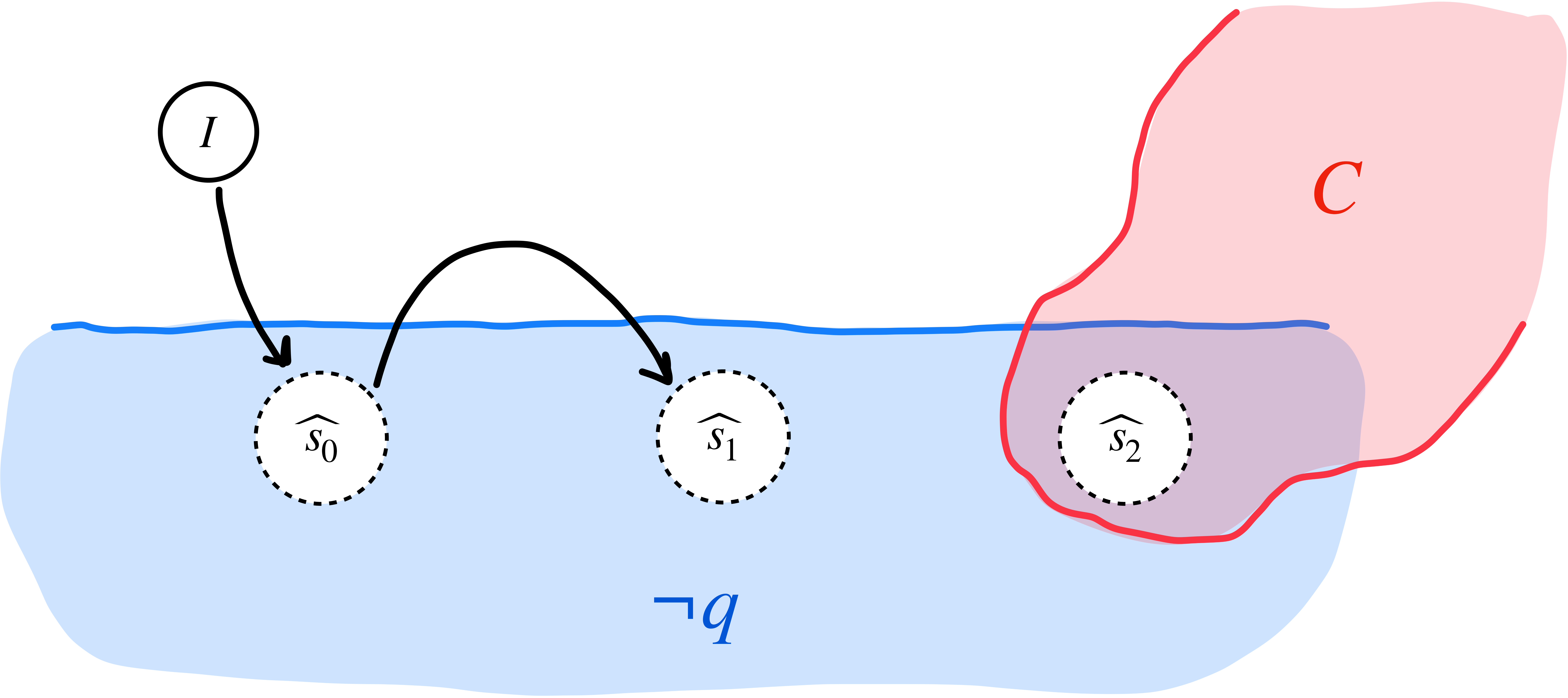
$\neg q$

$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

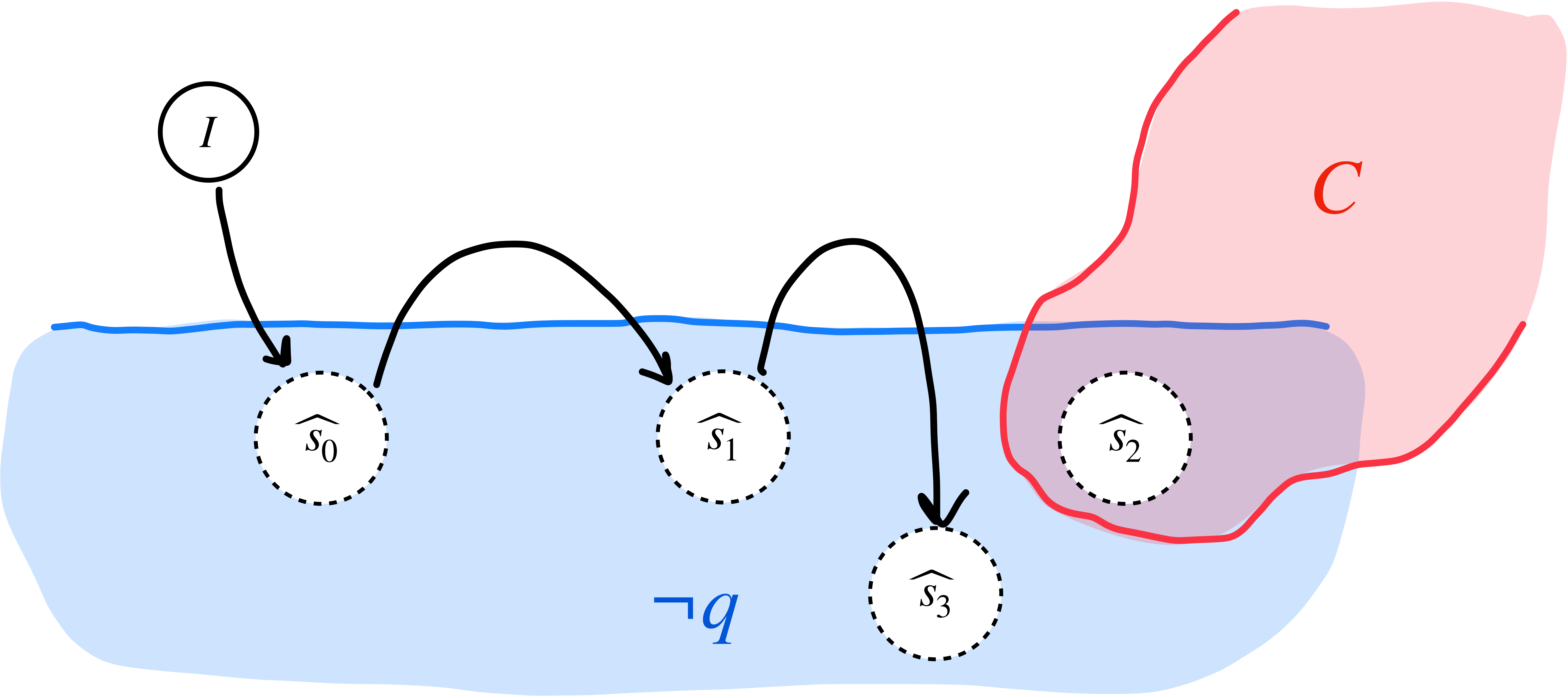


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

`Search Stack:` $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_3}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

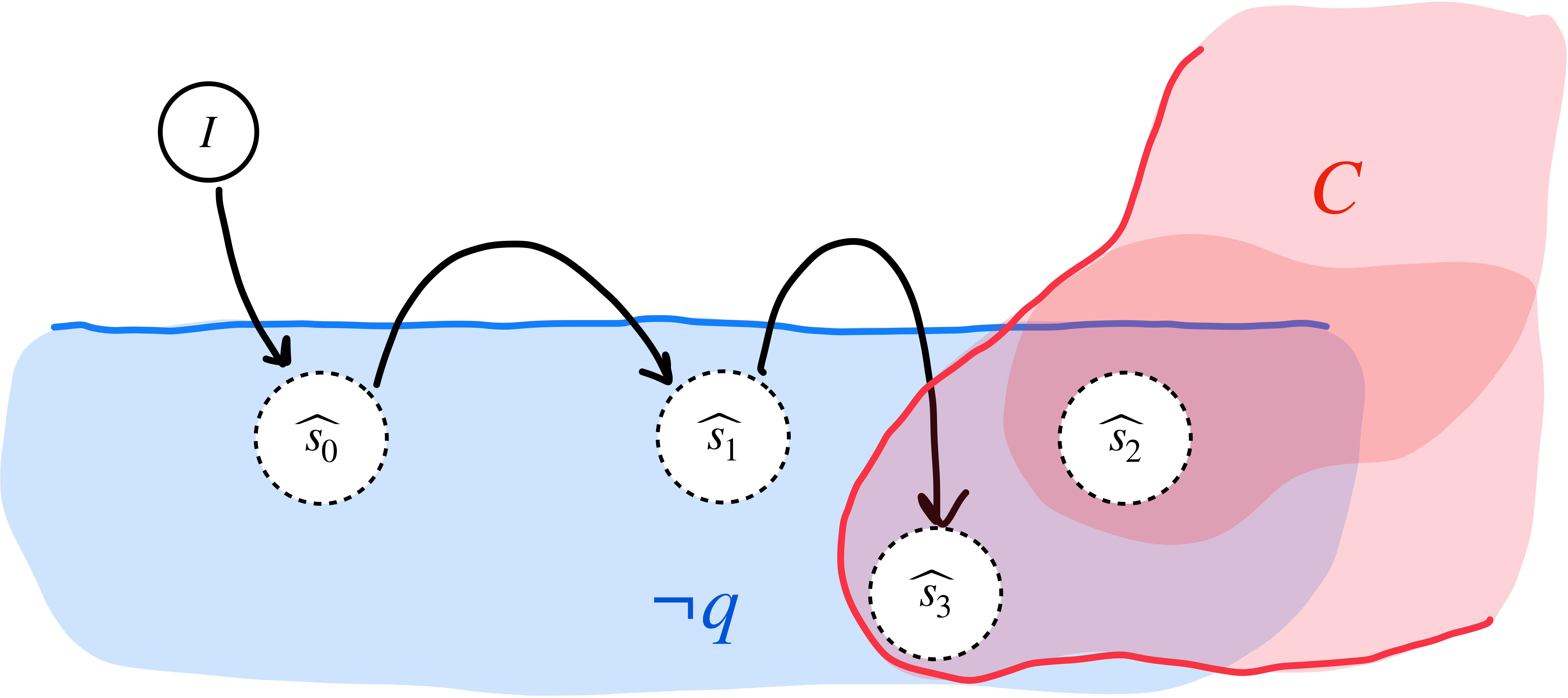


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_3}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

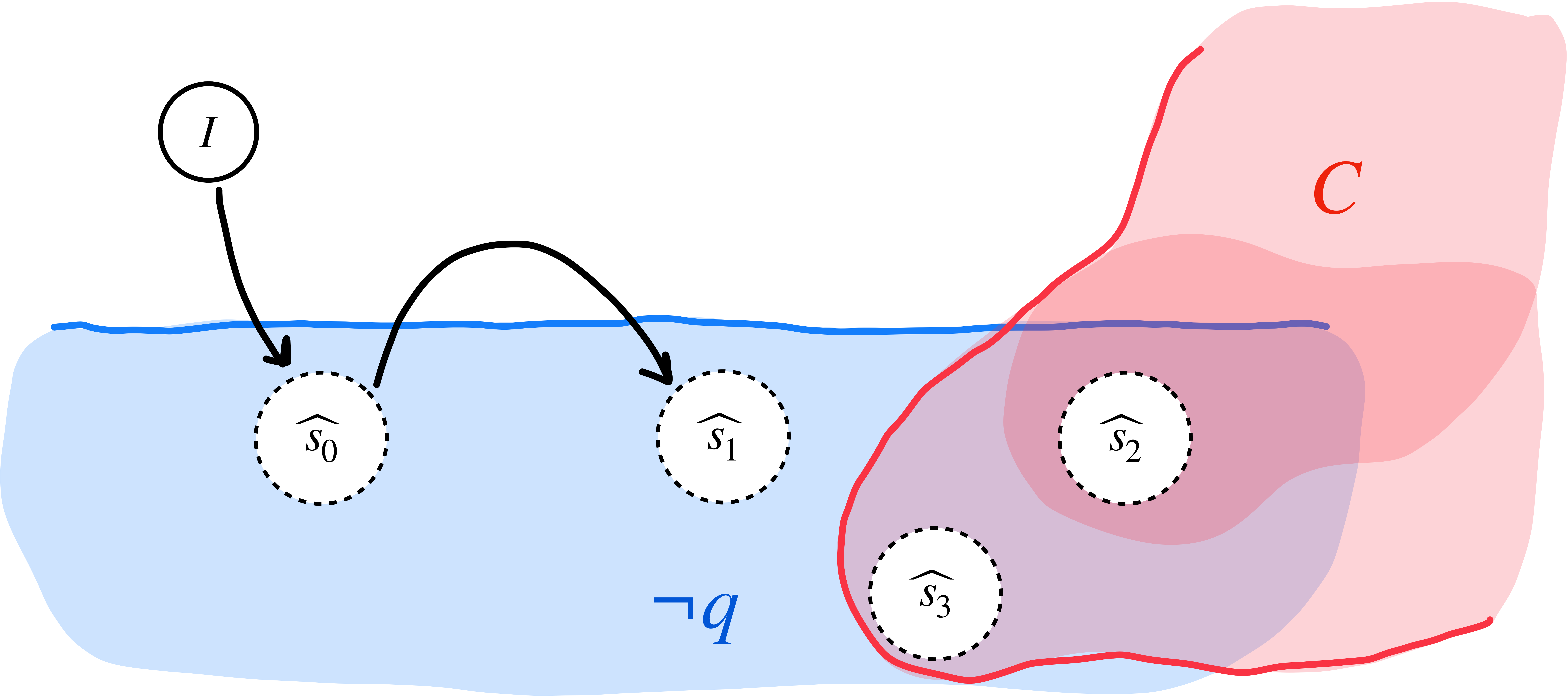


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

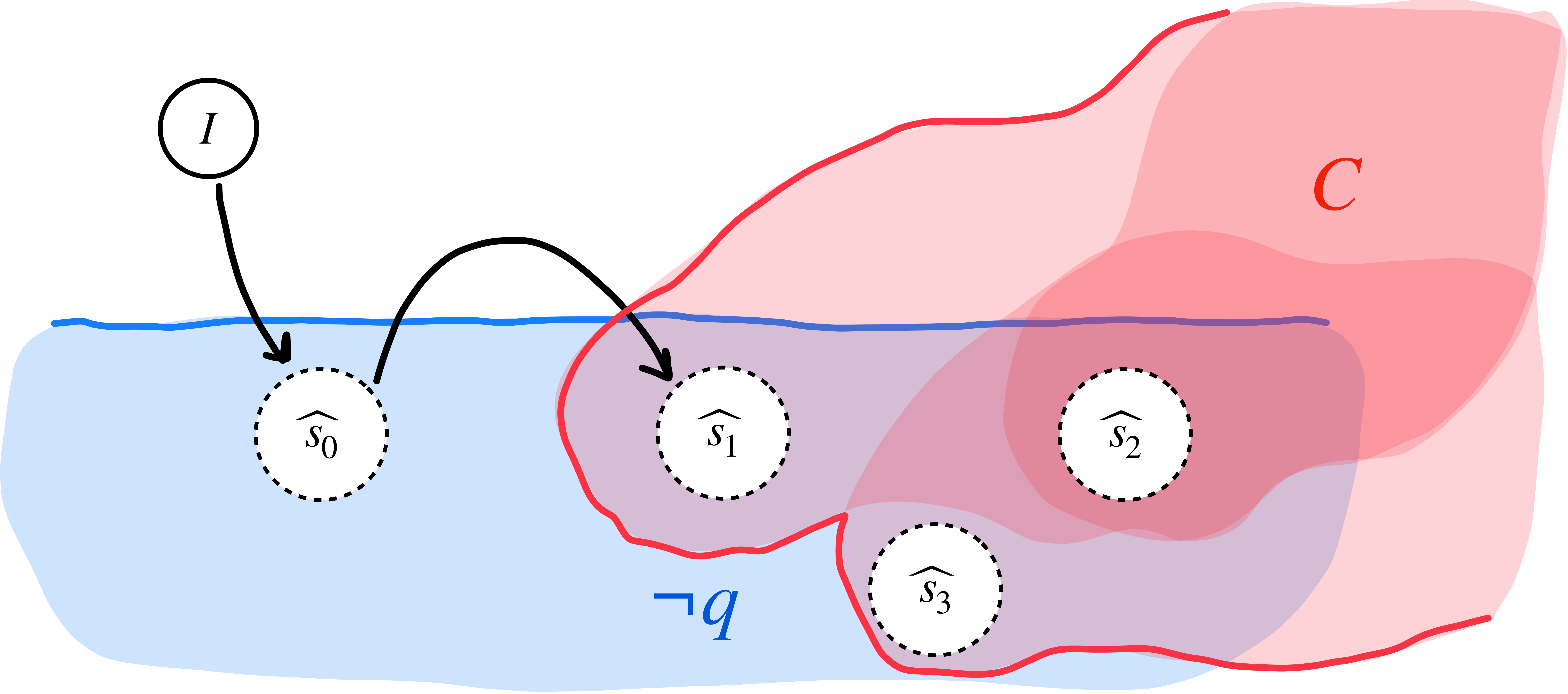


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

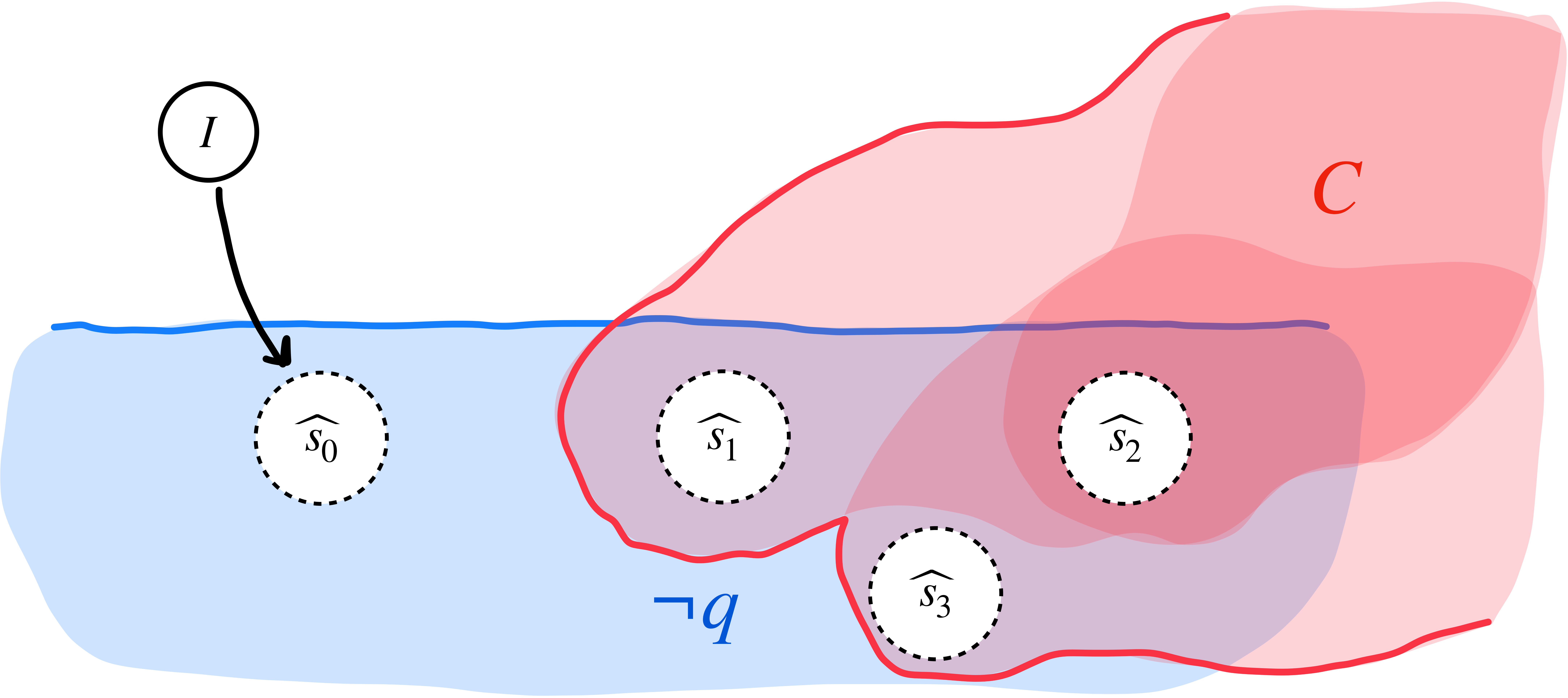


$$S = \langle X, I(X), T(X, X') \rangle$$

$$\varphi = \Diamond \Box q$$

Search Stack: $\widehat{s_0}$

$$\mathbb{P} = \{p_0, p_1, \dots\}$$

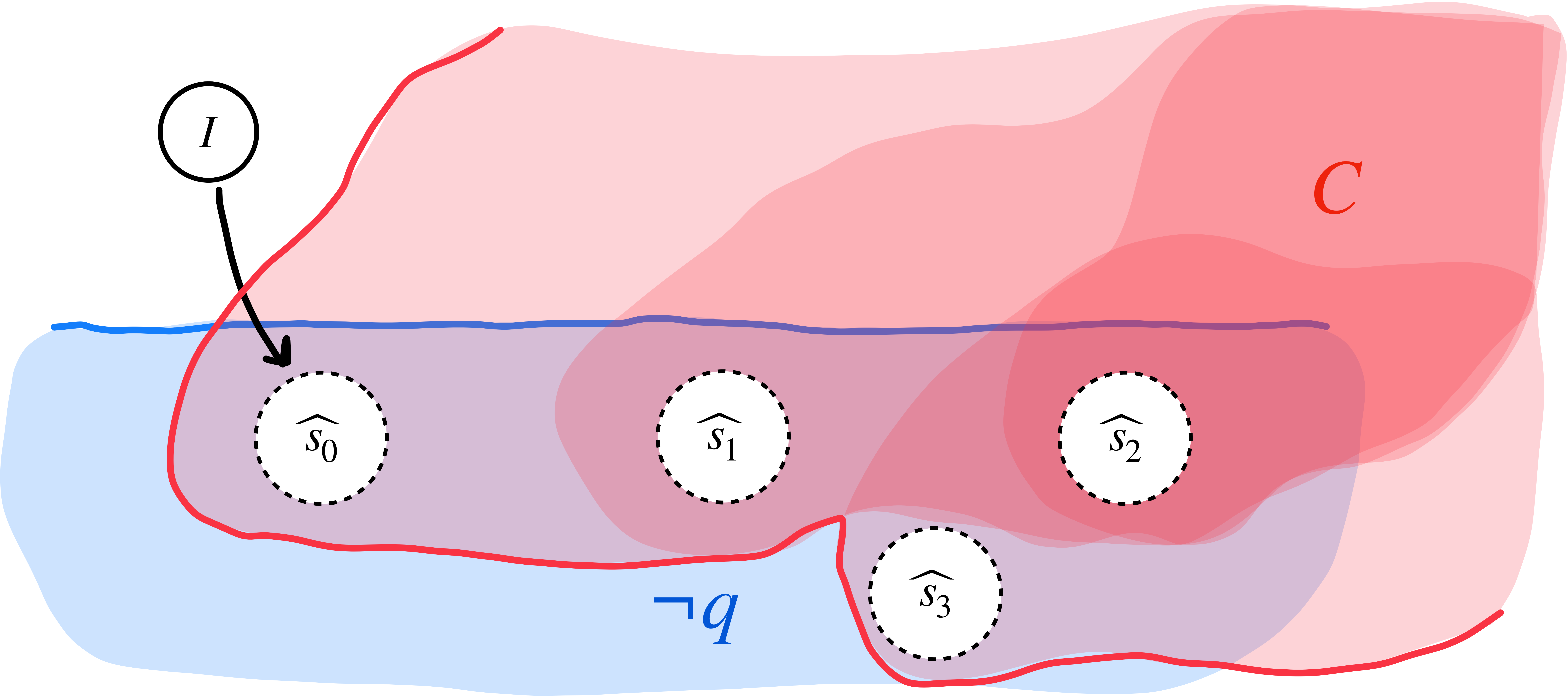


$$S = \langle X, I(X), T(X, X') \rangle$$

$$\varphi = \Diamond \Box q$$

`Search Stack:` $\widehat{s_0}$

$$\mathbb{P} = \{p_0, p_1, \ldots\}$$

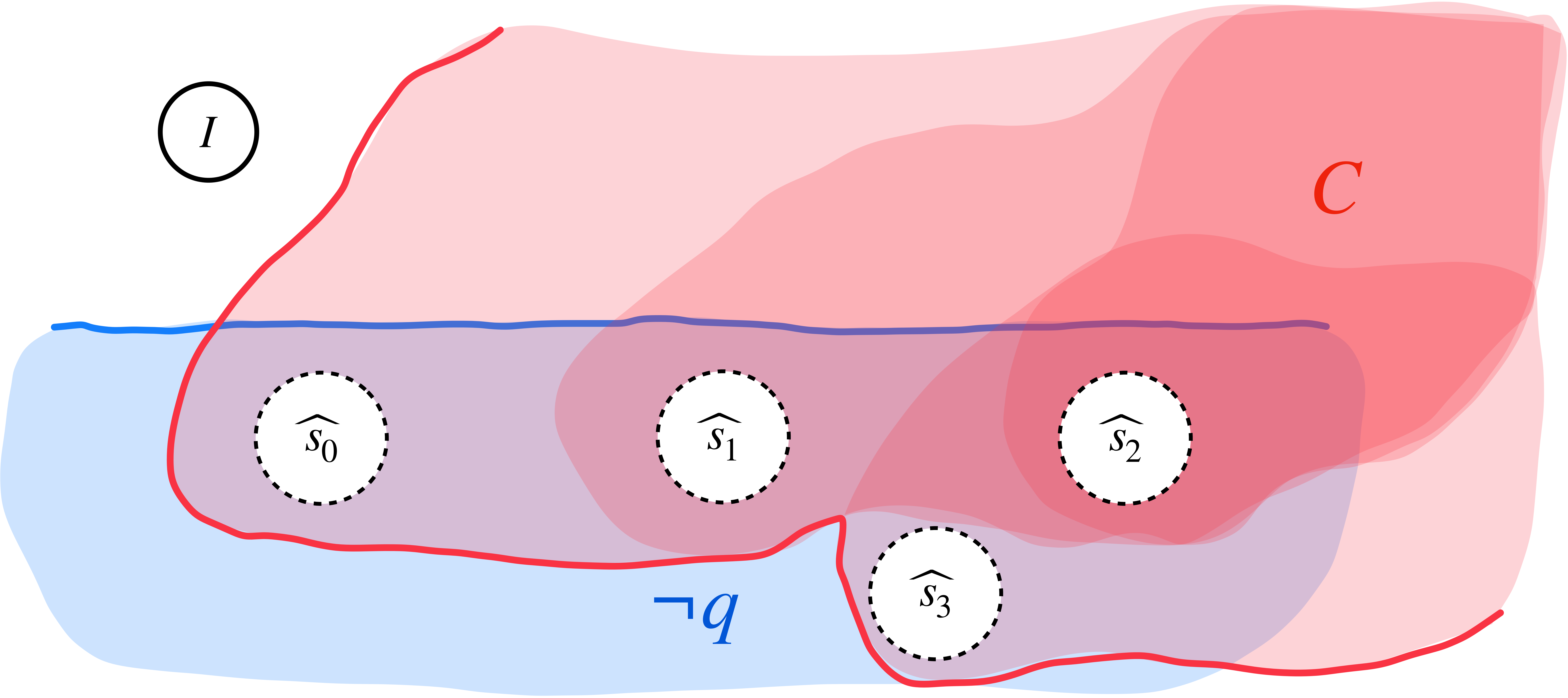


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack:

$\mathbb{P} = \{p_0, p_1, \ldots\}$

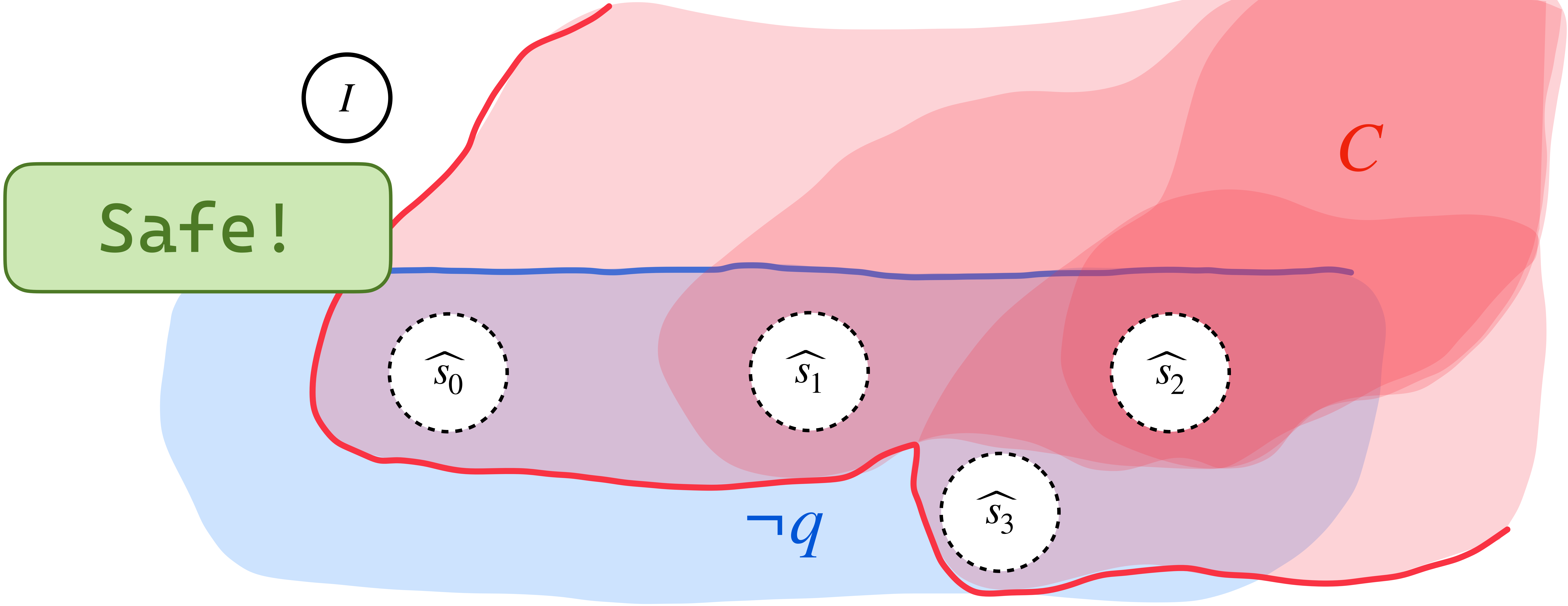


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack:

$\mathbb{P} = \{p_0, p_1, \dots\}$

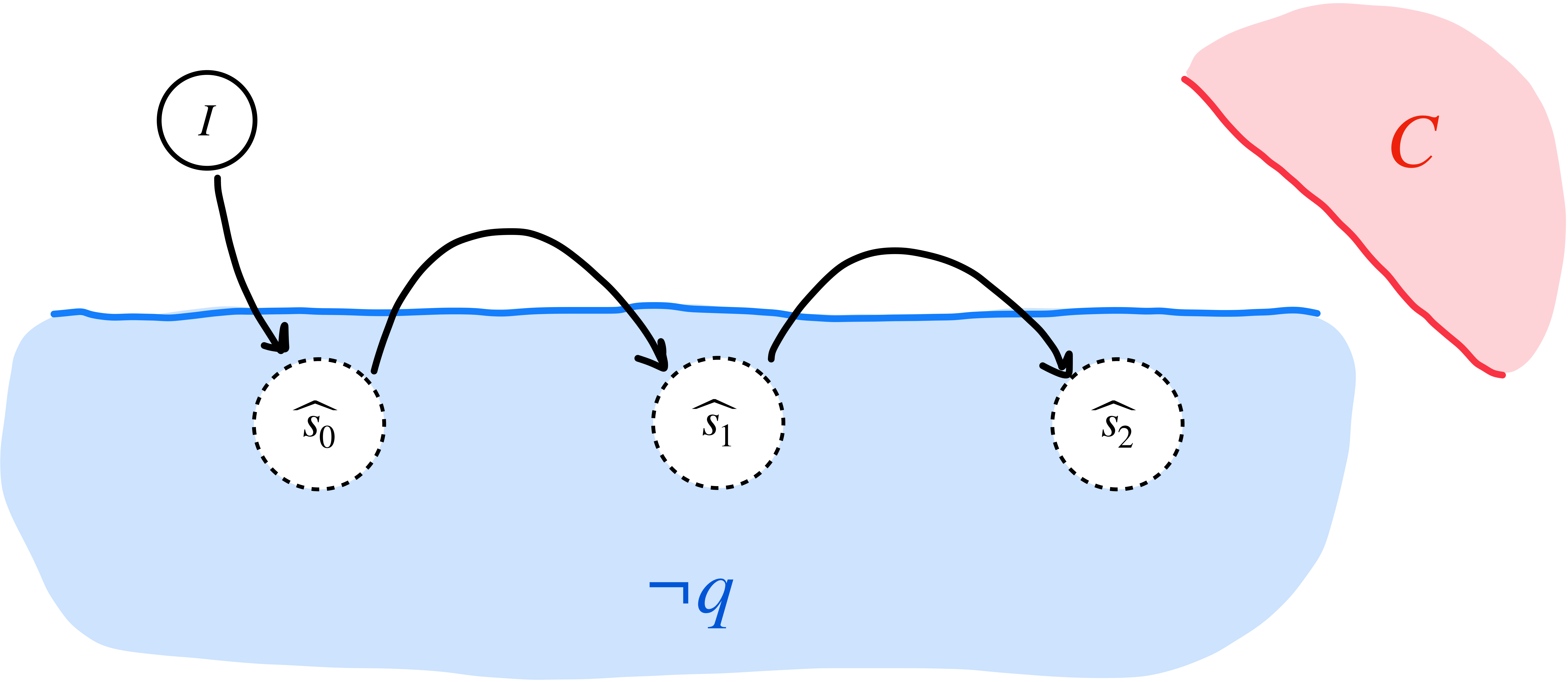


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

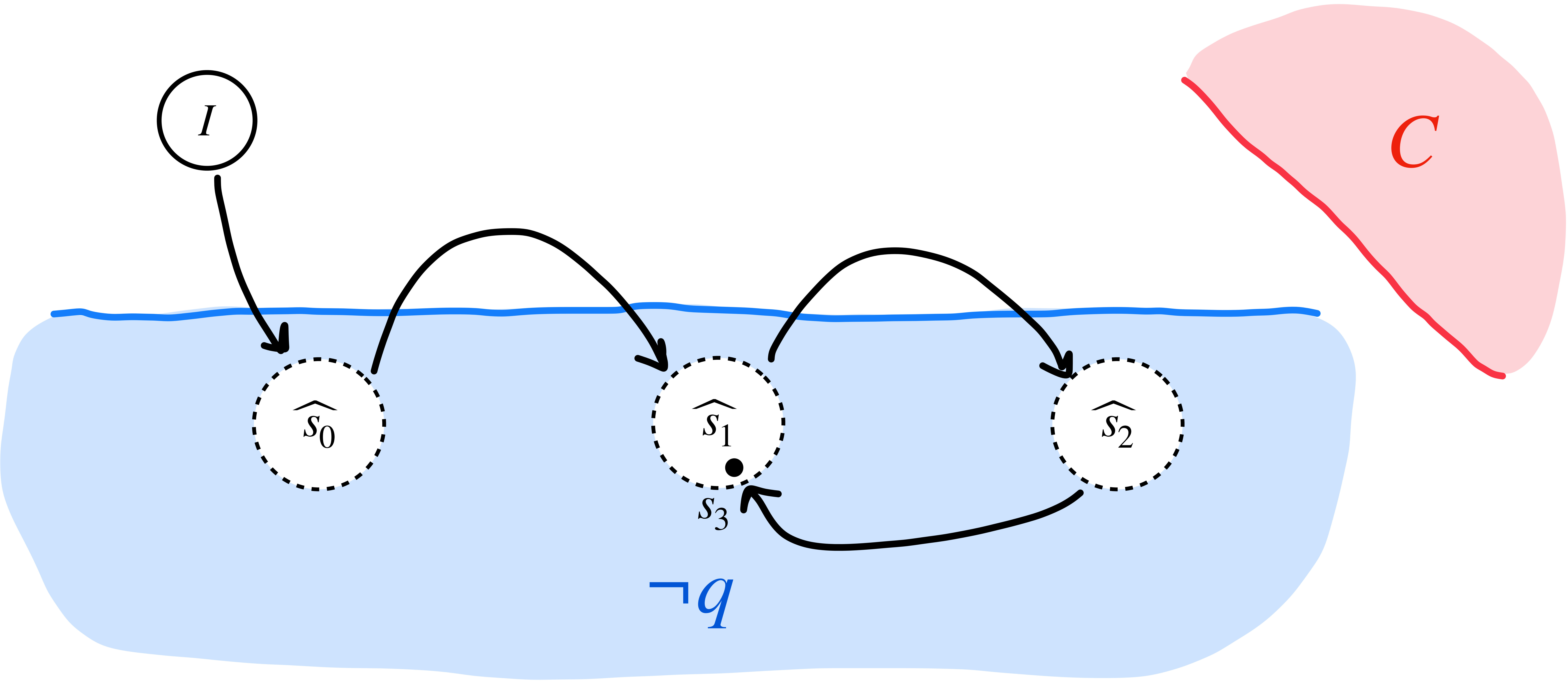


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

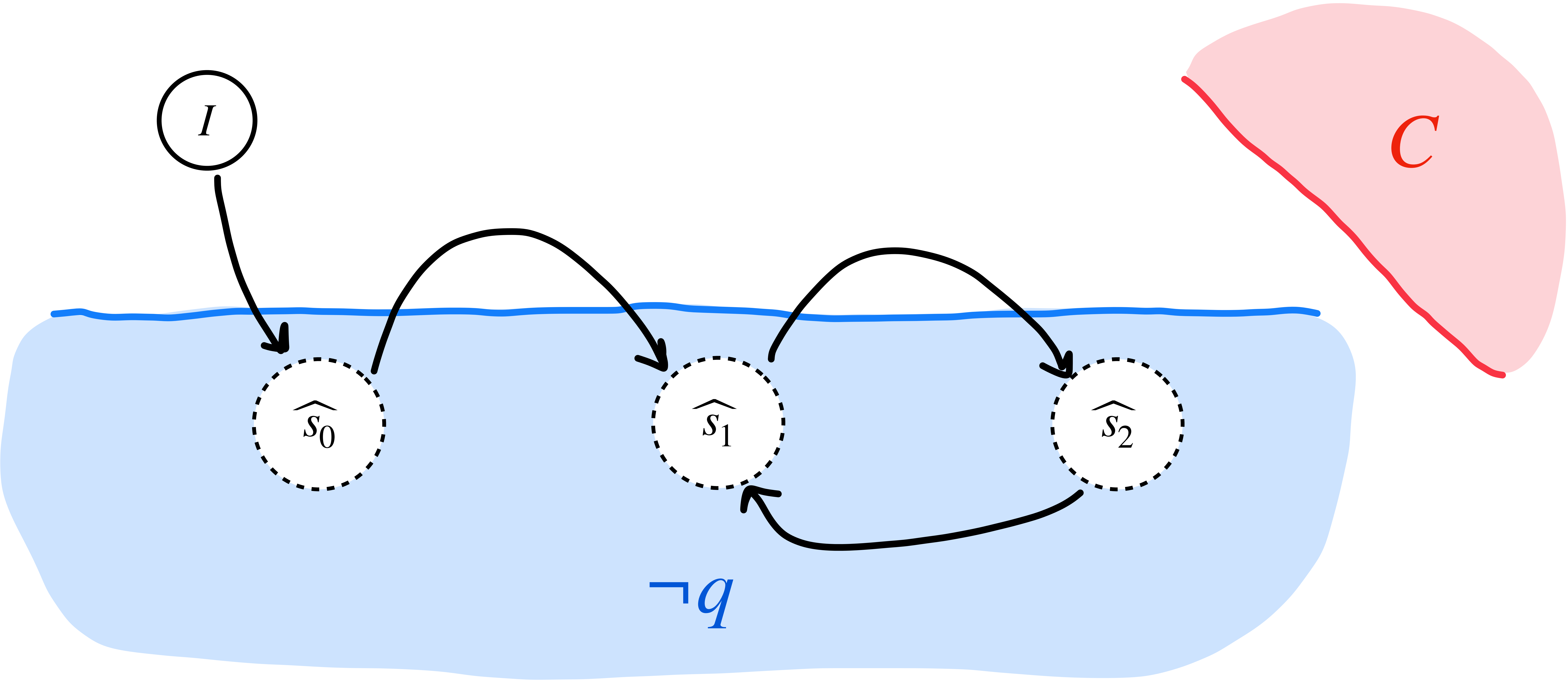


$S = \langle X, I(X), T(X, X') \rangle$

$\varphi = \Diamond \Box q$

Search Stack: $\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\mathbb{P} = \{p_0, p_1, \ldots\}$



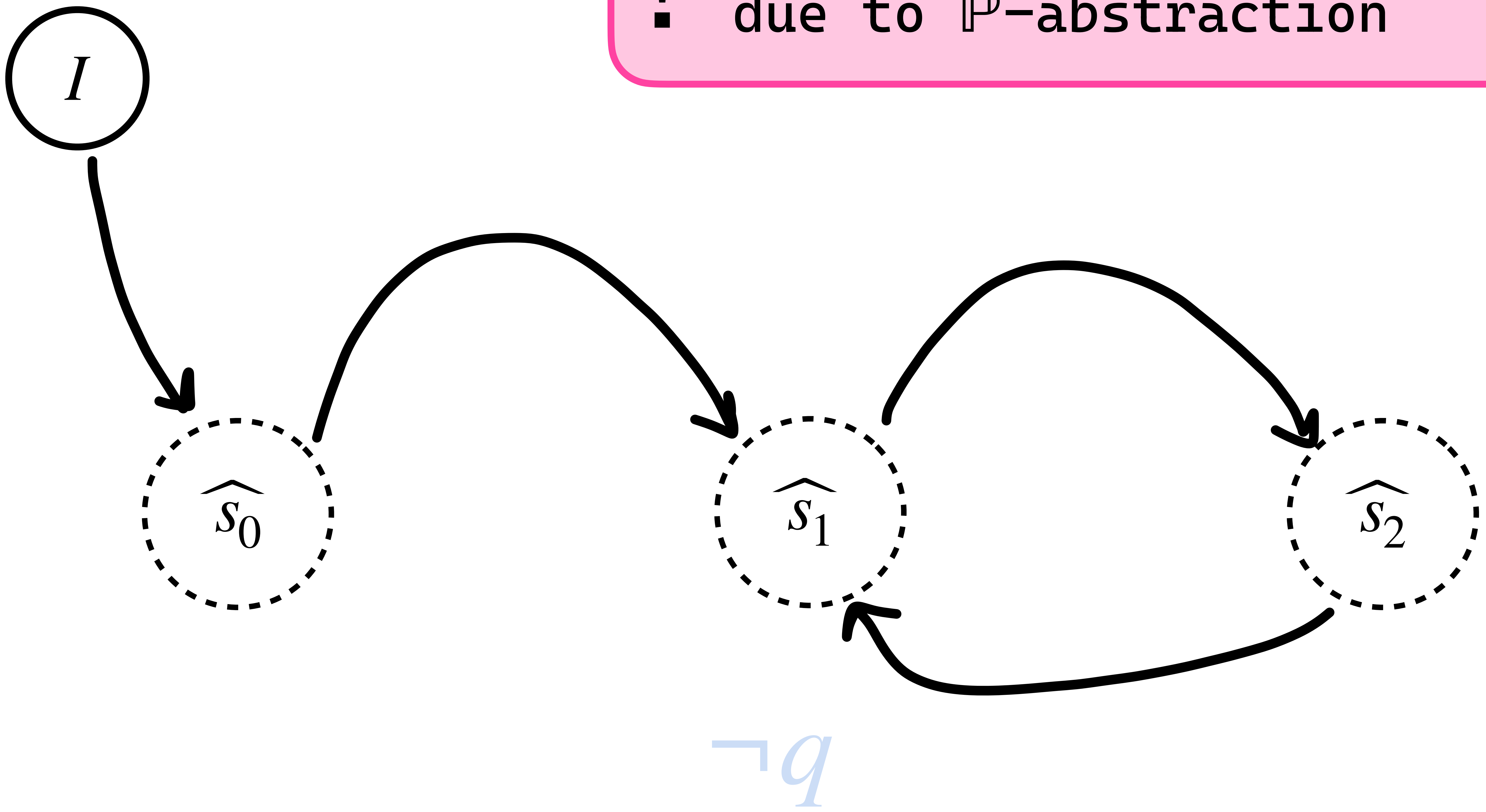
$\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\varphi = \Diamond\Box q$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

! Counterexample could be spurious due to $\mathbb{P}$-abstraction

C



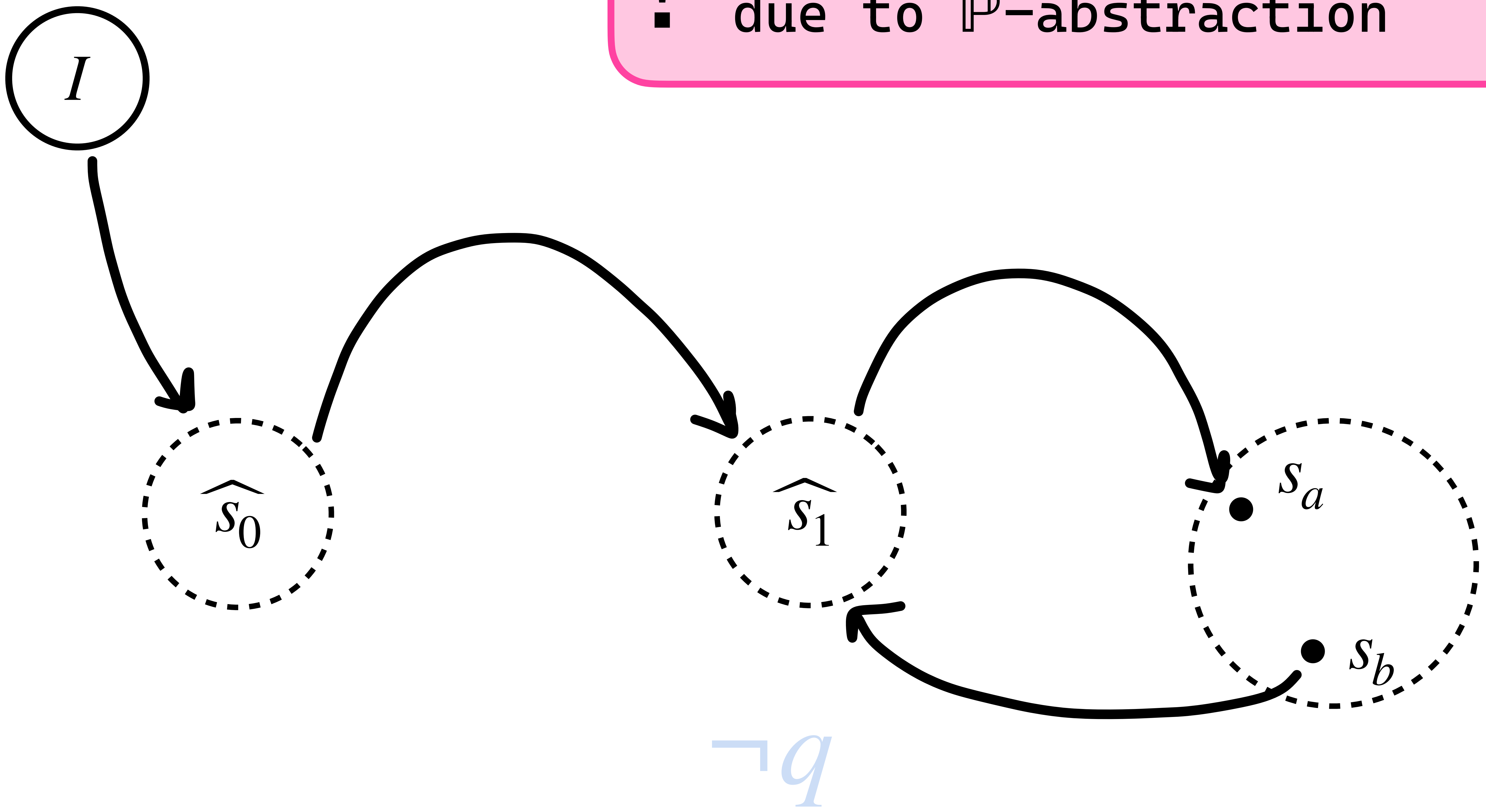
$\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\varphi = \Diamond \Box q$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

! Counterexample could be spurious due to $\mathbb{P}$-abstraction

C



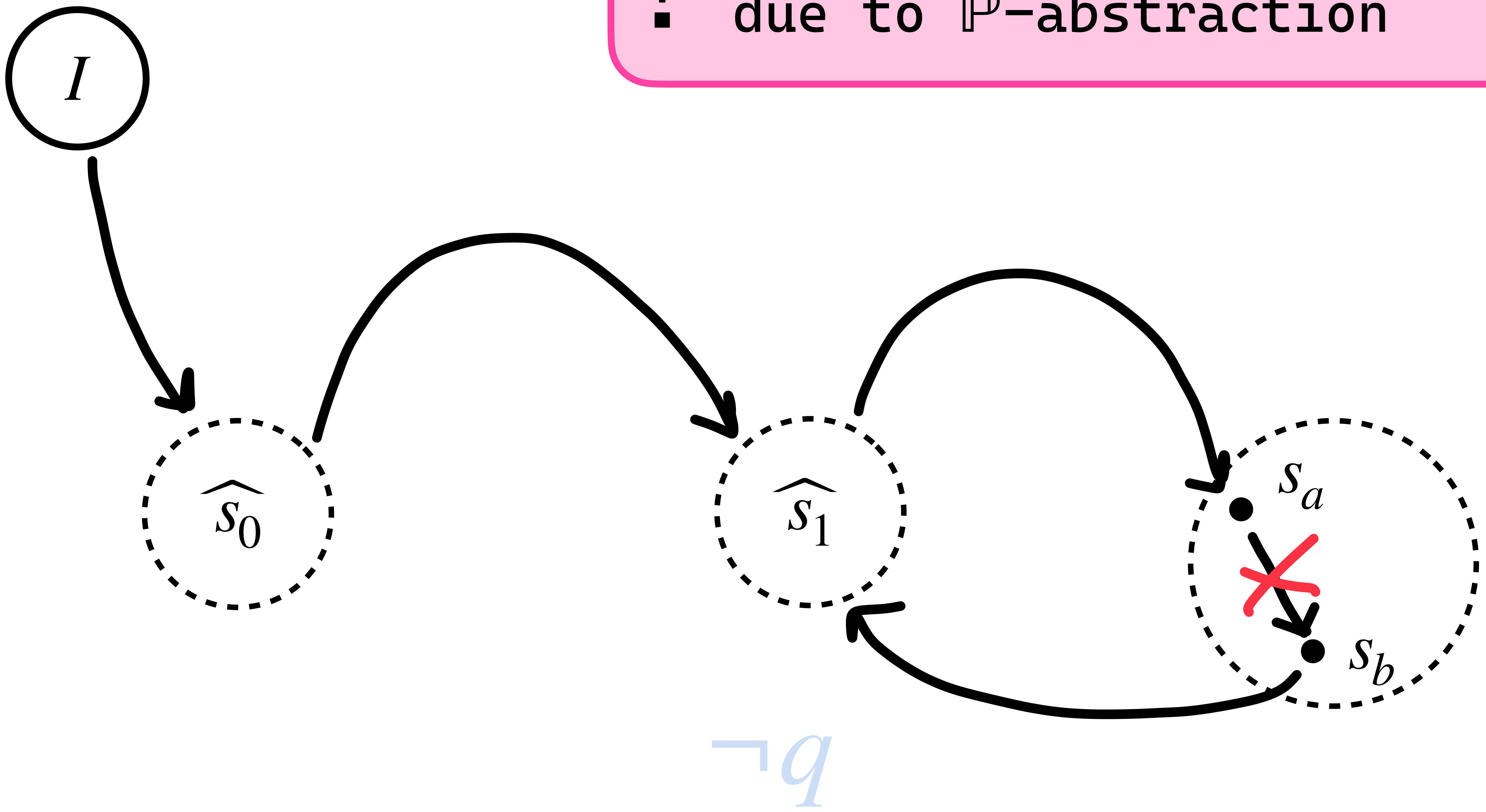
$\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\varphi = \Diamond \Box q$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

! Counterexample could be spurious due to $\mathbb{P}$-abstraction

C



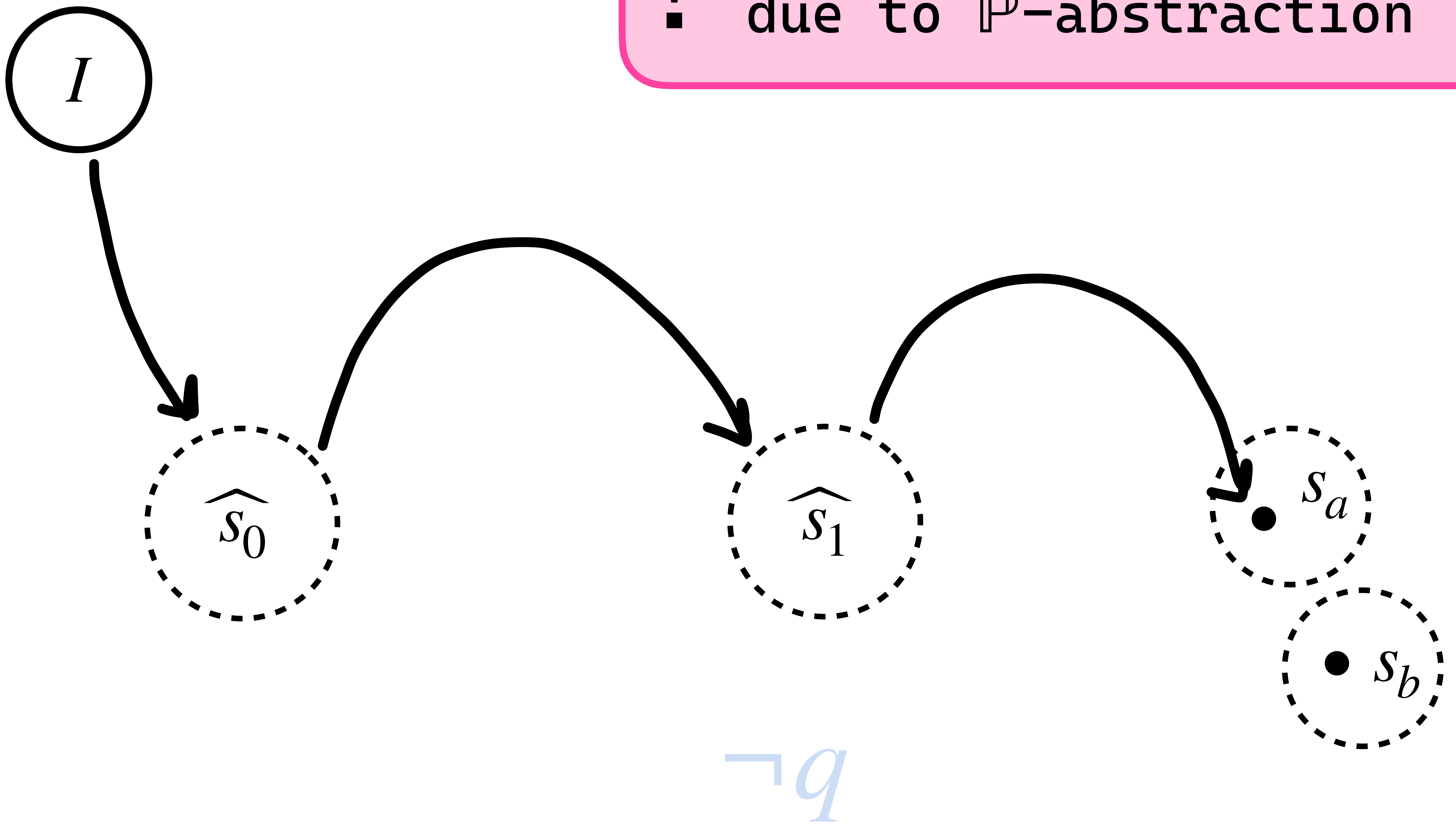
$\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\varphi = \Diamond \Box q$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

! Counterexample could be spurious due to $\mathbb{P}$-abstraction

C



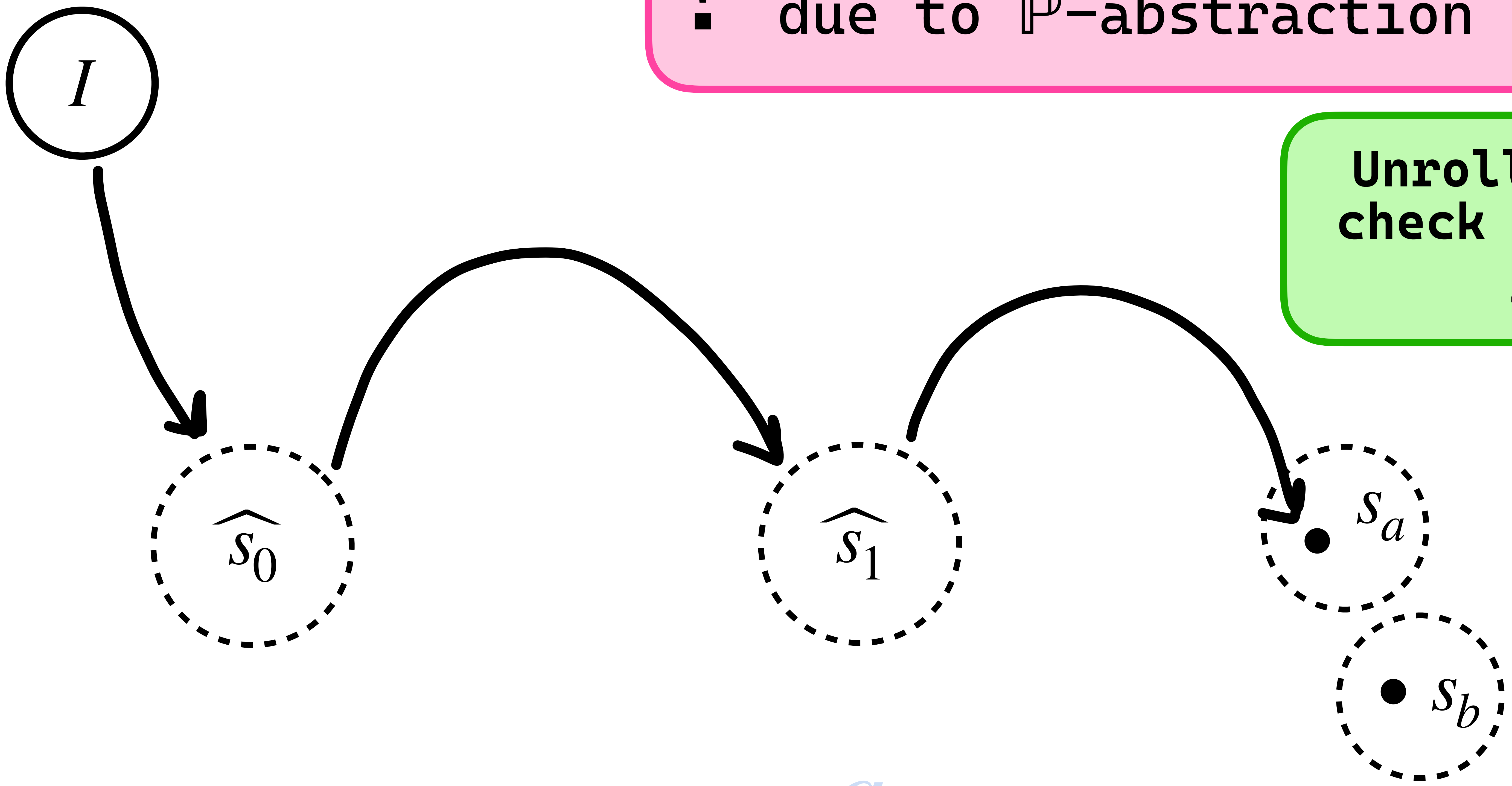
$\widehat{s_0}\ \widehat{s_1}\ \widehat{s_2}$

$\varphi = \Diamond \Box q$

$\mathbb{P} = \{p_0, p_1, \ldots\}$

! Counterexample could be spurious due to $\mathbb{P}$-abstraction

Unroll loop K times, check feasibility and refine $\mathbb{P}$.



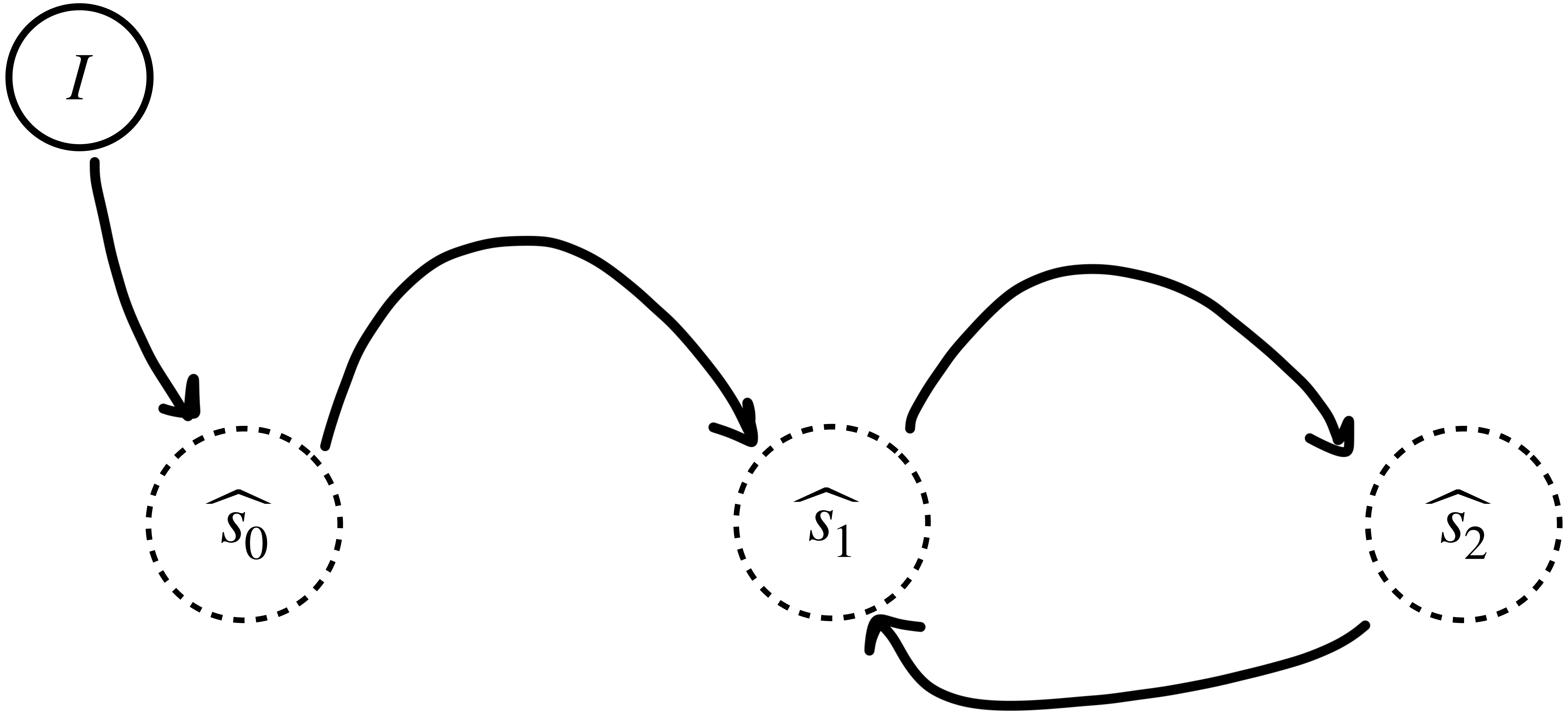
$\neg q$

! Loop could be spurious due to having only finite but unbounded unrollings.

$\widehat{s}_0 \quad \widehat{s}_1 \quad \widehat{s}_2$

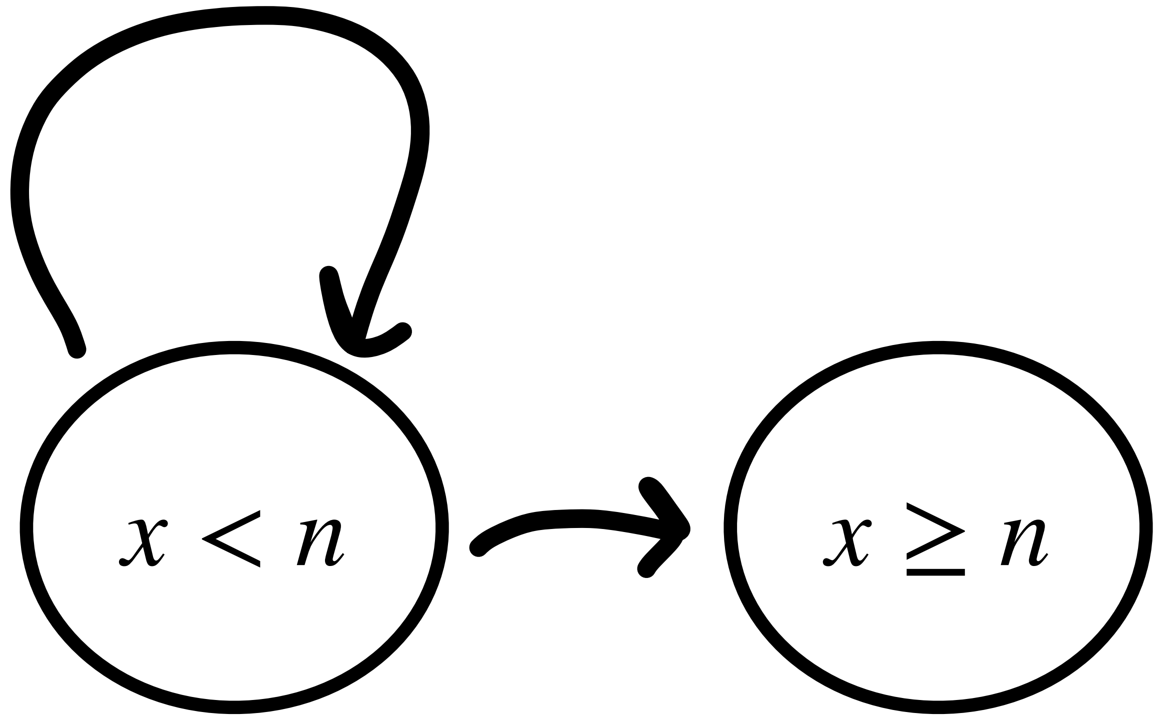
$\varphi = \Diamond \Box q$ $\quad \{p_0, p_1, \ldots\}$

C



$\neg q$

! Loop could be spurious due to having only finite but unbounded unrollings.



$$I := (x = 0) \land (n > x)$$
$$T := (x' = x + 1) \land (n' = n)$$
$$P := \Diamond \Box (x \geq n)$$

$\varphi = \Diamond \Box q$

$\{p_0, p_1, \ldots\}$

$\widehat{s_0}$ $\widehat{s_1}$ $\widehat{s_2}$

I

C

$\widehat{s_2}$

$\neg q$

! Loop could be spurious due to having only finite but unbounded unrollings.

Well-founded Relations

$\widehat{s_0}$ $\widehat{s_1}$ $\widehat{s_2}$

$\varphi = \Diamond \Box q$ $\{p_0, p_1, \ldots\}$

! Loop could be spurious due to having only finite but unbounded unrollings.

Well-founded Relations

Given a set of states Q, a **well-founded relation (WFR)** $w : Q \times Q$ is a binary relation such that every non-empty subset $U \subseteq Q$ has a minimal element $m \in U$ (there is no $u \in U$ such that $(m, u) \in w$).

C

$\widehat{s_0}$ $\widehat{s_1}$ $\widehat{s_2}$

! Loop could be spurious due to having only finite but unbounded unrollings.

$\varphi = \Diamond \Box q$ $\{p_0, p_1, \ldots\}$

Well-founded Relations

C

Given a set of states Q, a **well-founded relation (WFR)** $w : Q \times Q$ is a binary relation such that every non-empty subset $U \subseteq Q$ has a minimal element $m \in U$ (there is no $u \in U$ such that $(m, u) \in w$).

If there exists a WFR w over the states in π_{loop}, then π_{loop} cannot be unrolled infinitely (w is a proof of π_{loop}'s spuriousness).

! Loop could be spurious due to having only finite but unbounded unrollings.

Well-founded Relations

Given a set of states Q, a **well-founded relation (WFR)** $w : Q \times Q$ is a binary relation such that every non-empty subset $U \subseteq Q$ has a minimal element $m \in U$ (there is no $u \in U$ such that $(m, u) \in w$).

If there exists a WFR w over the states in π_{loop}, then π_{loop} cannot be unrolled infinitely (w is a proof of π_{loop}'s spuriousness).

Use off-the-shelf methods

! Loop could be spurious due to having only finite but unbounded unrollings.

$\widehat{s_0}$ $\widehat{s_1}$ $\widehat{s_2}$

$\varphi = \Diamond \Box q$ $\{p_0, p_1, \ldots\}$

Well-founded Relations

Given a set of states Q, a **well-founded relation (WFR)** $w : Q \times Q$ is a binary relation such that every non-empty subset $U \subseteq Q$ has a minimal element $m \in U$ (there is no $u \in U$ such that $(m, u) \in w$).

If there exists a WFR w over the states in π_{loop}, then π_{loop} cannot be unrolled infinitely (w is a proof of π_{loop}'s spuriousness).

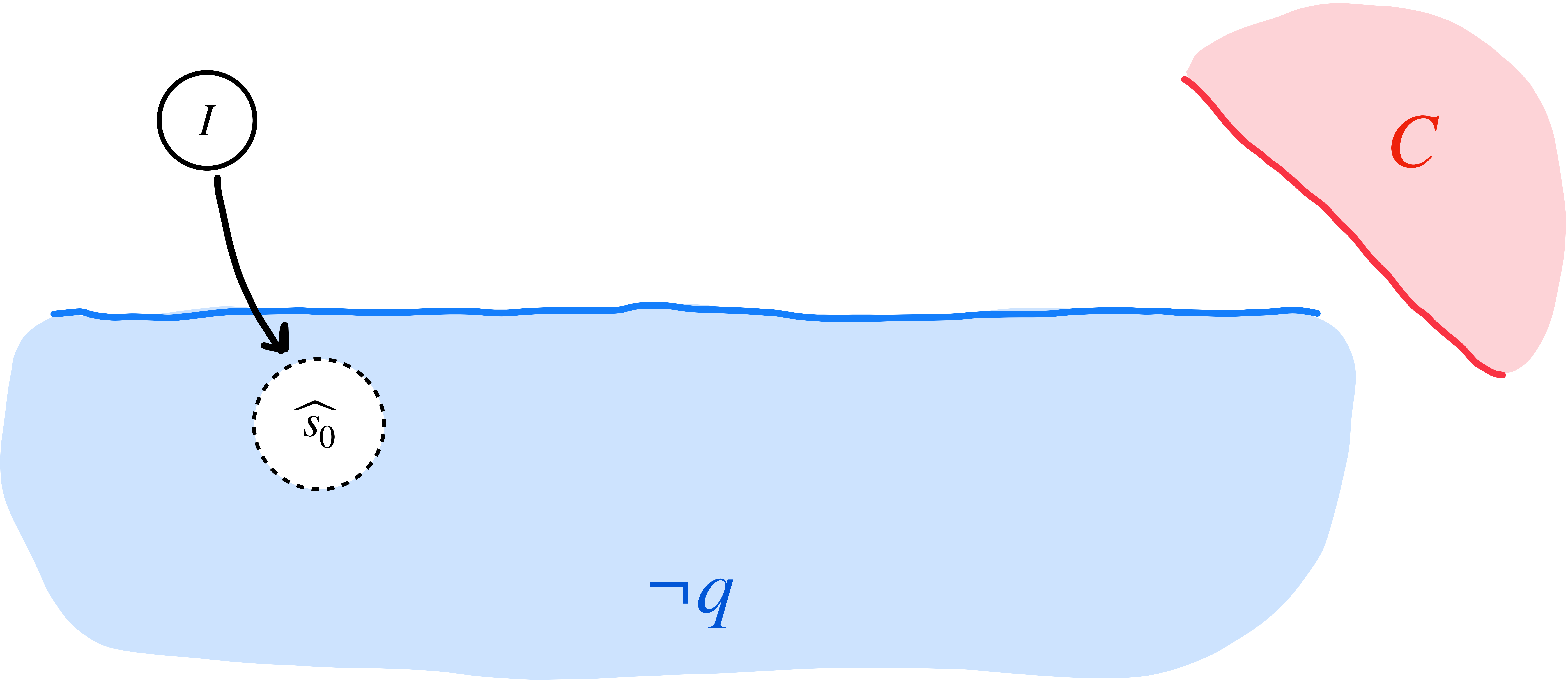
Given a set of WFRs $\mathbb{W}$, if π_{loop} is part of a genuine counterexample, then there is some pair of states $(s_0, s_1) \in \pi_{loop}$ such that (s_0, s_1) are not in any WFR $w \in \mathbb{W}$.

C

Use off-the-shelf methods

$\widehat{s_0}$

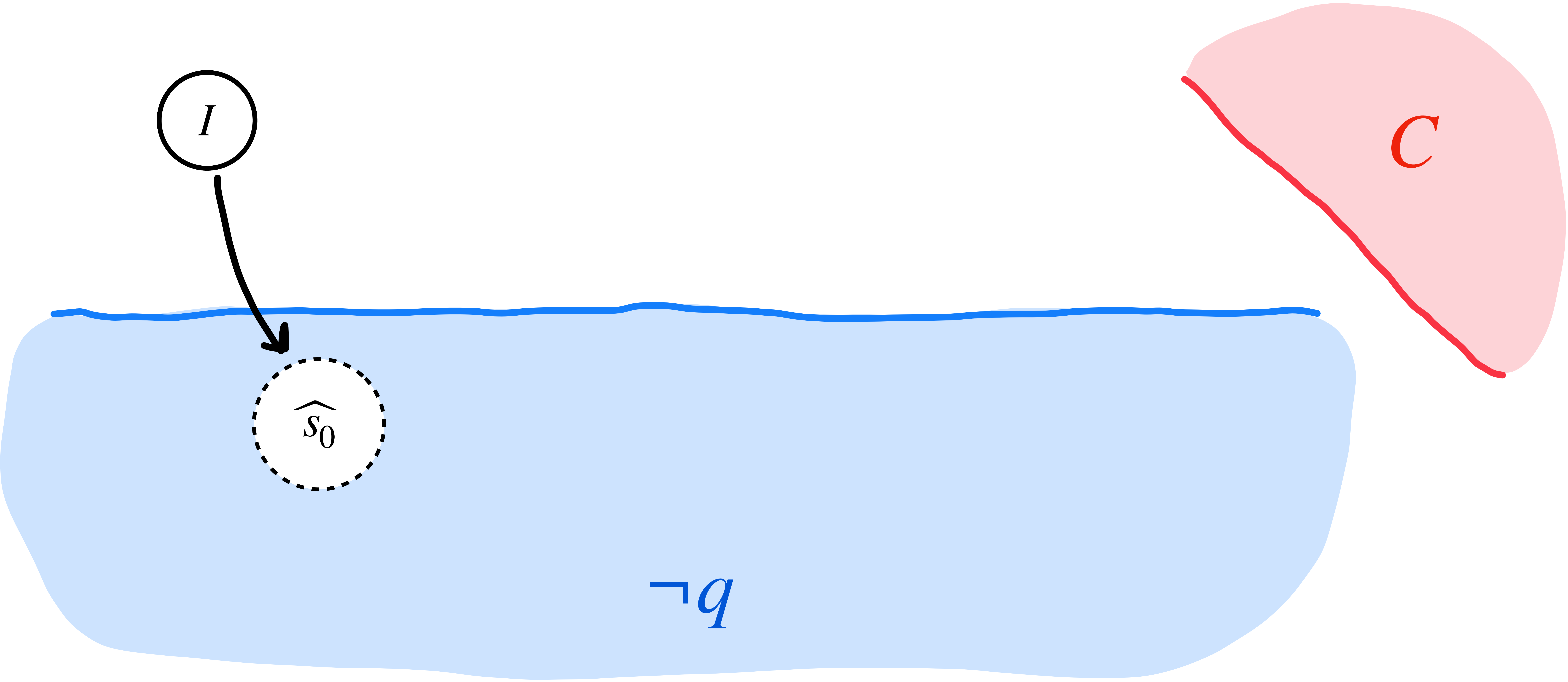
$\varphi = \Diamond \Box q \quad \mathbb{P} = \{p_0, p_1, \ldots, p_n\} \quad \mathbb{W} = \{w_0, w_1, \ldots w_m\}$



! How to block spurious loops using $\mathbb{W}$?

$\widehat{s_0}$

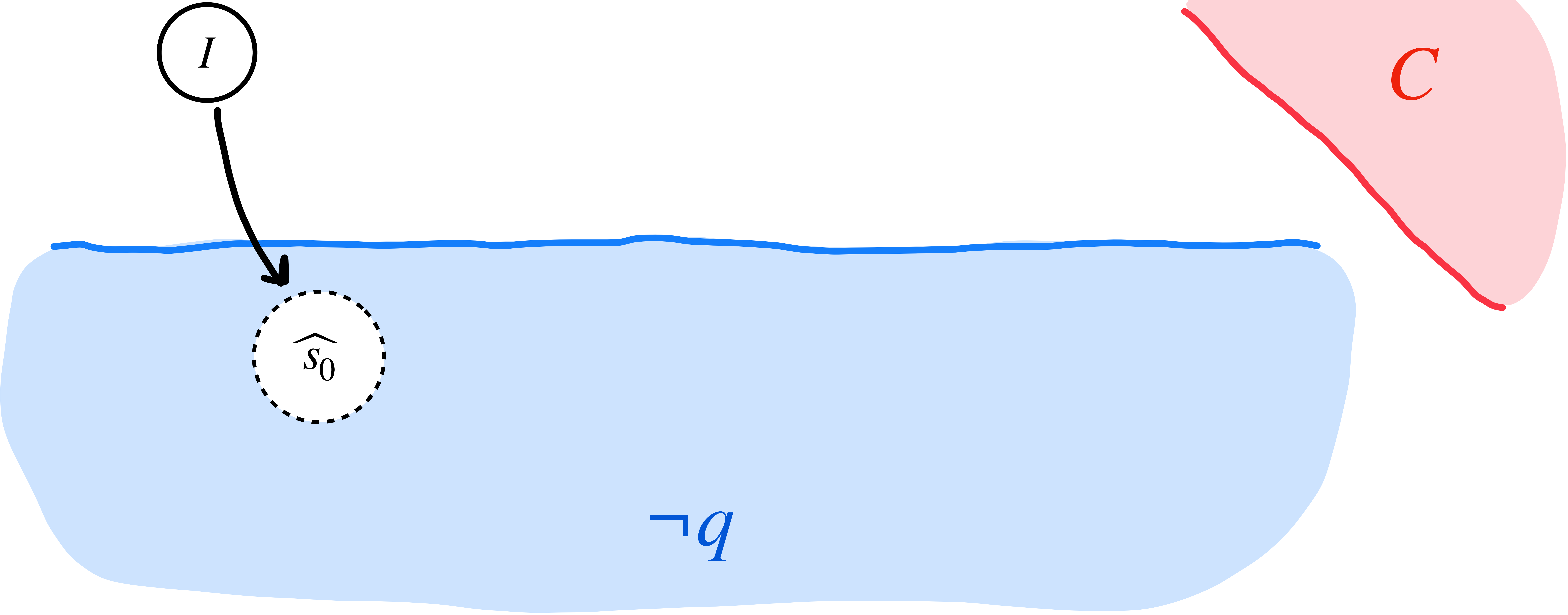
$\varphi = \Diamond\Box q$ $\quad$ $\{p_0, p_1, \ldots, p_n\}$ $\quad$ $\mathbb{W} = \{w_0, w_1, \ldots w_m\}$



! How to block spurious loops using $\mathbb{W}$?

Search for **pair** of $\neg q$-states not in any $w \in \mathbb{W}$.

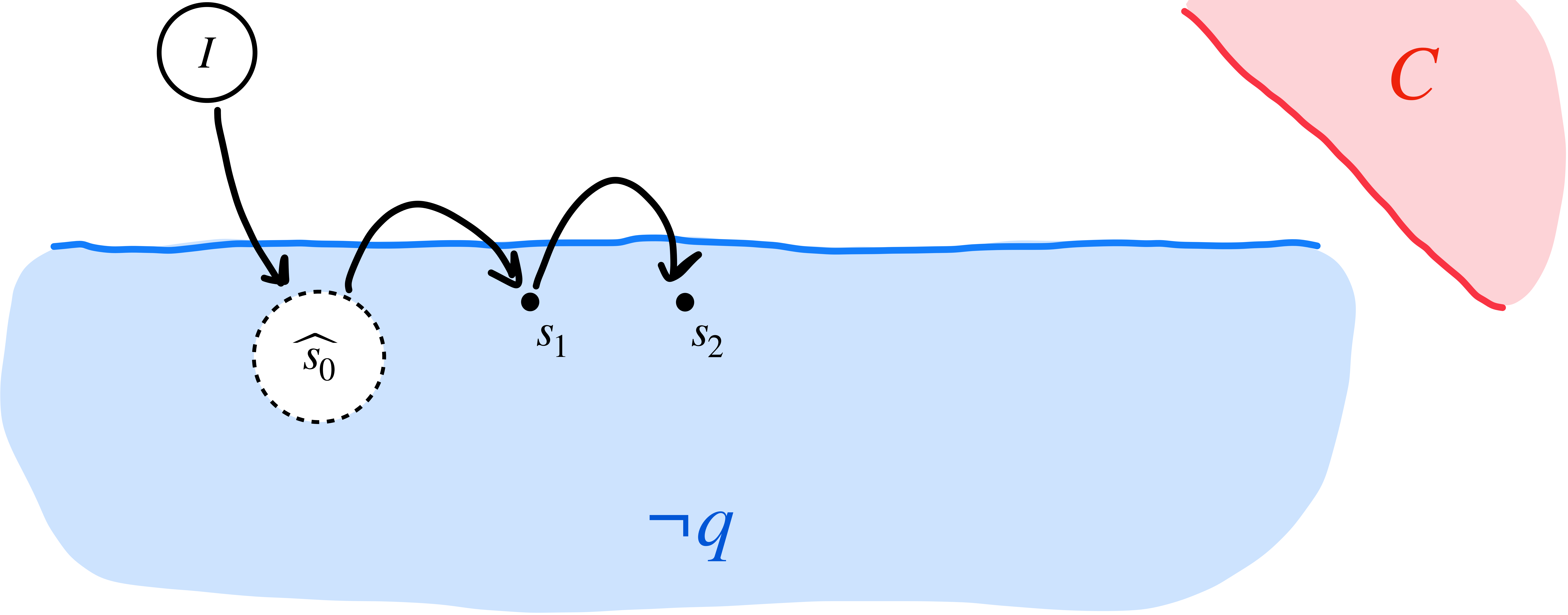
$$\mathbb{W} = \{w_0, w_1, \dots w_m\}$$



! How to block spurious loops using $\mathbb{W}$?

$$\mathbb{W} = \{w_0, w_1, \ldots w_m\}$$

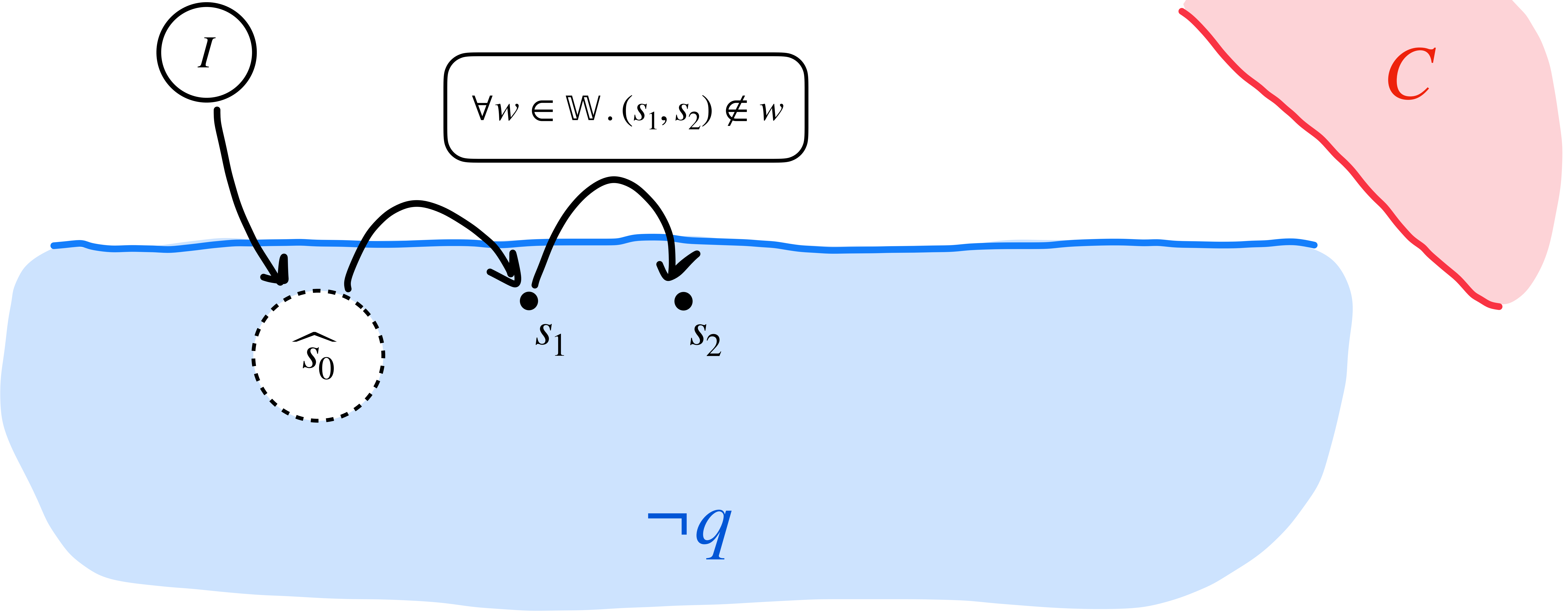
Search for **pair** of $\neg q$-states not in any $w \in \mathbb{W}$.



! How to block spurious loops using $\mathbb{W}$?

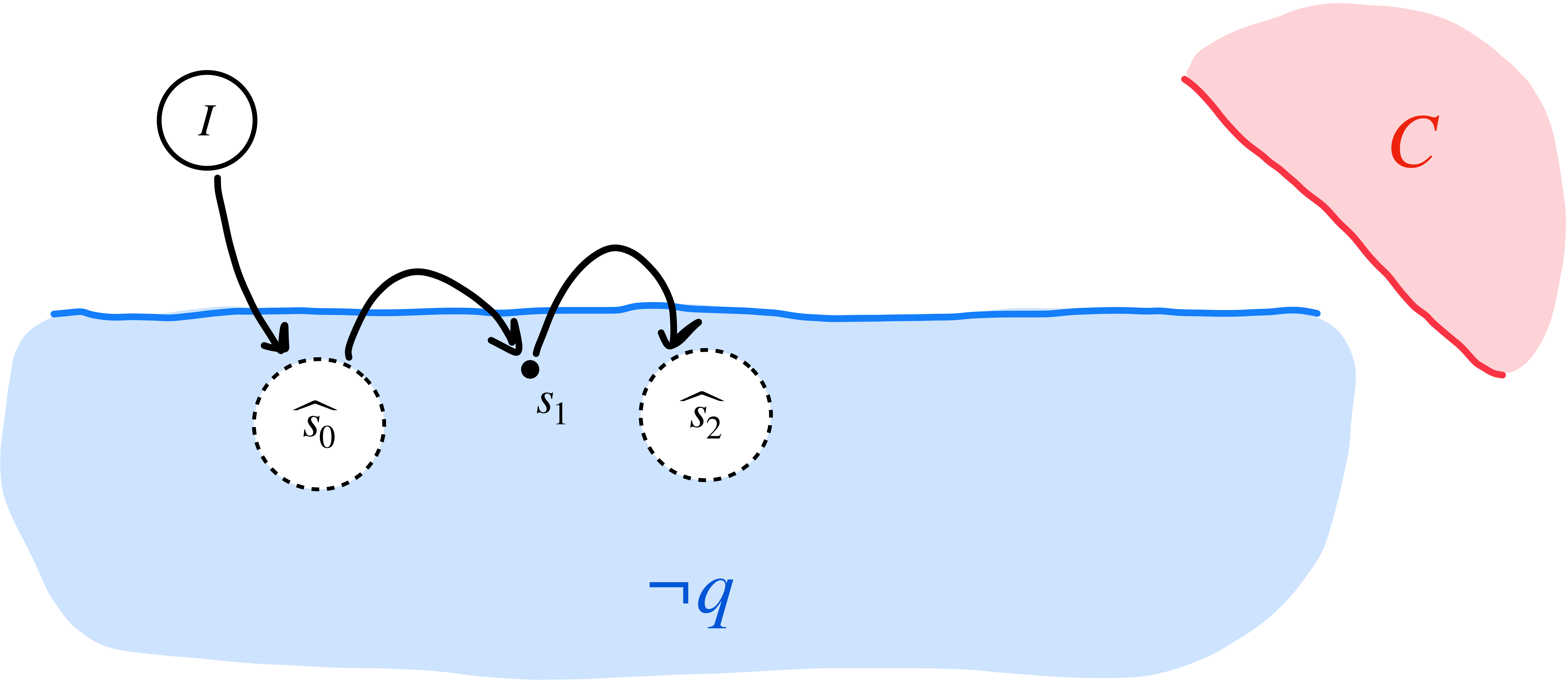
$$\mathbb{W} = \{w_0, w_1, \ldots w_m\}$$

Search for **pair** of $\neg q$-states not in any $w \in \mathbb{W}$.



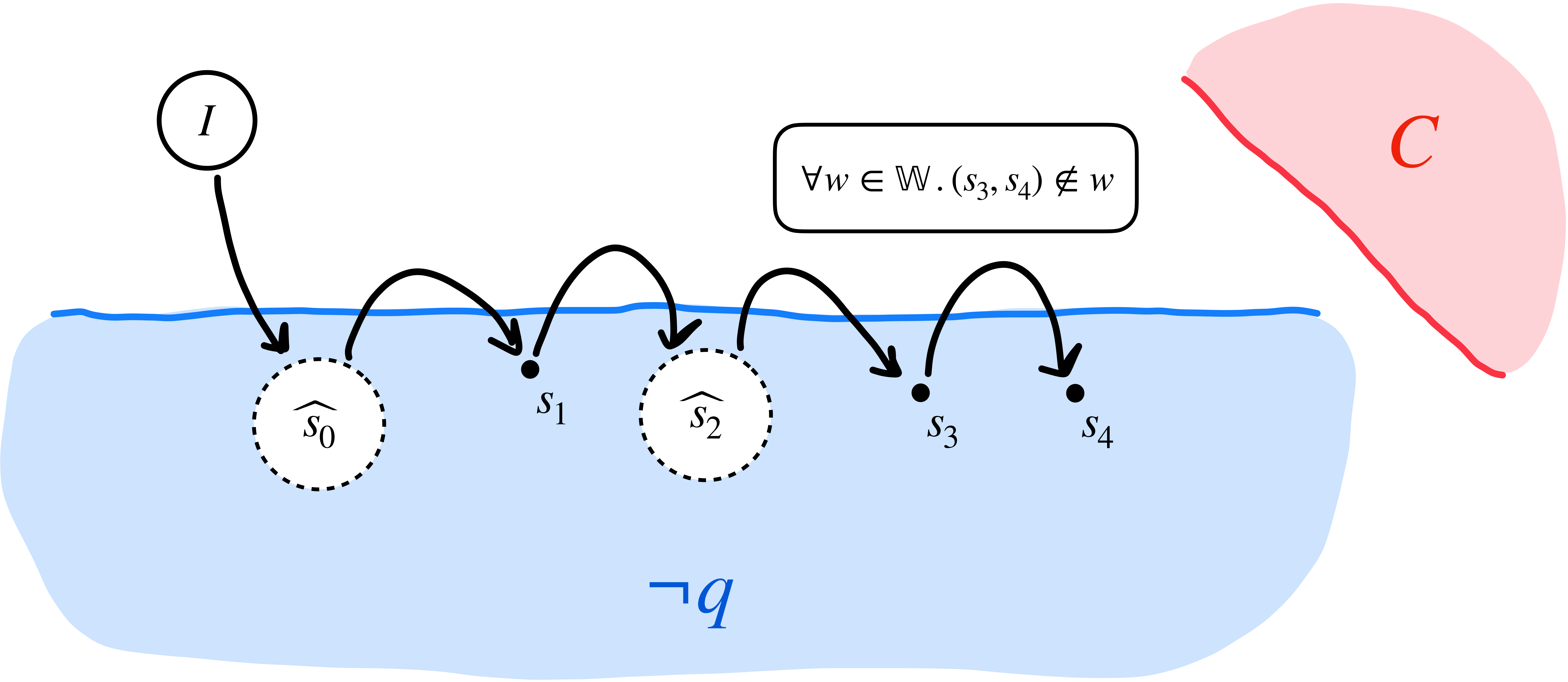
$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** $\widehat{s_0}$ $\widehat{s_2}$

$\varphi = \Diamond \Box q$ $\mathbb{P} = \{p_0, p_1, \ldots, p_n\}$ $\mathbb{W} = \{w_0, w_1, \ldots w_m\}$



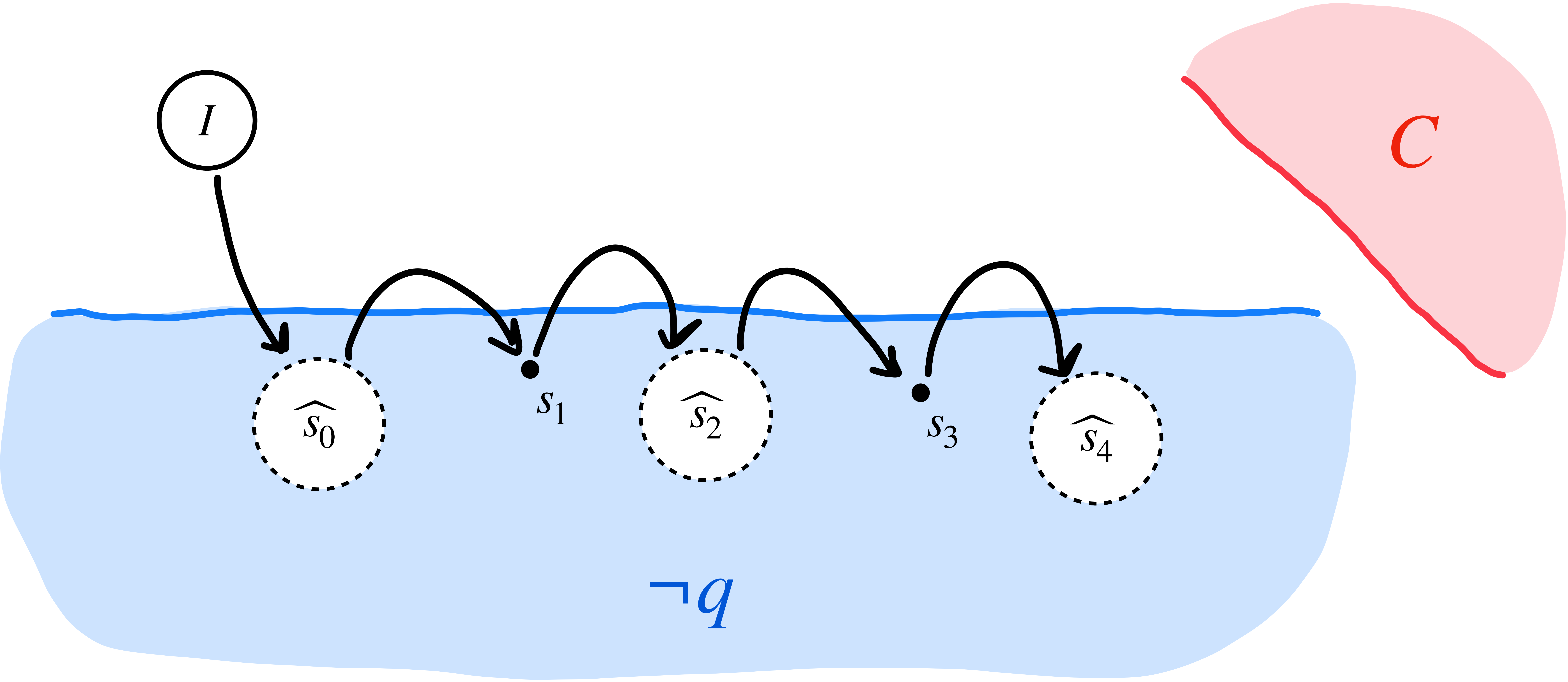
$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** $\widehat{s_0}$ $\widehat{s_2}$

$\varphi = \Diamond \Box q$ $\mathbb{P} = \{p_0, p_1, \ldots, p_n\}$ $\mathbb{W} = \{w_0, w_1, \ldots w_m\}$



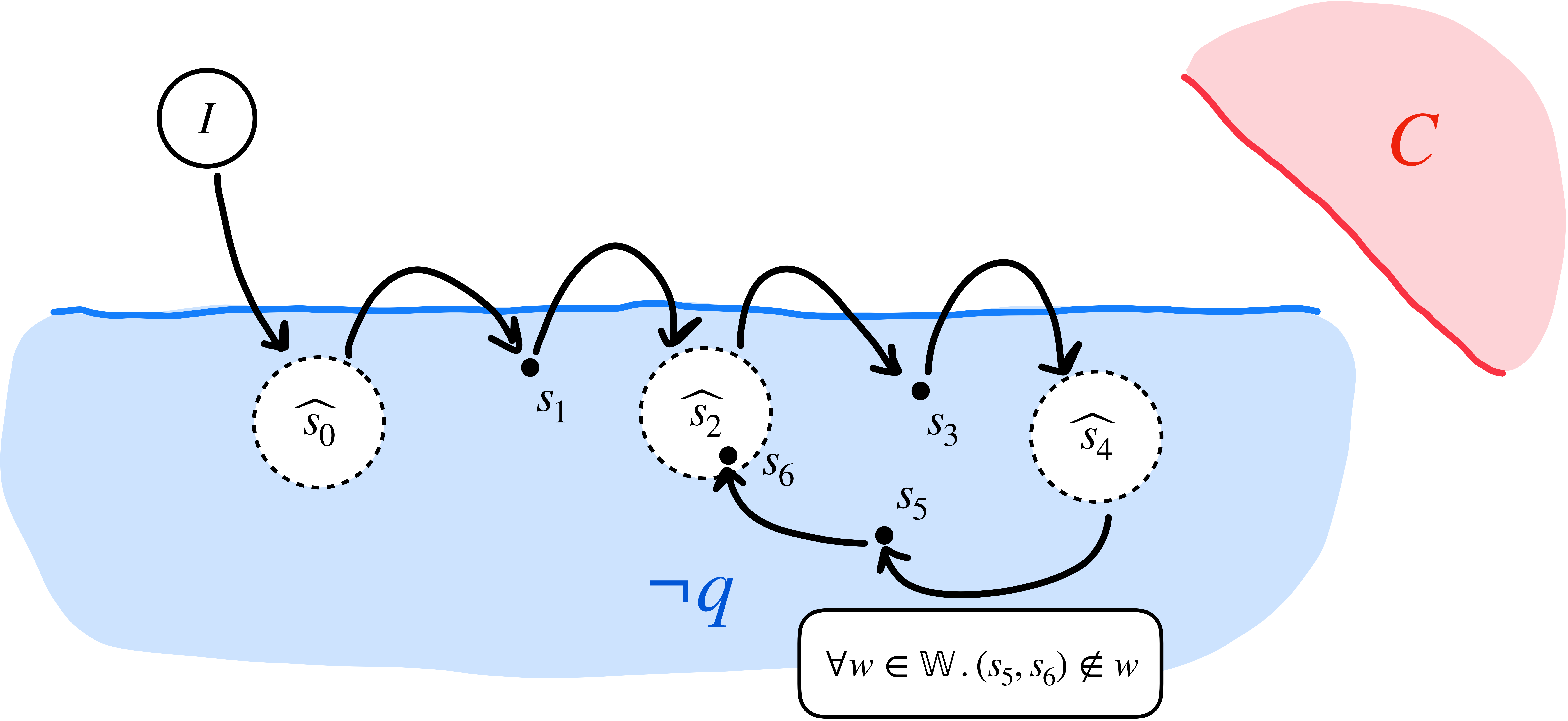
$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** $\widehat{s_0}\ \widehat{s_2}\ \widehat{s_4}$

$\varphi = \Diamond \Box q$ $\mathbb{P} = \{p_0, p_1, \ldots, p_n\}$ $\mathbb{W} = \{w_0, w_1, \ldots w_m\}$



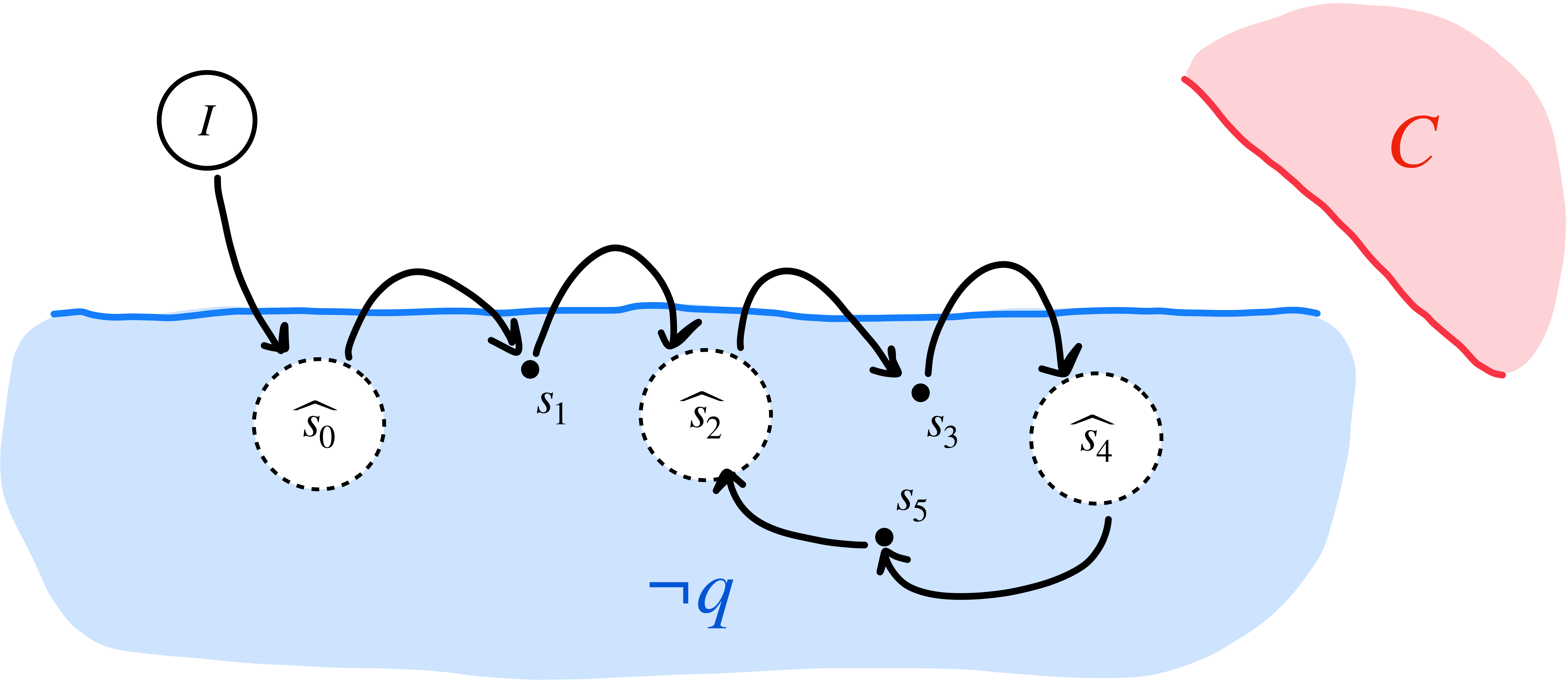
$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** $\widehat{s_0}$ $\widehat{s_2}$ $\widehat{s_4}$

$\varphi = \Diamond \Box q$ $\mathbb{P} = \{p_0, p_1, \ldots, p_n\}$ $\mathbb{W} = \{w_0, w_1, \ldots w_m\}$



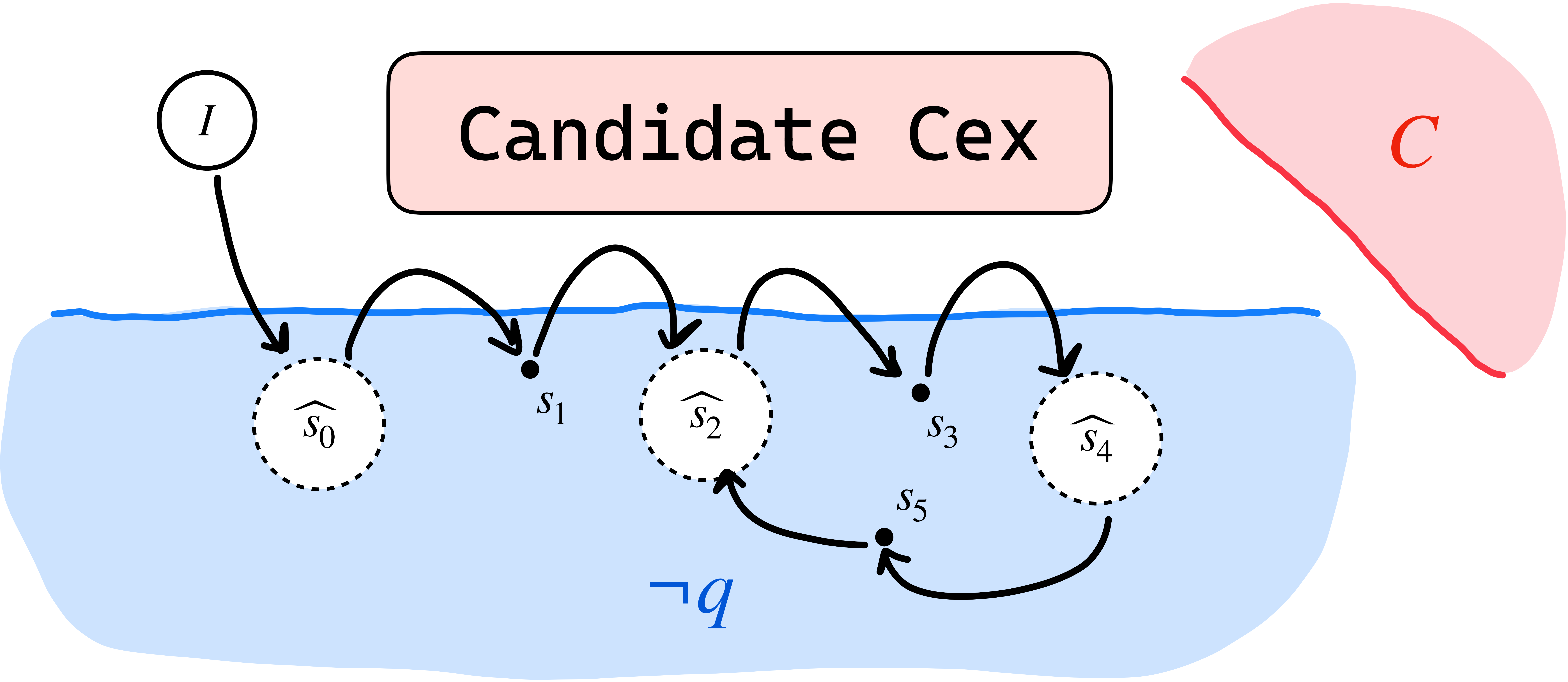
$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** $\widehat{s_0}$ $\widehat{s_2}$ $\widehat{s_4}$

$\varphi = \Diamond \Box q$ $\mathbb{P} = \{p_0, p_1, \ldots, p_n\}$ $\mathbb{W} = \{w_0, w_1, \ldots w_m\}$



$S = \langle X, I(X), T(X, X') \rangle$ **Search Stack:** $\widehat{s_0}\ \widehat{s_2}\ \widehat{s_4}$

$\varphi = \Diamond \Box q \quad \mathbb{P} = \{p_0, p_1, \ldots, p_n\} \quad \mathbb{W} = \{w_0, w_1, \ldots w_m\}$



RLive-inf Overview

- While find-counterexample($I, T, \Diamond\Box q, C, \mathbb{P}, \mathbb{W}$) returns π:

RLive-inf Overview

- While find-counterexample($I, T, \Diamond\Box q, C, \mathbb{P}, \mathbb{W}$) returns π:
  - If $\pi_{prefix} \cdot \pi_{loop}^{K}$ is infeasible for $K \in \mathbb{N}$:
    - Perform Craig interpolation, update $\mathbb{P}$, and continue

RLive-inf Overview

- While find-counterexample$(I, T, \Diamond\Box q, C, \mathbb{P}, \mathbb{W})$ returns π:
  - If $\pi_{prefix} \cdot \pi_{loop}^{K}$ is infeasible for $K \in \mathbb{N}$:
    - Perform Craig interpolation, update $\mathbb{P}$, and continue
  - Else if there exists WFR w that blocks π_{loop}:
    - Add w to $\mathbb{W}$ and continue

RLive-inf Overview

- While find-counterexample($I, T, \Diamond\Box q, C, \mathbb{P}, \mathbb{W}$) returns π:
  - If $\pi_{prefix} \cdot \pi_{loop}^{K}$ is infeasible for $K \in \mathbb{N}$:
    - Perform Craig interpolation, update $\mathbb{P}$, and continue
  - Else if there exists WFR w that blocks π_{loop}:
    - Add w to $\mathbb{W}$ and continue
  - Else return unsafe

RLive-inf Overview

- While find-counterexample($I, T, \Diamond\Box q, C, \mathbb{P}, \mathbb{W}$) returns π:
  - If π is feasible:
    - Return unsafe
  - Else if $\pi_{prefix} \cdot \pi_{loop}^{K}$ is infeasible for $K \in \mathbb{N}$:
    - Perform Craig interpolation, update $\mathbb{P}$, and continue
  - Else if there exists WFR w that blocks π_{loop}:
    - Add w to $\mathbb{W}$ and continue
- Return safe

Experimental Setup

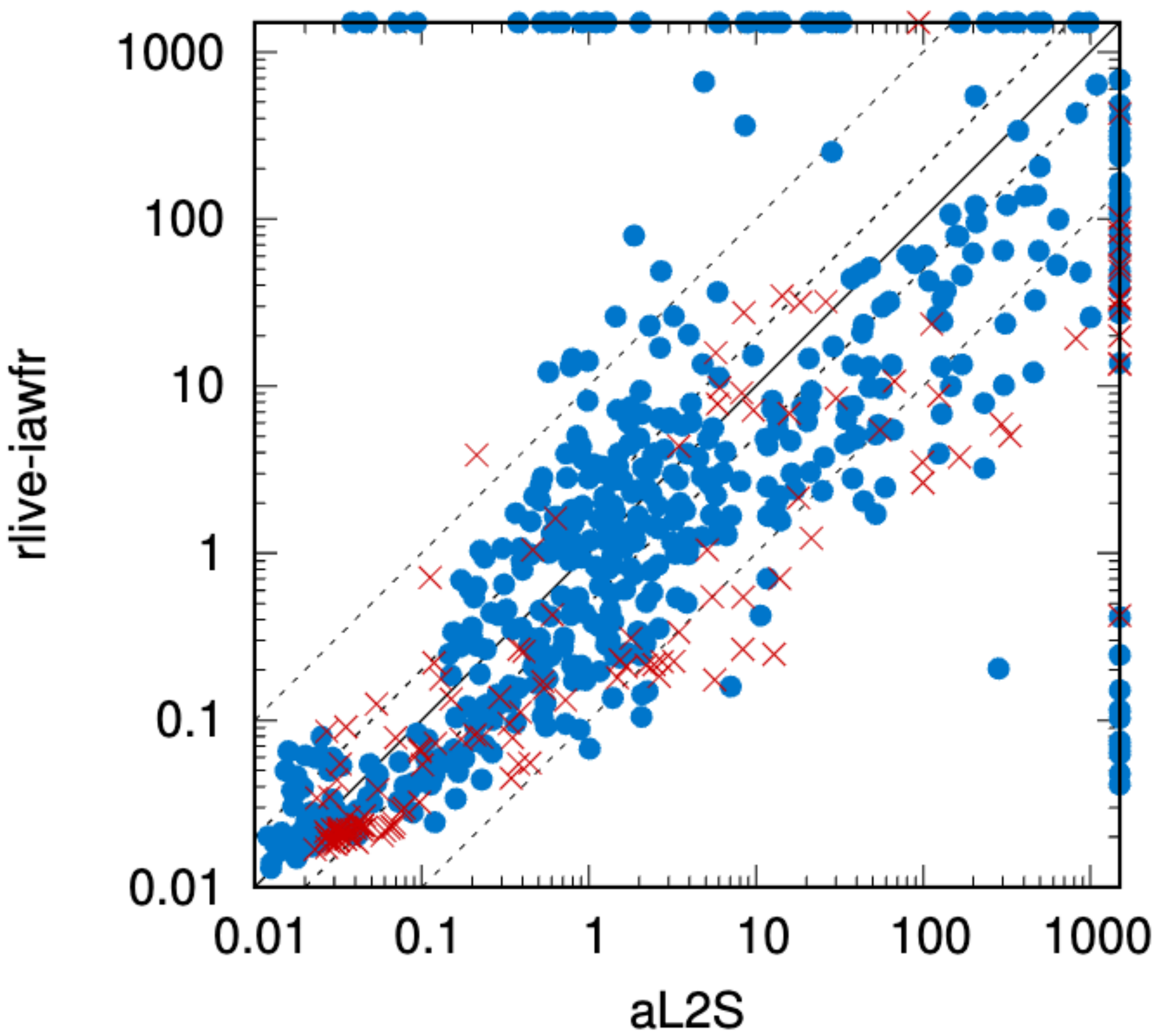
- Implemented in ic3ia, compared against Abstract L2S

Experimental Setup

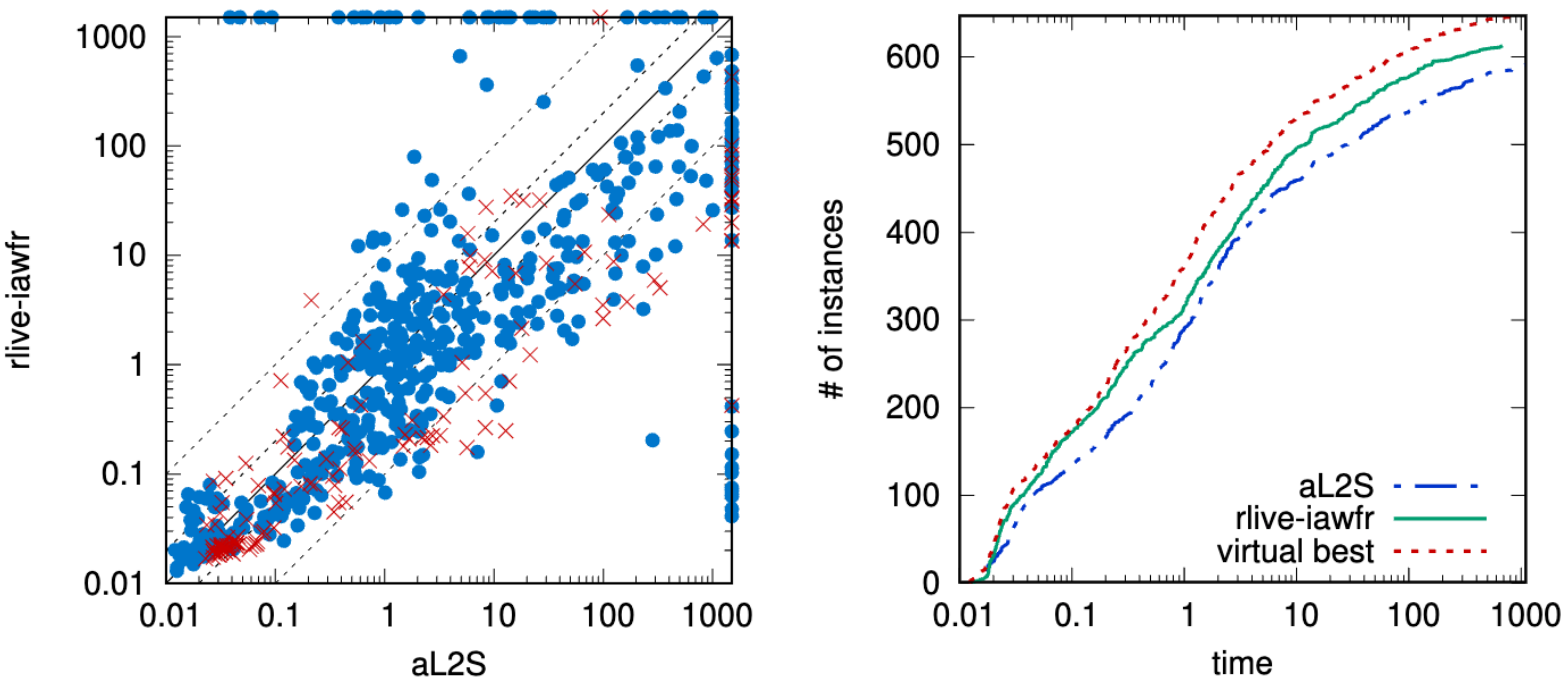
- Implemented in ic3ia, compared against Abstract L2S
- Abstract L2S:
  - Similar CEGAR loop to rlive-inf, monolithic encoding
  - Outperforms HSF, T2-CTL*, Ultimate-LTL

Experimental Setup

- Implemented in ic3ia, compared against Abstract L2S
- Abstract L2S:
  - Similar CEGAR loop to rlive-inf, monolithic encoding
  - Outperforms HSF, T2-CTL*, Ultimate-LTL
- Benchmarks:
  - Symbolic transition systems (556)
  - Imperative-style programs (86)
  - Termination benchmarks (193)



rlive-iawfr
aL2S
1000
100
10
1
0.1
0.01
0.01
0.1
1
10
100
1000



1000
100
10
1
0.1
0.01
rlive-iawfr
0.01
0.1
1
10
100
1000
aL2S
600
500
400
300
200
100
0
# of instances
0.01
0.1
1
10
100
1000
time
aL2S
rlive-iawfr
virtual best

RLive-inf Summary

- New algorithm for **infinite-state liveness checking**.
- Finds candidate counterexamples via **incremental, depth-first** search with predicate abstraction.
- Uses **well-founded relations** to block finite but unbounded loops in target system.
- Implementation **outperforms aL2S** on variety of benchmarks.

Artifact

	Num Solved (/835)	Num Safe Solved	Num Unsafe Solved	Time (s) (Non-timeouts)			Num Predicates			Num WFRs		
				Min	Max	Avg	Min	Max	Avg	Min	Max	Avg
Abstract L2S	591	470	121	0.008	1088.96	37.328	4	255	46.091	0	14	0.680
RLive-inf	613	480	133	0.006	703.248	21.983	1	94	24.278	0	15	0.675
Virtual Best	647	513	134	0.008	972.825	23.418	1	173	33.023	0	13	0.625