Black-box Code Analysis for Reverse Engineering

Grégoire Menguy, CEA LIST, France
Sébastien Bardin, CEA LIST, France
Nadjib Lazaar, LIRMM, France
Arnaud Gotlieb, Simula, Norway

Richard Bonichon, Tweag IO, France
Cauim de Souza Lima, CEA LIST, France

Speaker

Grégoire Menguy

PhD student at CEA LIST @BinsecTool

@grmenguy

https://gregoiremenguy.github.io/

Context

Software size +

- Hard to verify code
- Hard to test code
- Hard to understand code

Need for automatic software analysis

White vs Black-box

White-box analysis

- Rely on the syntax
  - Symbolic exec.
  - Abstract interp.
- Scale issues

Black-box analysis

- Rely on code executions
- **Insensitive to syntactic complexity**

Scenarios

Deobfuscation

- Malware analysis
- Assess obfuscation strength

30 MIN

Contract inference

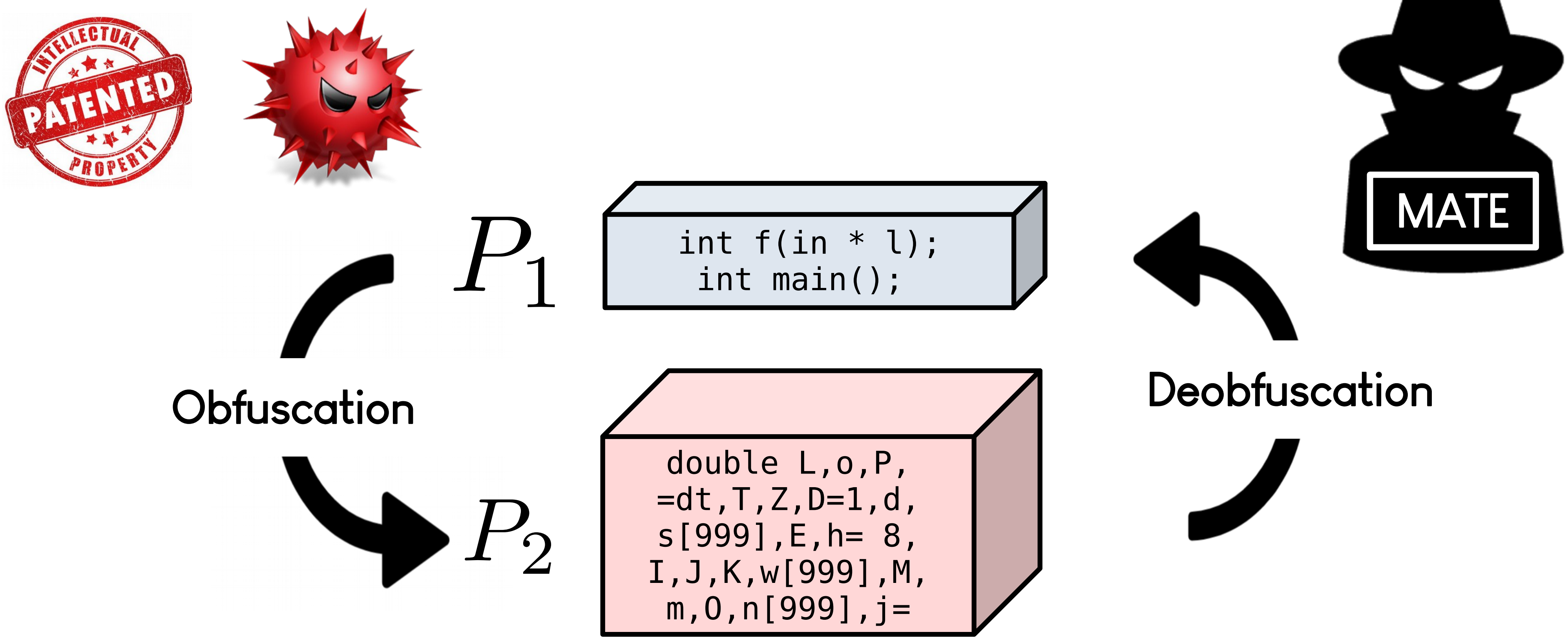
- Core refactoring
- Code understanding

10 MIN

Search-based Local Blackbox Deobfuscation: Understand, Improve and Mitigate

Grégoire Menguy, CEA LIST, France
Sébastien Bardin, CEA LIST, France
Richard Bonichon, Tweag IO, France
Cauim de Souza Lima, CEA LIST, France

Obfuscation



Deobfuscation

Protecting Software through Obfuscation: Can It Keep Pace with Progress in Code Analysis?

SEBASTIAN SCHRITTWIESER, St. Pölten University of Applied Sciences, Austria
STEFAN KATZENBEISSER, Technische Universität Darmstadt, Germany
JOHANNES KINDER, Royal Holloway, University of London, United Kingdom
GEORG MERZDOVNIK and EDGAR WEIPPL, SBA Research, Vienna, Austria

A Generic Approach to Automatic Deobfuscation of Executable Code

Babak Yadegari, Brian Johannesmeyer, Benjamin Whitely, Saumya Debray
Department of Computer Science
The University of Arizona
Tucson, AZ 85721
{babaky, bjohannesmeyer, whitely, debray}@cs.arizona.edu

Symbolic deobfuscation: from virtualized code back to the original*

Jonathan Salwan[1], Sébastien Bardin[2], and Marie-Laure Potet[3]

Backward-Bounded DSE: Targeting Infeasibility Questions on Obfuscated Codes*

Sébastien Bardin
CEA, LIST,
91191 Gif-Sur-Yvette, France
sebastien.bardint@cea.fr

Robin David
CEA, LIST,
91191 Gif-Sur-Yvette, France
robin.david@cea.fr

Jean-Yves Marion
Université de Lorraine,
CNRS and Inria, LORIA, France
jean-yves.marion@loria.fr

BINSEC

TRITON
Dynamic Binary Analysis

MIASM

Deobfuscation

Protecting Software through Obfuscation: Can It Keep Pace with Progress in Code Analysis?

SEBASTIAN SCHRITTWIESER, St. Pölten University of Applied Sciences, Austria
STEFAN KATZENBEISSER, Technische Universität Da
JOHANNES KINDER, Royal Holloway, University of Lo
GEORG MERZDOVNIK and EDGAR WEIPPL, SE

A Generic Approach to Automatic Deobfusca

Babak Yadegari Brian Johannesmeyer Benjamin
Department of Computer Scie
The University of Arizona
Tucson, AZ 85721
{babaky, bjohannesmeyer, whitely, deb

Symbolic deobfus
from virtualized code back the original*

Jonathan Salwan[1], Sébastien Bardin[2], and Marie-Laure Potet[3]

Backward-Bounded DSE:
Targeting Infeasibility Questions
on Obfuscated Codes*

Robin David
CEA, LIST,
f-Sur-Yvette, France
n.david@cea.fr

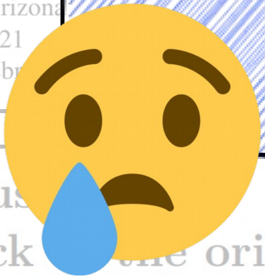
Jean-Yves Marion
Université de Lorraine,
CNRS and Inria, LORIA, France
jean-yves.marion@loria.fr

SEC

TRITON
Dynamic Binary Analysis

MIASM

White-box deobfuscation is highly efficient



Anti-White-Box Deobfuscation

But efficient countermeasures

Information Hiding in Software with Mixed Boolean-Arithmetic Transforms

Yongxin Zhou, Alec Main, Yuan X. Gu, and Harold Johnson

Cloakware Inc., USA
{yongxin.zhou,alec.main,yuan.gu,harold.johnson}@cloakware.com

How to Kill Symbolic Deobfuscation for Free
(or: Unleashing the Potential of Path-Oriented Protections)

Mathilde Ollivier
CEA, LIST,
Paris-Saclay, France
mathilde.ollivier2@cea.fr

Sébastien Bardin
CEA, LIST,
Paris-Saclay, France
sebastien.bardin@cea.fr

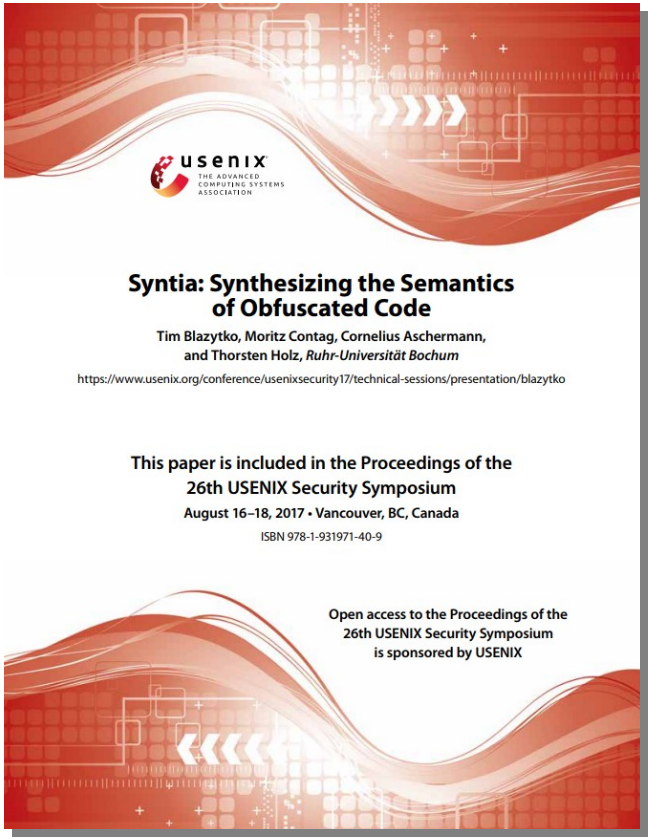
Richard Bonichon
CEA, LIST,
Paris-Saclay, France
richard.bonichon@cea.fr

Jean-Yves Marion
Université de Lorraine, CNRS, LORIA
Nancy, France
Jean-Yves.Marion@loria.fr

Probabilistic Obfuscation through Covert Channels

Jon Stephens Babak Yadegari Christian Collberg Saumya Debray Carlos Scheidegger
Department of Computer Science
The University of Arizona
Tucson, AZ 85721, USA
Email: {stephensj2, babaky, collberg, debray, cscheid}@cs.arizona.edu

New Threat: Black-box Deobfuscation


usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

Syntia: Synthesizing the Semantics of Obfuscated Code

Tim Blazytko, Moritz Contag, Cornelius Aschermann, and Thorsten Holz, *Ruhr-Universität Bochum*

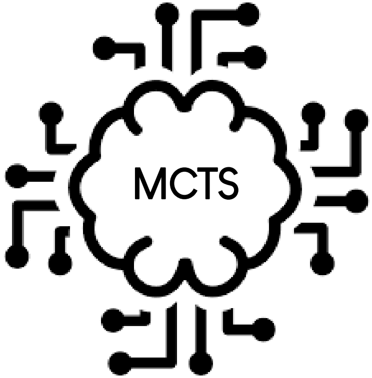
https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/blazytko

This paper is included in the Proceedings of the 26th USENIX Security Symposium

August 16–18, 2017 • Vancouver, BC, Canada

ISBN 978-1-931971-40-9

Open access to the Proceedings of the 26th USENIX Security Symposium is sponsored by USENIX



Bypasses white-box methods limitations

Open Questions

Understand

- Strenghts ?
- Weaknesses ?
- Why ?

Improve

- Why MCTS ?
- Can be improved ?
- Impacted by SOTA protections ?

Mitigate

- How to protect ?

Contributions

Understand

- Propose missing formalisation
- Refine Syntia Xps: new strenghs & weaknesses
- Show & explain why MCTS not appropriate

Improve

- S-metaheursitics > MCTS
- Implement our approach: Xyntia
- Evaluation of Xyntia

Mitigate

- Propose 2 protections
- Evaluate them against Xyntia and Syntia

Black-box deobfuscation
What's that ?

Black-box Deobfuscation

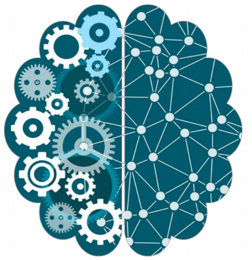
1. Sample

(t = 1, T = 2)
(t = 2, T = 5)
(t = 0, T = 6)
...

-1
-3
-6
...

2. Learn

(t = 1, T = 2) → -1
(t = 2, T = 5) → -3
(t = 0, T = 6) → -6
...



$$t - T$$

Learning Engine

$U \times U$ $t + T$ $t - U$

$\longrightarrow$ $expr$ $\longrightarrow$

$$\Delta\begin{pmatrix} expr(t=1, T=2), -1 \\ expr(t=2, T=5), -3 \\ expr(t=0, T=6), -6 \\ \dots \end{pmatrix}$$

Expression grammar

U := U + U | U – U | U * U …
| t | T | 1

Why Black-box ?

Given a language L and an expression "e" in L

Syntactic complexity	Semantic complexity
Size of the expression "e"	Size of the smallest expr. in L equivalent to "e"

Example

$t - T$ is syntactically simpler than $(t \vee -2T) \times 2 - (t \oplus -2T) + T$

but they share the same semantic complexity (being equivalent)

Why Black-box ?

Given a language L and an expression "e" in L

Syntactic complexity	Semantic complexity
Size of the expression "e"	Size of the smallest expr. in L equivalent to "e"

Example

$t - T$ is syntactically simpler than $(t \vee -2T) \times 2 - (t \oplus -2T) + T$

but they share the same semantic complexity (being equivalent)

Obfuscation increases syntactitc complexity
→ **No impact on black-box methods**

Understand

Zoom of SOTA: Syntia

Dig into Syntia and deepen its evaluation

– RQ1: stability of Syntia

– RQ2: efficiency of Syntia

– RQ3: impact of operators set

Syntia: New Results

Stable

Quality

Correctness

Speed

Robustness

Experimental Design

B1 (Syntia)

- 500 expressions
- Up to 3 inputs
- redundancy
- Unbalances w.r.t. type

B2 (ours)

- 1110 expressions
- 2 to 6 inputs
- No redundancy
- Balances w.r.t. type

	Type			# Inputs				
	Bool.	Arith.	MBA	2	3	4	5	6
#Expr.	370	370	370	150	600	180	90	90

Table 1: Distribution of samples in benchmark B2

Evaluation of Syntia

B1 (Syntia)

– With a 60 s/expr. timeout: 75% of success rate

– With a 1 h/expr. timeout: 88% of success rate

– With a **12 h/expr. timeout: 97% of success rate**

B2 (Ours)

Table 2: Syntia depending on the timeout per expression (B2)

	1s	10s	60s	600s
Succ. Rate	16.5%	25.6%	34.5%	42.3%
Equiv. Range	16.3%	25.1 - 25.3%	33.7 - 34.0%	41.4 - 41.6%
Mean Qual	0.35	0.49	0.59	0.67

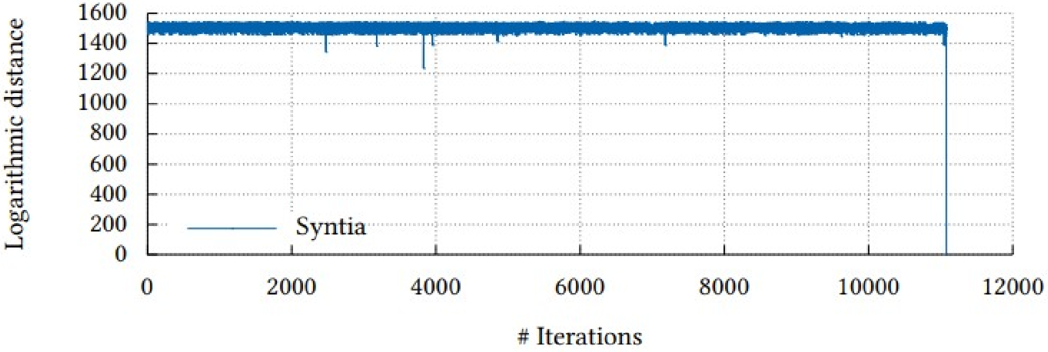
Why ?

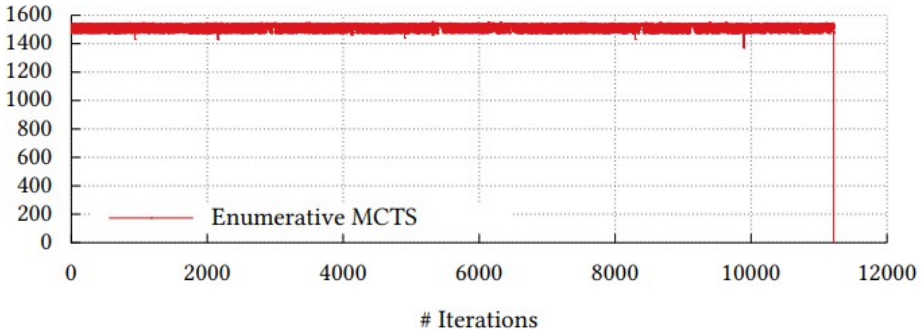
- Syntia manipulates non terminal expressions, e.g., $U - V$
- Scoring of non terminal expressions can be misleading

 $t - T$

$$U - V \rightsquigarrow \begin{cases} t - T \\ t - 1 \\ 1 - 1 \end{cases}$$

- Syntia (i.e., MCTS) = “almost BFS”

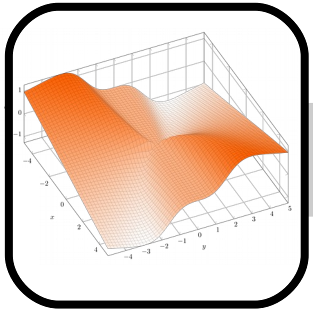




Improve

Black-box Deobf: An Optimization Pb

Syntia sees blackbox deobfuscation as a **single player game**

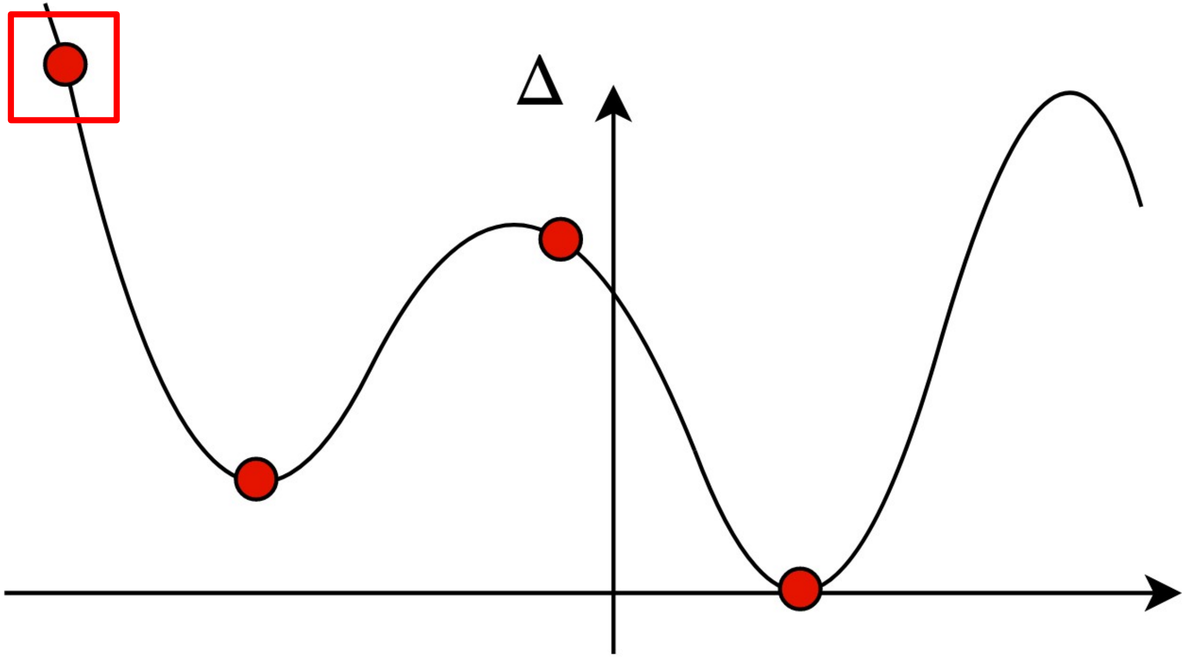


We propose to see it as an **optimization problem**

Goal : find $\underbrace{s^*}_{\text{An expr.}}$ s.t. $\underbrace{f}_{\Delta}(s^*) \leq f(s), \forall s \in S$

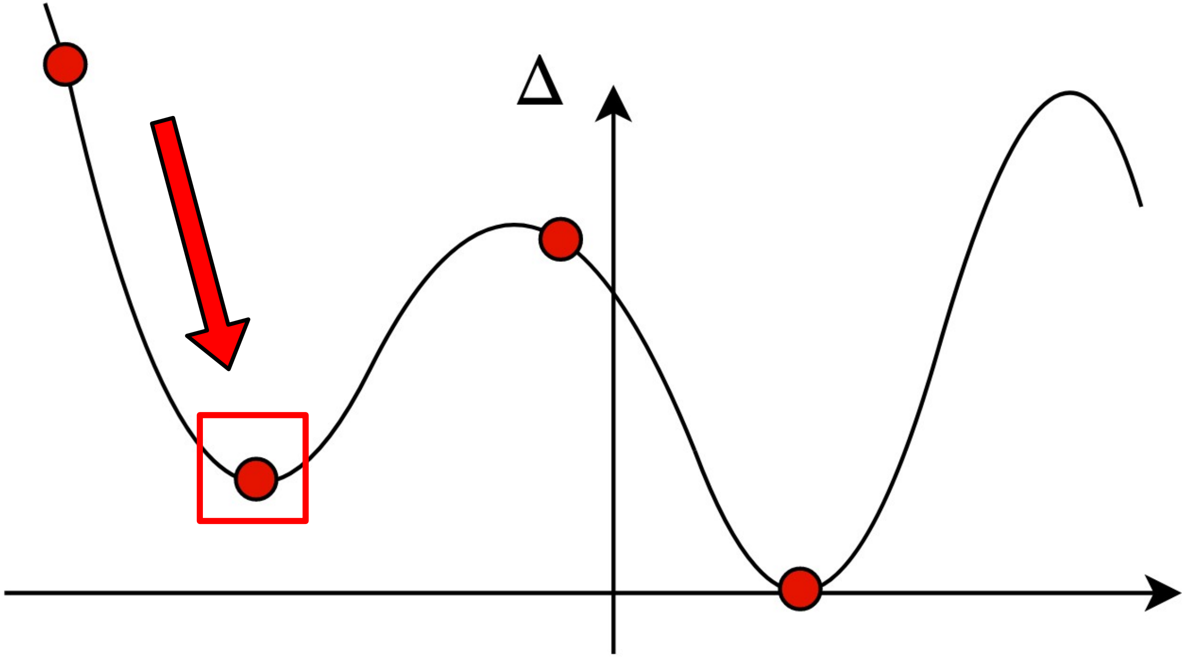
S-metaheuristics

– Solve optimization problems



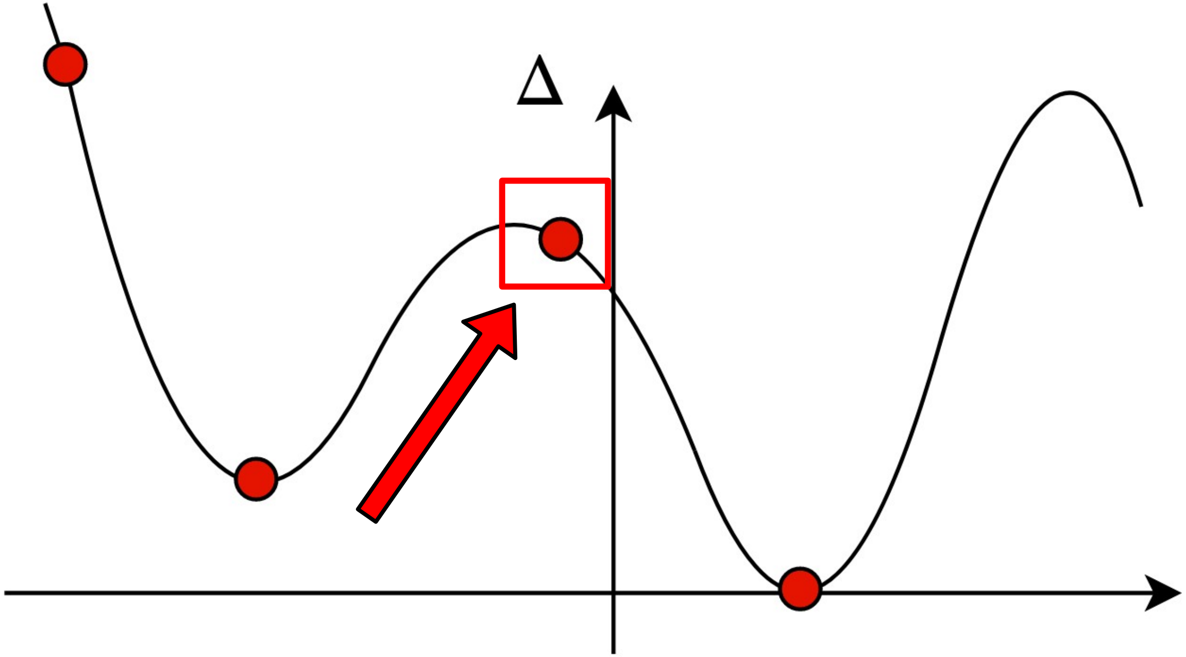
S-metaheuristics

– Solve optimization problems



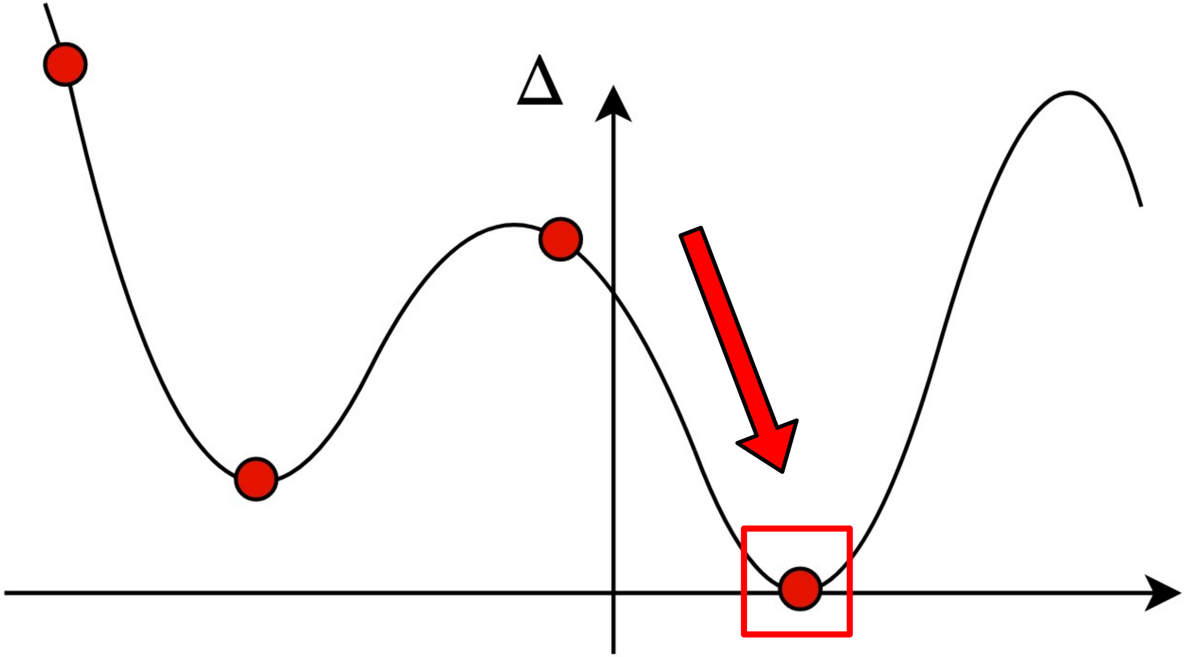
S-metaheuristics

– Solve optimization problems



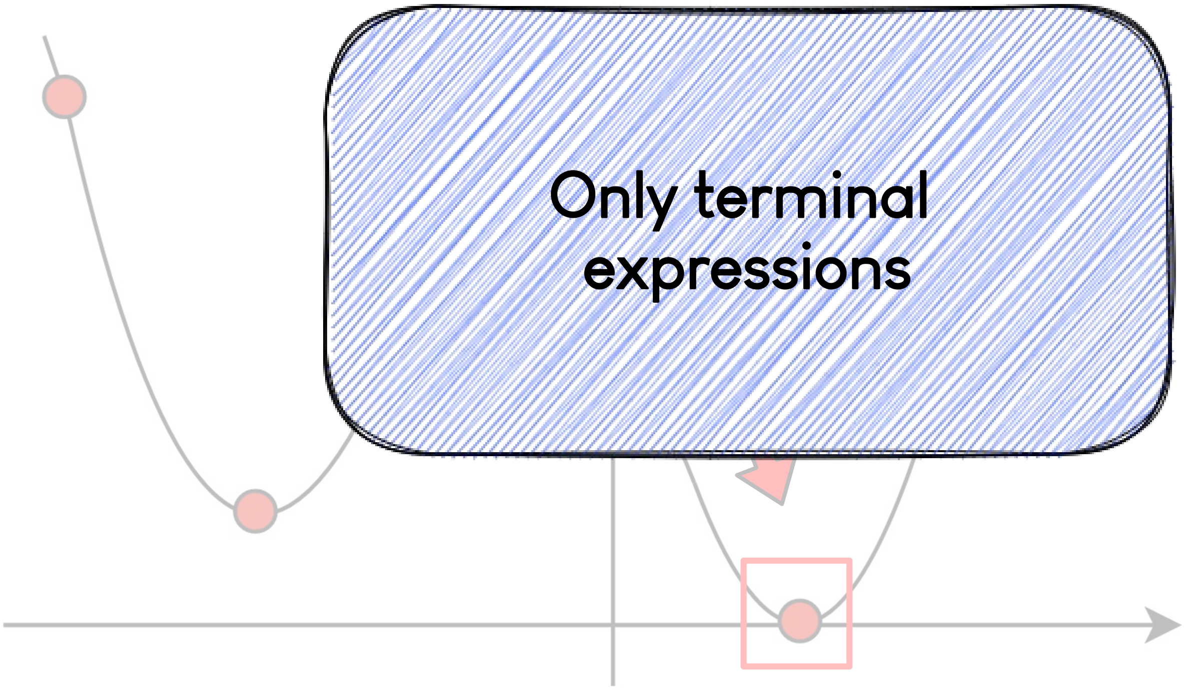
S-metaheuristics

– Solve optimization problems

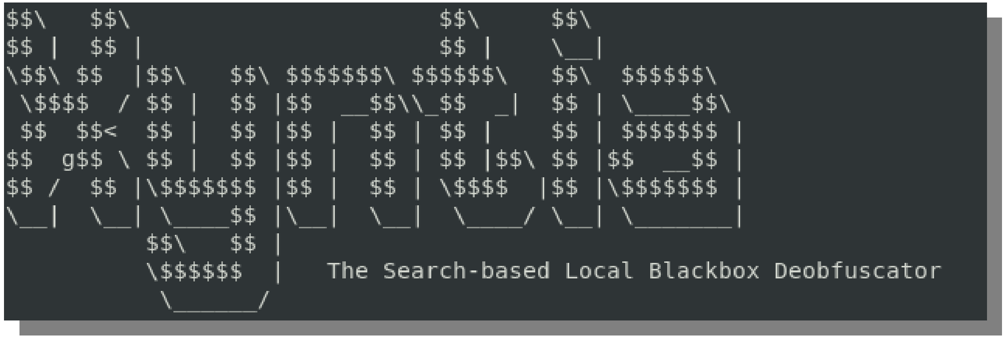


S-metaheuristics

– Solve optimization problems



New Prototype: Xyntia



↳ **S-metaheuristics**

↳ **Can choose between:**
- Hill Climbing
- Simulated Annealing
- Metropolis Hasting
- Iterated Local Search

} > **MCTS**

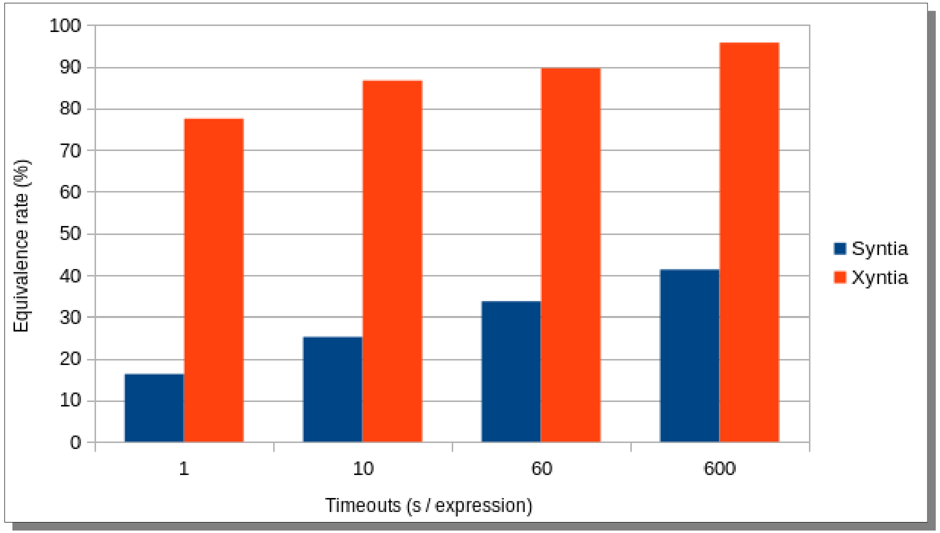
Xyntia vs Syntia

B1 (Syntia)

– 100 % success rate in 1s/expr.

Syntia: 75% in 60 s/expr.

B2 (Ours)



Xyntia vs Syntia

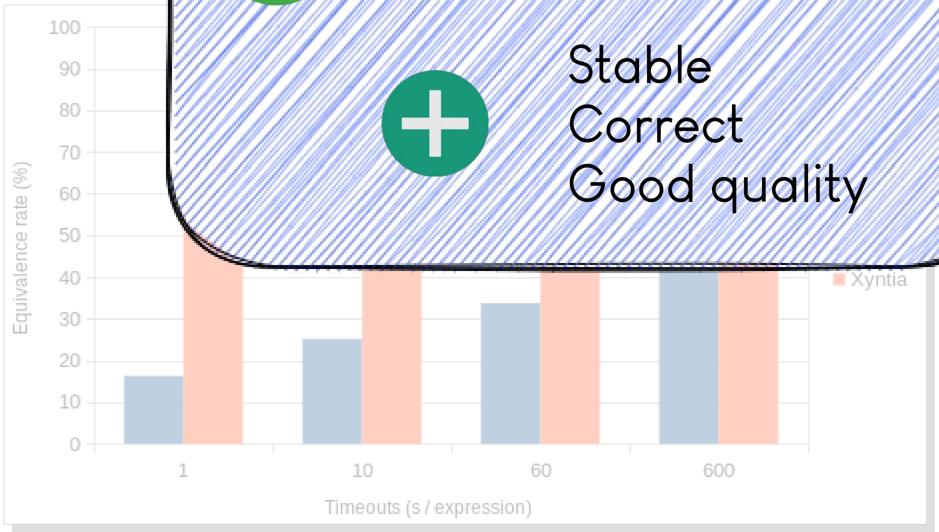
B1 (Syntia)

– 100 % success rate in 1s/expr

Syntia: 75% in 60 s/expr.

Robust & Fast

Stable
Correct
Good quality



Other Experiments

– Xyntia vs Qsynth

– Xyntia vs “compiler like simplifications”

– Xyntia vs program synthesizer CVC4

– Xyntia vs superoptimizer STOKE

404

Not Found

The resource requested could not be found on this server!

– Use-cases
 - ↳ State-of-the-art protections
 - ↳ VM-based obfuscation

What's Next ?


PROTECTIONS ARE BROKEN!


DON'T WORRY...
I HAVE A PLAN
memegenerator.net

Mitigate

Context: Virtualization

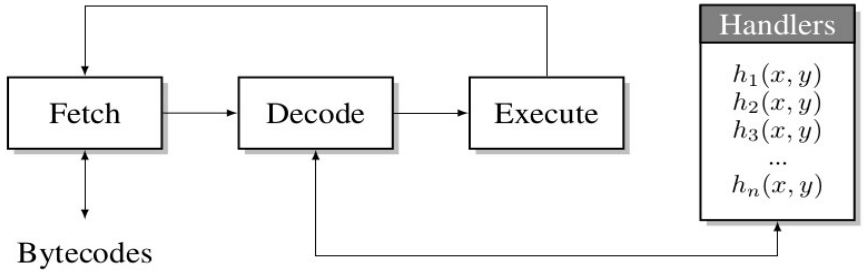


Fig. 1: Virtualization based obfuscation

Proved to be sensitive to black-box deobfuscation

Themida® ADVANCED WINDOWS SOFTWARE PROTECTION

Why VM-based Obf. Is Vulnerable ?

- Handlers are too semantically simple
  - ↳ e.g., $+, -, \times, \wedge, \vee$
- Obfuscation increases syntactic complexity
  - ↳ Black-box deobf. is **not impacted**

We need to move ...

From syntactic to semantic complexity

Semantically Complex Expressions

Goal

- ↳ Increase the semantic complexity of each handlers
- ↳ Keep a Turing complete set of handlers

Example

$$
\begin{array}{rrcl}
 & h_0 & = & (x+y) + -((a-x^2) - (xy)) \\
+ & h_1 & = & (a-x^2) - xy + (-(y-(a \wedge x)) \times (y \otimes x)) \\
+ & h_2 & = & (y-(a \wedge x)) \times (y \otimes x) \\
\hline
 & h & = & x+y
\end{array}
$$

Merged Handlers

Goal : Increase handlers semantic complexity + sampling harder

Example : $h_1(x, y) = x + y$ and $h_2(x, y) = x \wedge y$

$$\longrightarrow h(x, y, c) = \text{if } (c = cst) \text{ then } h_1(x, y) \text{ else } h_2(x, y)$$

Hide conditionals :

```c
int32_t h(int32_t a, int32_t b, int32_t c) {
    // if (c == cst) then h1(a,b,c) else h2(a,b,c);
    int32_t res = c - cst ;
    int32_t s = res >> 31;
    res = (-((res ^ s) -s) >> 31) & 1;
    return h1(a, b, c)*(1 - res) + res*h2(a, b, c);
}
```

Semantically Complex Handlers: Results

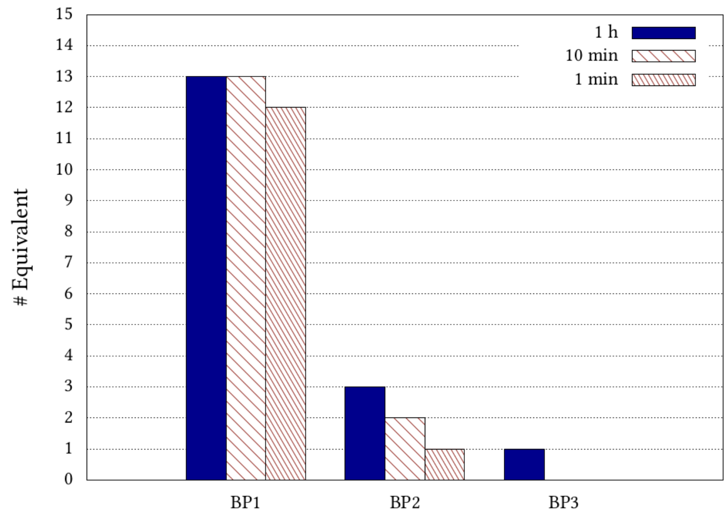


Figure 8: Xyntia ($\text{Xyntia}_{\text{OPT}}$) on BP1,2, 3 – varying timeouts

More results: Syntia with 12h/exprs. → 1/15 on BP1

Merged Handlers: Results

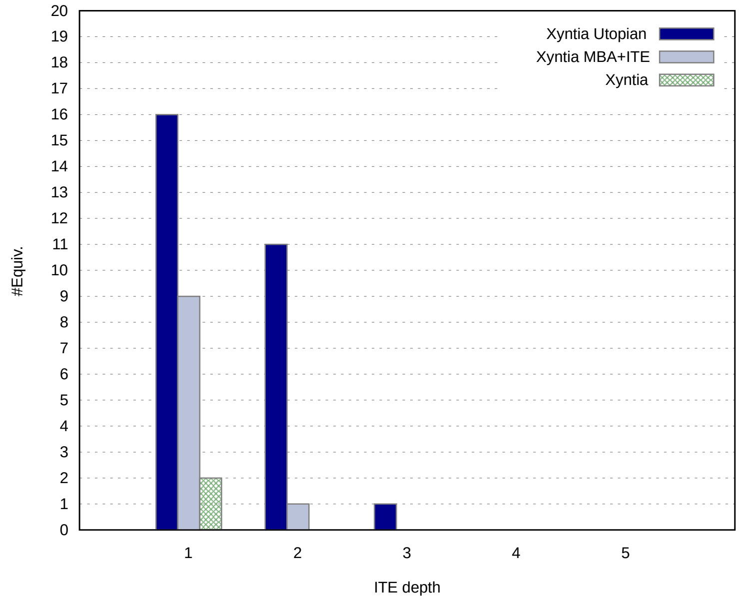


Figure 10: Merged handlers: Xyntia (timeout=60s)

More results: Syntia finds nothing for ≥ 2 nested ITE

Xyntia: The Recap

MCTS is not appropriate for blackbox deobfuscation
→ Search space too unstable
→ Estimation of non terminal expressions pertinence is misleading

S-metaheuristics yields a significant improvement
→ More robust
→ Much Faster

Moving for syntactic to semantic complexity
→ 2 efficient methods to protect against blackbox deobfuscation

Discussion: No Guarantees

– Xyntia & Syntia have **no correctness guarantees**

– But **some contexts need it:**

↳ Code verification

↳ Code refactoring

Can black-box approaches have clear guarantees ?

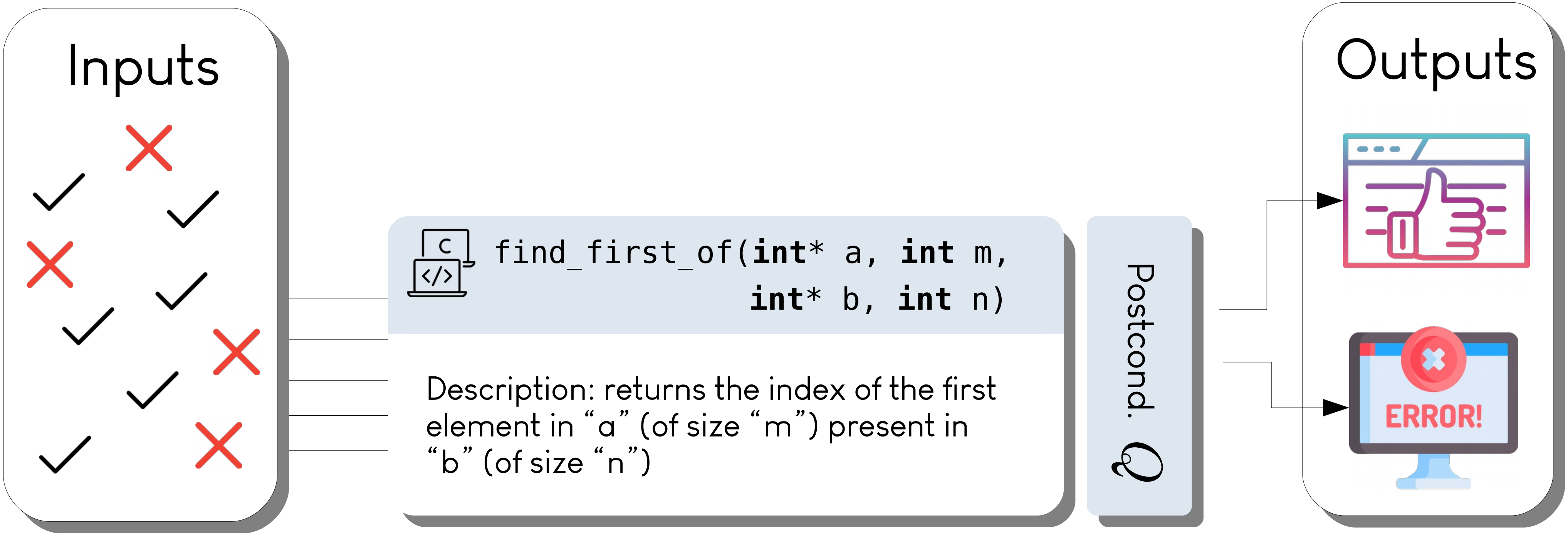
H2020 TAILOR -Foundations of Trustworthy AI -
Integrating Reasoning, Learning and Optimization

Automated Program Analysis: Revisiting Precondition Inference through Constraint Acquisition

Grégoire Menguy, CEA LIST, France
Sébastien Bardin, CEA LIST, France
Nadjib Lazaar, LIRMM, France
Arnaud Gotlieb, Simula, Norway

IJCAI ECAI
VIENNA 22

Dream: Infer Preconditions



Dream: Infer Preconditions

Inputs

$$[m > 0 \Rightarrow valid(a)]$$
$$\wedge$$
$$[m > 0 \wedge n > 0 \Rightarrow valid(b)]$$

```c
find_first_of(int* a, int m,
              int* b, int n)
```

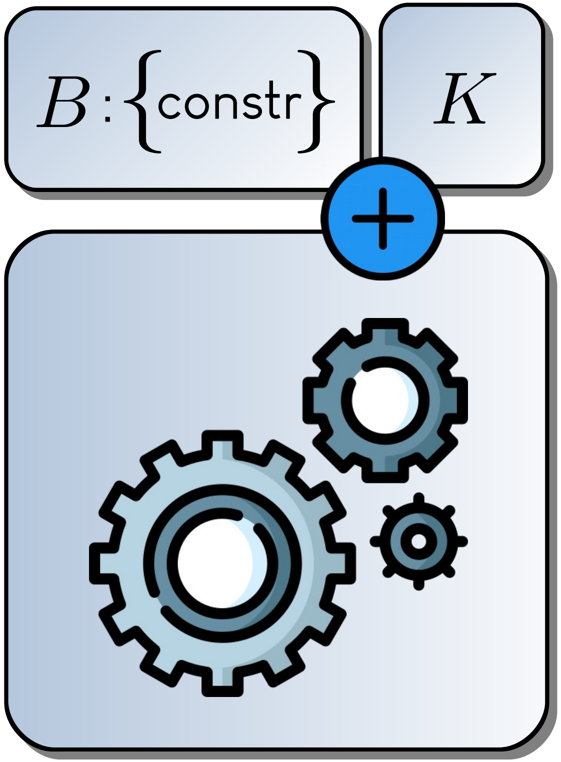
Description: returns the index of the first element in “a” (of size “m”) present in “b” (of size “n”)

$Q = true$

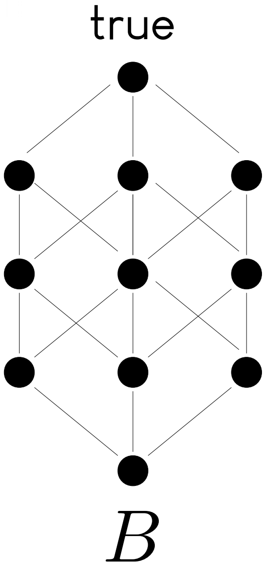
Outputs

Undecidable problem: Rice theorem (1953)

Active Constraint Acquisition



Background knowledge:
rules to speed up learning



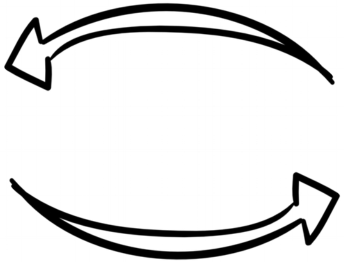
Active Constraint Acquisition

Query

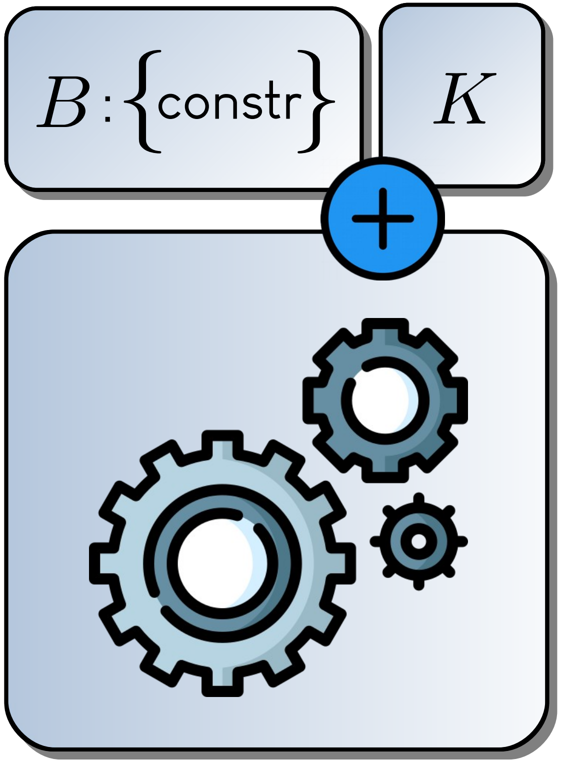
Elise: 8h – 12h

Paul: 10h – 11h

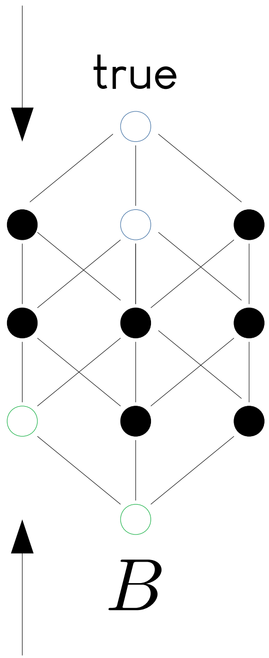
...



yes /no

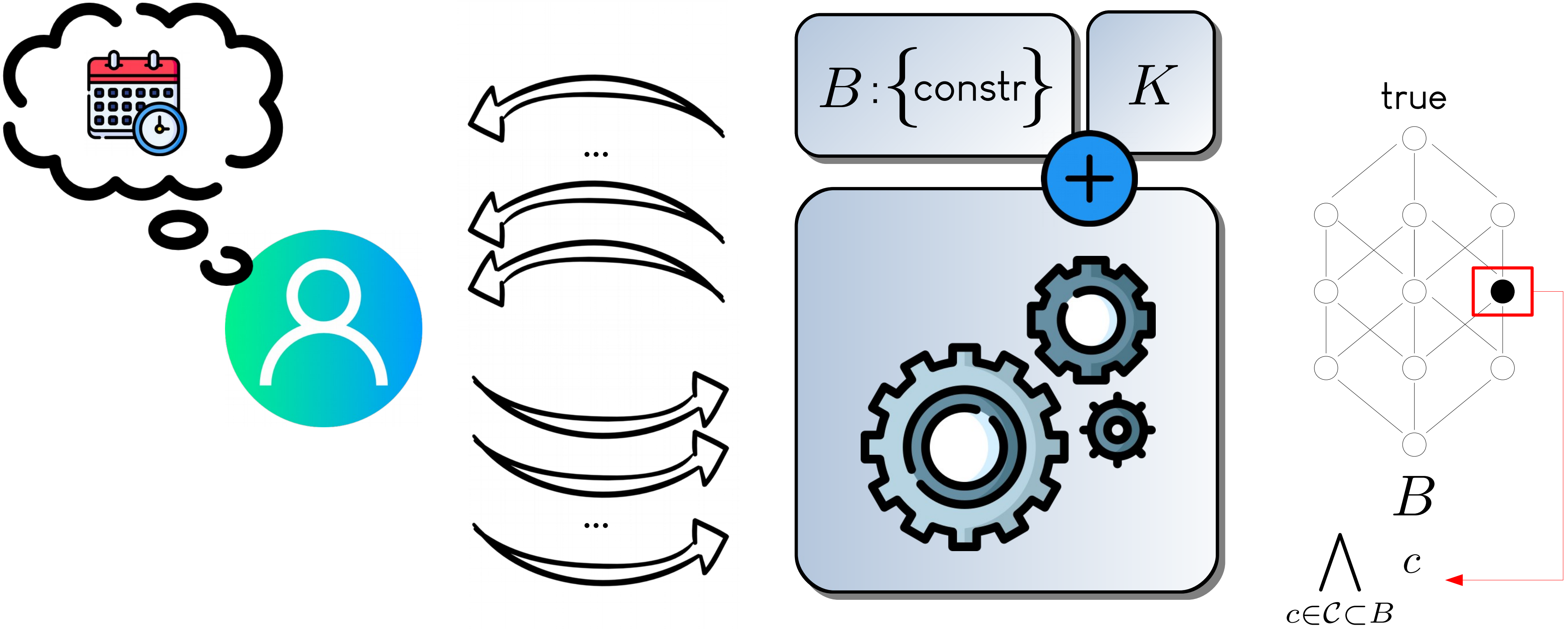


no: Top-down

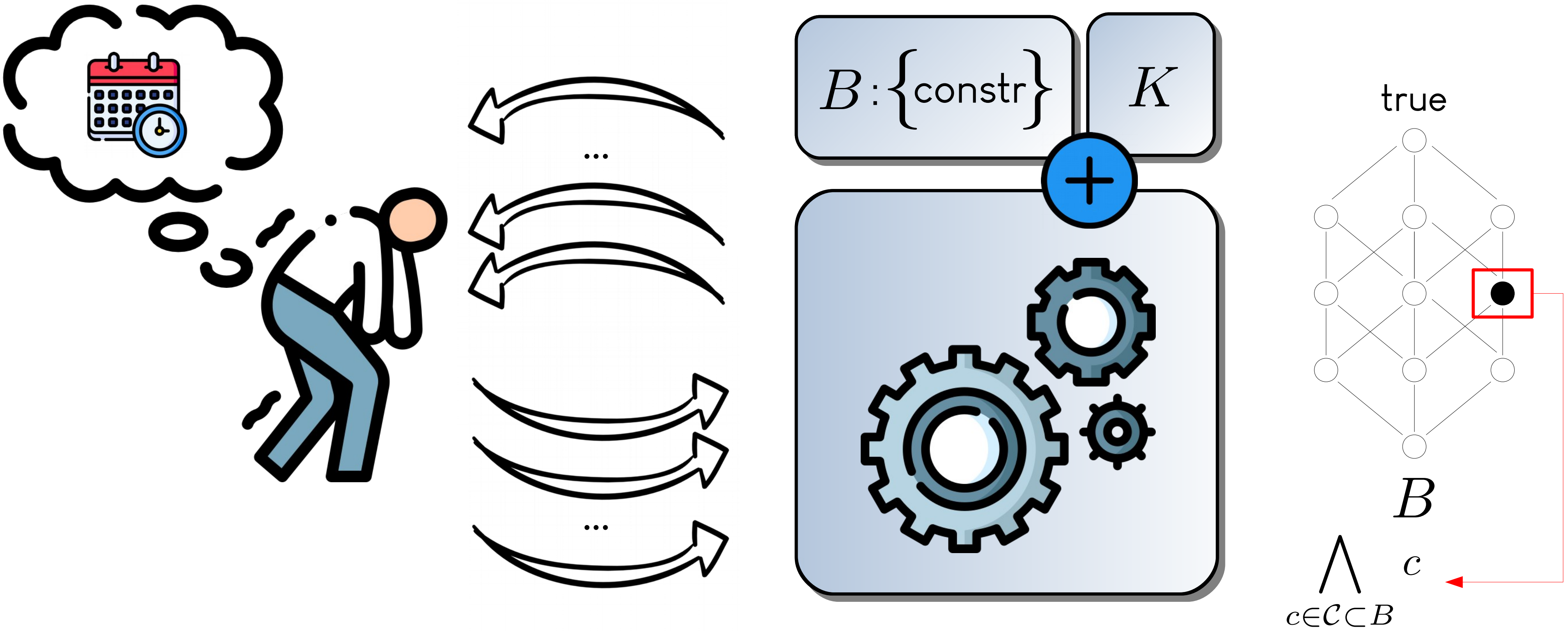


yes: Bottom-up

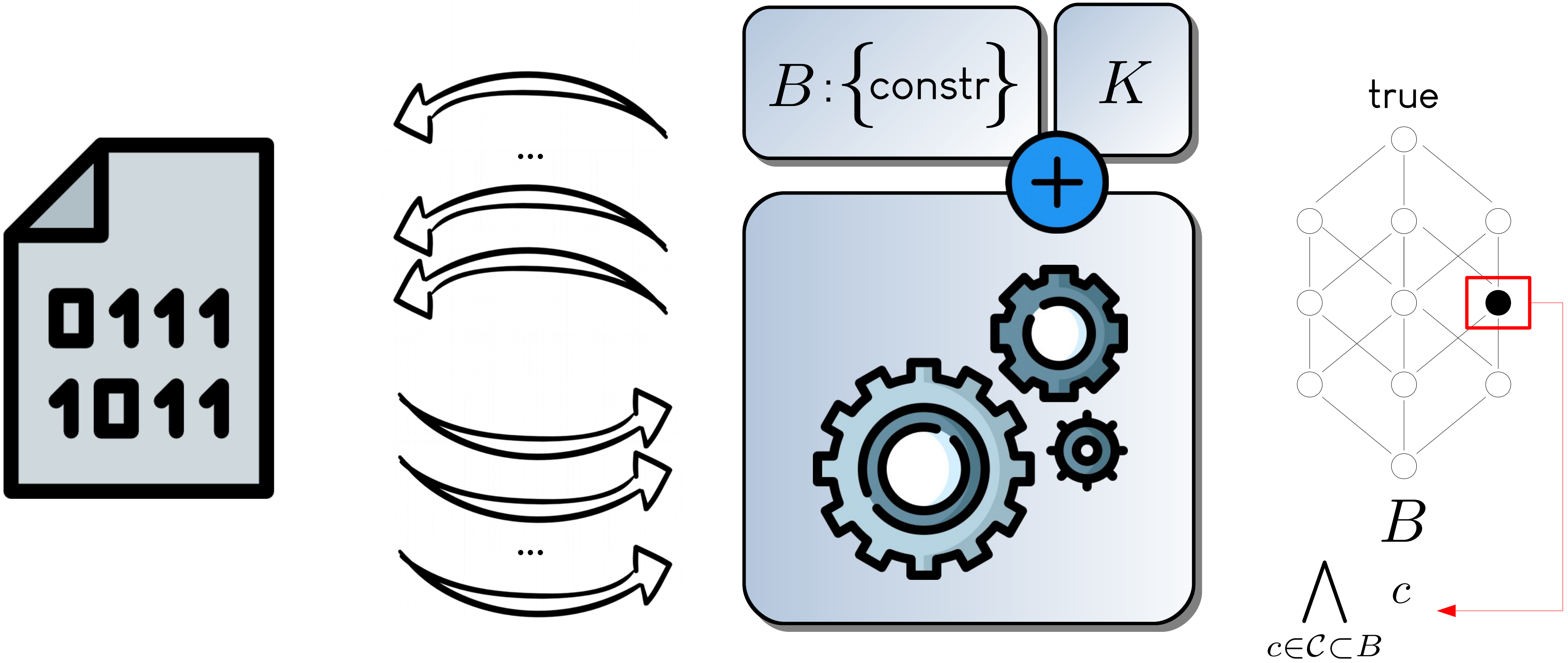
Active Constraint Acquisition



Careful: too many queries



PreCA: Precondition Constr. Acq.



Theoretical Analysis

PreCA guarantees

↳ If B is expressive enough → ∅ or Precond.

↳ ⊕ If code exec. always terminates → The most general precondition

These are good theoretical guarantees

↳ SOTA executions based methods, from programming language community, have no clear guarantees

Evaluation

Dataset: 94 learning tasks • compiled C functions (string.h, arrays, arithmetic ...)

Evaluation:

	PreCA		1 hour Daikon, PIE, Gehr et al		P-Gen
$Q = true$	92%	VS	At most 52%	VS	74%
$Q \neq true$	41%		At most 23%		34%

⊕ PreCA better in 5s than concurrent tools in 1 hour

Conclusion

Black-box methods can be used in broad contexts

↪ Deobfuscation – Contracts inference

Balance between robustness and guarantees

Thank you for your attention

@grmenguy

https://gregoiremenguy.github.io/