Bounded Quantifier Instantiation for Checking Inductive Invariants

TACAS'17

April 24, 2017

Yotam Feldman
Tel Aviv University

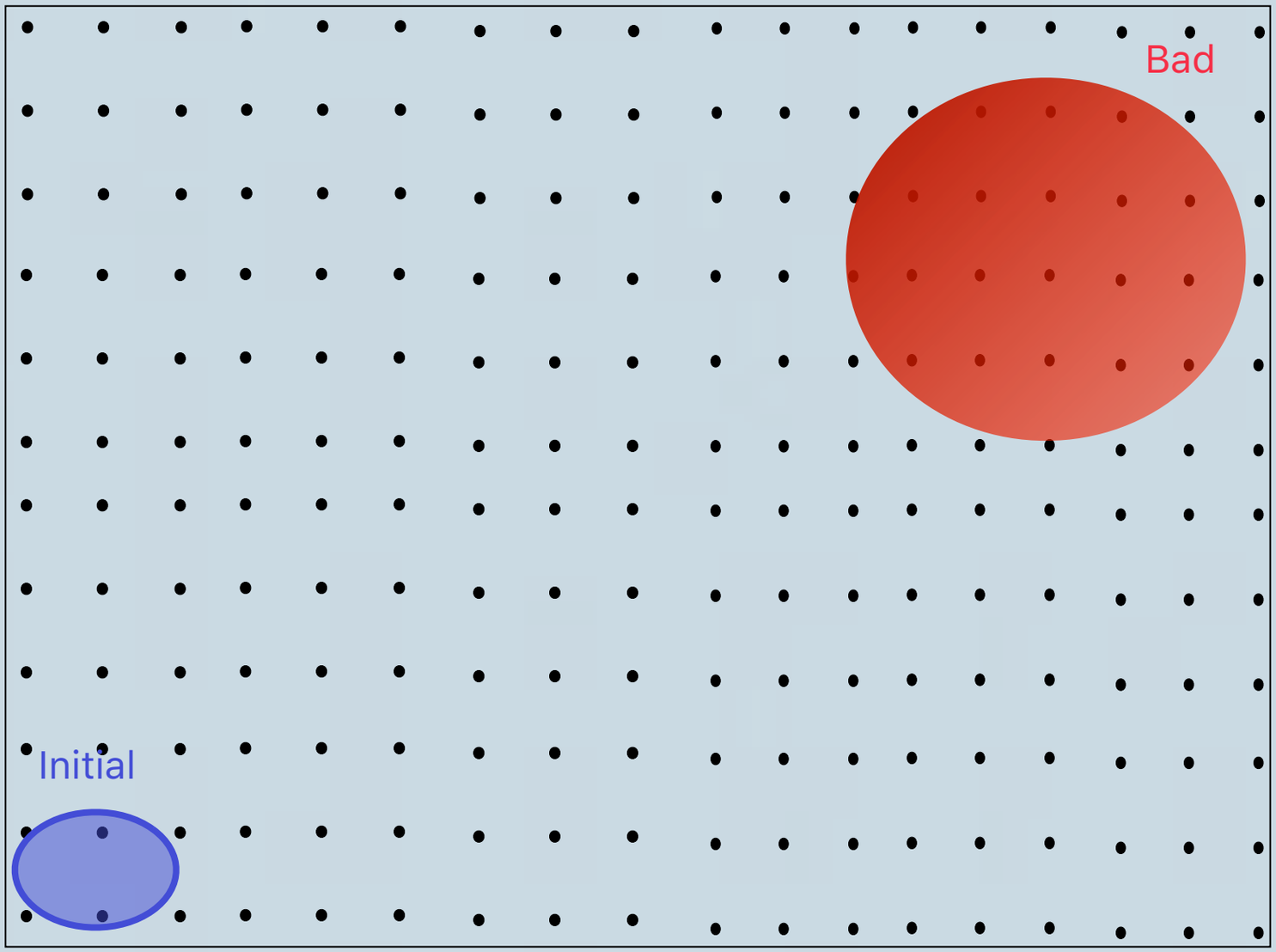
Oded Padon
Tel Aviv University

Neil Immerman
UMass, Amherst

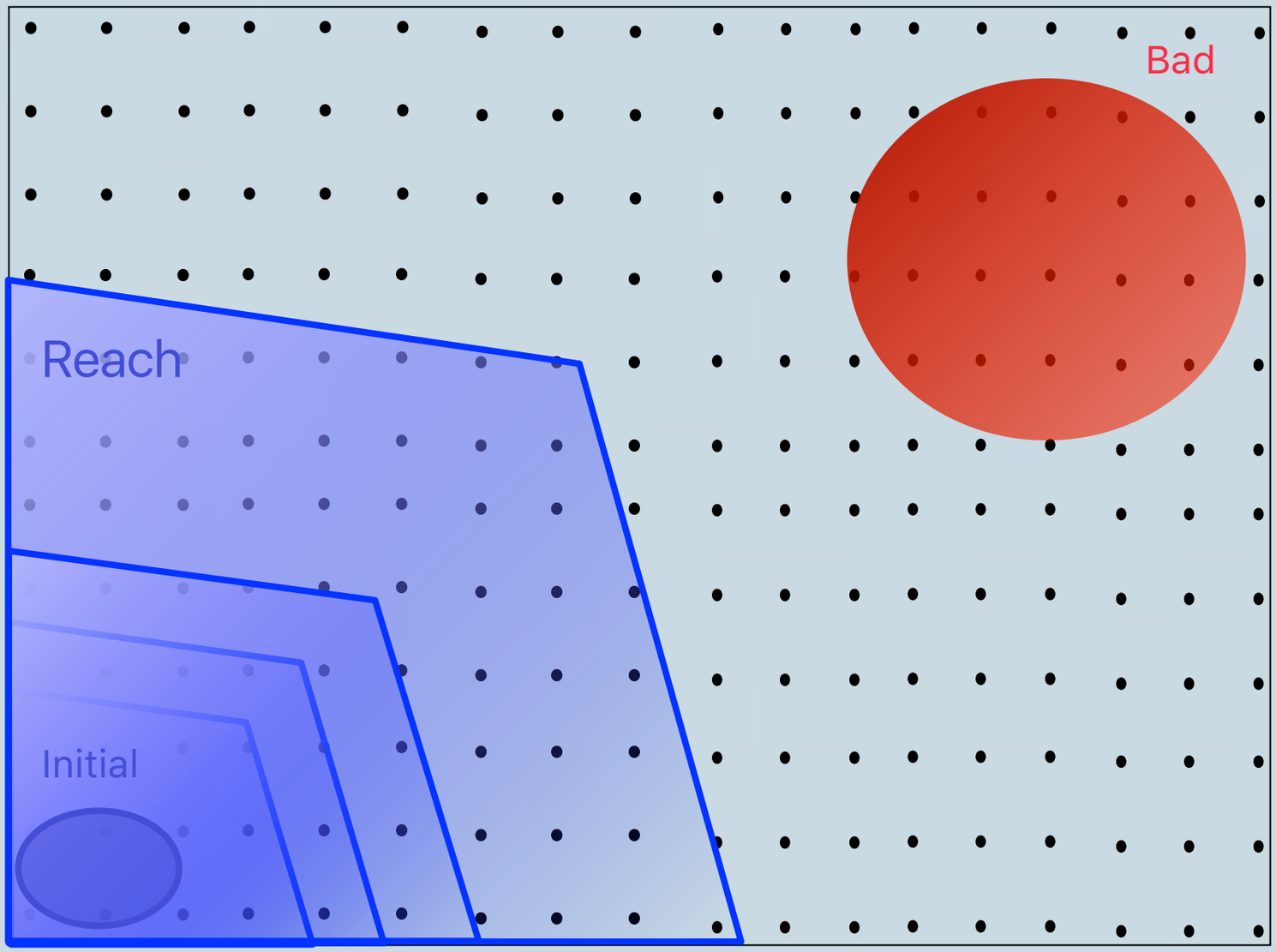
Mooly Sagiv
Tel Aviv University

Sharon Shoham
Tel Aviv University

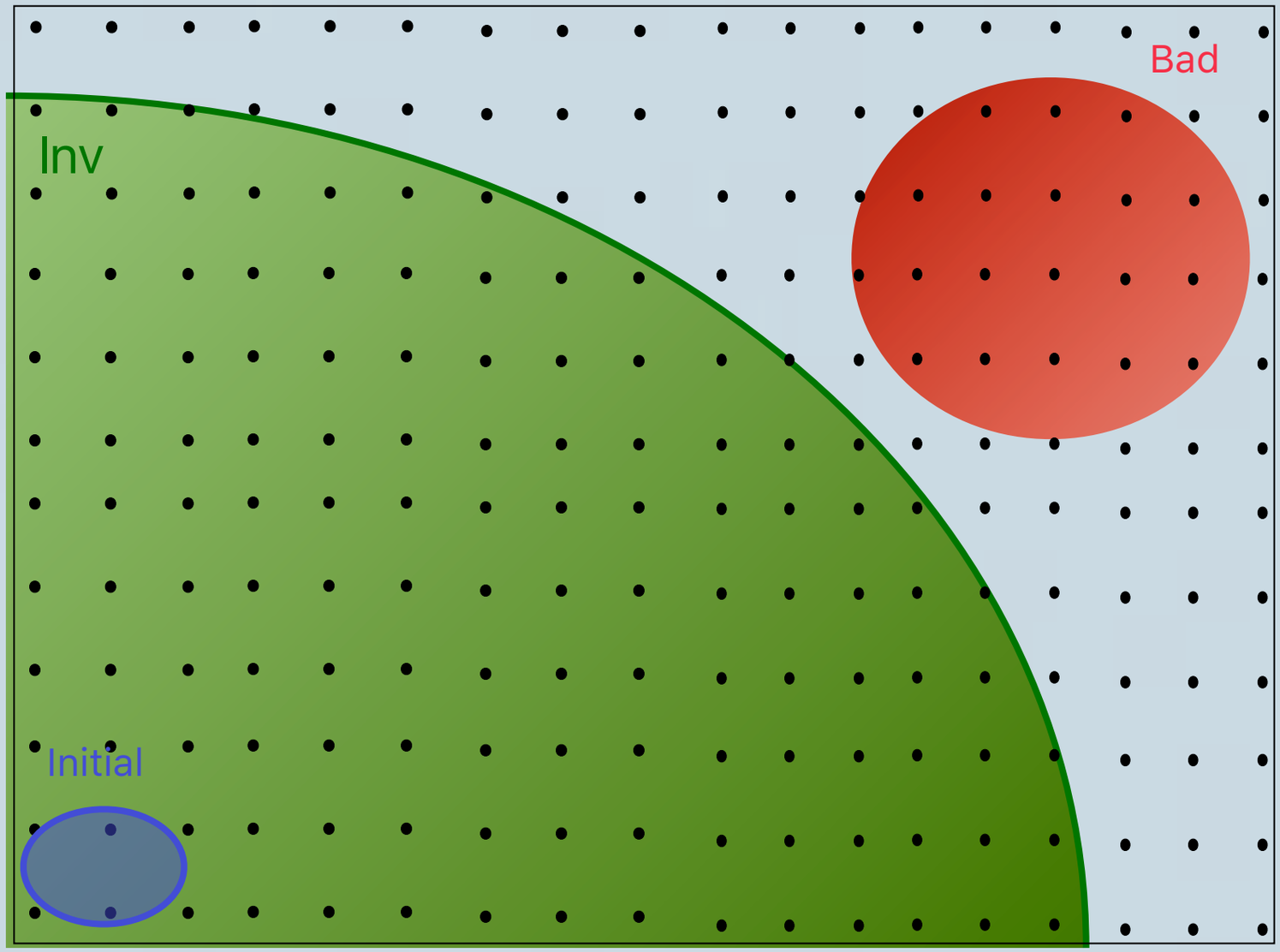
Safety of Infinite-State Systems



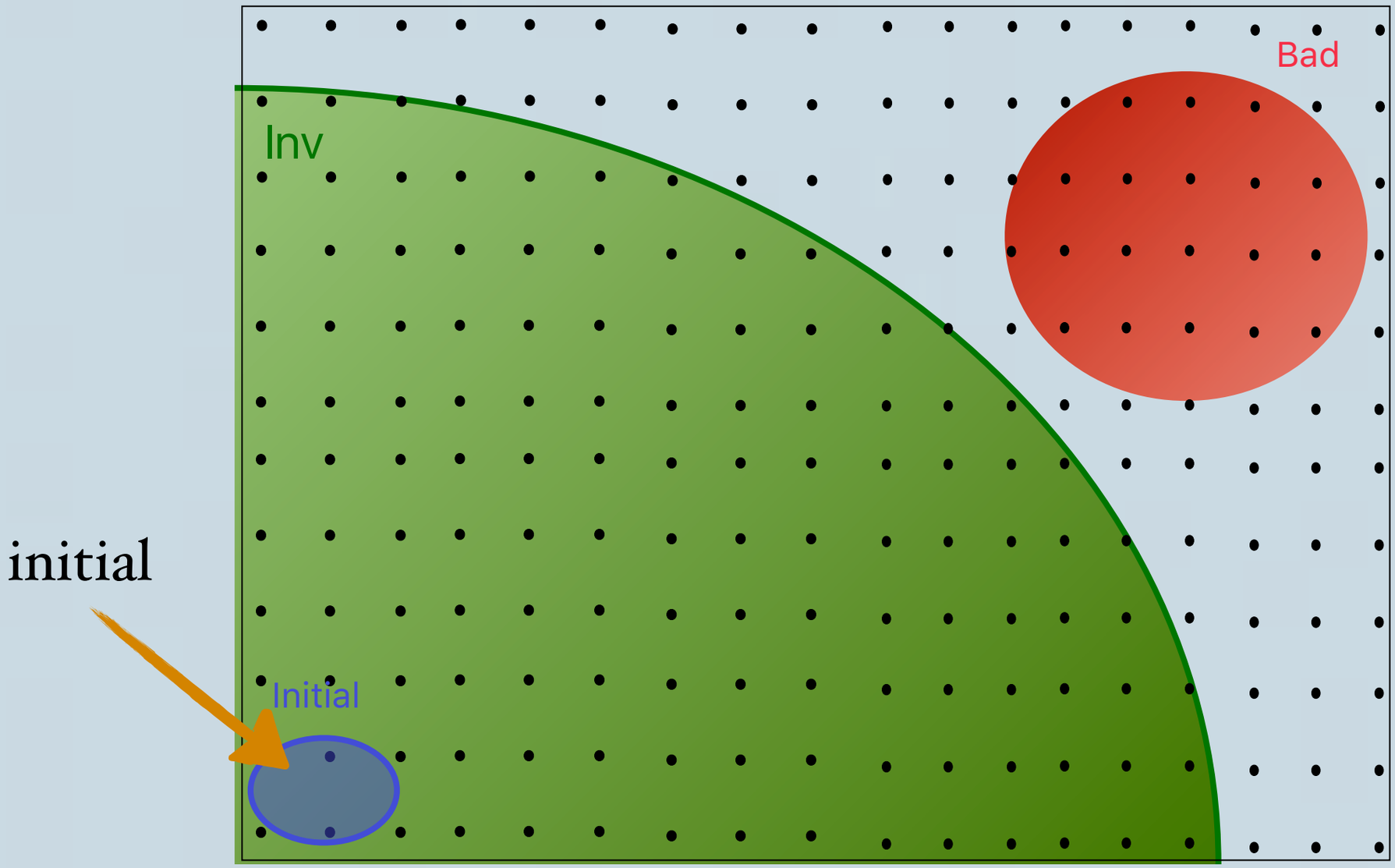
Safety of Infinite-State Systems



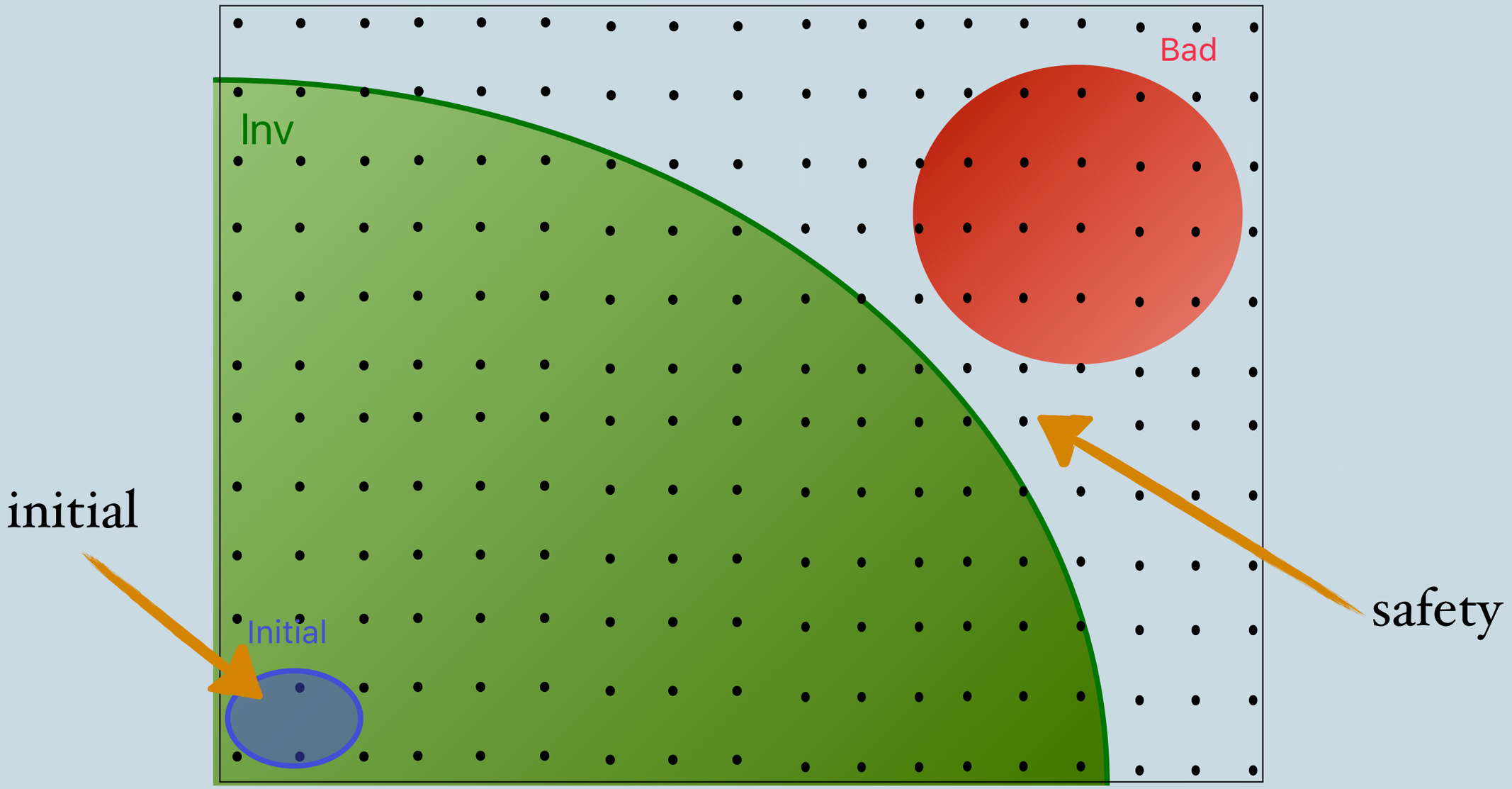
Inductive Invariants



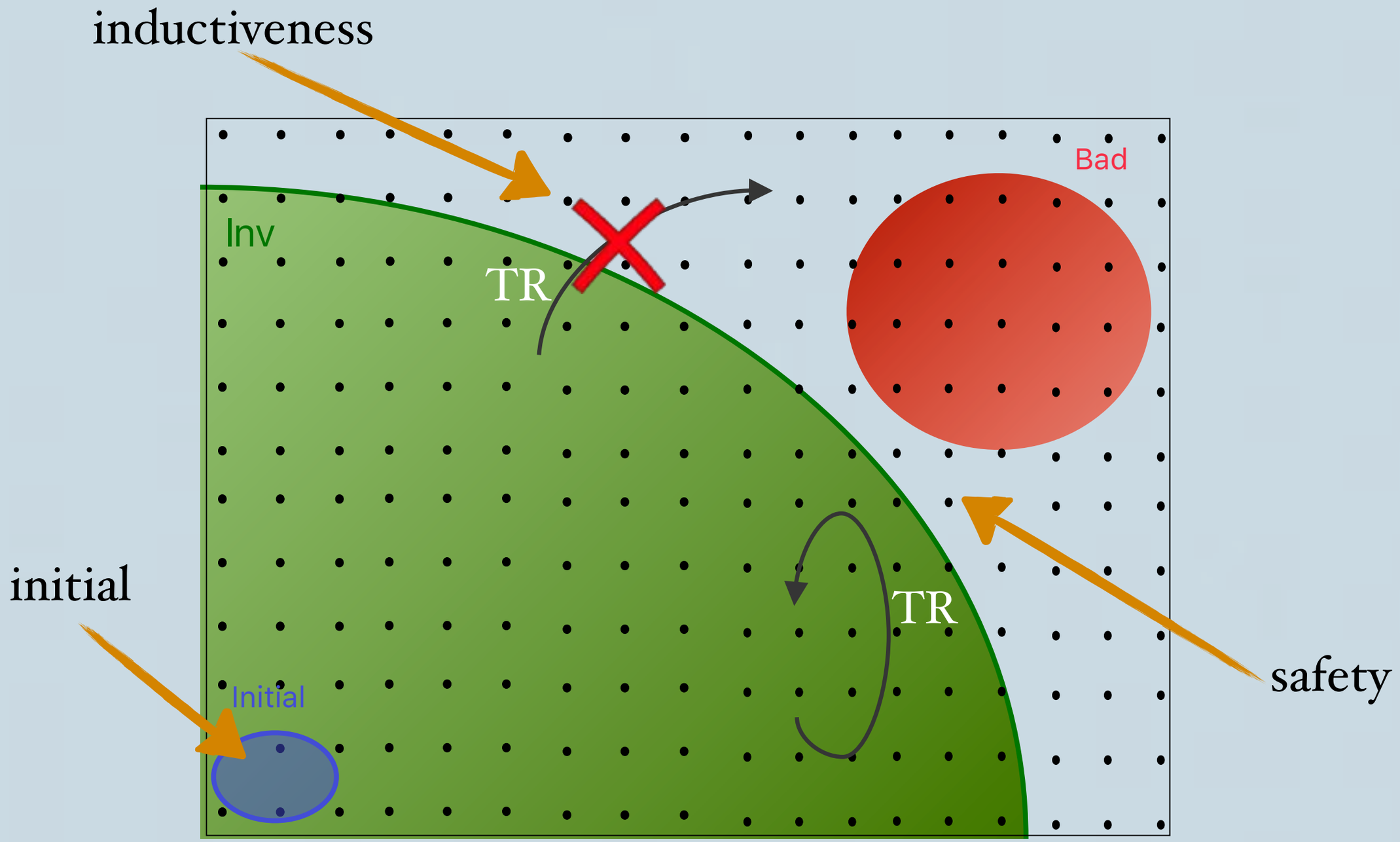
Inductive Invariants



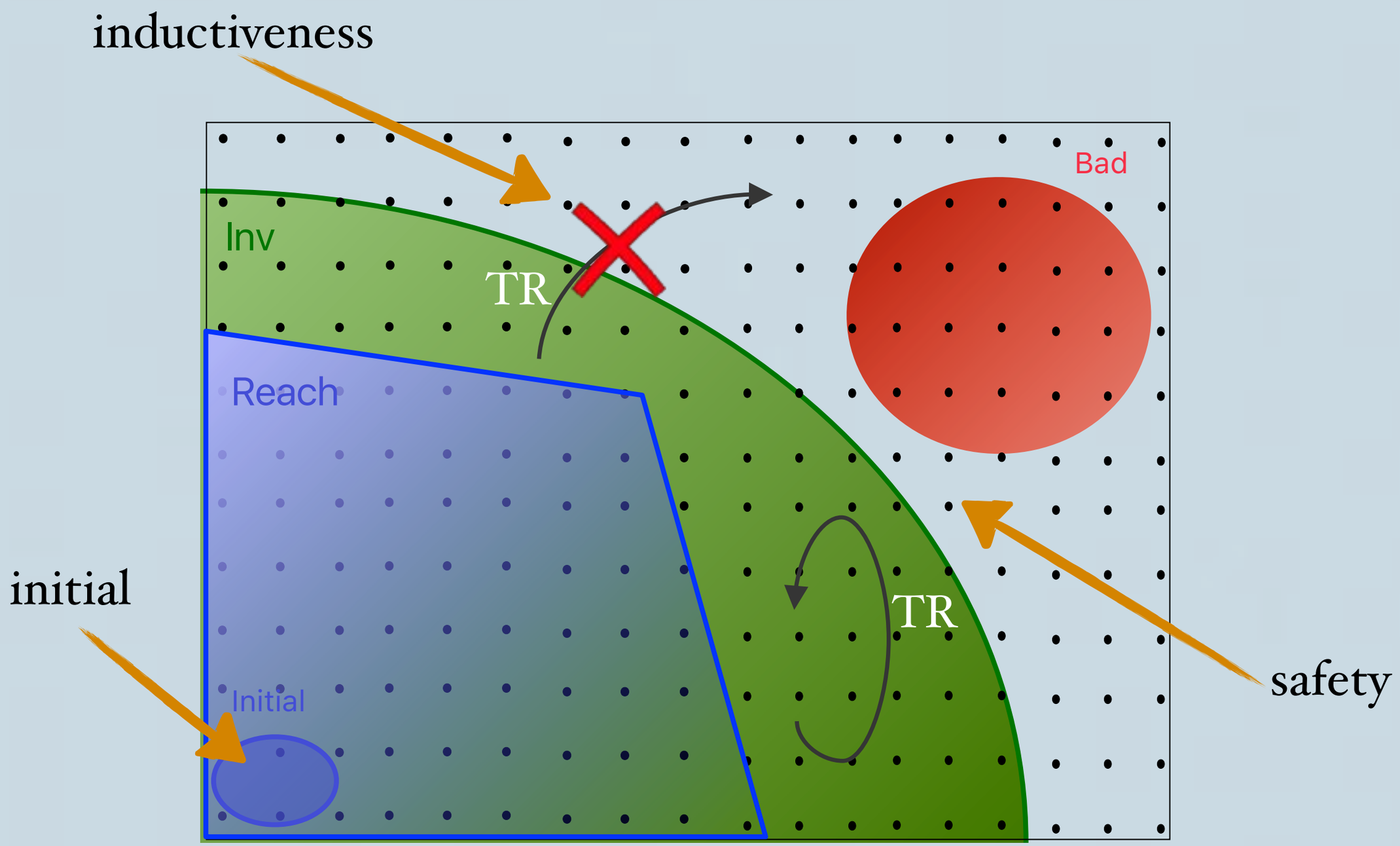
Inductive Invariants



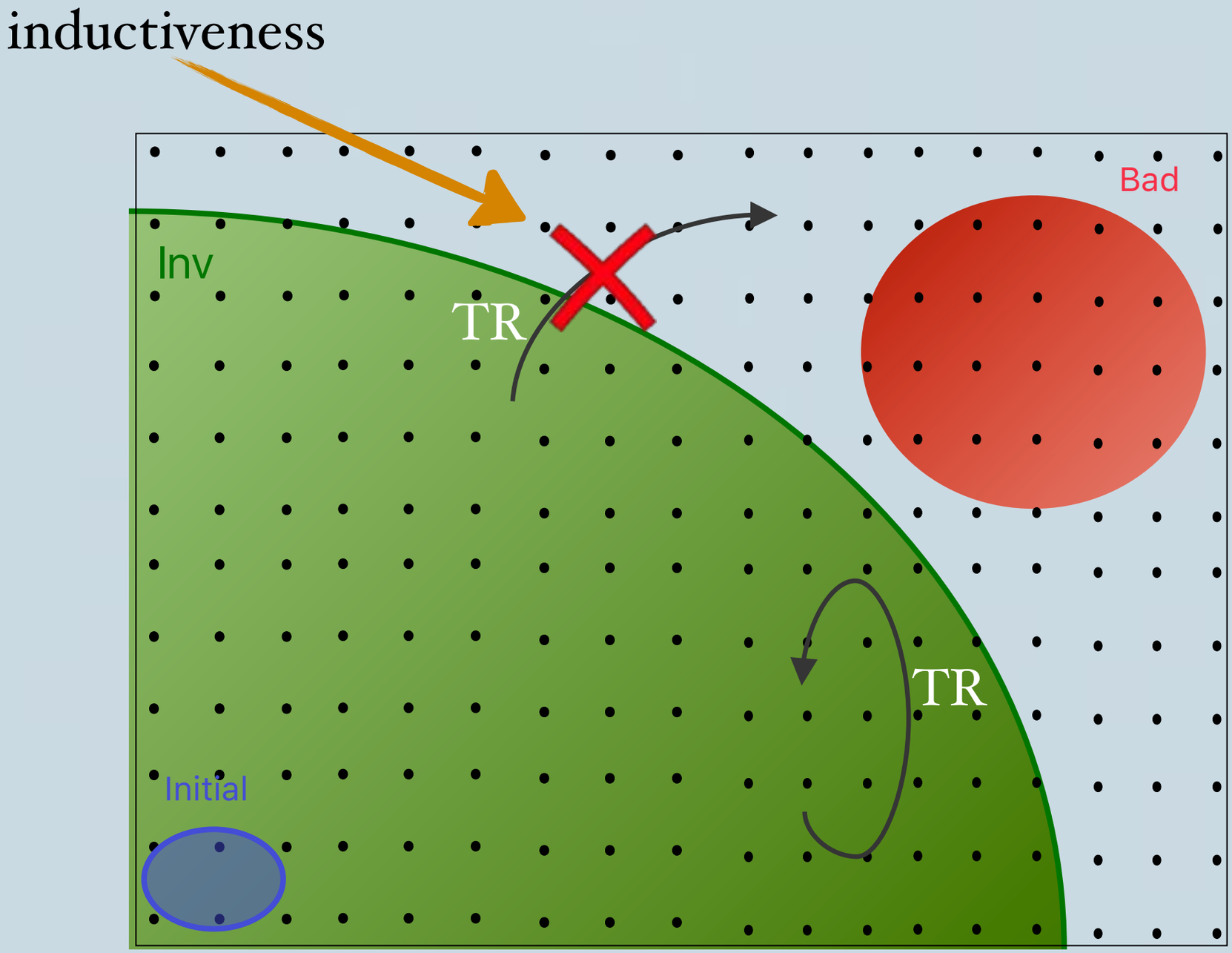
Inductive Invariants



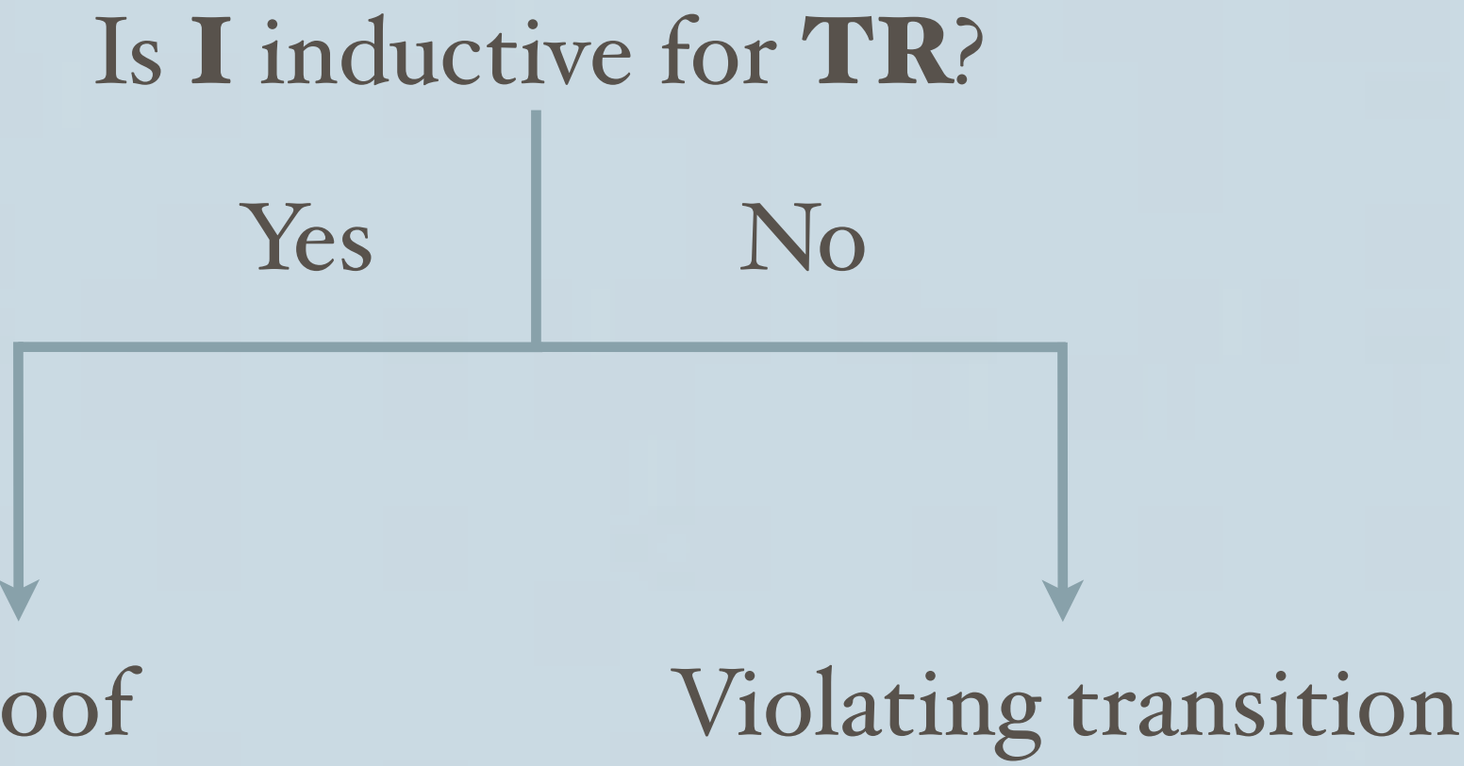
Inductive Invariants



Checking Inductive Invariants

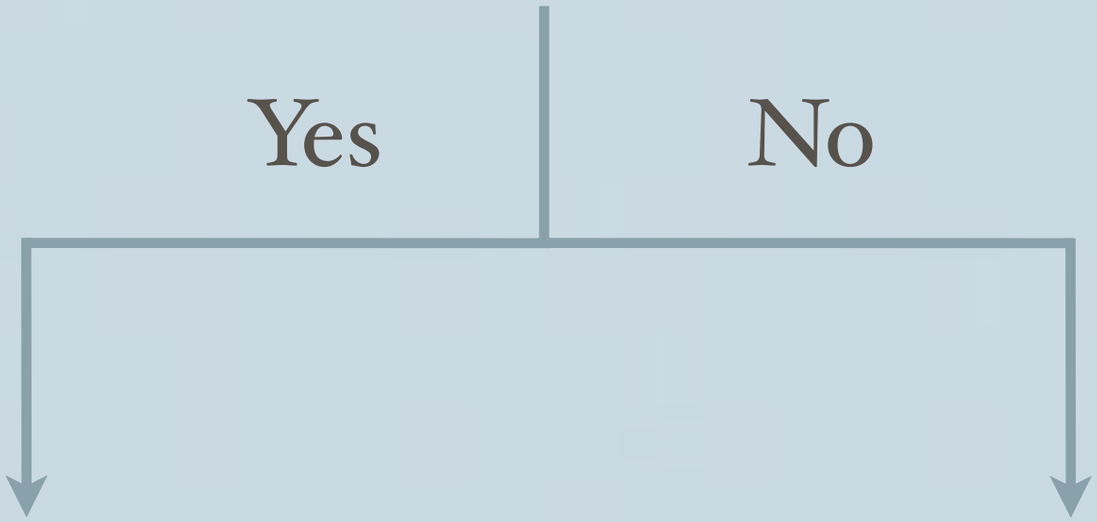


Checking Inductive Invariants



Checking Inductive Invariants

Is **I** inductive for **TR**?



Goal: check automatically

Checking Inductive Invariants

$$\boldsymbol{I}(V) \wedge \boldsymbol{TR}(V, V') \wedge \neg \boldsymbol{I}(V')$$

invariant inductive iff formula unsat

Checking Inductive Invariants

$$\boldsymbol{I}(V) \wedge \boldsymbol{TR}(V, V') \wedge \neg \boldsymbol{I}(V')$$

invariant inductive iff formula unsat

Pure First-Order Logic Modeling

+ Quantification - Arithmetic

- Shape Analysis [Itzhaky et al. CAV'13, POPL'14, CAV'14, CAV'15]
- Software-Defined Networks [Ball et al. PLDI'14]
- Distributed protocols [Padon et al. PLDI'16]
- Concurrent Modification Errors in Java programs [Frumkin et al. VMCAI'17]

Effectively Propositional Logic (EPR)

- No function symbols: constants + relations
- No arithmetic
- Restricted quantifier prefix: ∃*∀*
- Satisfiability is decidable

Effectively Propositional Logic (EPR)

- No function symbols: constants + relations
- No arithmetic
- Restricted quantifier prefix: $\exists^*\forall^*$
- Satisfiability is decidable

$$\mathrm{TR} = \exists^*\forall^* \qquad\qquad \mathrm{Inv} = \forall^*$$

Effectively Propositional Logic (EPR)

- No function symbols: constants + relations
- No arithmetic
- Restricted quantifier prefix: $\exists^*\forall^*$
- Satisfiability is decidable

$$\mathrm{TR} = \exists^*\forall^* \qquad\qquad \mathrm{Inv} = \forall^*$$

$$\mathrm{Inv} \wedge \mathrm{TR} \wedge \neg\mathrm{Inv}'$$
$$\forall^* \wedge \exists^*\forall^* \wedge \exists^*$$

Effectively Propositional Logic (EPR)

- No function symbols: constants + relations
- No arithmetic
- Restricted quantifier prefix: $\exists^*\forall^*$
- Satisfiability is decidable

$$\mathrm{TR} = \exists^*\forall^*$$

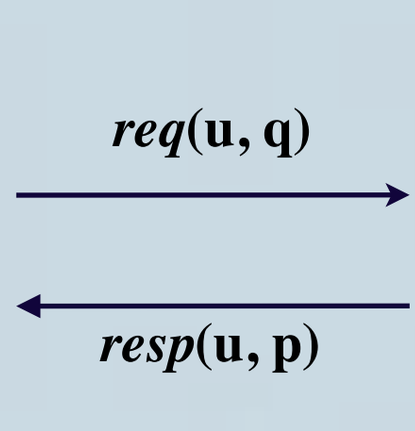
$$\mathrm{Inv} = \forall^*$$

$$\mathrm{Inv} \wedge \mathrm{TR} \wedge \neg \mathrm{Inv}'$$

$$\forall^* \wedge \exists^*\forall^* \wedge \exists^*$$

Example

```
req := ∅; resp := ∅; match := ∅;
action new_request(u) {
    q := new request;
    req := req ∪ {(u, q)}
}
```




```
action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};
    resp := resp ∪ {(u, p)}
}

action check(u, p) {
     if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

I = ∀u, p. *resp*(u, p) → ∃q. *req*(u, q) ∧ *match*(q, p)

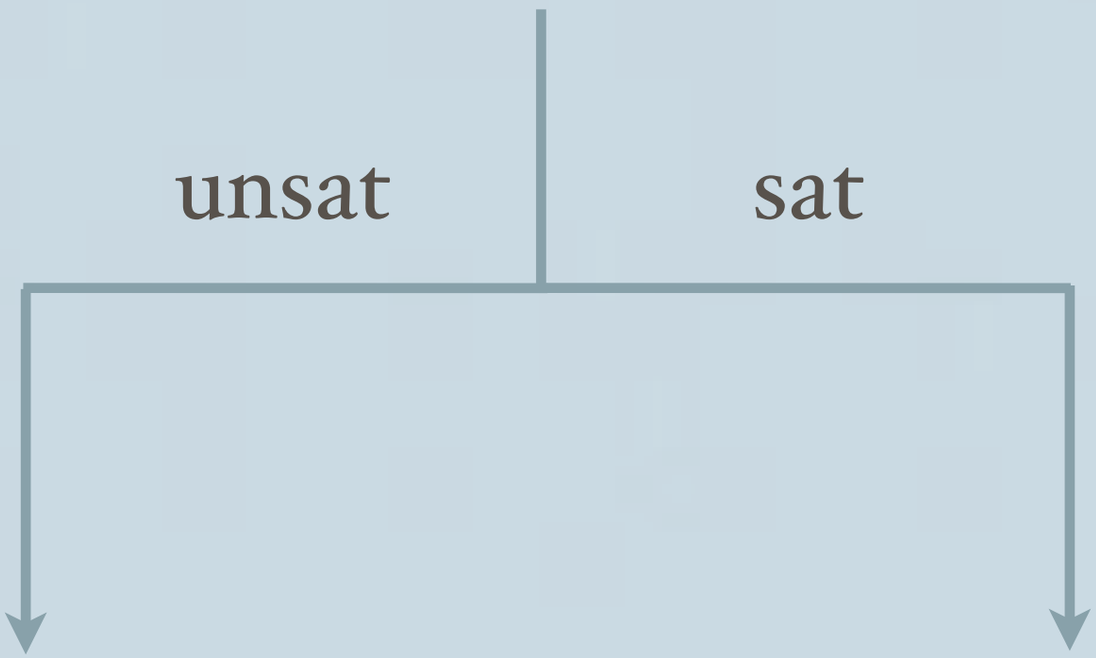
No Longer Decidable

$\text{Inv} = \forall^*$  $\text{Inv} = \forall^*\exists^*$

- Checking inductiveness of $\forall^*\exists^*$ invariants for EPR programs is **undecidable**.

Solving with Quantifier Alternation

$$I \wedge TR \wedge \neg I'$$



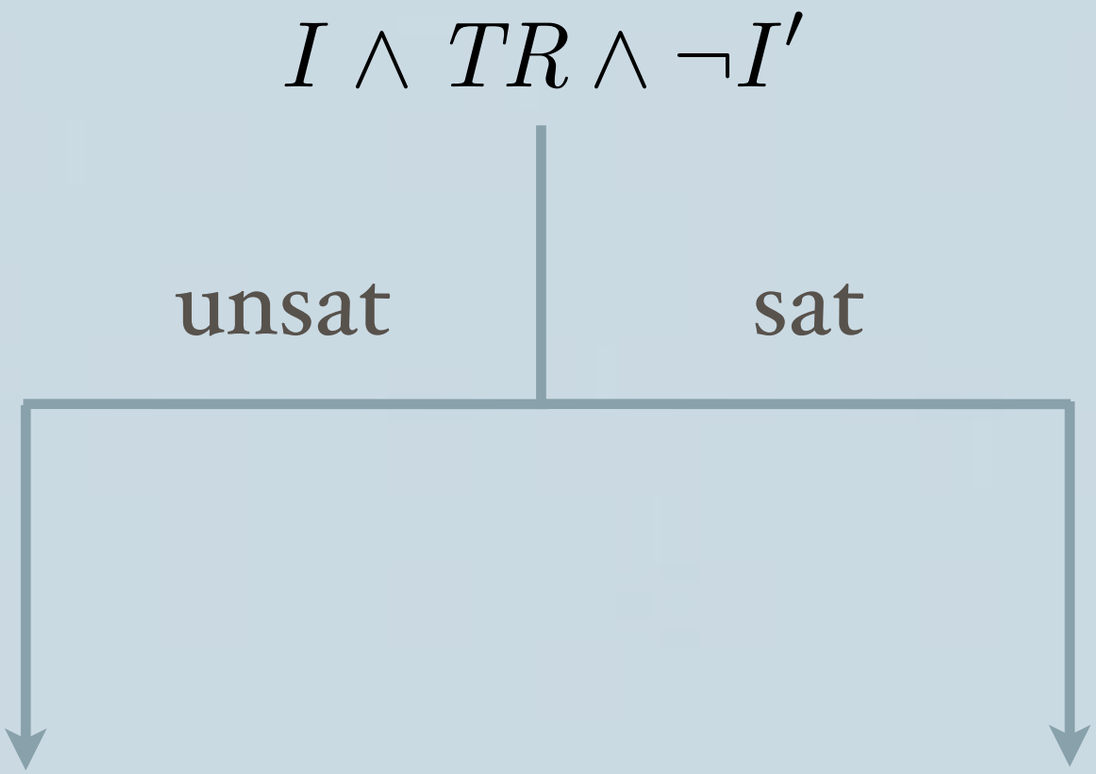
how to find
a proof?

matching loops

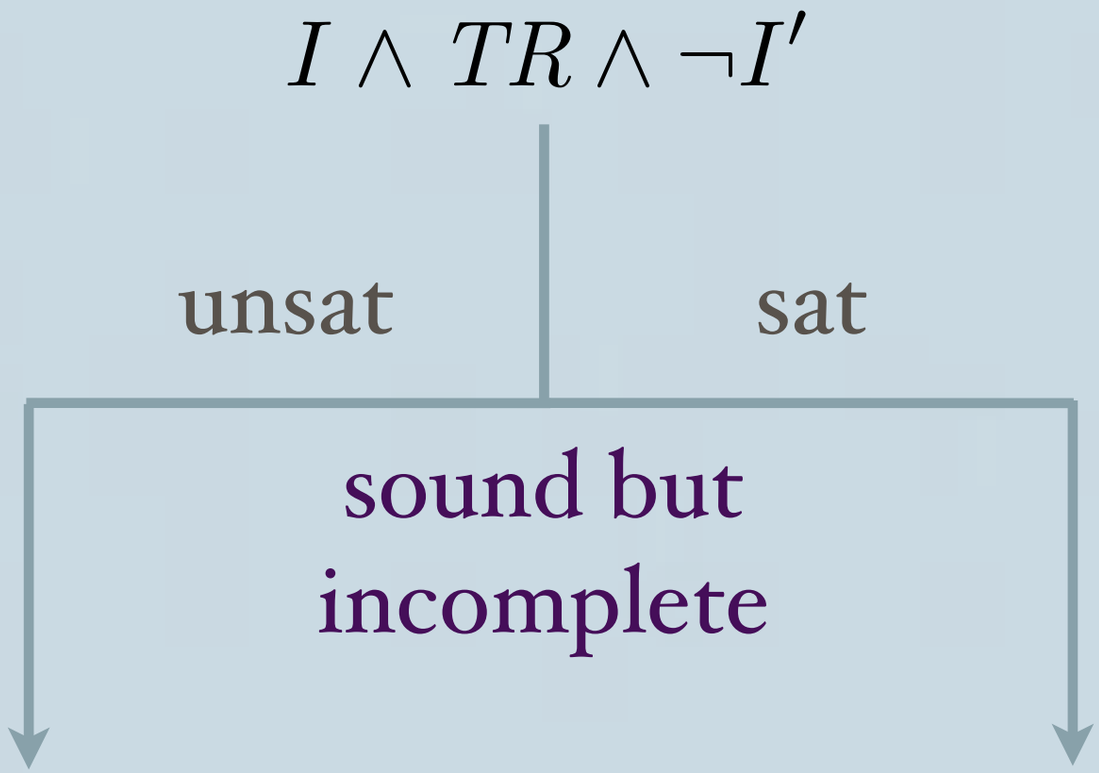
how to find
a cex?

infinite cex

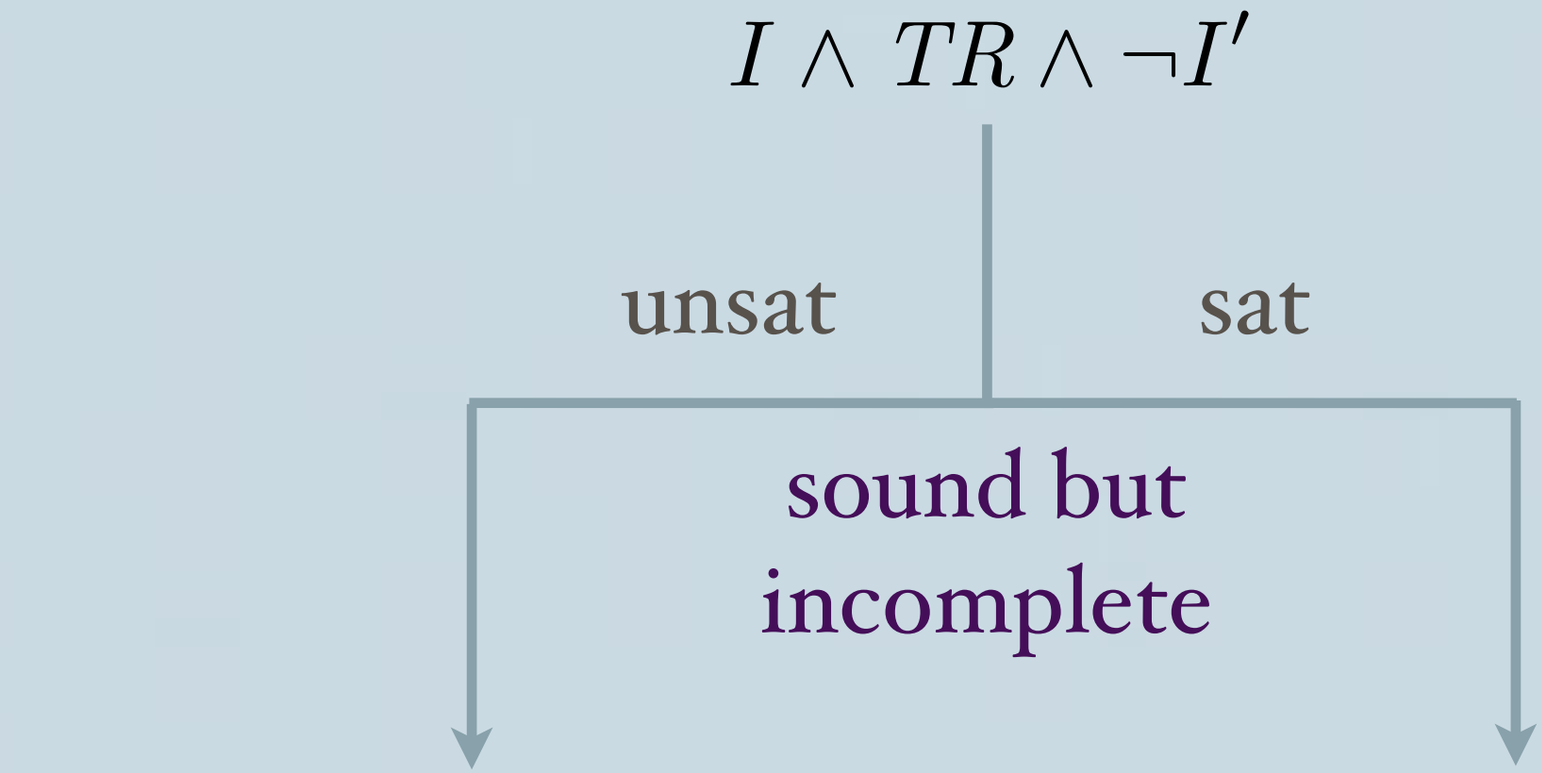
Our Approach: Bounded-Horizon



Our Approach: Bounded-Horizon



Our Approach: Bounded-Horizon



powerful as
manual reductions
to a decidable class
via **instrumentation**

Our Approach: Bounded-Horizon

$$I \wedge TR \wedge \neg I'$$

unsat

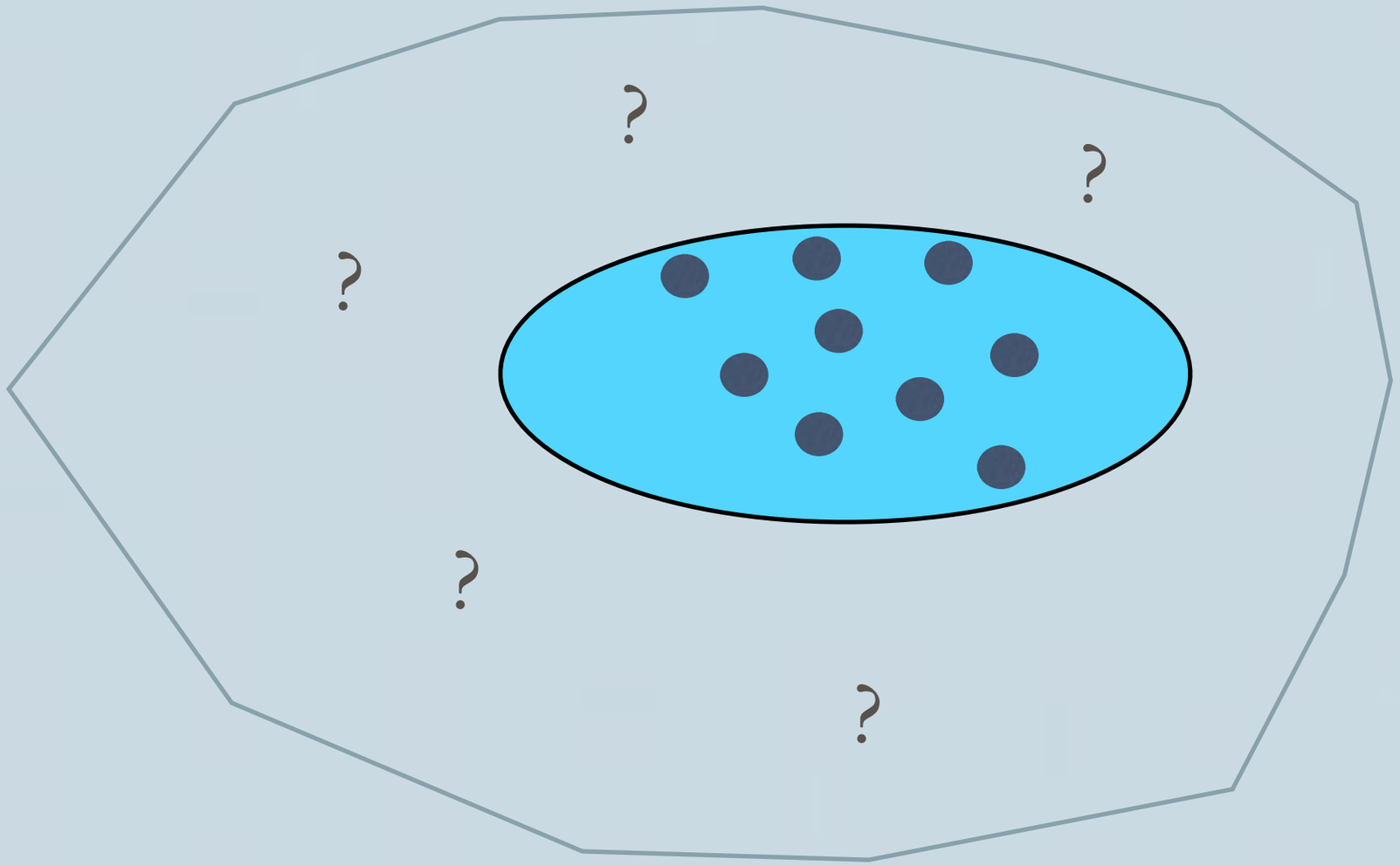
sat

sound but
incomplete

powerful as
manual reductions
to a decidable class
via **instrumentation**

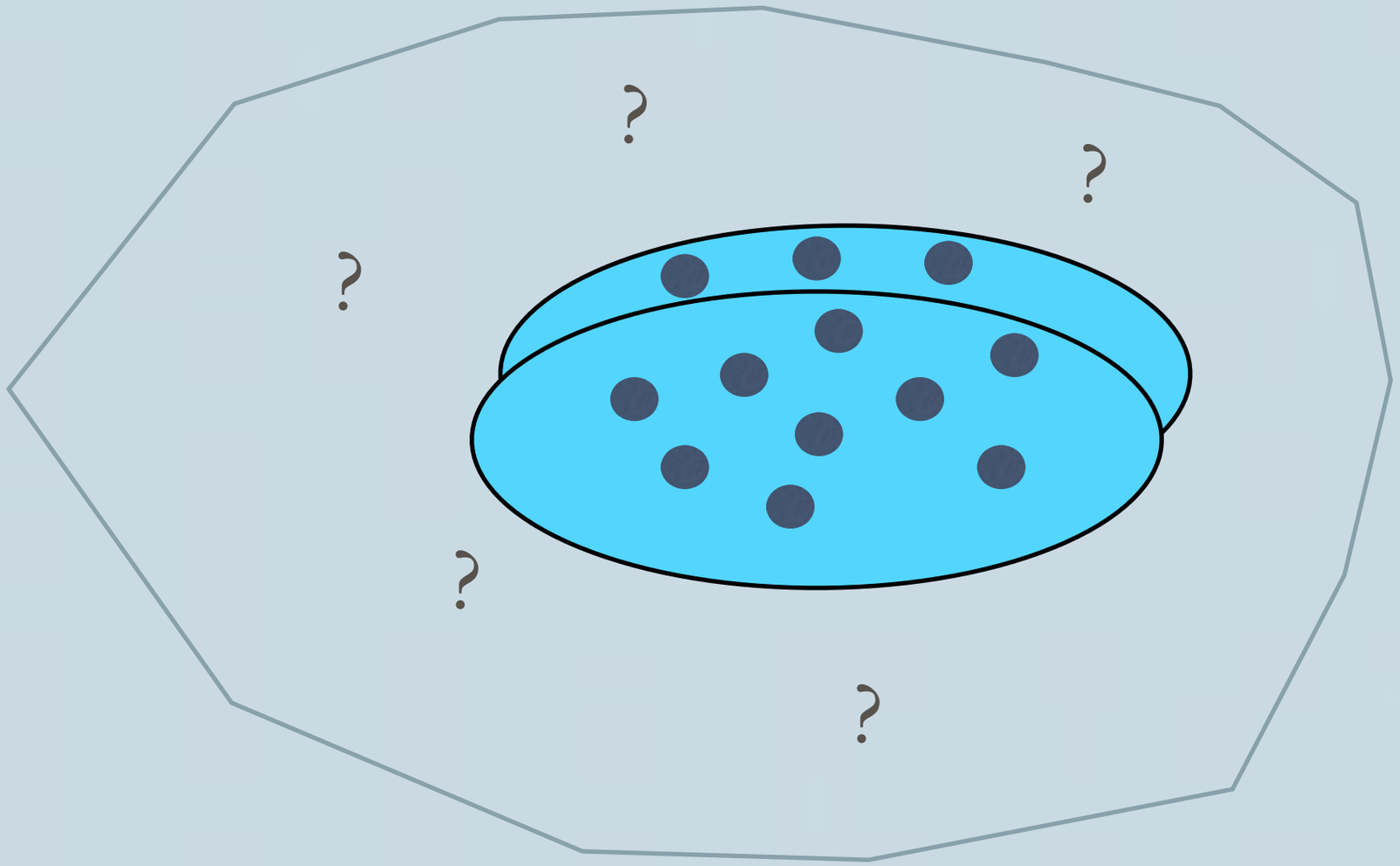
ability of producing
partial cexs
to overcome infinite cexs

Bounded-Horizon: Intuition



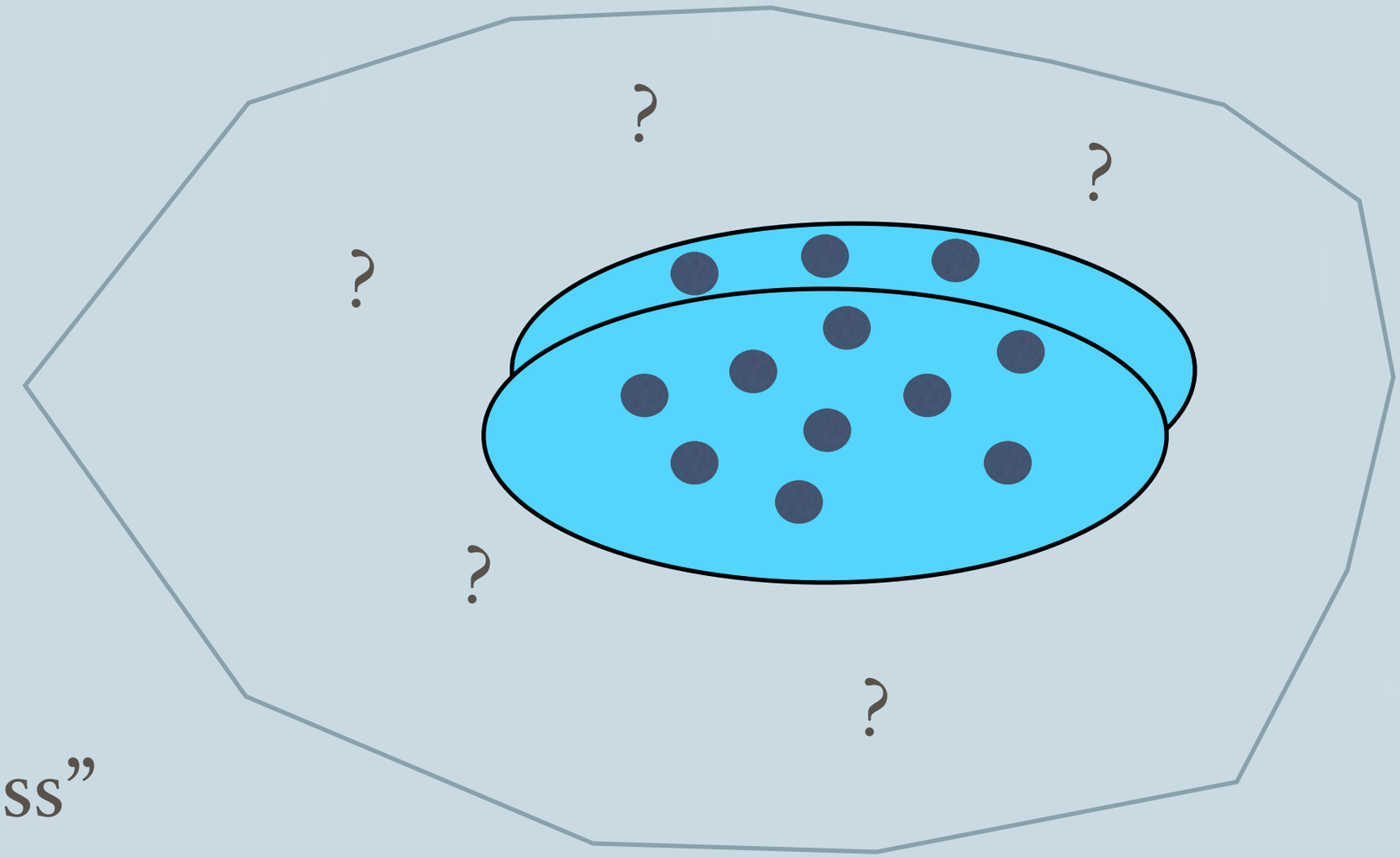
$I \wedge \mathrm{TR} \wedge \neg I'$ is unsat

Bounded-Horizon: Intuition



$I \wedge \mathrm{TR} \wedge \neg I'$ is unsat

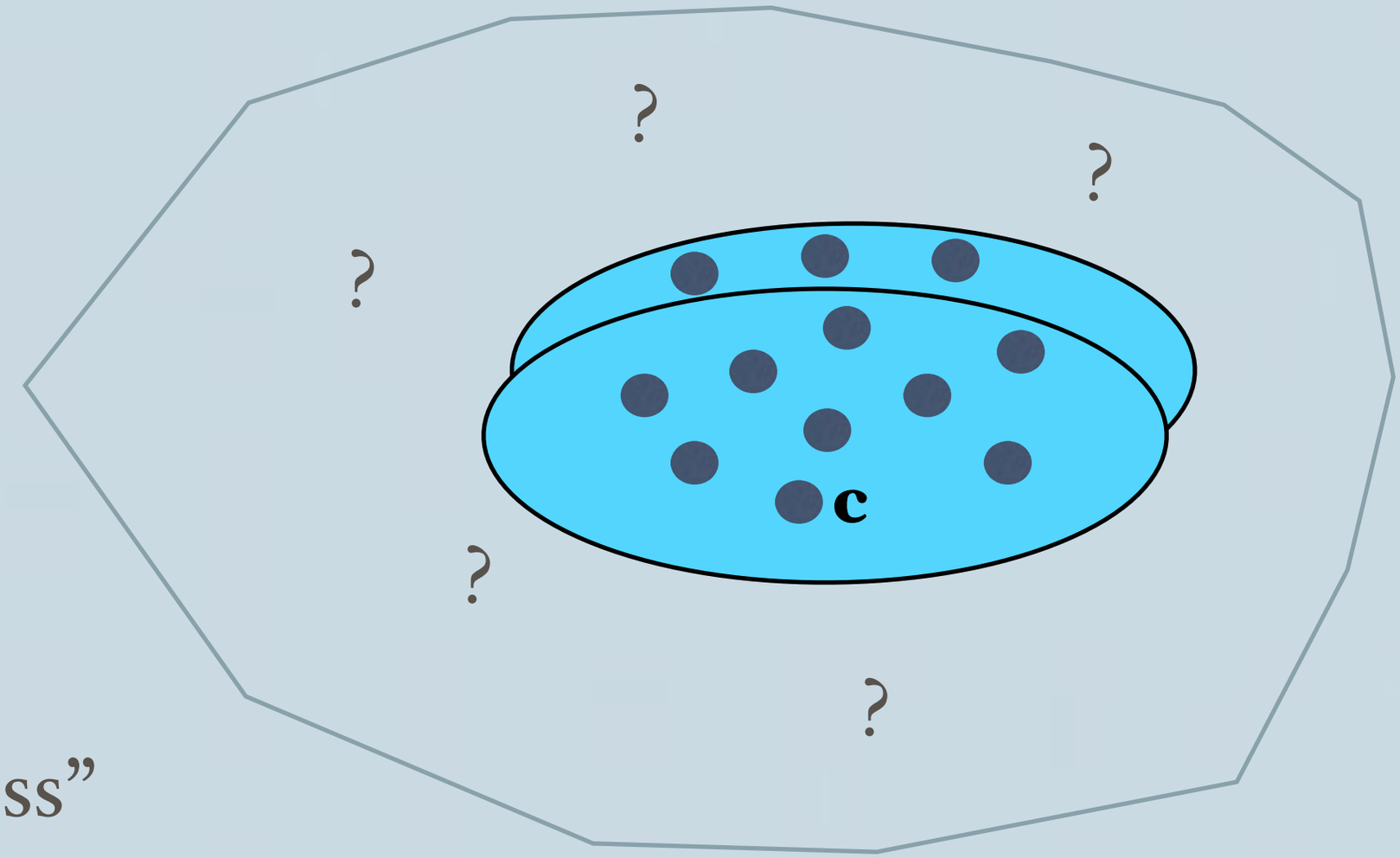
Bounded-Horizon: Intuition



"witness"

$(\forall x.\, \exists y.\, \alpha(x, y)) \wedge \mathrm{TR} \wedge \neg I'$ is unsat

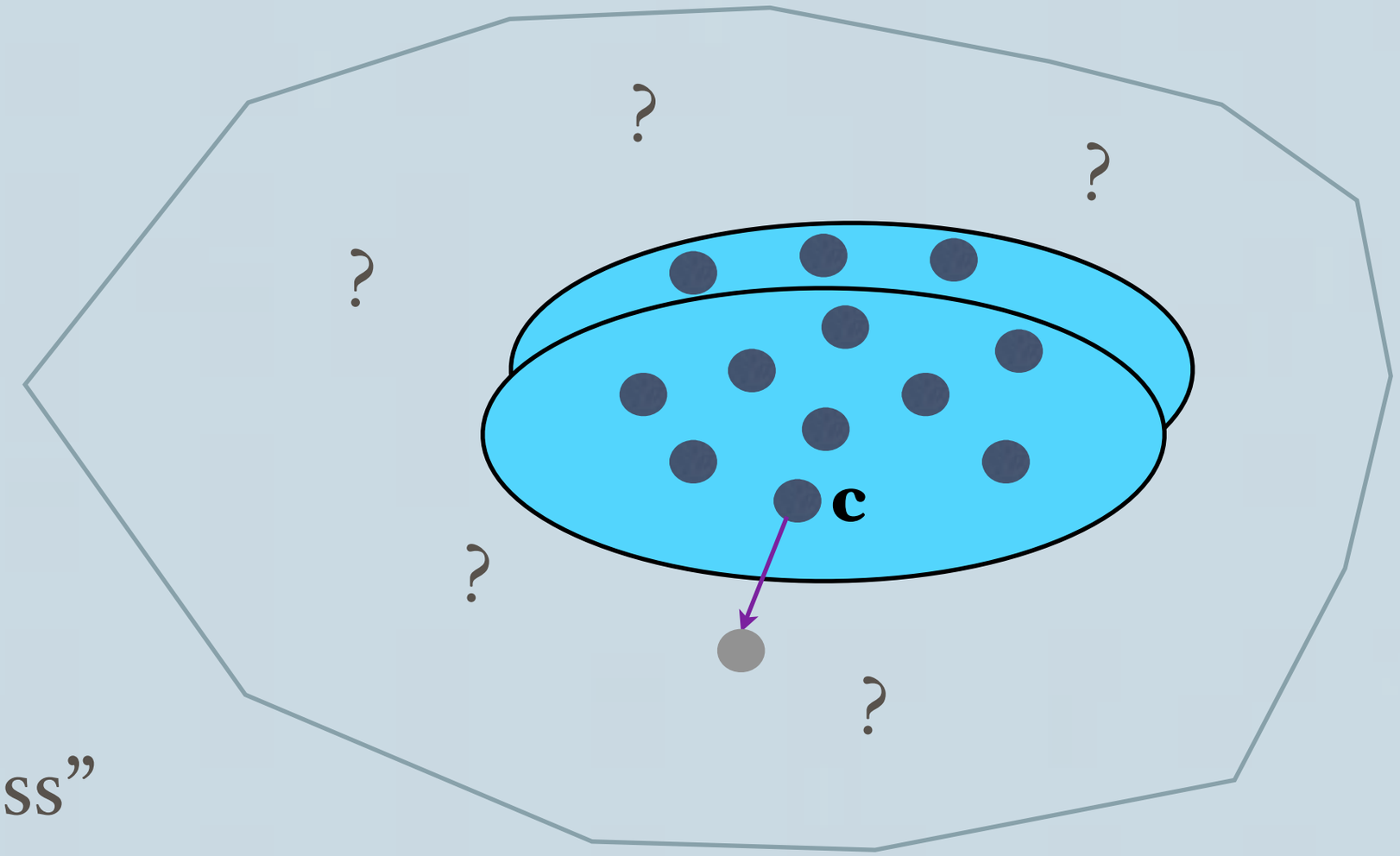
Bounded-Horizon: Intuition



"witness"

$(\forall x.\, \exists y.\, \alpha(x, y)) \wedge \mathrm{TR} \wedge \neg I'$ **is unsat**

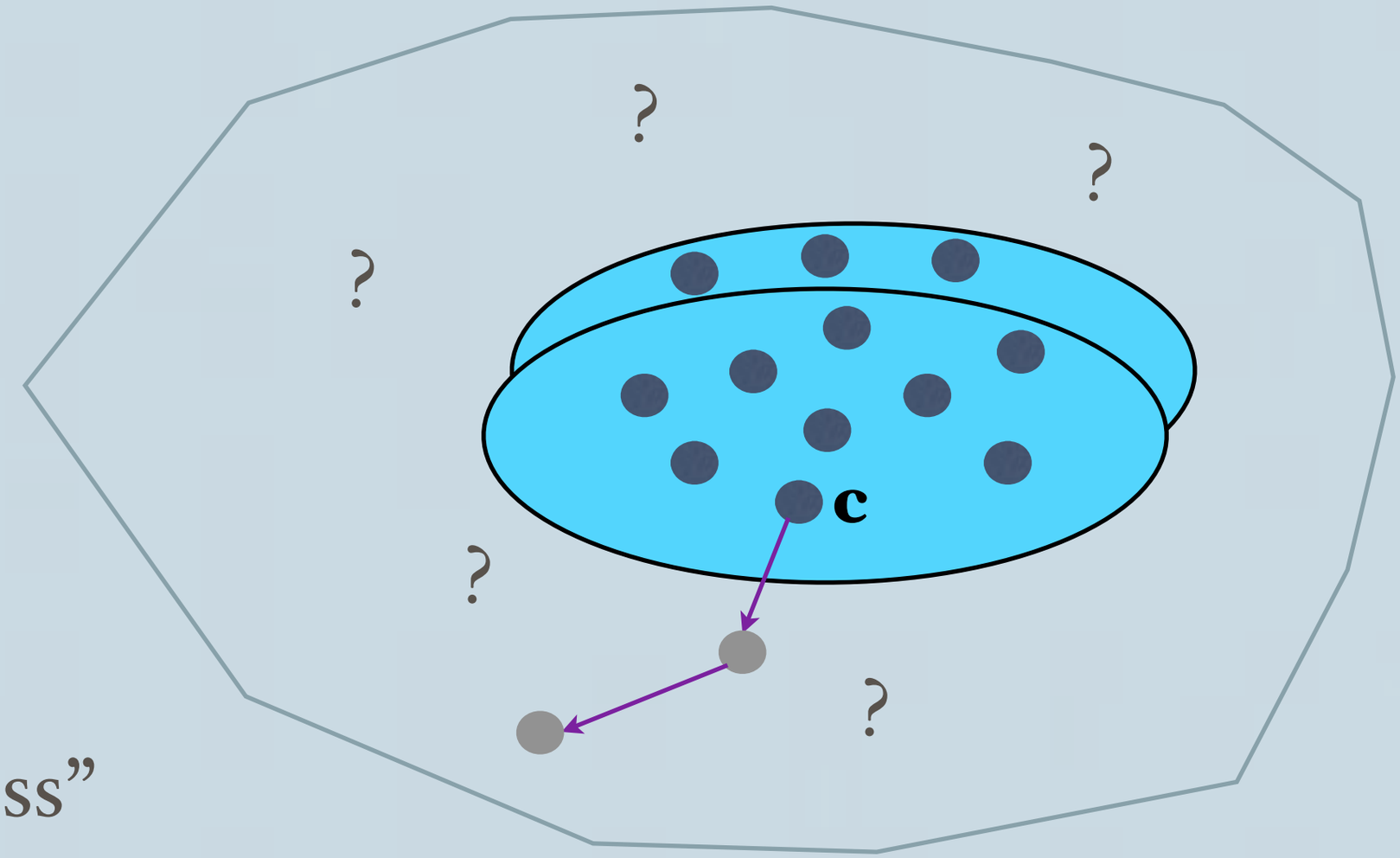
Bounded-Horizon: Intuition



"witness"

$(\forall x.\, \exists y.\, \alpha(x, y)) \wedge \mathrm{TR} \wedge \neg I'$ is unsat

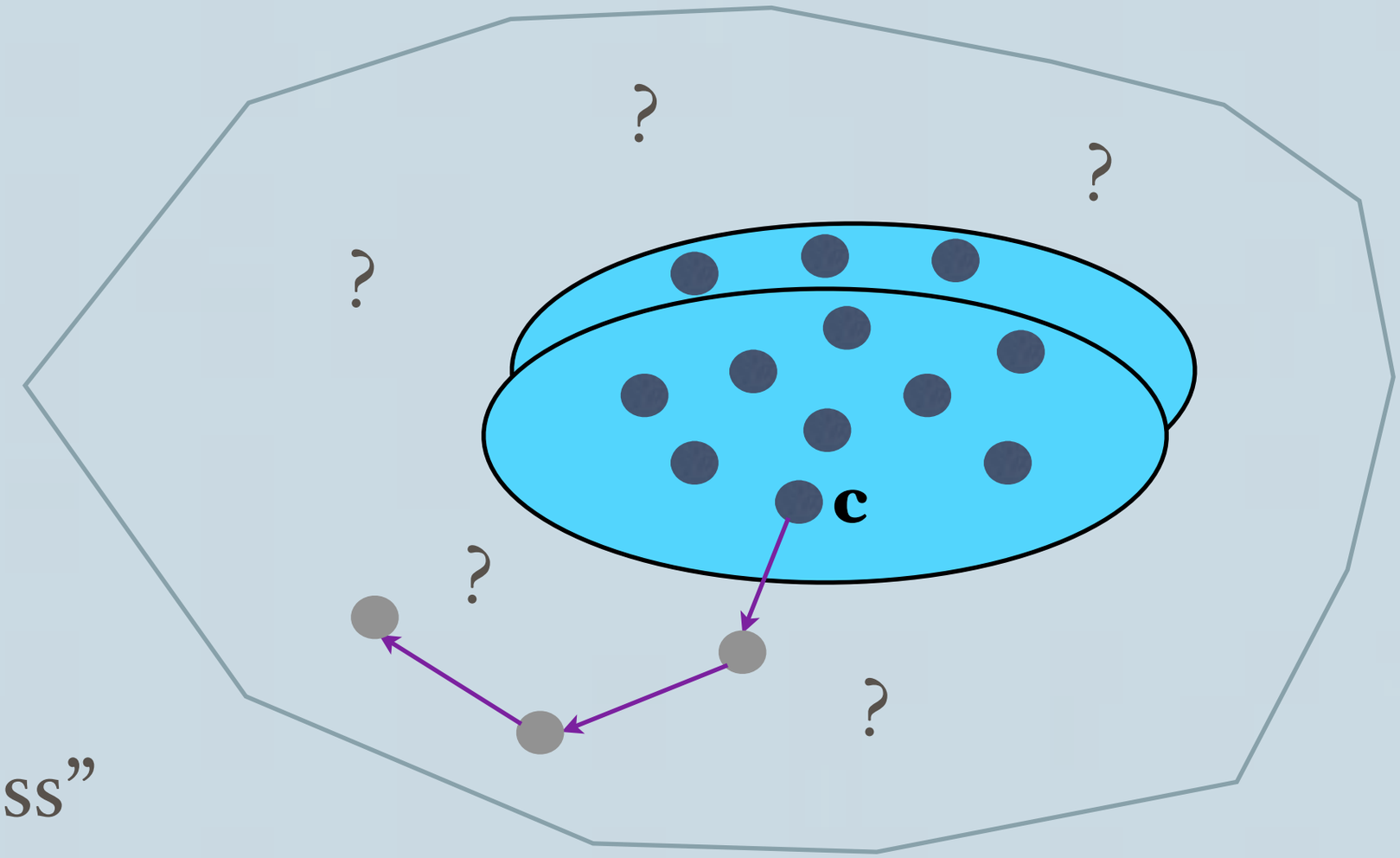
Bounded-Horizon: Intuition



"witness"

$(\forall x.\, \exists y.\, \alpha(x, y)) \wedge \mathrm{TR} \wedge \neg I'$ **is unsat**

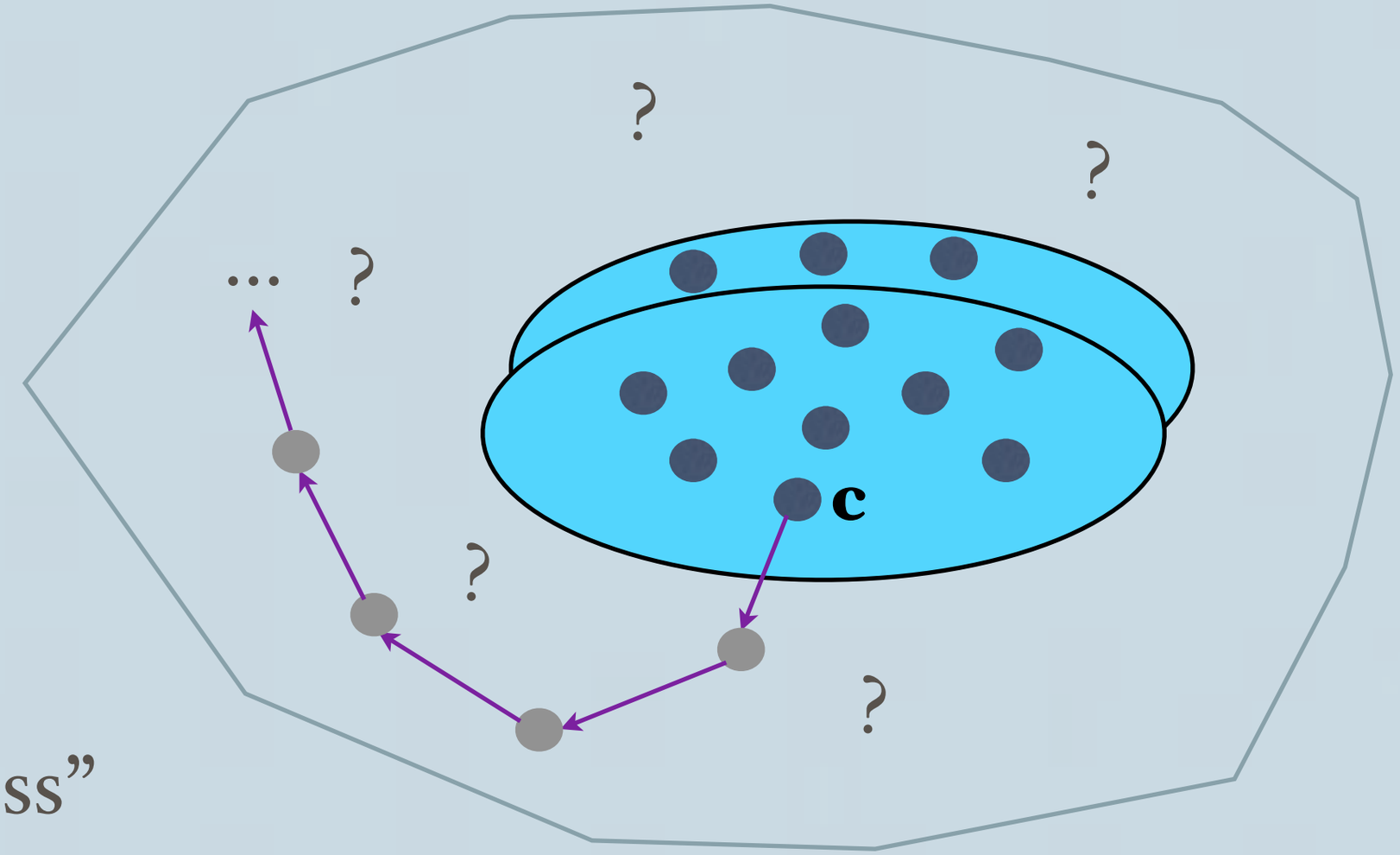
Bounded-Horizon: Intuition



"witness"

$(\forall x.\, \exists y.\, \alpha(x, y)) \wedge \mathrm{TR} \wedge \neg I'$ **is unsat**

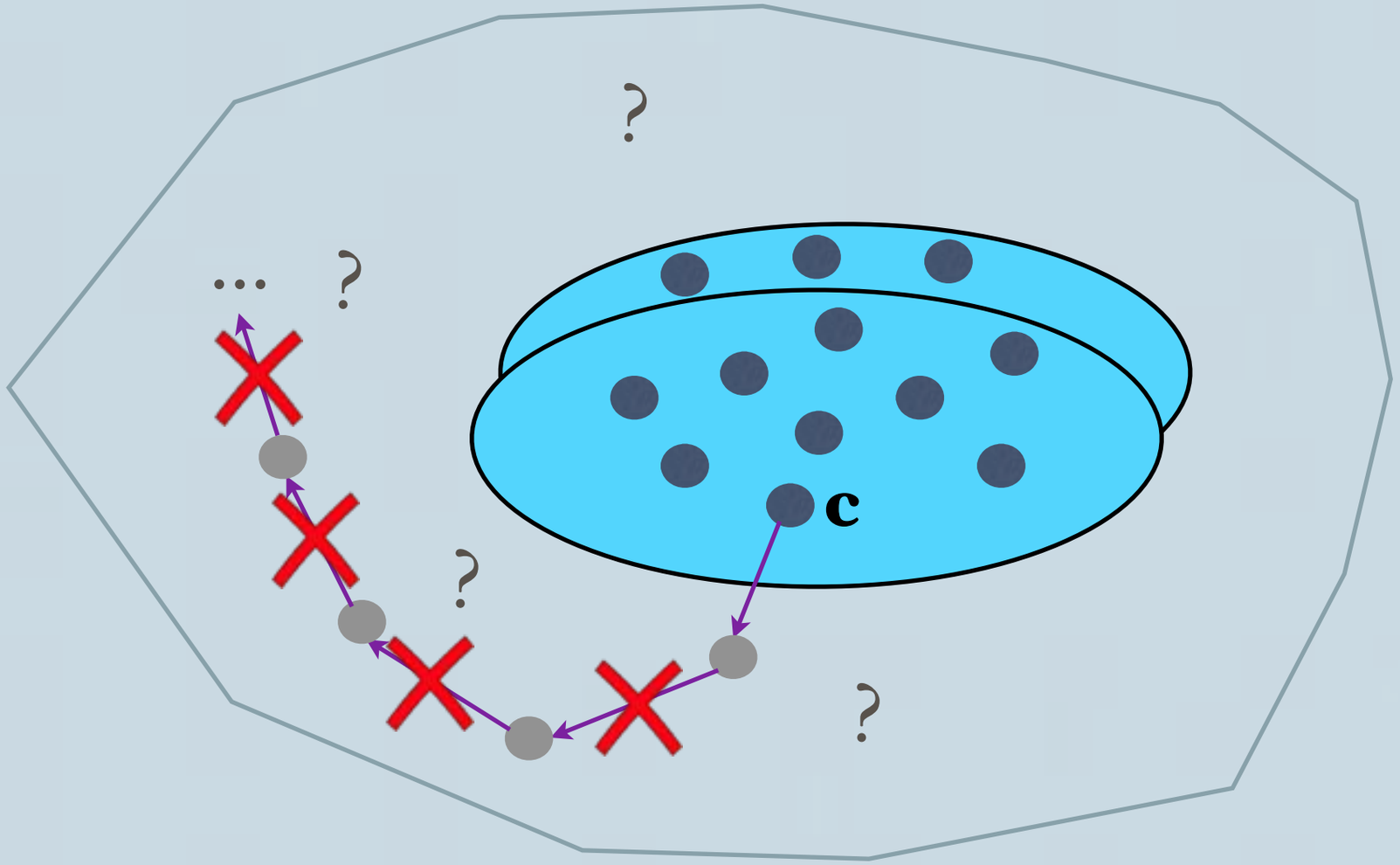
Bounded-Horizon: Intuition



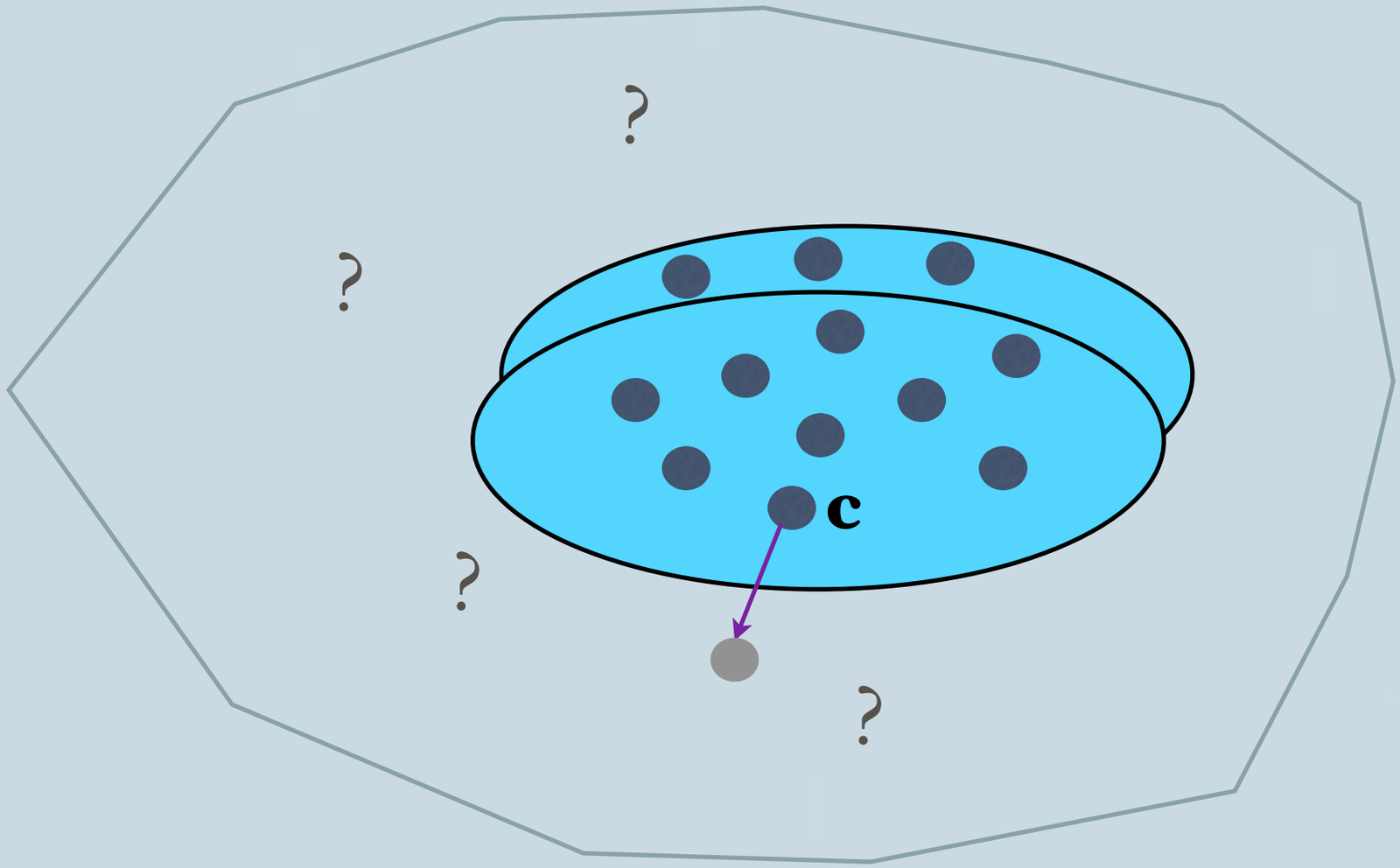
"witness"

$(\forall x.\, \exists y.\, \alpha(x, y)) \wedge \mathrm{TR} \wedge \neg I'$ is unsat

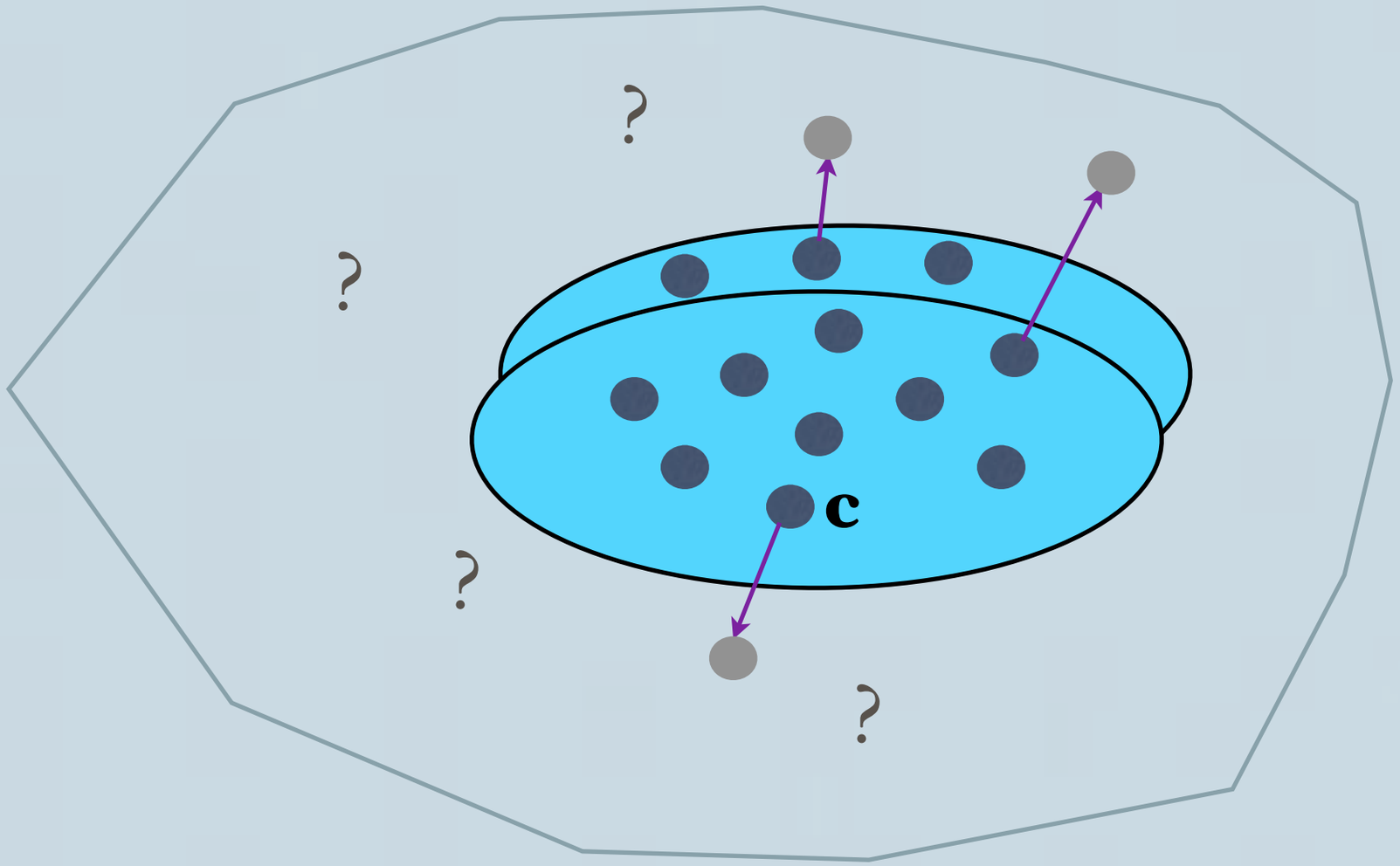
Bounded-Horizon: Intuition



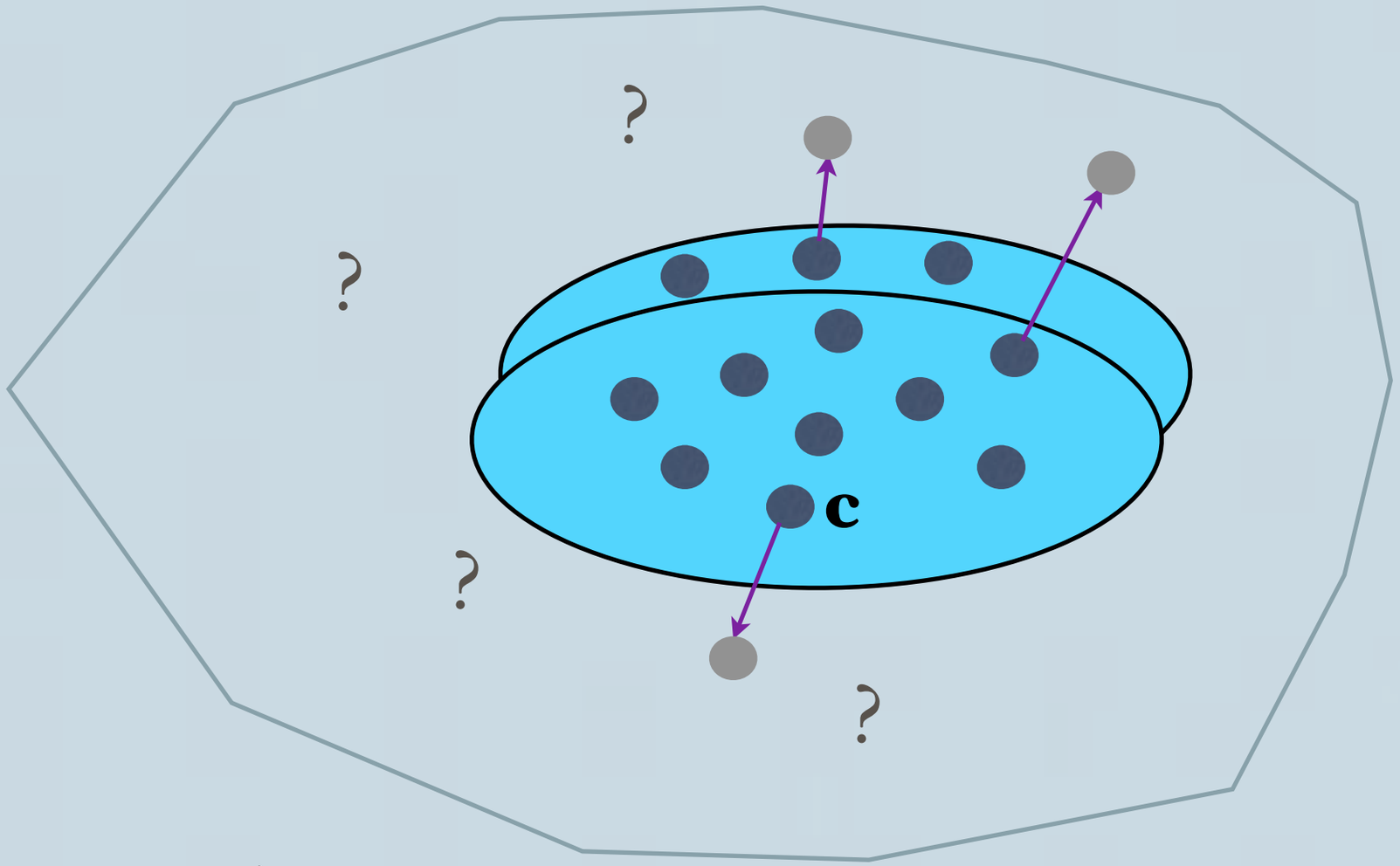
Bounded-Horizon: Intuition



Bounded-Horizon: Intuition

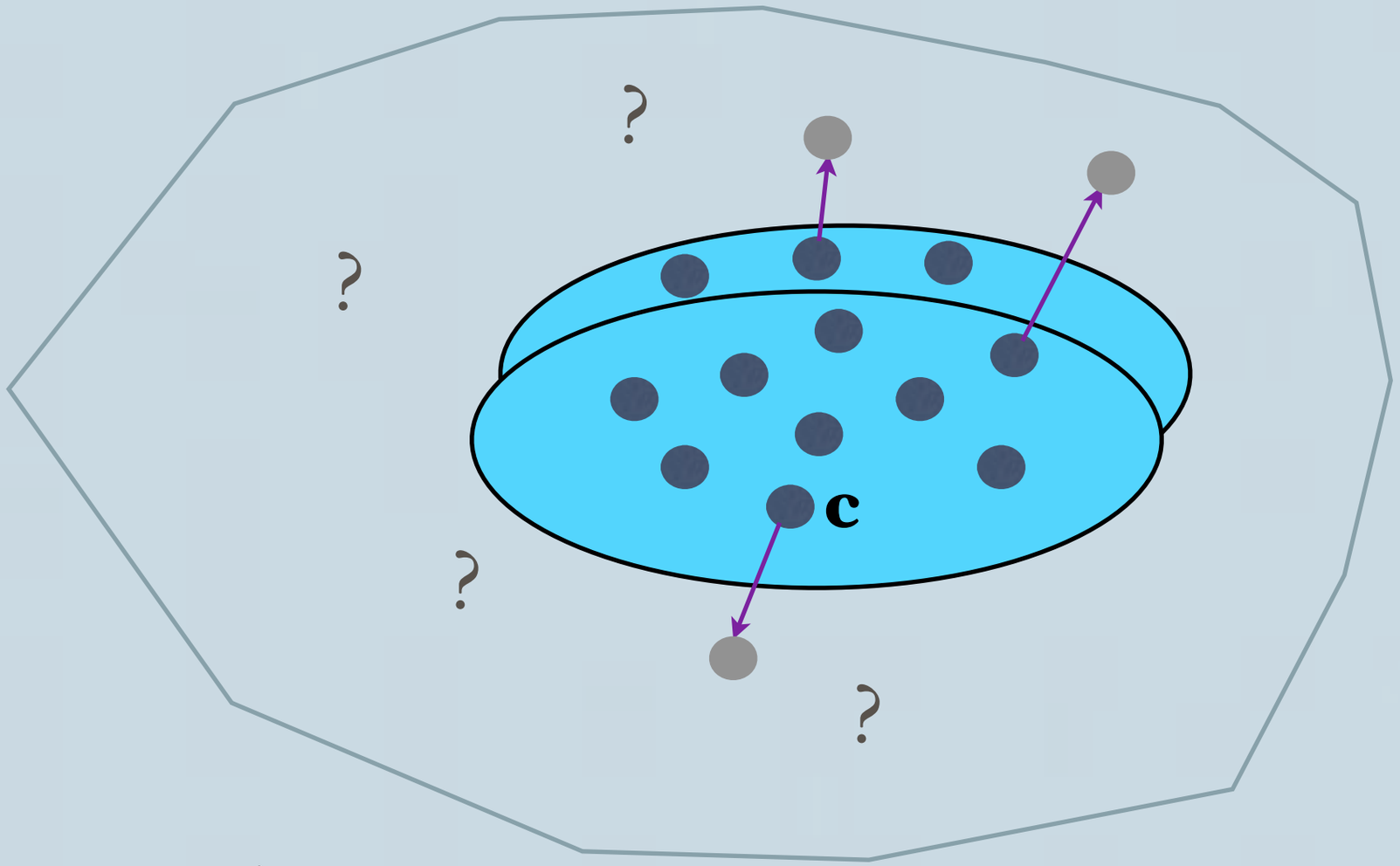


Bounded-Horizon: Intuition



$$\left(\bigwedge_{c} \exists y.\, \alpha(c, y)\right) \wedge \mathrm{TR} \wedge \neg I' \text{ is unsat}$$

Bounded-Horizon: Intuition



$$\left(\bigwedge_{c} \exists y.\, \alpha(c, y)\right) \wedge \mathrm{TR} \wedge \neg I' \text{ is unsat}$$

decidable!

Quantifier Instantiation

Check SAT:

$$\forall x.\ \exists y.\ \alpha(x, y) \wedge \mathrm{TR} \wedge \neg I'$$

Quantifier Instantiation

Check SAT:

$$\forall x.\ \exists y.\ \alpha(x, y)$$

Quantifier Instantiation

Check SAT:

$$\forall x.\ \exists y.\ \alpha(x, y)$$

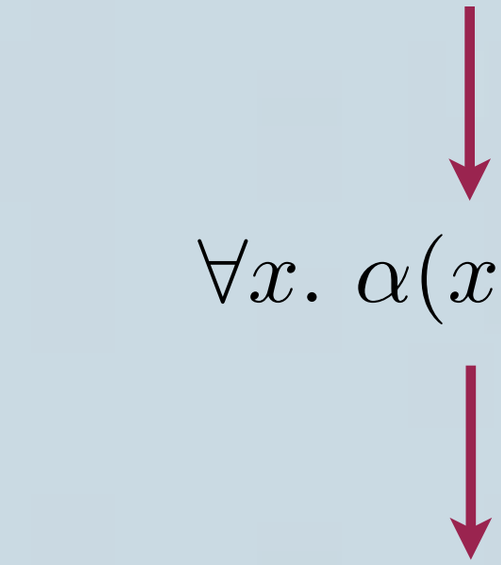
Skolemization

$$\forall x.\ \alpha(x, f(x))$$

Quantifier Instantiation

Check SAT:

$$\forall x.\ \exists y.\ \alpha(x, y)$$



Skolemization

$$\forall x.\ \alpha(x, f(x))$$

Herbrand's Theorem

$$\left\{\alpha(t, f(t)) \mid t \text{ ground term}\right\}$$

Bounded Instantiation

Check SAT:

$$\forall x.\ \exists y.\ \alpha(x, y)$$

Skolemization

$$\forall x.\ \alpha(x, f(x))$$

Herbrand's Theorem

$$\leq \mathbf{k}$$

decidable!

$$\left\{ \alpha(t, f(t)) \;\middle|\; t \text{ ground term} \right\}$$

Bounded Instantiation

Check SAT:

$$\forall x.\ \exists y.\ \alpha(x, y)$$

Skolemization

$$\forall x.\ \alpha(x, f(x))$$

Herbrand's Theorem

$\leq \mathbf{k}$

decidable!

$$\left\{ \alpha(t, f(t)) \;\middle|\; t \text{ ground term} \right\}$$

sound but incomplete

Our Approach: Bounded-Horizon

$$I \wedge TR \wedge \neg I'$$

unsat

how to find
a proof?

**at least as powerful
as instrumentation**

sat

how to find
a cex?

ability of producing
partial cexs

Reduction by Instrumentation

TR $\qquad\qquad$ $\widehat{\mathrm{TR}}$

$\mathrm{Inv} = \forall^*\exists^* \longrightarrow \widehat{\mathrm{Inv}} = \forall^*$

S. Sagiv, T. W. Reps, and R. Wilhelm. Parametric Shape Analysis via 3-Valued Logic (TOPLAS'02)
S. Itzhaky, A. Banerjee, N. Immerman, A. Nanevski, and M. Sagiv. Effectively-Propositional Reasoning About Reachability in Linked Data Structures (CAV'13)
A. Karbyshev, N. Bjørner, S. Itzhaky, N. Rinetzky, and S. Shoham. Property-Directed Inference of Universal Invariants or Proving their Absence (CAV'15)
O. Padon, K. L. McMillan, A. Panda, M. Sagiv, and S. Shoham. Ivy: Safety Verification by Interactive Generalization (PLDI'16)

Reduction by Instrumentation

TR — auxiliary state for derived relation → $\widehat{\mathrm{TR}}$

$\mathrm{Inv} = \forall^*\exists^*$ ⟶ $\widehat{\mathrm{Inv}} = \forall^*$

"$r(x) \equiv \psi(x)$"

S. Sagiv, T. W. Reps, and R. Wilhelm. Parametric Shape Analysis via 3-Valued Logic (TOPLAS'02)

S. Itzhaky, A. Banerjee, N. Immerman, A. Nanevski, and M. Sagiv. Effectively-Propositional Reasoning About Reachability in Linked Data Structures (CAV'13)

A. Karbyshev, N. Bjørner, S. Itzhaky, N. Rinetzky, and S. Shoham. Property-Directed Inference of Universal Invariants or Proving their Absence (CAV'15)

O. Padon, K. L. McMillan, A. Panda, M. Sagiv, and S. Shoham. Ivy: Safety Verification by Interactive Generalization (PLDI'16)

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}

}
```

I = ∀u, p. *resp*(u, p) → ∃**q. *req*(u, q)** ∧ ***match*(q, p)**

```
action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};

    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}

}
```

I = ∀u, p. *resp*(u, p) → **∃q. *req*(u, q) ∧ *match*(q, p)**

```
action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};

    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}

}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → **∃q. *req*(u, q) ∧ *match*(q, p)**

```
action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};

    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}

}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};

    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → **∃q. *req*(u, q) ∧ *match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}

}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};

    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → **∃q. *req*(u, q) ∧ *match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}

}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};

    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → ∃**q. *req*(u, q)** ∧ ***match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}
    r := r ∪ {(u,y) | match(q,y)}
}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};
    r := r ∪ {(x,p) | req(x,q)};
    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → ∃**q. *req*(u, q) ∧ *match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}
    r := r ∪ {(u,y) | match(q,y)}
}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};
    r := r ∪ {(x,p) | req(x,q)};
    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → ∃**q. *req*(u, q) ∧ *match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}
    r := r ∪ {(u,y) | match(q,y)}
}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};
    r := r ∪ {(x,p) | req(x,q)};
    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → ∃**q. *req*(u, q) ∧ *match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

if resp(u, p) ∧ ¬r(u, p)

Instrumentation Example

```
req := ∅; resp := ∅; match := ∅;
action new request(u) {
    q := new request;
    req := req ∪ {(u, q)}
    r := r ∪ {(u,y) | match(q,y)}
}

action respond(u,q) {
    assume req(u , q);
    p := new response;
    match := match ∪ {(q, p)};
    r := r ∪ {(x,p) | req(x,q)};
    resp := resp ∪ {(u, p)}
}

action check(u, p) {
    if resp(u, p) ∧ ¬(∃q. req(u, q) ∧ match(q, p))
        then abort
}
```

r(u, p) "≡" ∃q. *req*(u, q) ∧ *match*(q, p)

I = ∀u, p. *resp*(u, p) → ∃**q. *req*(u, q) ∧ *match*(q, p)**

I = ∀u, p. *resp*(u, p) → **r(u, p)**

Possibly unsound!

if resp(u, p) ∧ ¬r(u, p)

Instrumentation for Verification

Given TR, $I \in \forall^*\exists^*$

Instrumentation for Verification

Given $\mathrm{TR},\ I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}},\ \widehat{I} \in \forall^*$ and ψ, r such that

Instrumentation for Verification

Given $\mathrm{TR},\ I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}},\ \widehat{I} \in \forall^*$ and ψ, r such that

$$I = \widehat{I}[\psi/r]$$

Instrumentation for Verification

Given $\mathrm{TR},\ I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}},\ \widehat{I} \in \forall^*$ and ψ, r such that

$$I = \widehat{I}[\psi/r]$$

$\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

Instrumentation for Verification

Given $\mathrm{TR},\ I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}},\ \widehat{I} \in \forall^*$ and ψ, r such that

$$I = \widehat{I}[\psi/r]$$

$\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

$\mathrm{TR} \rightarrow \widehat{\mathrm{TR}}[\psi/r, \psi'/r']$ is valid (sound instrumentation)

Instrumentation for Verification

Given $\mathrm{TR},\ I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}},\ \widehat{I} \in \forall^*$ and ψ, r such that

$$I = \widehat{I}[\psi/r]$$

$\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

$\mathrm{TR} \rightarrow \widehat{\mathrm{TR}}[\psi/r, \psi'/r']$ is valid (sound instrumentation)

then

I is **inductive** for TR

Implies Bounded-Horizon

Given TR, $I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}}$, $\widehat{I} \in \forall^*$ and ψ, r such that

$I = \widehat{I}[\psi/r]$

$\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

$\mathrm{TR} \rightarrow \widehat{\mathrm{TR}}[\psi/r, \psi'/r']$ is valid (sound instrumentation)

then

I is **inductive** for TR

using Bounded-Horizon with bound 1 or 2

Implies Bounded-Horizon

Given $\mathrm{TR},\ I \in \forall^*\exists^*$

if exists $\widehat{\mathrm{TR}},\ \widehat{I} \in \forall^*$ and ψ, r such that

$I = \widehat{I}[\psi/r]$

$\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

$\mathrm{TR} \to \widehat{\mathrm{TR}}[\psi/r, \psi'/r']$ is valid (sound instrumentation)

$\mathrm{TR}, \widehat{\mathrm{TR}}$ have the same existentials

then

I is **inductive** for TR

using Bounded-Horizon with bound 1 or 2

Bounded-Horizon vs. Instrumentation

Given TR, $I \in \forall^*\exists^*$ **if exists** sound instrumentation without additional existentials

$I = \widehat{I}[\psi/r]$ $\quad$ $\widehat{I} \in \forall^*$ $\quad$ $\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

Then

- Theorem:
  For ψ that is a Boolean combination of ∀* and ∃*,
  for proving that I is inductive for TR it suffices to use **Bounded-Horizon with bound=2**.

Bounded-Horizon vs. Instrumentation

Given TR, $I \in \forall^*\exists^*$ **if exists** sound instrumentation without additional existentials

$I = \widehat{I}[\psi/r]$ $\quad$ $\widehat{I} \in \forall^*$ $\quad$ $\widehat{I}$ is inductive for $\widehat{\mathrm{TR}}$

Then

- Theorem:
  For $\psi \in \exists^*$ and r appears only positively in the invariant
  (or $\psi \in \forall^*$ and r appears only negatively in the invariant),
  Bounded-Horizon with bound=1 suffices.

- Invariants that are combinations of $\forall^*\exists^*$ and $\exists^*\forall^*$ also handled

Our Approach: Bounded-Horizon

$$I \wedge TR \wedge \neg I'$$

unsat

how to find
a proof?

at least as powerful
as instrumentation

sat

how to find
a cex?

**ability of producing
partial cexs**

Infinite Counterexample

$$I \wedge \mathrm{TR} \wedge \neg I'$$

has **infinite** model

but **no finite** models

Infinite Counterexample

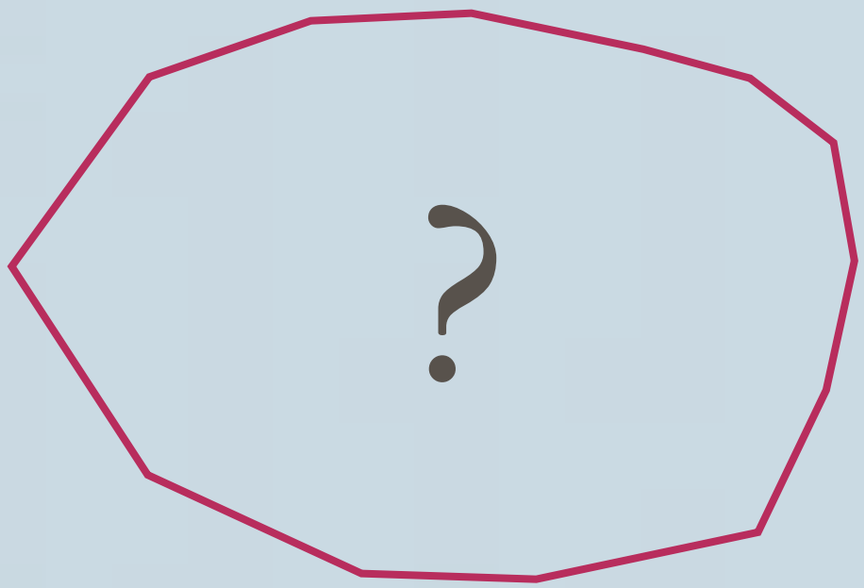
$$I \wedge \mathrm{TR} \wedge \neg I'$$

has **infinite** model

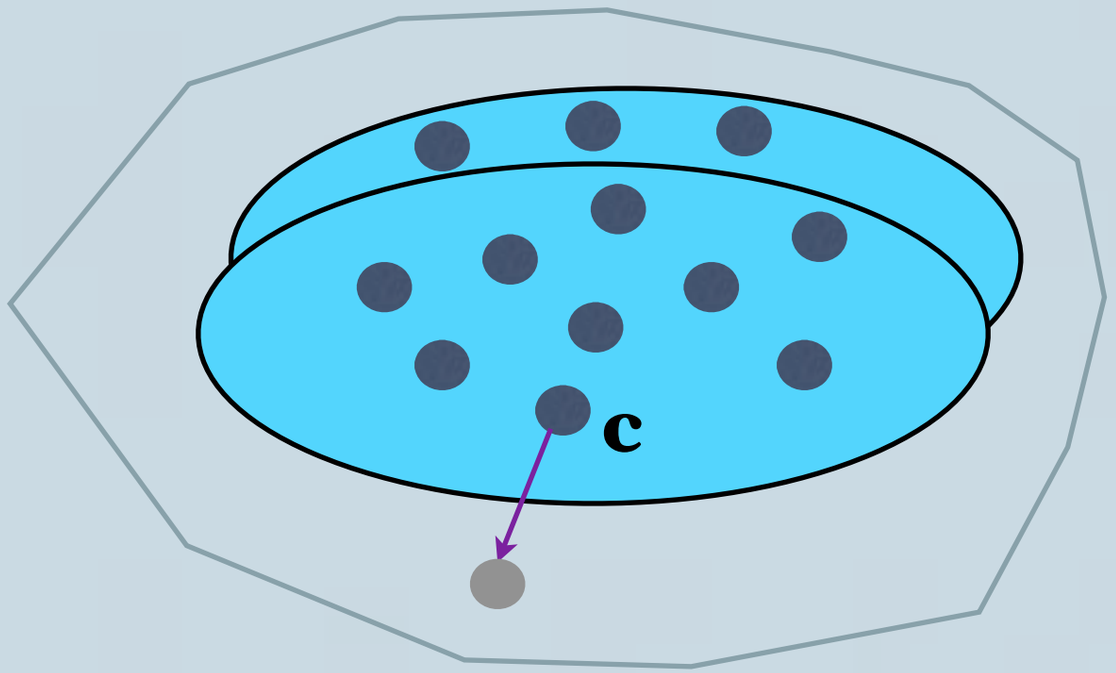
but **no finite** models

usually **no support** for **infinite cexs**

in general first-order logic

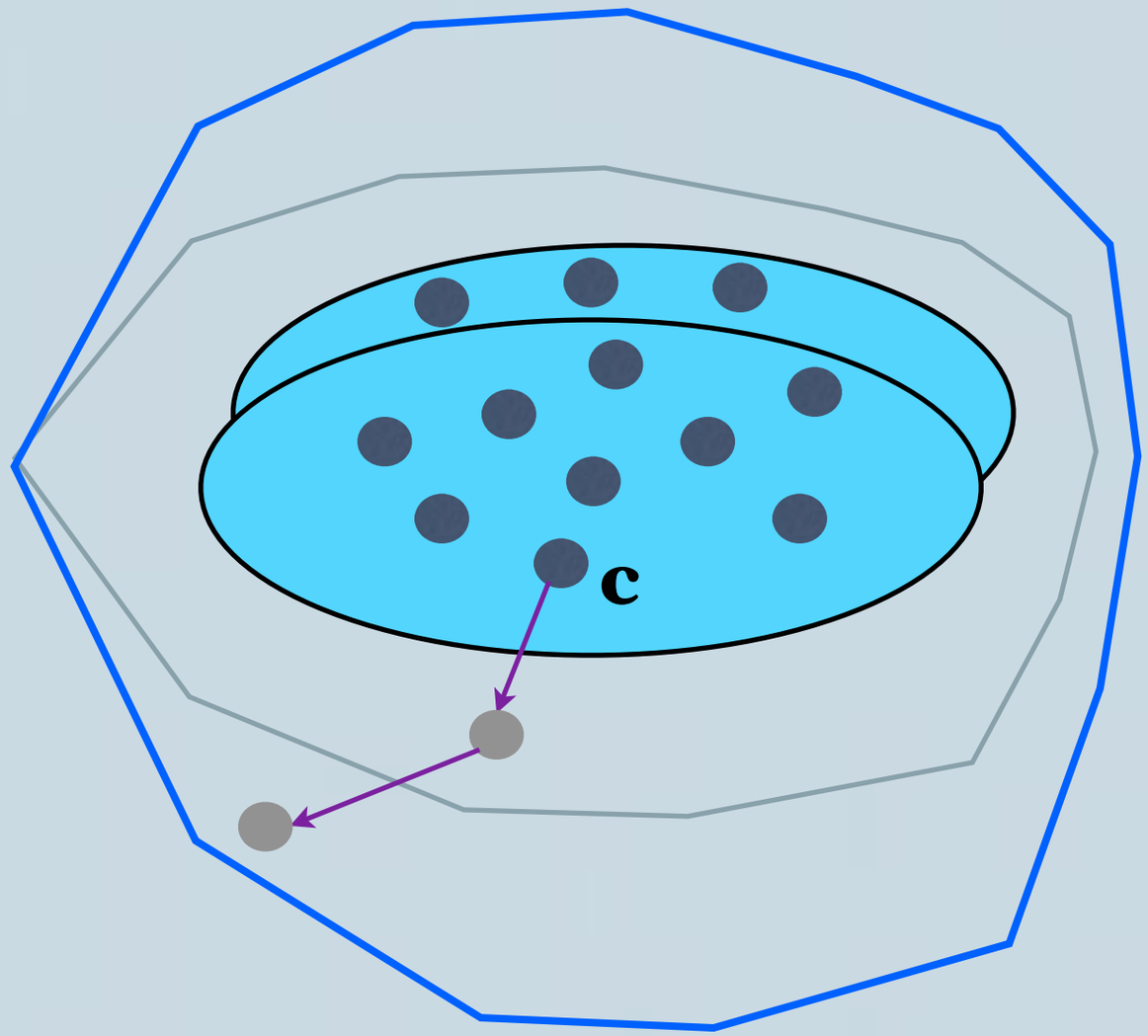


Partial Models



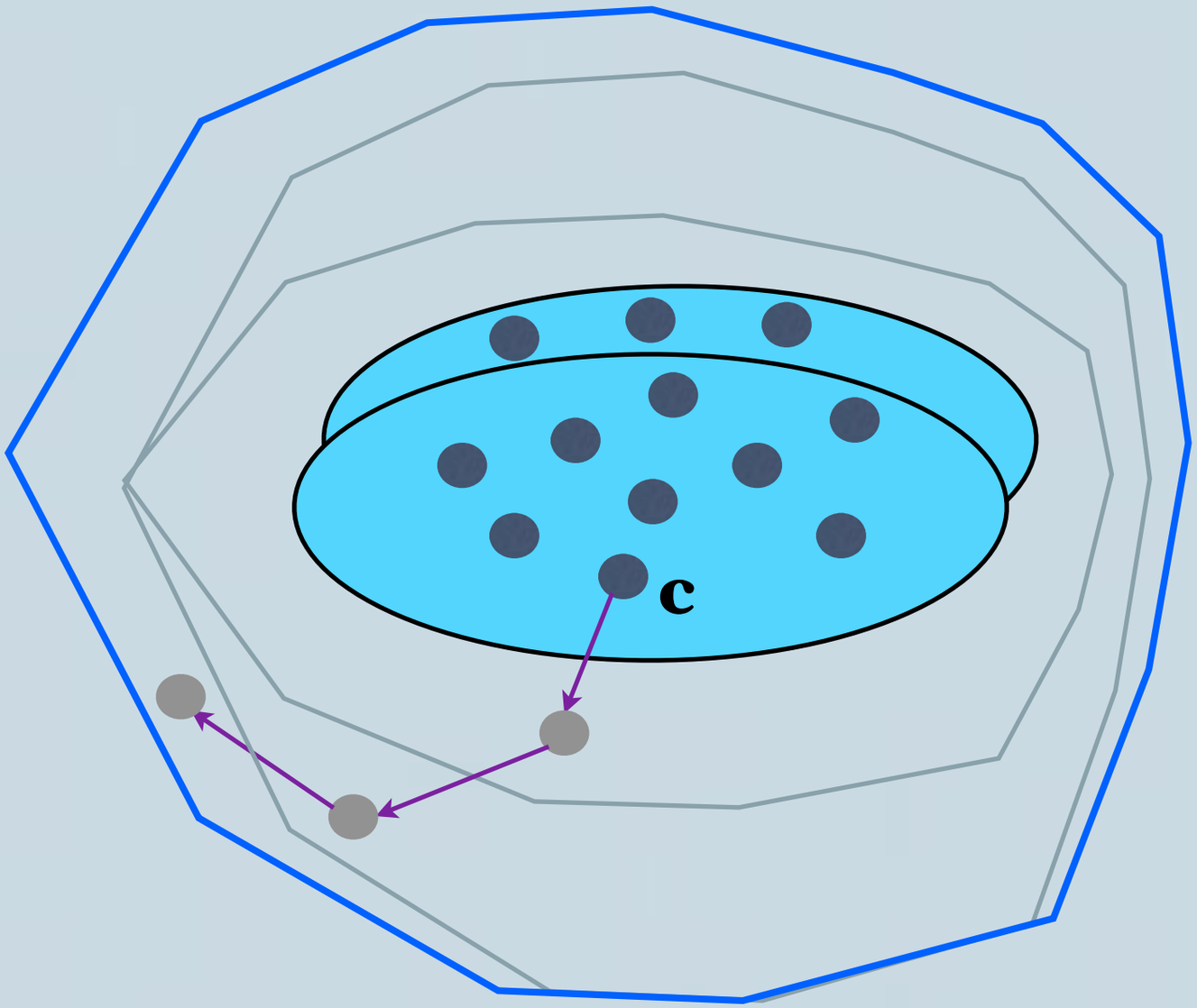
Bound 1

Partial Models



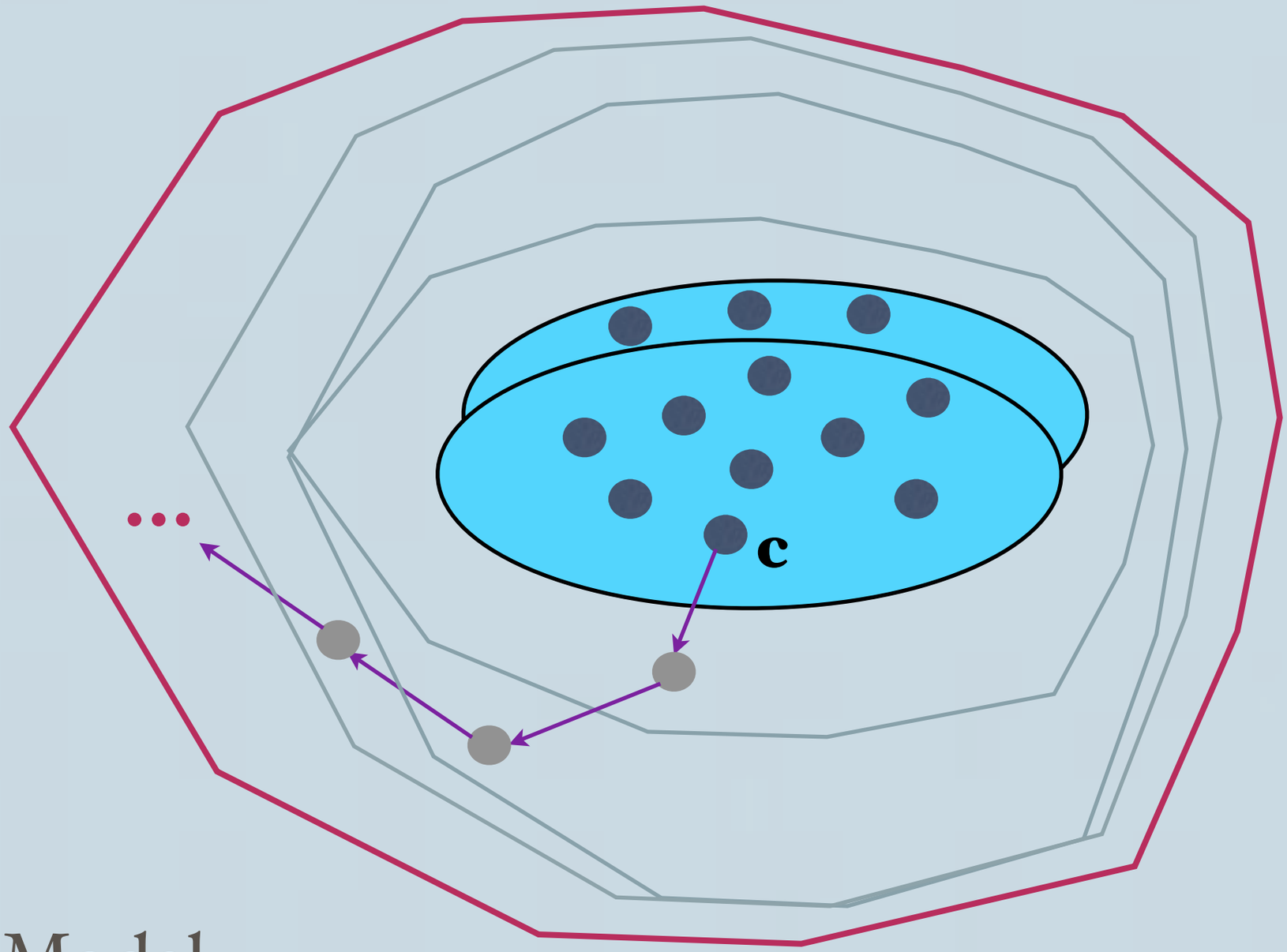
Bound 2

Partial Models



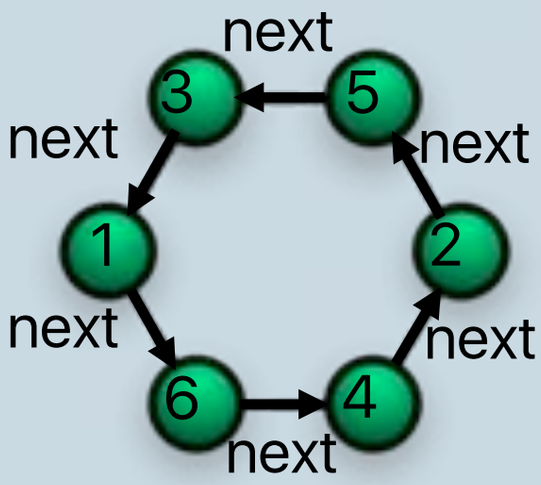
Bound 3

Partial Models



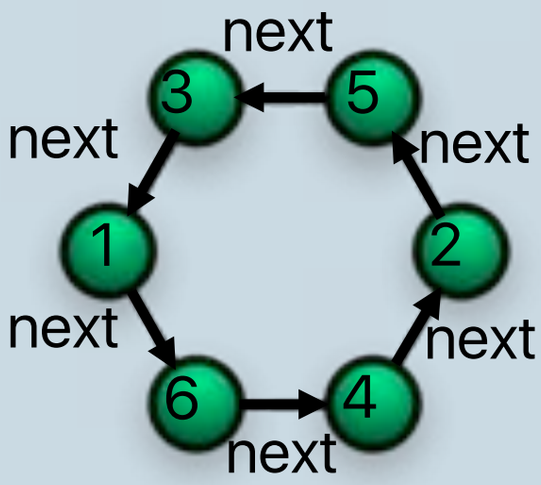
Total Model

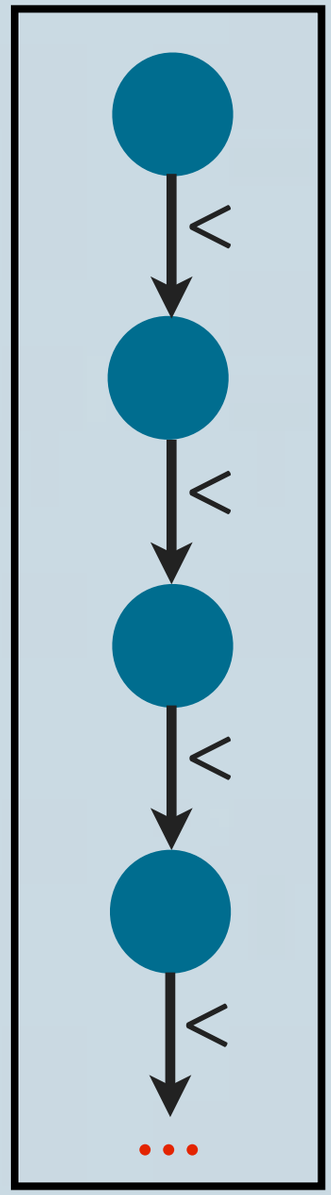
Example: Leader Election in a Ring



E. Chang and R. Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes, CACM’79

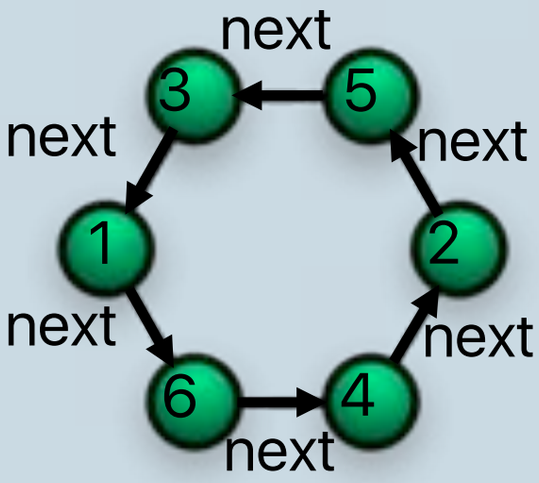
Example: Leader Election in a Ring

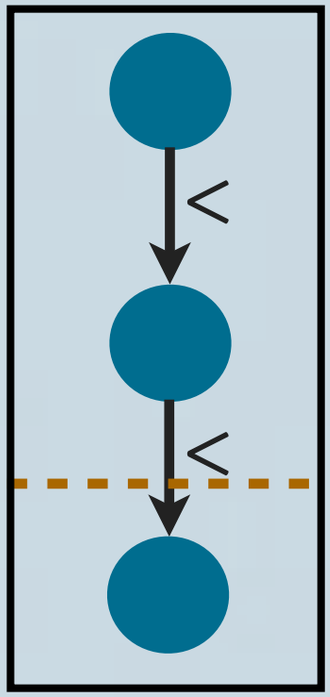


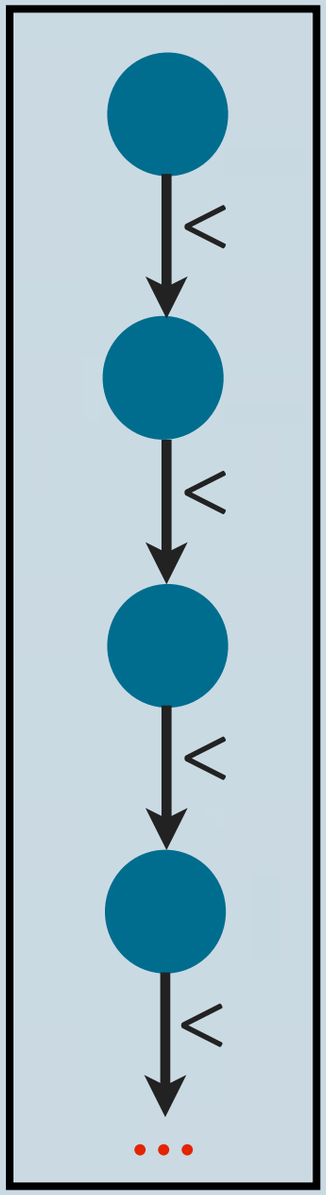


E. Chang and R. Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes, CACM'79

Example: Leader Election in a Ring

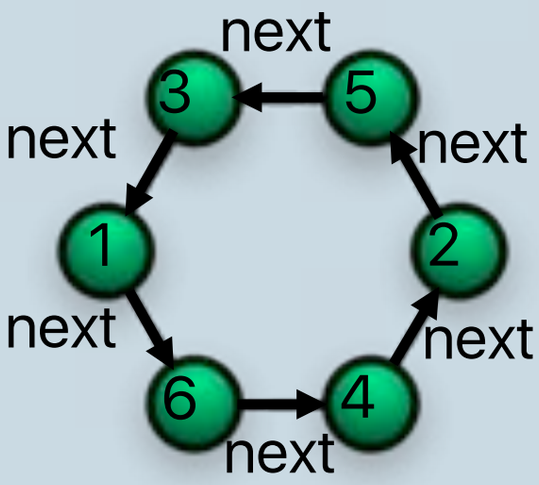


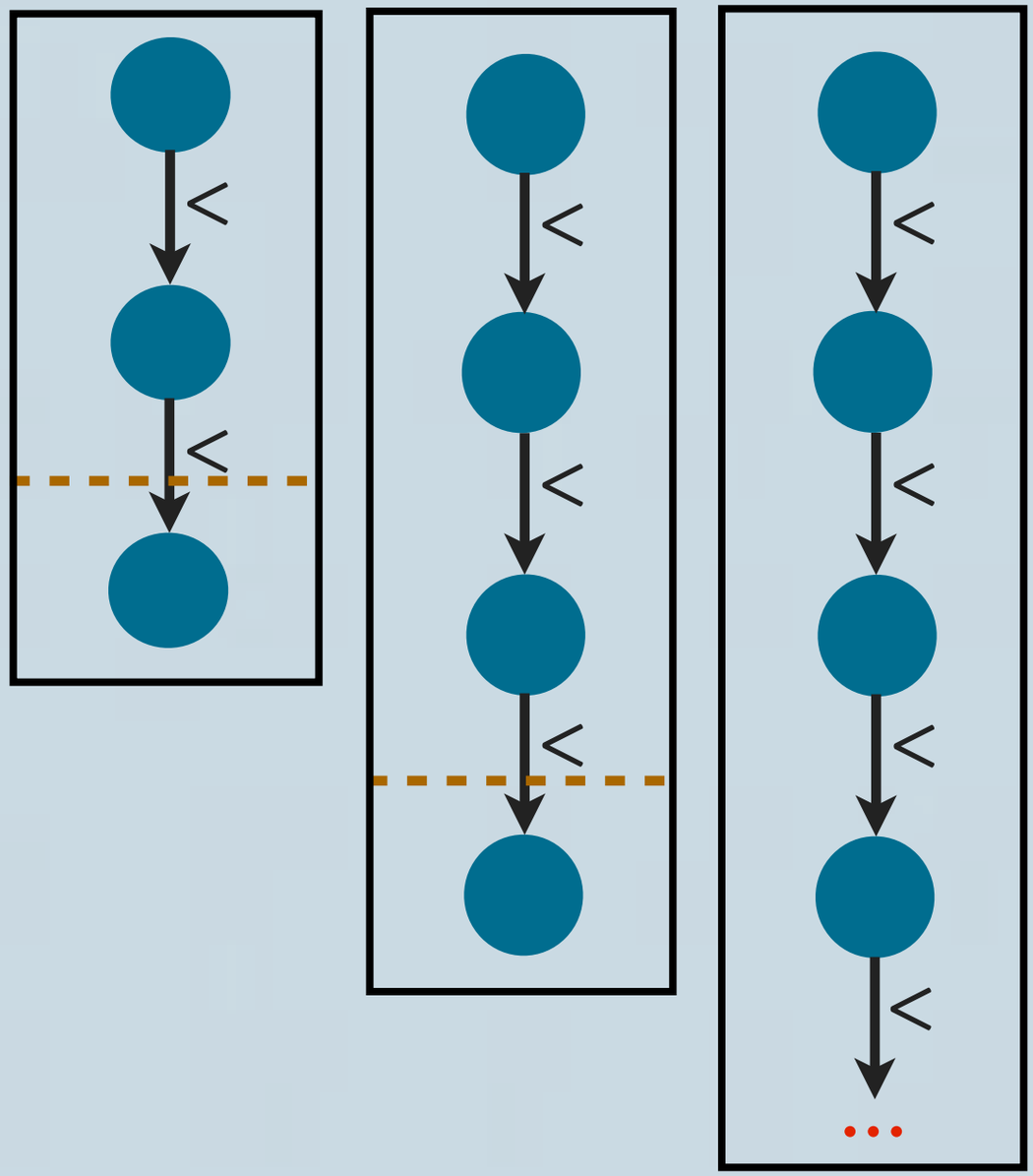




E. Chang and R. Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes, CACM'79

Example: Leader Election in a Ring





E. Chang and R. Roberts. An improved algorithm for decentralized extrema-finding in circular configurations of processes, CACM'79

Implementation of Bound 1

$$\forall x.\ \theta_S$$

$$\downarrow$$

$$\forall x.\left(\bigvee_{c} x = c\right) \rightarrow\ \theta_S$$

$$\downarrow$$

Z3: sat?

Evaluation

Program	#∀	#f-sk	#Consts	B1 Total	B1 Solve	Baseline Z3
Client-server	14	1	15	58 ms	3 ms	3 ms
List reverse	47	3	15	319 ms	211 ms	50 ms
Learning switch	70	1	7	245 ms	65 ms	33 ms
Hole-punching firewall with ghost	15	1	18	75 ms	4 ms	4 ms
Hole-punching firewall ∃ time	32	2	21	102 ms	4 ms	4 ms
Leader-election in a ring (correct)	41	1	21	113 ms	36 ms	27 ms
Leader-election in a ring (incorrect)	40	1	20	1112 ms	1008 ms	—

Summary

Checking $\forall^*\exists^*$ Inductive Invariants

Proof

sound but incomplete

powerful as certain **manual reductions** to a decidable class

Cex

sound but incomplete

ability of producing **partial cexs** to facilitate **progress**