Accelerating Protocol Synthesis and Detecting Unrealizability with Interpretation Reduction

Derek Egolf and Stavros Tripakis

Northeastern University, Boston

We synthesize TLA+ protocols

- TLA^+ (Temporal Logic of Actions) [Lamport]
    - Properties: linear, with theories
    - Model checker: TLC; Proof system: TLAPS
- Symbolic representation (not explicit state)
- Used in both academia and industry [Commun. ACM'15]
- Allows "parameterized" protocols (e.g. set of nodes)
- Prior work [FMCAD'24] only other paper on TLA^+ synthesis

Sketching
[Solar-Lezama]

Example property: not shown

Example sketch

$$\begin{aligned} &\text{Send}(\text{src}, \text{dst}) := \\ &\quad \land\ ???_1 \\ &\quad \land\ \text{message}' = ???_2 \\ &\quad \land\ \text{has\_lock}' = ???_3 \end{aligned}$$

$$\begin{aligned} &\text{Receive}(\text{src}, \text{dst}) := \\ &\quad \land\ ???_4 \\ &\quad \land\ \text{message}' = ???_5 \\ &\quad \land\ \text{has\_lock}' = ???_6 \end{aligned}$$

Example completion

$$\begin{aligned} &\text{Send}(\text{src}, \text{dst}) := \\ &\quad \land\ \text{has\_lock}[\text{src}] \\ &\quad \land\ \text{message}' = \text{message} \cup \{(\text{src}, \text{dst})\} \\ &\quad \land\ \text{has\_lock}' = \text{has\_lock}[\text{src} \leftarrow \text{false}] \end{aligned}$$

$$\begin{aligned} &\text{Receive}(\text{src}, \text{dst}) := \\ &\quad \land\ (\text{src}, \text{dst}) \in \text{message} \\ &\quad \land\ \text{message}' = \text{message} \setminus \{(\text{src}, \text{dst})\} \\ &\quad \land\ \text{has\_lock}' = \text{has\_lock}[\text{dst} \leftarrow \text{true}] \end{aligned}$$

Sketching, SyGuS
[Solar-Lezama], [Alur et. al.]

Example property: not shown

Example sketch

$$\begin{aligned}\mathrm{Send}(\mathrm{src},\mathrm{dst}) :=\ & \\ \wedge\ & ???_1 \\ \wedge\ & \mathrm{message}' = ???_2 \\ \wedge\ & \mathrm{has\_lock}' = ???_3\end{aligned}$$

$$\begin{aligned}\mathrm{Receive}(\mathrm{src},\mathrm{dst}) :=\ & \\ \wedge\ & ???_4 \\ \wedge\ & \mathrm{message}' = ???_5 \\ \wedge\ & \mathrm{has\_lock}' = ???_6\end{aligned}$$

Example completion

$$\begin{aligned}\mathrm{Send}(\mathrm{src},\mathrm{dst}) :=\ & \\ \wedge\ & \mathrm{has\_lock}[\mathrm{src}] \\ \wedge\ & \mathrm{message}' = \mathrm{message} \cup \{(\mathrm{src},\mathrm{dst})\} \\ \wedge\ & \mathrm{has\_lock}' = \mathrm{has\_lock}[\mathrm{src} \leftarrow \mathrm{false}]\end{aligned}$$

$$\begin{aligned}\mathrm{Receive}(\mathrm{src},\mathrm{dst}) :=\ & \\ \wedge\ & (\mathrm{src},\mathrm{dst}) \in \mathrm{message} \\ \wedge\ & \mathrm{message}' = \mathrm{message} \setminus \{(\mathrm{src},\mathrm{dst})\} \\ \wedge\ & \mathrm{has\_lock}' = \mathrm{has\_lock}[\mathrm{dst} \leftarrow \mathrm{true}]\end{aligned}$$

Example grammar: not shown

Key Contributions

Key Contributions

- Improve state of art in TLA^+ synthesis (100x)

Key Contributions

- Improve state of art in TLA$^+$ synthesis (100x)
- Halt when no solution: *unrealizability*

Key Contributions

- Improve state of art in TLA$^+$ synthesis (100x)
- Halt when no solution: *unrealizability*
- New search space reduction technique: *Interpretation Reduction*

Key Contributions

- Improve state of art in TLA$^+$ synthesis (100x)
- Halt when no solution: *unrealizability*
- New search space reduction technique: *Interpretation Reduction*
- Improved counterexample generalization for pruning

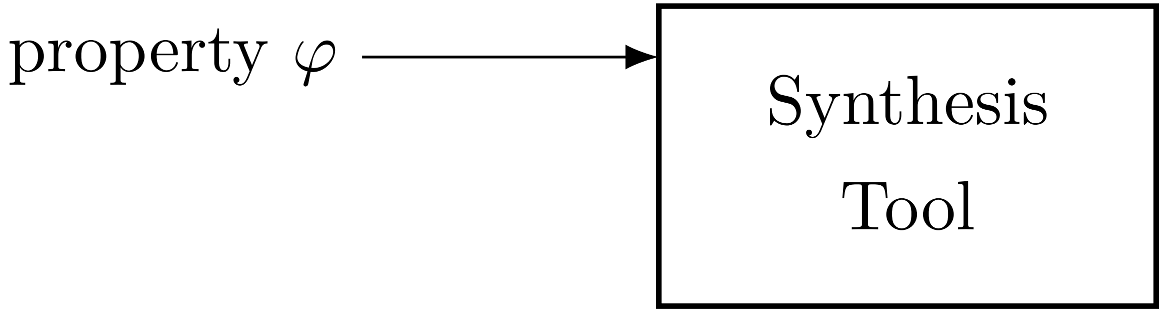
Problem Statement

Problem

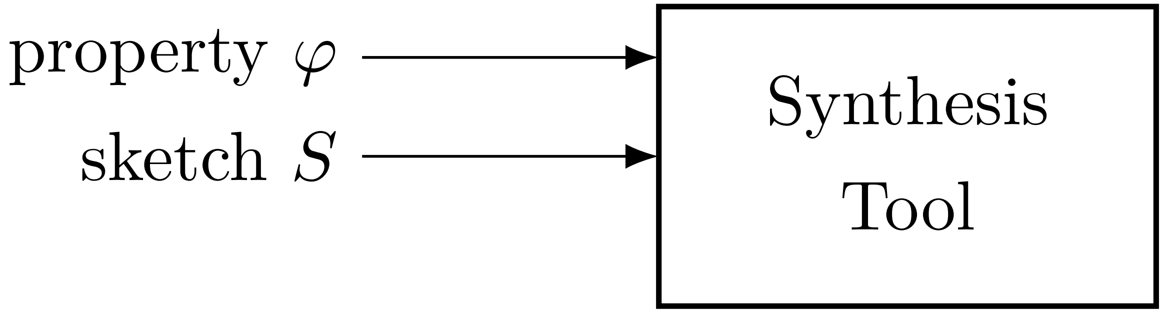
Problem

Synthesis
Tool

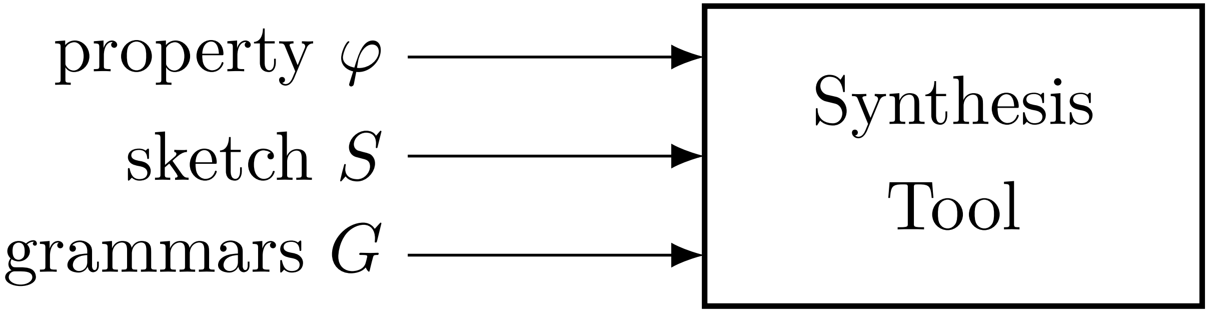
Problem



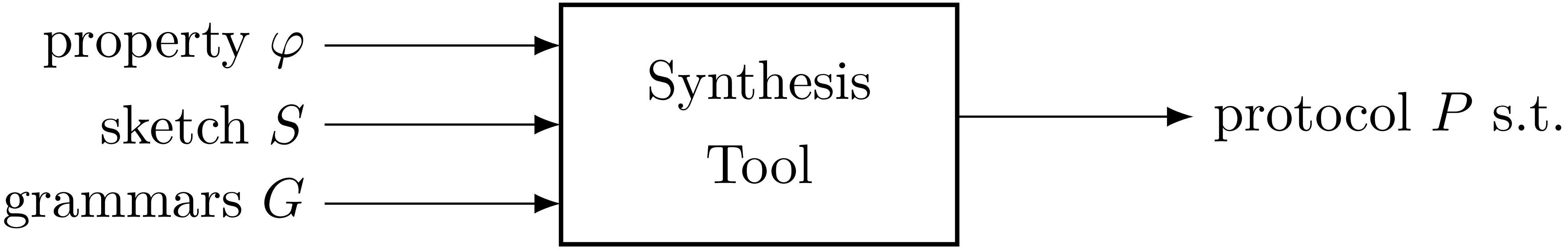
Problem



Problem

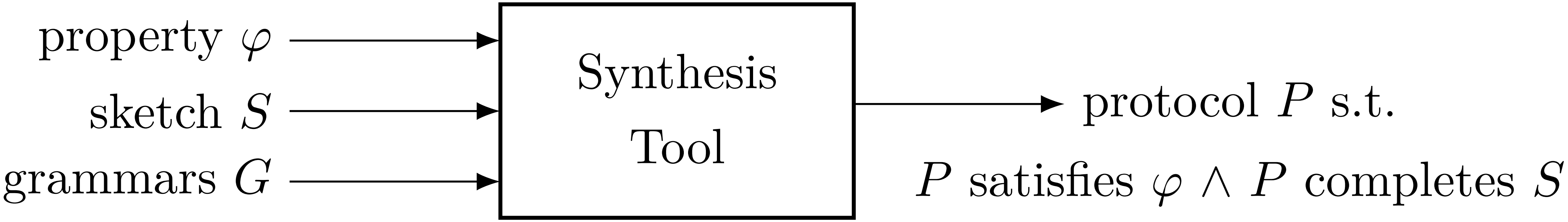


Problem

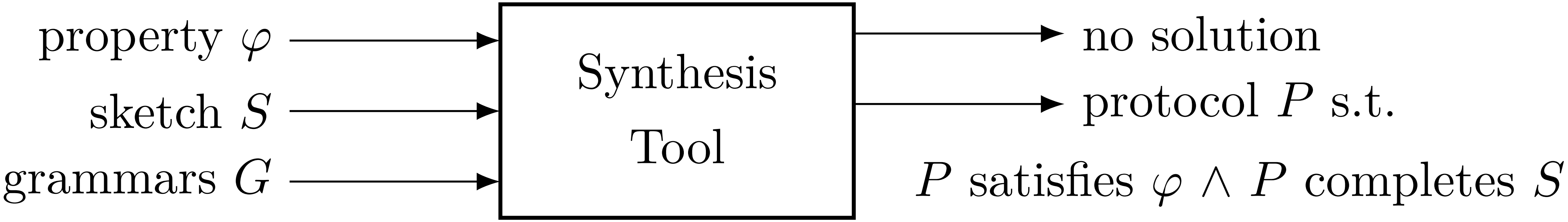


property φ
sketch S
grammars G
Synthesis Tool
protocol P s.t.

Problem


property φ
sketch S
grammars G
Synthesis Tool
protocol P s.t.
P satisfies $\varphi \land P$ completes S

Problem


property φ
sketch S
grammars G
Synthesis Tool
no solution
protocol P s.t.
P satisfies $\varphi \wedge P$ completes S

Example Input

Example Input

$$\text{Safe} := \forall x, y \in \text{Node} : (\text{has\_lock}[x] \land \text{has\_lock}[y]) \Rightarrow (x = y)$$

Example Input

$$\text{Safe} := \forall x, y \in \text{Node} : (\text{has\_lock}[x] \wedge \text{has\_lock}[y]) \Rightarrow (x = y)$$

$$\text{Live} := \ \ldots$$

Example Input

$$\text{Safe} := \forall x, y \in \text{Node} : (\text{has\_lock}[x] \land \text{has\_lock}[y]) \Rightarrow (x = y)$$

$$\text{Live} := \ \ldots$$

$$\begin{aligned} &\text{Send}(\text{src}, \text{dst}) := \\ &\quad \land \ ???_1 \\ &\quad \land \ \text{message}' = \ ???_2 \\ &\quad \land \ \text{has\_lock}' = \ ???_3 \end{aligned}$$

Example Input

$$\text{Safe} := \forall x, y \in \text{Node} : (\text{has\_lock}[x] \land \text{has\_lock}[y]) \Rightarrow (x = y)$$

$$\text{Live} := \ \ldots$$

$$\text{Send}(\text{src}, \text{dst}) :=$$

$$\land \ ???_1$$

$$\land \ \text{message}' = \ ???_2$$

$$\land \ \text{has\_lock}' = \ ???_3$$

$$\text{Receive}(\text{src}, \text{dst}) := \ \ldots$$

Example Input

$$\text{Safe} := \forall x, y \in \text{Node} : (\text{has\_lock}[x] \land \text{has\_lock}[y]) \Rightarrow (x = y)$$

$$\text{Live} := \ \ldots$$

$$\begin{aligned}
&\text{Send}(\text{src}, \text{dst}) := \\
&\quad \land\ ???_1 \\
&\quad \land\ \text{message}' =\ ???_2 \\
&\quad \land\ \text{has_lock}' =\ ???_3
\end{aligned}$$

$$\text{Receive}(\text{src}, \text{dst}) := \ \ldots$$

$$\begin{aligned}
&???_2 \rightarrow S_2 \\
&S_2 \rightarrow T \mid S_2 \setminus S_2 \mid S_2 \cup S_2 \\
&T \rightarrow \text{message} \mid \{(N, N)\} \\
&N \rightarrow \text{src} \mid \text{dst}
\end{aligned}$$

Example Input

$$\text{Safe} := \forall x, y \in \text{Node} : (\text{has\_lock}[x] \wedge \text{has\_lock}[y]) \Rightarrow (x = y)$$

$$\text{Live} := \ \ldots$$

$$\begin{aligned}
&\text{Send}(\text{src}, \text{dst}) := \\
&\quad \wedge \ ???_1 \\
&\quad \wedge \ \text{message}' = \ ???_2 \\
&\quad \wedge \ \text{has_lock}' = \ ???_3
\end{aligned}$$

$$\text{Receive}(\text{src}, \text{dst}) := \ \ldots$$

$$\begin{aligned}
???_2 &\to S_2 \\
S_2 &\to T \mid S_2 \setminus S_2 \mid S_2 \cup S_2 \\
T &\to \text{message} \mid \{(N, N)\} \\
N &\to \text{src} \mid \text{dst}
\end{aligned}$$

$$\begin{aligned}
???_1 &\to \ \ldots \\
???_3 &\to \ \ldots
\end{aligned}$$

Solution

CEGIS

Counterexample-Guided Inductive Synthesis [Solar-Lezama]

CEGIS

Counterexample-Guided Inductive Synthesis [Solar-Lezama]

Learner

CEGIS

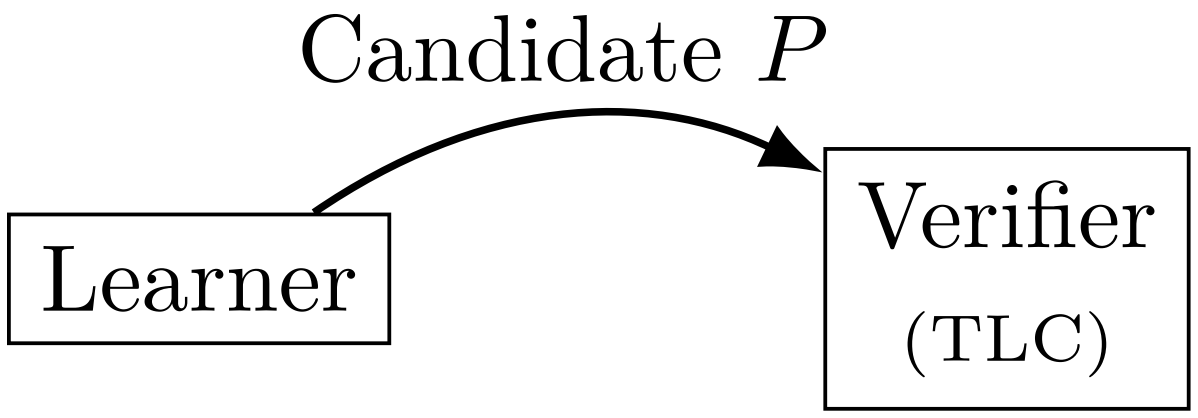
Counterexample-Guided Inductive Synthesis [Solar-Lezama]

Learner

Verifier
(TLC)

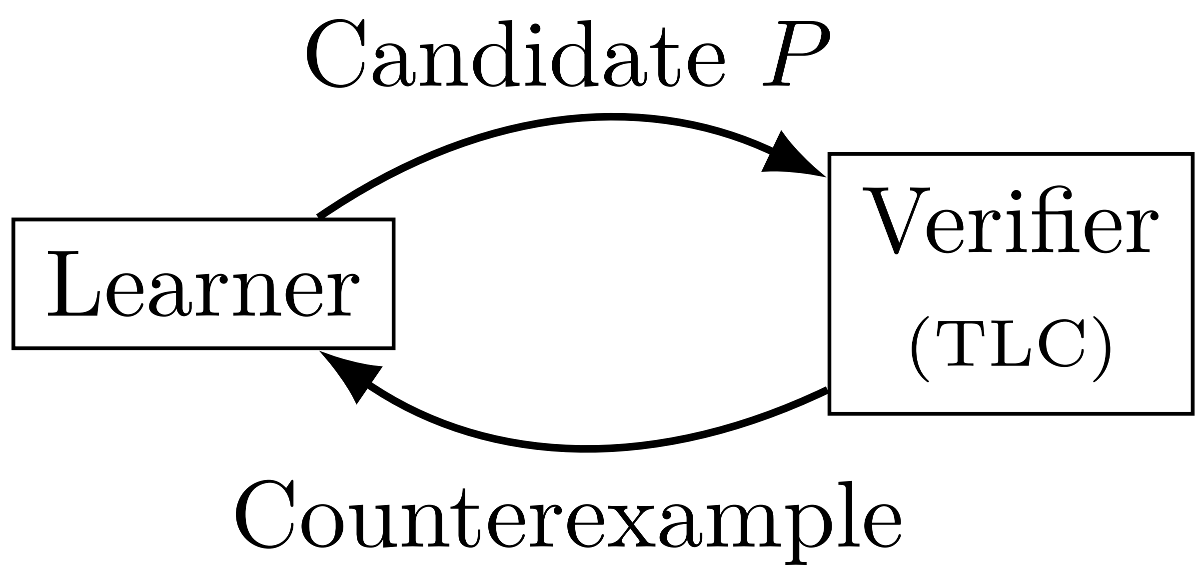
CEGIS

Counterexample-Guided Inductive Synthesis [Solar-Lezama]



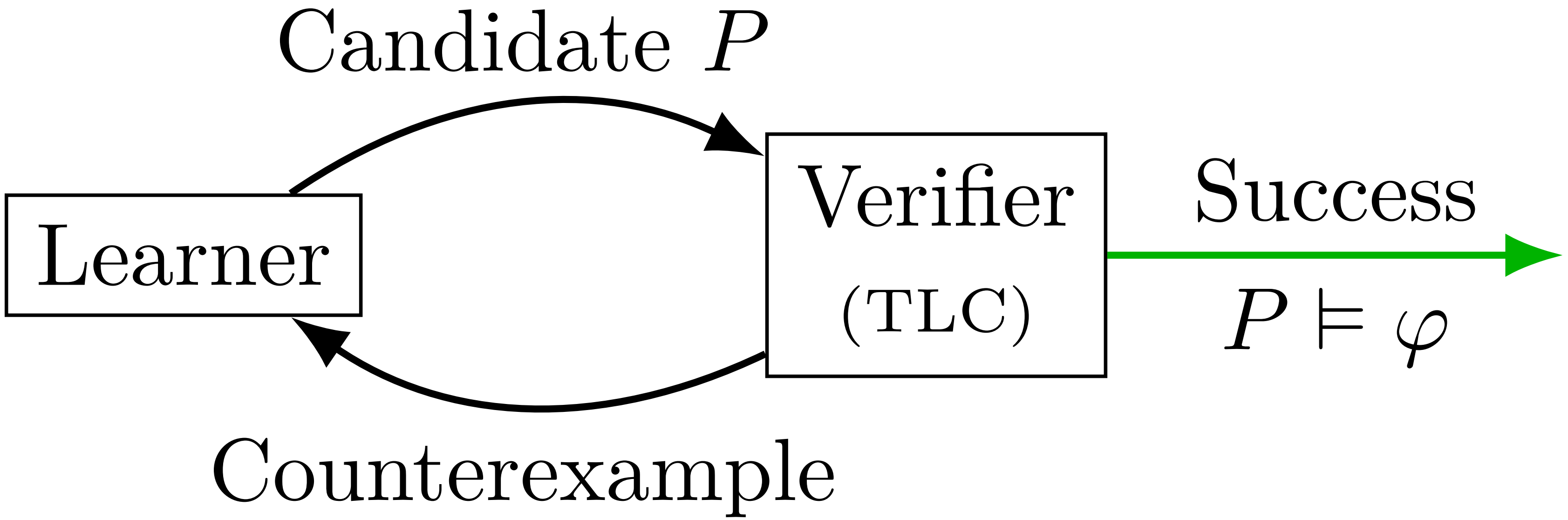
CEGIS

Counterexample-Guided Inductive Synthesis [Solar-Lezama]



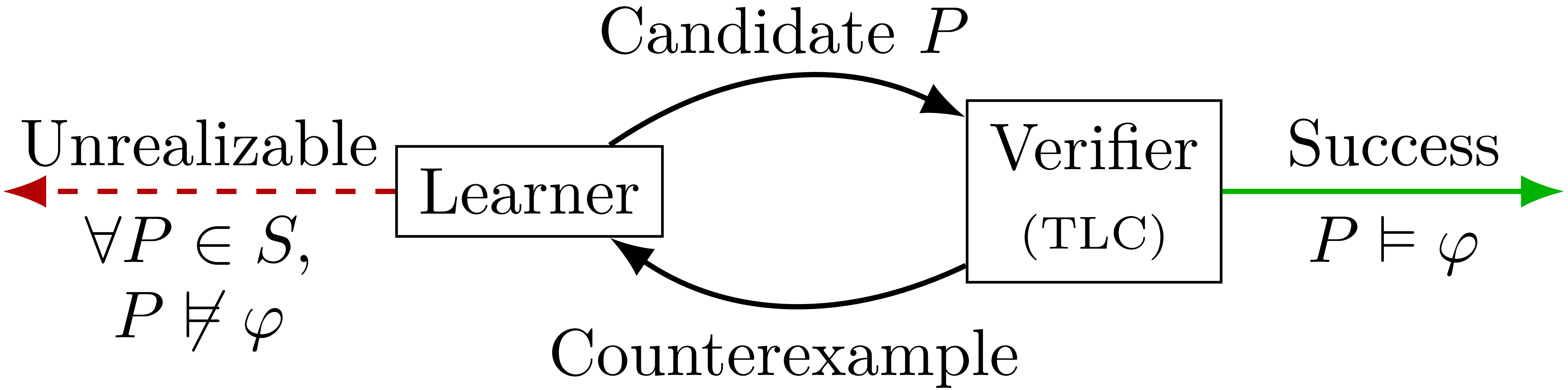
CEGIS

Counterexample-Guided Inductive Synthesis [Solar-Lezama]



CEGIS

Counterexample-Guided Inductive Synthesis [Solar-Lezama]



Improving the Learner

- Naive Learner: ignore cex, enumerate all protocols
    - *many protocols; model checking expensive*
- Idea 1; **discard protocols before model checking**
    - Pruning constraints: generalize counterexamples
- Idea 2; **avoids enumerating protocols in the first place**
    - Equivalence reduction: do not use equiv. subexpressions
- Idea 2++; Interpretation reduction: **use weak eq. relation**

Generalizing Counterexamples

Generalizing Counterexamples w/ Pruning Constraints

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \wedge x' = ???_2)$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \wedge x' = ???_2)$

Bad Completion: $(A_1 := \texttt{true} \wedge x' = b)$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \land x' = ???_2)$

Bad Completion: $(A_1 := \text{true} \land x' = b)$

Bad Completion: $(A_1 := x = a \land x' = b)$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \land x' = ???_2)$

Bad Completion: $(A_1 := \texttt{true} \land x' = b)$

Bad Completion: $(A_1 := x = a \land x' = b)$

Good Completion: $(A_1 := x = a \land x' = a)$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \land x' = ???_2)$

Bad Completion: $(A_1 := \mathsf{true} \land x' = b)$

Bad Completion: $(A_1 := x = a \land x' = b)$

Good Completion: $(A_1 := x = a \land x' = a)$

Good Completion: $(A_1 := x \neq a \land x' = b)$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \land x' = ???_2)$

Bad Completion: $(A_1 := \texttt{true} \land x' = b)$

Bad Completion: $(A_1 := x = a \land x' = b)$

Good Completion: $(A_1 := x = a \land x' = a)$

Good Completion: $(A_1 := x \neq a \land x' = b)$

Idea: treat cex as a path. Disable an action or point it somewhere else.

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \land x' = ???_2)$

Bad Completion: $(A_1 := \text{true} \land x' = b)$

Bad Completion: $(A_1 := x = a \land x' = b)$

Good Completion: $(A_1 := x = a \land x' = a)$

Good Completion: $(A_1 := x \neq a \land x' = b)$

Idea: treat cex as a path. Disable an action or point it somewhere else.

$\pi_{\text{cex}} :=$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \land x' = ???_2)$

Bad Completion: $(A_1 := \text{true} \land x' = b)$

Bad Completion: $(A_1 := x = a \land x' = b)$

Good Completion: $(A_1 := x = a \land x' = a)$

Good Completion: $(A_1 := x \neq a \land x' = b)$

Idea: treat cex as a path. Disable an action or point it somewhere else.

$\pi_{\text{cex}} := ???_1([x \mapsto a]) \neq \text{true}$

Generalizing Counterexamples w/ Pruning Constraints

cex: $[x \mapsto a] \xrightarrow{A_1} [x \mapsto b]$ (safety violation to, e.g., always $x \neq b$)

Sketch: $(A_1 := ???_1 \wedge x' = ???_2)$

Bad Completion: $(A_1 := \text{true} \wedge x' = b)$

Bad Completion: $(A_1 := x = a \wedge x' = b)$

Good Completion: $(A_1 := x = a \wedge x' = a)$

Good Completion: $(A_1 := x \neq a \wedge x' = b)$

Idea: treat cex as a path. Disable an action or point it somewhere else.

$\pi_{\text{cex}} := ???_1([x \mapsto a]) \neq \text{true} \vee ???_2([x \mapsto a]) \neq b$

Generalizing Counterexamples (Cont.)

- In general: many completions violate π_{cex}
- (Ideally) **Theorem**: cex belongs to P iff $P \not\models \pi_{\text{cex}}$
- (Reality) π_{cex} underprunes when cex is stuttering violation
- π_{cex} exact for safety, deadlock, and liveness cex
- Prior work [FMCAD'24] only exact for safety
- Checking $P \models \pi_{\text{cex}}$ cheaper than model checking.

Interpretation Reduction

Standard Equivalence Reduction

Example grammar

$???_1 \to Y$

$Y \to u \mid X + Y$

$X \to 0 \mid v \mid X + X$

Standard Equivalence Reduction

Example grammar

$$???_1 \to Y$$

$$Y \to u \mid X + Y$$

$$X \to 0 \mid v \mid X + X$$

$$u : Y$$

Standard Equivalence Reduction

Example grammar

$???_1 \to Y$

$Y \to u \mid X + Y$

$X \to 0 \mid v \mid X + X$

$u : Y$

$v : X$

Standard Equivalence Reduction

Example grammar

$$???_1 \rightarrow Y$$

$$Y \rightarrow u \mid X + Y$$

$$X \rightarrow 0 \mid v \mid X + X$$

$$u : Y$$

$$v : X$$

$$v + u : Y$$

Standard Equivalence Reduction

Example grammar

$??? _1 \to Y$

$Y \to u \mid X + Y$

$X \to 0 \mid v \mid X + X$

$u : Y$

$v : X$

$v + u : Y$

$0 : X$

Standard Equivalence Reduction

Example grammar

$???_1 \to Y$

$Y \to u \mid X + Y$

$X \to 0 \mid v \mid X + X$

$u : Y$

$v : X$

$v + u : Y$

$0 : X$

$0 + u : Y$

Standard Equivalence Reduction

Example grammar

$$???_1 \to Y$$

$$Y \to u \mid X + Y$$

$$X \to 0 \mid v \mid X + X$$

$u : Y$

$v : X$

$v + u : Y$

$0 : X$

$0 + u : Y$

Standard Equivalence Reduction

Example grammar

$???_1 \to Y$

$Y \to u \mid X + Y$

$X \to 0 \mid v \mid X + X$

Discard $0 + u$.

Do not enumerate, e.g.,

$0 + (0 + u)$.

$u : Y$

$v : X$

$v + u : Y$

$0 : X$

$0 + u : Y$

Standard Equivalence Reduction

Example grammar

$??? _1 \to Y$

$Y \to u \mid X + Y$

$X \to 0 \mid v \mid X + X$

Discard $0 + u$.

Do not enumerate, e.g., $0 + (0 + u)$.

$u : Y$

$v : X$

$v + u : Y$

$0 : X$

$0 + u : Y$

[FMCAD'24] uses standard equivalence reduction

Absolute Equivalence

Absolute Equivalence

E.g., $x + y \equiv y + x$

Absolute Equivalence

E.g., $x + y \equiv y + x$

$e_1 \equiv e_2$ means: for all α, $e_1(\alpha) = e_2(\alpha)$

Absolute Equivalence

E.g., $x + y \equiv y + x$

$e_1 \equiv e_2$ means: for all α, $e_1(\alpha) = e_2(\alpha)$

Interpretation Equivalence

Absolute Equivalence

E.g., $x + y \equiv y + x$

$e_1 \equiv e_2$ means: for all α, $e_1(\alpha) = e_2(\alpha)$

Interpretation Equivalence

E.g., $x + y \equiv_{\mathcal{A}} x + x$ for $\mathcal{A} = \{[x \mapsto 0, y \mapsto 0], [x \mapsto 42, y \mapsto 42]\}$

Absolute Equivalence

E.g., $x + y \equiv y + x$

$e_1 \equiv e_2$ means: for all α, $e_1(\alpha) = e_2(\alpha)$

Interpretation Equivalence

E.g., $x + y \equiv_{\mathcal{A}} x + x$ for $\mathcal{A} = \{[x \mapsto 0, y \mapsto 0], [x \mapsto 42, y \mapsto 42]\}$

$e_1 \equiv_{\mathcal{A}} e_2$ means: for all $\alpha \in \mathcal{A}$, $e_1(\alpha) = e_2(\alpha)$

Relevant Interpretations

Relevant Interpretations

- Q: Which interpretations are in $\mathcal{A}$?

Relevant Interpretations

- Q: Which interpretations are in $\mathcal{A}$?
- A: All interpretations (states) appearing in pruning constraints.

Relevant Interpretations

- Q: Which interpretations are in $\mathcal{A}$?
- A: All interpretations (states) appearing in pruning constraints.

For instance:

$$\Pi = \{\ ???_1([x \mapsto 0, y \mapsto 1]) \neq 1\ \}$$

$$\mathcal{A} = \{\ [x \mapsto 0, y \mapsto 1]\ \}$$

Relevant Interpretations

- Q: Which interpretations are in $\mathcal{A}$?
- A: All interpretations (states) appearing in pruning constraints.

For instance:

$$\Pi = \{\ ???_1([x \mapsto 0, y \mapsto 1]) \neq 1\ \}$$

$$\mathcal{A} = \{\ [x \mapsto 0, y \mapsto 1]\ \}$$

Note: $y \not\models \Pi$, since y evaluates to 1.

Relevant Interpretations

- Q: Which interpretations are in $\mathcal{A}$?
- A: All interpretations (states) appearing in pruning constraints.

For instance:

$$\Pi = \{\ ???_1([x \mapsto 0, y \mapsto 1]) \neq 1\ \}$$
$$\mathcal{A} = \{\ [x \mapsto 0, y \mapsto 1]\ \}$$

Note: $y \not\models \Pi$, since y evaluates to 1.

Also note: $x + y \not\models \Pi$, since $0 + 1 = 1$.

Relevant Interpretations

- Q: Which interpretations are in $\mathcal{A}$?
- A: All interpretations (states) appearing in pruning constraints.

For instance:

$$\Pi = \{\ ???_1([x \mapsto 0, y \mapsto 1]) \neq 1\ \}$$

$$\mathcal{A} = \{\ [x \mapsto 0, y \mapsto 1]\ \}$$

Note: $y \not\models \Pi$, since y evaluates to 1.

Also note: $x + y \not\models \Pi$, since $0 + 1 = 1$.

Finally, note $y \equiv_{\mathcal{A}} x + y$.

Interpretation Reduction

- Let $\mathcal{A} = \{\ [x \mapsto 0, y \mapsto 1]\ \}$ have all relevant interpretations
- Suppose we've enumerated the subexpressions x and y
- Should we enumerate subexpression $x + y$?
- No, $y \equiv_{\mathcal{A}} x + y$
- **Theorem.** one representative per eq. class mod $\mathcal{A}$ is sufficient to maintain completeness of the synthesis algorithm
- **Lemma.** $e_1 \equiv_{\mathcal{A}} e_2 \implies (e_1 \models \Pi \Leftrightarrow e_2 \models \Pi)$

Summary: Improving the Learner

- Idea 1; **discard protocols before model checking**
  - Pruning constraints: generalize counterexamples
- Idea 2; **avoids enumerating protocols in the first place**
  - Equivalence reduction: do not use equiv. subexpressions
- Idea 2++; Interpretation reduction: **use weak eq. relation**
- More details in paper!

Unrealizability

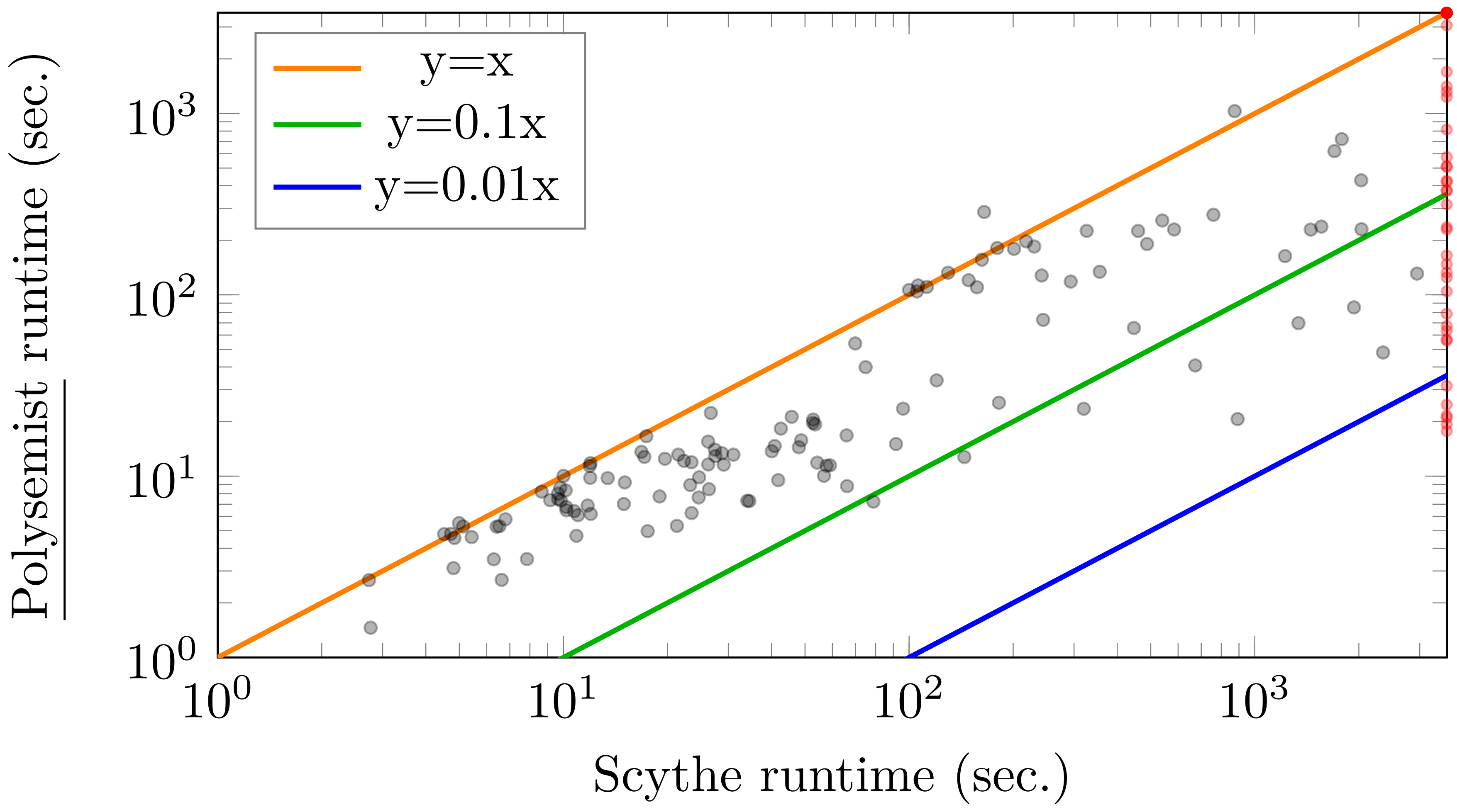
- Detected when pool of enumerated expression reaches fixed point
- Requires sufficiently weak equivalence relation
- weaker = faster fixed point

Experimental Evaluation

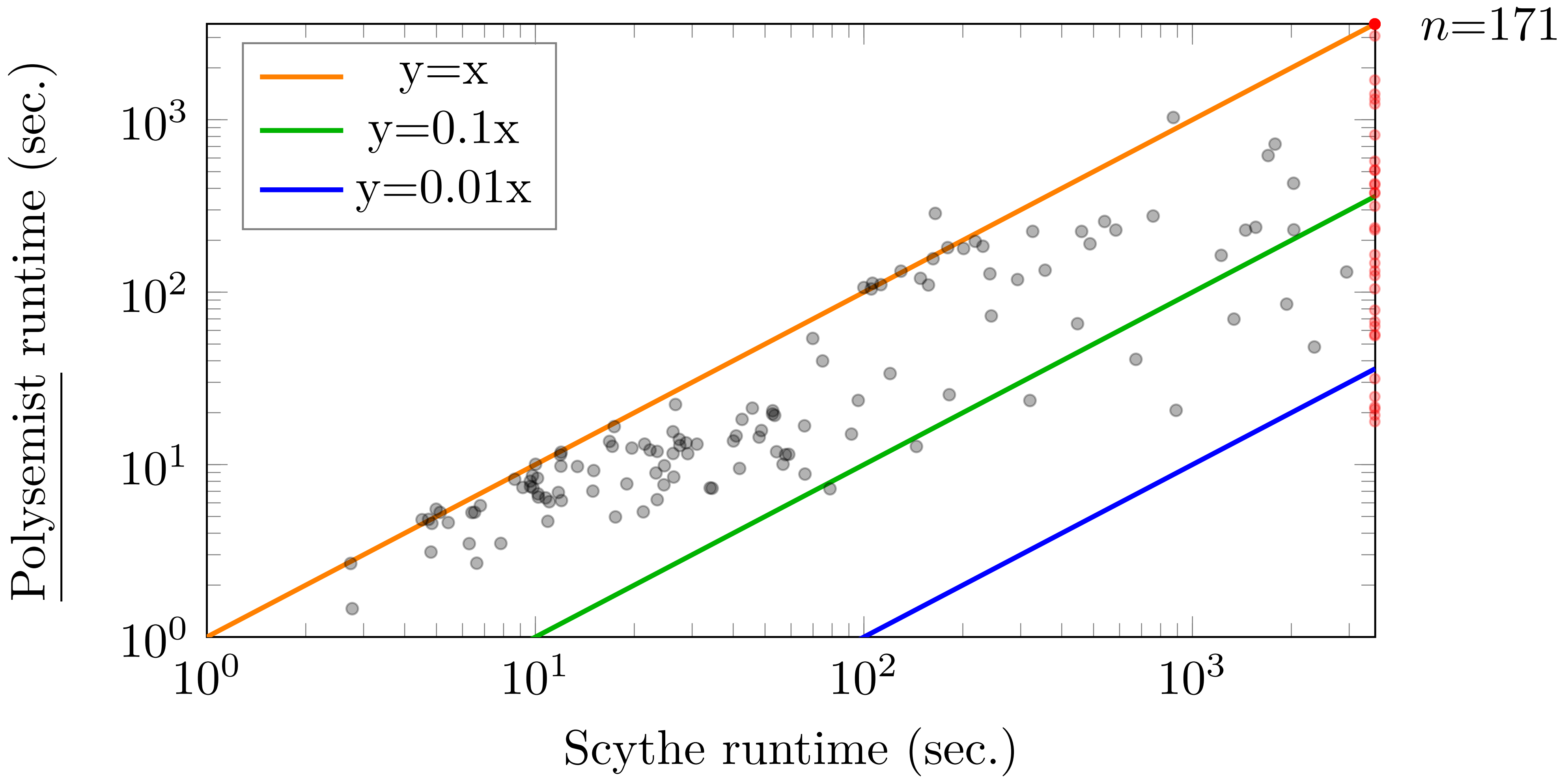
Setup

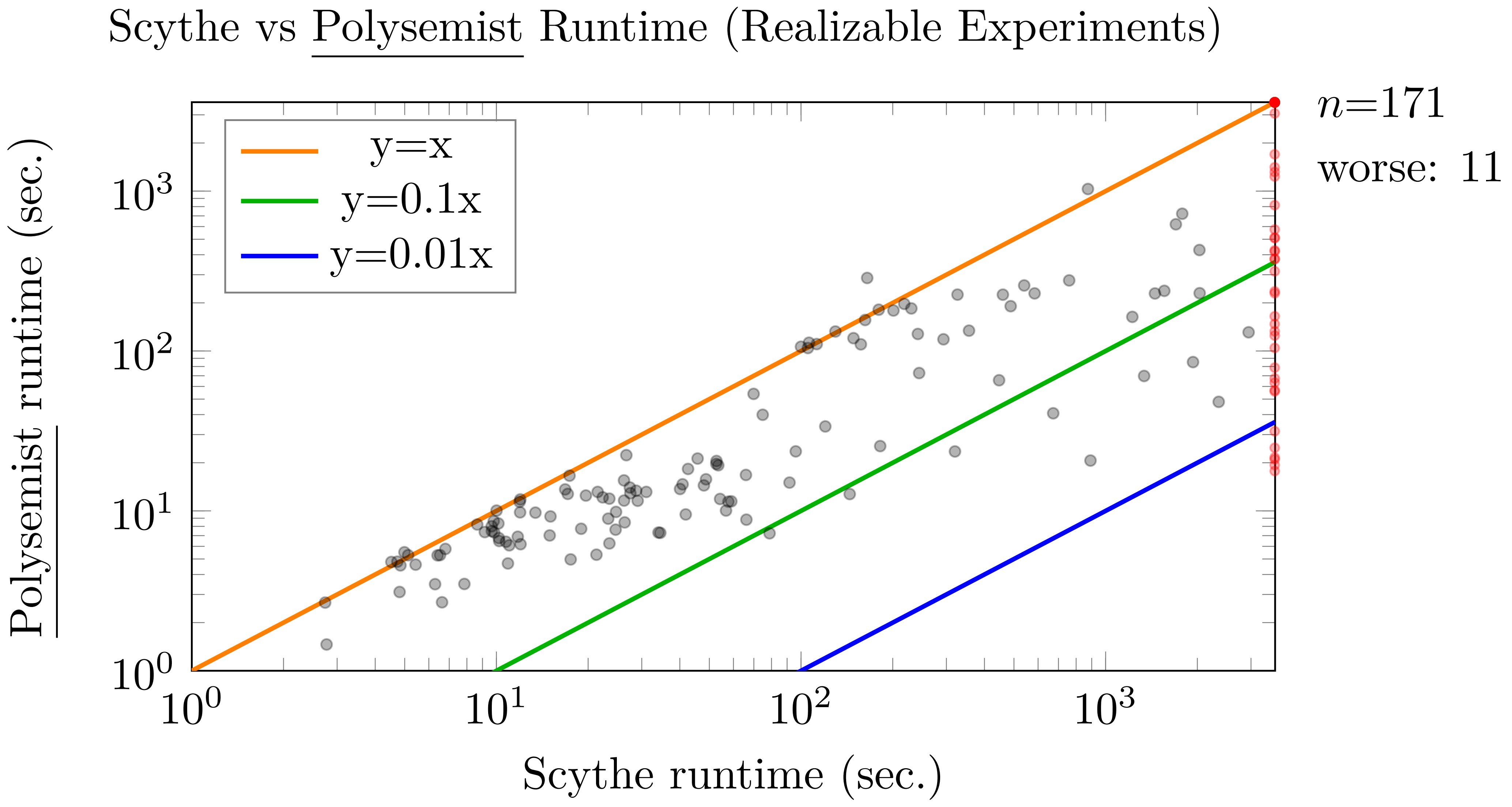
- Our tool: Polysemist
- Compare with state of art tool: Scythe (FMCAD 2024)
- Benchmarks based on those of Scythe
- 7 case studies, including: reconfig-raft, 2PC, SKV, locks
- Realizable ($n = 171$) and Unrealizable ($n = 123$) experiments
- Easiest: all pre- or post-conditions in 1 action
- Hardest: all pre- and post-conditions in 2 actions

Scythe vs Polysemist Runtime (Realizable Experiments)



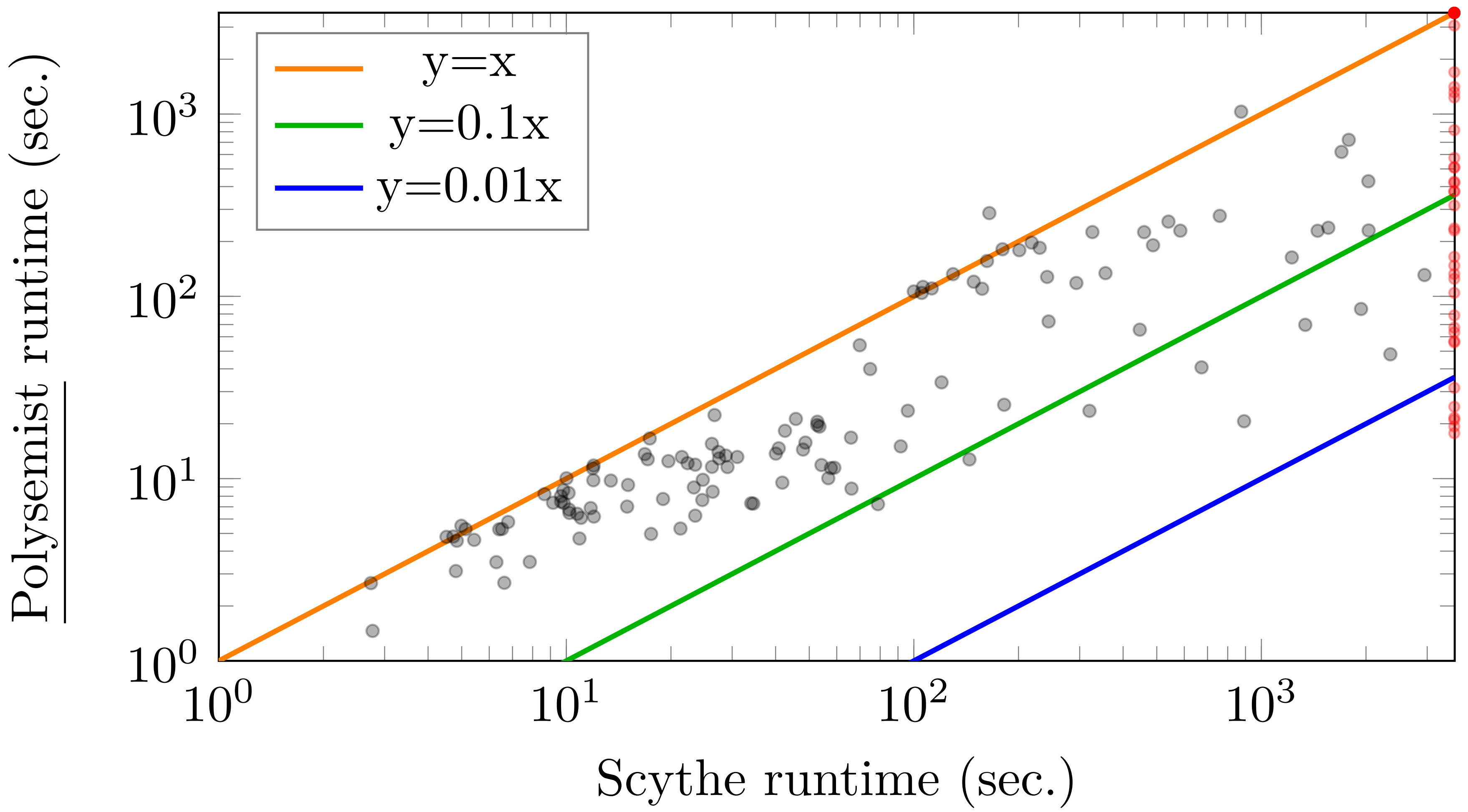
Scythe vs Polysemist Runtime (Realizable Experiments)





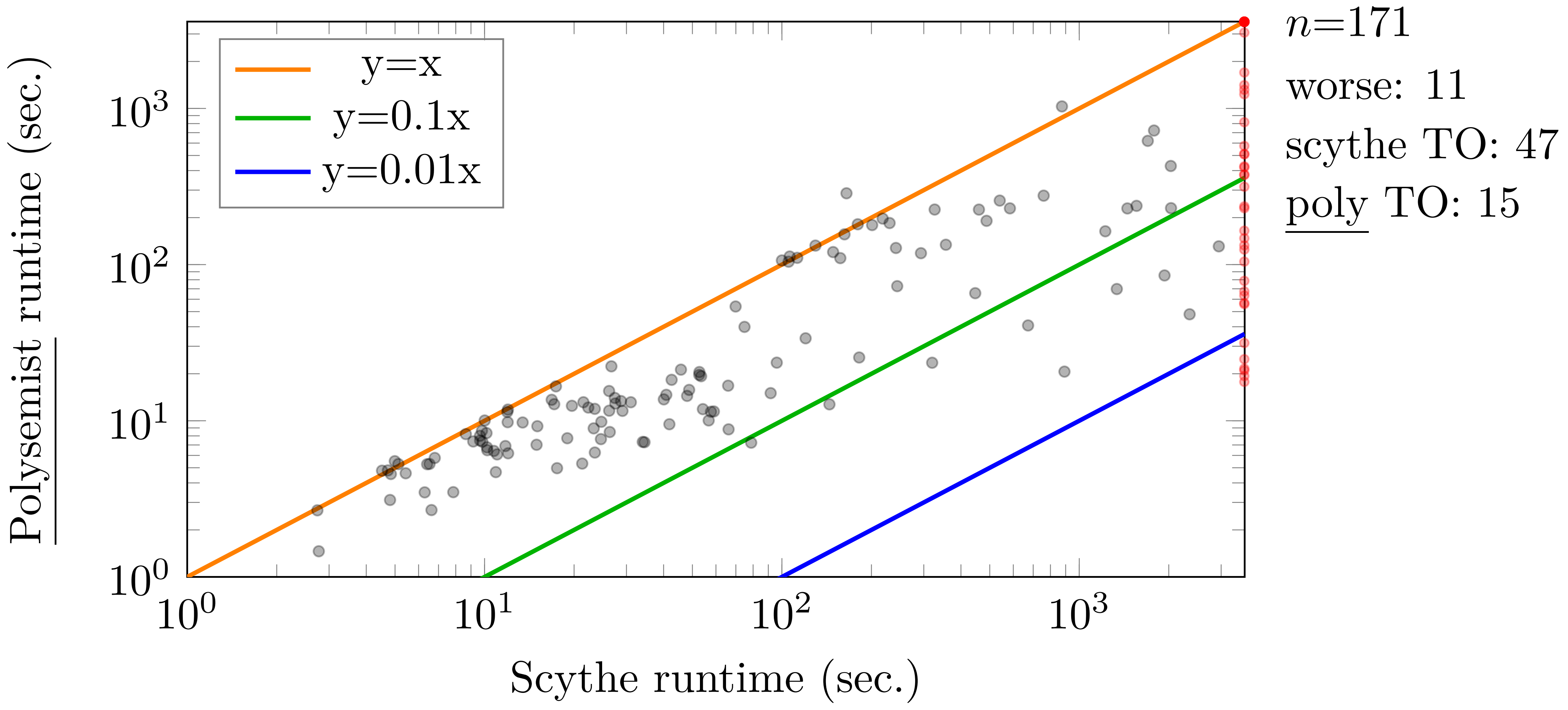
Scythe vs Polysemist Runtime (Realizable Experiments)
y=x
y=0.1x
y=0.01x
n=171
worse: 11
Polysemist runtime (sec.)
Scythe runtime (sec.)
10^0
10^1
10^2
10^3

Scythe vs Polysemist Runtime (Realizable Experiments)



n=171
worse: 11
scythe TO: 47

Scythe vs Polysemist Runtime (Realizable Experiments)



Unrealizable Experiments

- $n = 123$
- **Scythe** : TO=107 / HALT=16
- **Polysemist** : TO=43 / HALT=80
    - Usually halted in <60 seconds
    - Did not TO unless Scythe did

Conclusion

Summary

Summary

- CEGIS for TLA^+ protocols

Summary

- CEGIS for TLA$^+$ protocols
- Counterexample generalization avoids redundant model checking

Summary

- CEGIS for TLA$^+$ protocols
- Counterexample generalization avoids redundant model checking
- Interpretation reduction avoids redundant enumeration

Summary

- CEGIS for TLA$^+$ protocols
- Counterexample generalization avoids redundant model checking
- Interpretation reduction avoids redundant enumeration
- 100x improvements over state of art

Summary

- CEGIS for TLA$^+$ protocols
- Counterexample generalization avoids redundant model checking
- Interpretation reduction avoids redundant enumeration
- 100x improvements over state of art
- Recognize 5x more unrealizable instances than state of art

Thank you!

egolf.d@northeastern.edu

Paper, Tool, Slides, Poster