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

Speaker

Grégoire Menguy

PhD student at CEA LIST @BinsecTool

@grmenguy

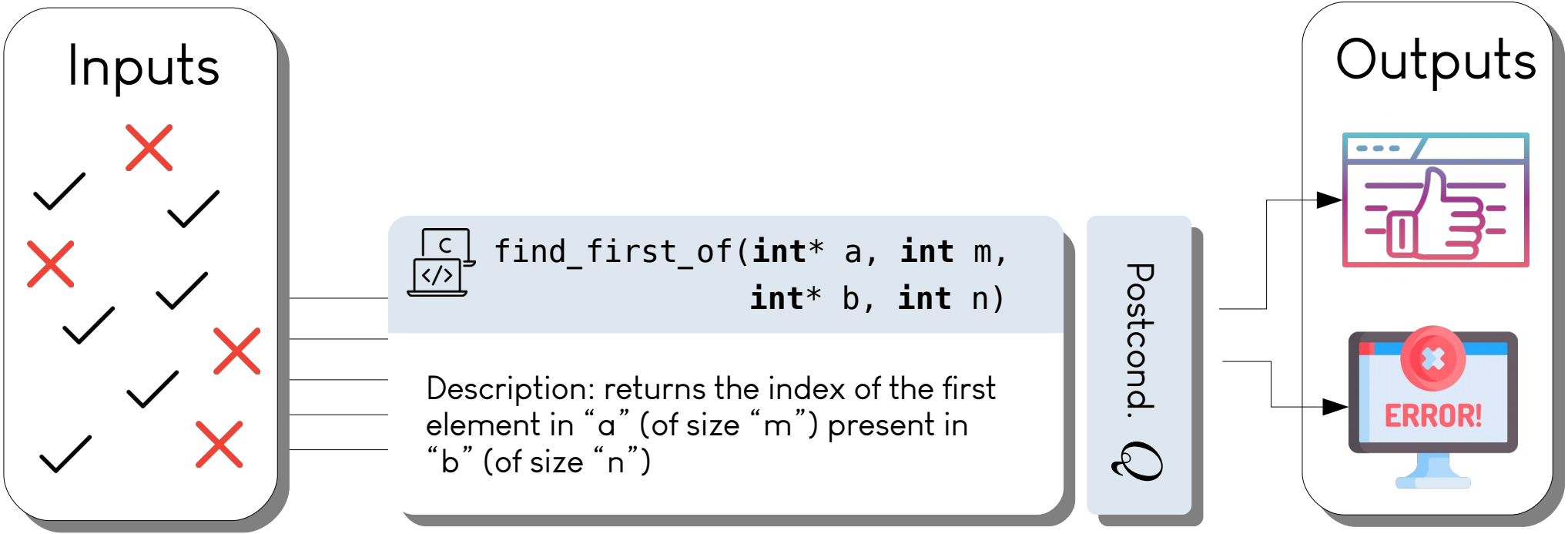
https://gregoiremenguy.github.io/

On the Way to Secure Code

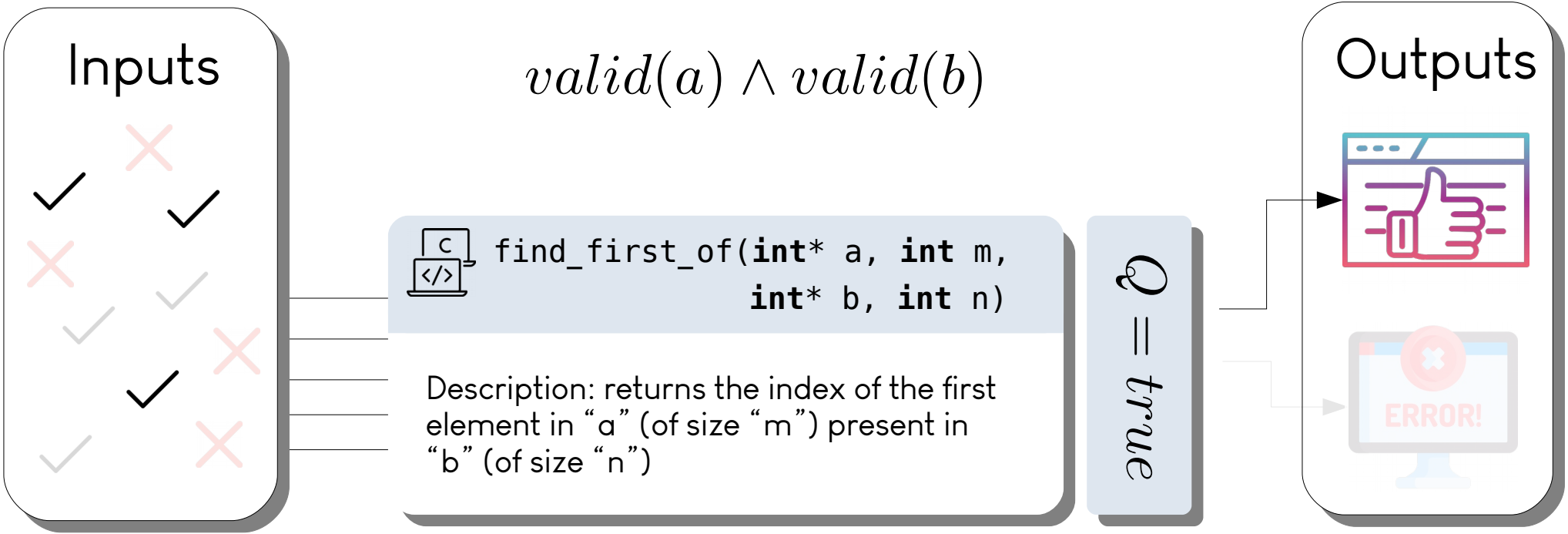
Improve Confidence in Software

- Testing
- Formal Verification
  - E.g., Precondition / postconditon
    - Enable to scale to big code
    - Almost never given in practice

Dream: Infer Preconditions

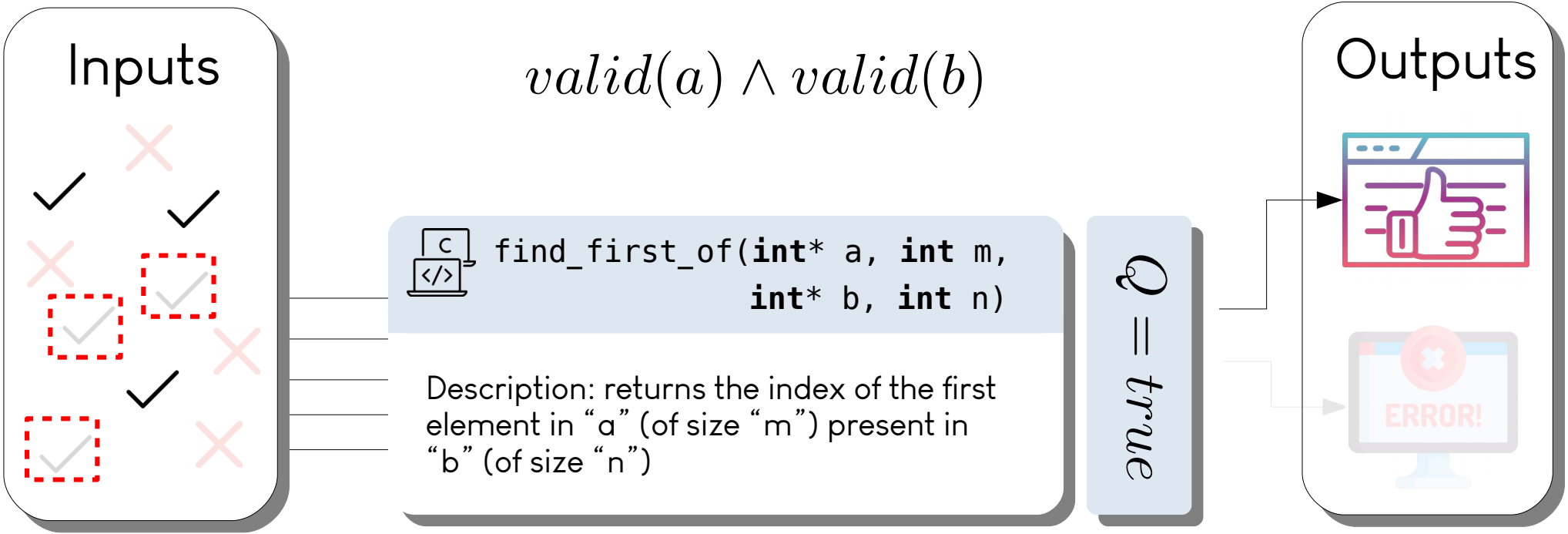


Dream: Infer Preconditions



Undecidable problem: Rice theorem (1953)

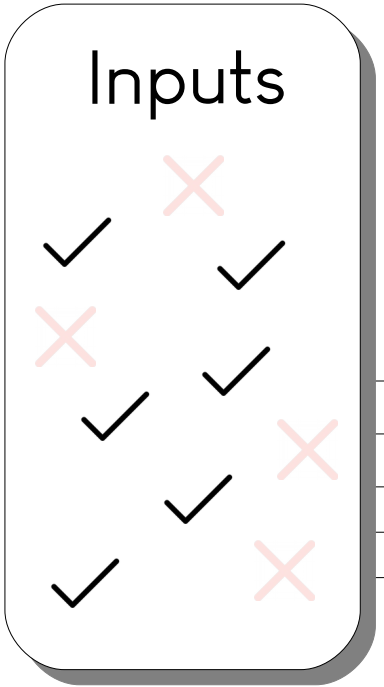
Dream: Infer Preconditions



Undecidable problem: Rice theorem (1953)

Dream: Infer The Weakest Precond.

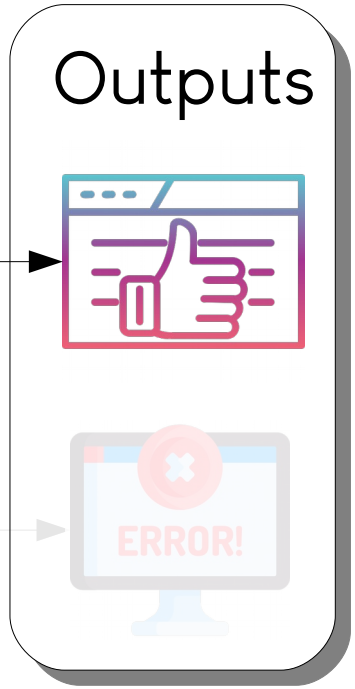
$$[m > 0 \Rightarrow valid(a)]$$
$$\wedge$$
$$[m > 0 \wedge n > 0 \Rightarrow valid(b)]$$




```
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”)

$Q = true$



Undecidable problem: Rice theorem (1953)

State-of-the-art

Execution Based (Daikon, PIE, Gehr et al.):

- 👍 Does not need the source code
- 👎 No clear guarantees

Data-Driven Precondition Inference with Learned Features

Saswat Padhi
Univ. of California, Los Angeles, USA
padhi@cs.ucla.edu

Rahul Sharma
Stanford University, USA
sharmar@cs.stanford.edu

Todd Millstein
Univ. of California, Los Angeles, USA
todd@cs.ucla.edu

Code Based:

- 👎 Need the source code
  - scalability issues • code not available
- 👍 Clear guarantees

Counterexample-Guided Precondition Inference*

Mohamed Nassim Seghir and Daniel Kroening

Computer Science Department, University of Oxford

Goal

Execution Based (Daikon, PIE, Gehr et al.):

- Does not need the source code
- Clear guarantees

Constraint Acquisition Based Precond. Inference

Code Based:

- Need the source code
  - scalability issues • code not available
- Clear guarantees

Data-Driven Precondition Inference with Learned Features

Saswat Padhi
Univ. of California, Los Angeles, USA
padhi@cs.ucla.edu

Rahul Sharma
Stanford University, USA
sharmar@cs.stanford.edu

Todd Millstein
Univ. of California, Los Angeles, USA
todd@cs.ucla.edu

Counterexample-Guided Precondition Inference*

Mohamed Nassim Seghir and Daniel Kroening

Computer Science Department, University of Oxford

Constraint Acquisition

Constraint Programming

↳ Hard to design models

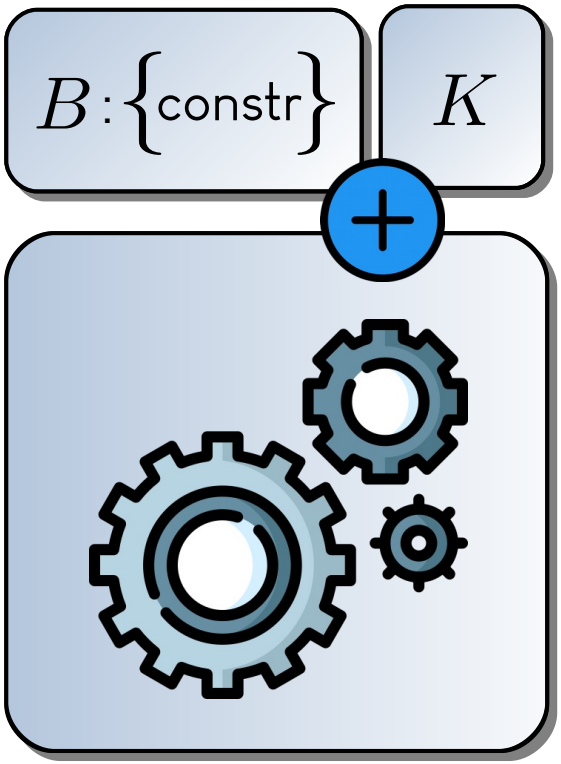
Constraint Acquisition

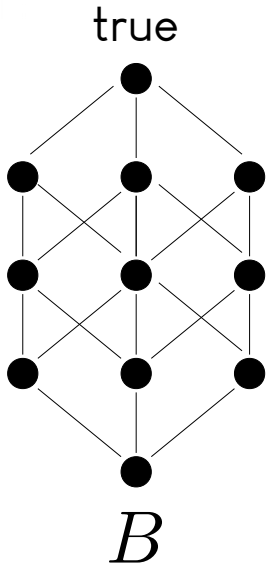
↳ Version Space Learning (Mitchell, 82)

↳ Bessiere, C., Koriche, F., Lazaar, N., & O'Sullivan, B. (2017). Constraint Acquisition. Artificial Intelligence, 244, 315-342.

Active Constraint Acquisition

Background knowledge:
rules to speed up learning





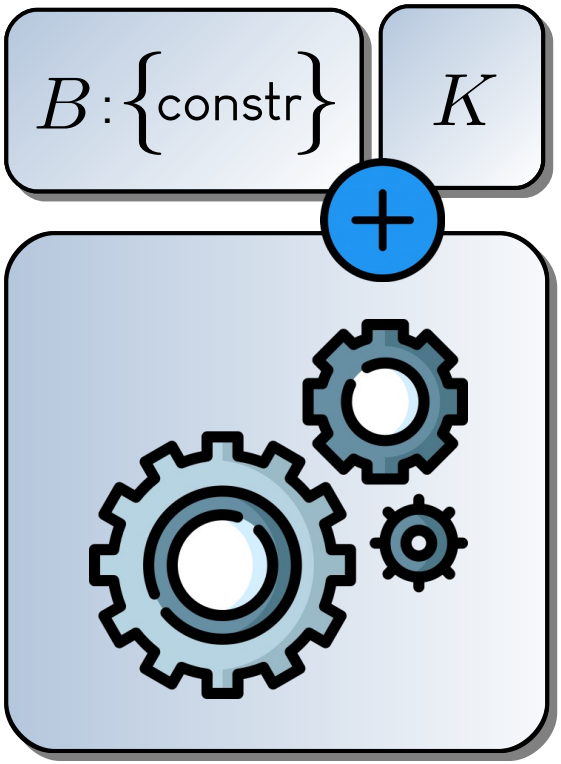
Active Constraint Acquisition

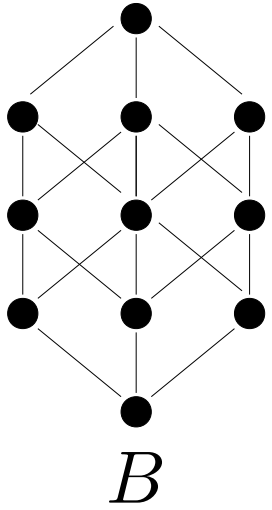
Query

Elise: 8h – 12h

Paul: 10h – 11h

...





Active Constraint Acquisition

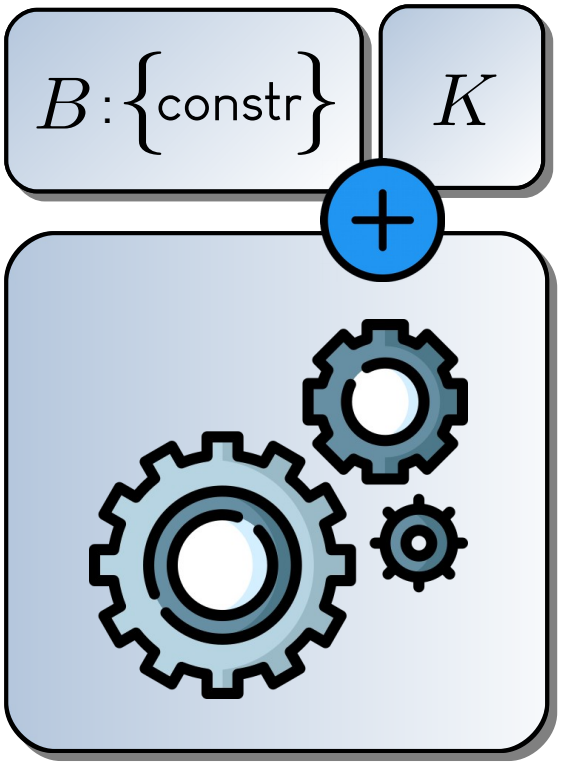
Query

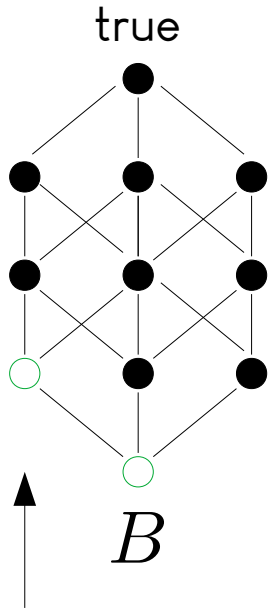
Elise: 8h – 12h

Paul: 10h – 11h

...

yes





yes: Bottom-up

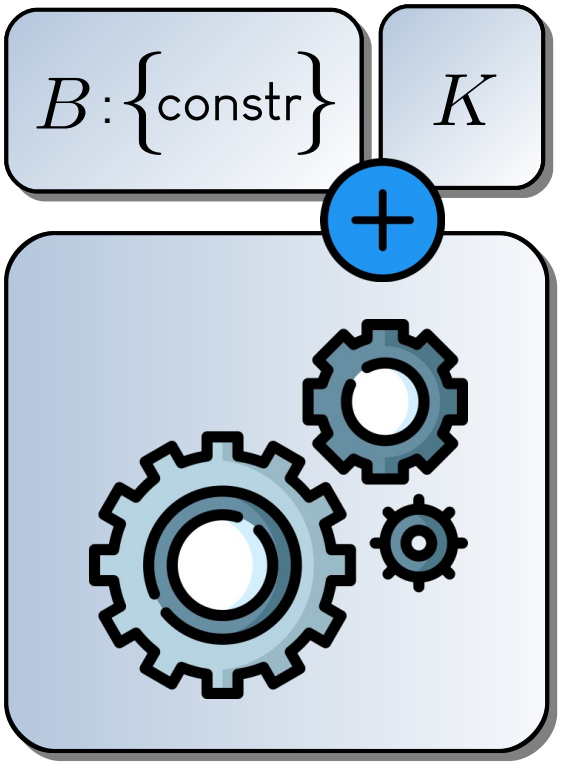
Active Constraint Acquisition

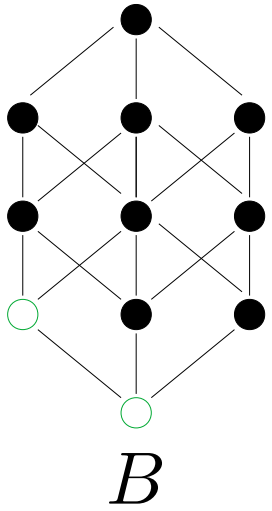
Query

Elise: 8h – 12h

Paul: 10h – 23h

...





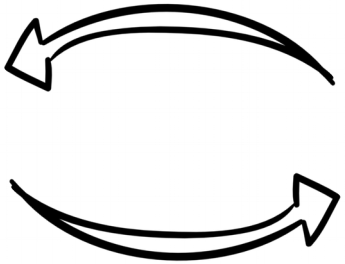
Active Constraint Acquisition

Query

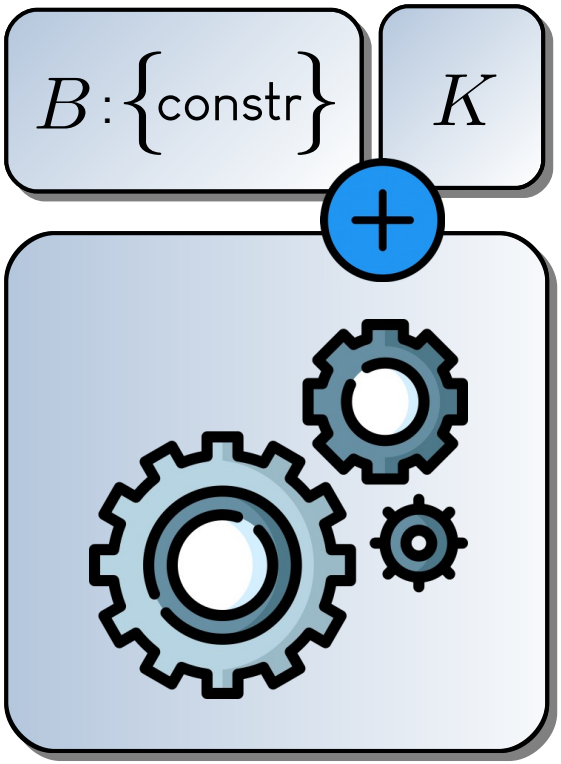
Elise: 8h – 12h

Paul: 10h – 23h

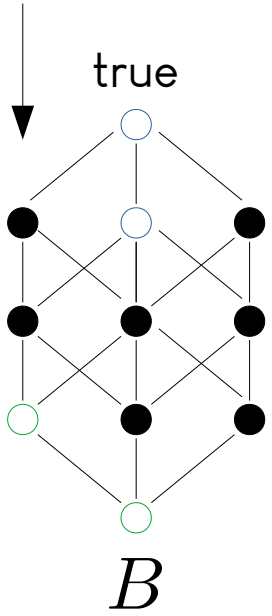
...



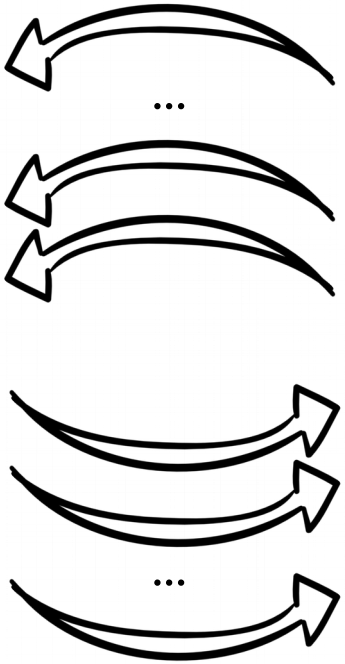
no

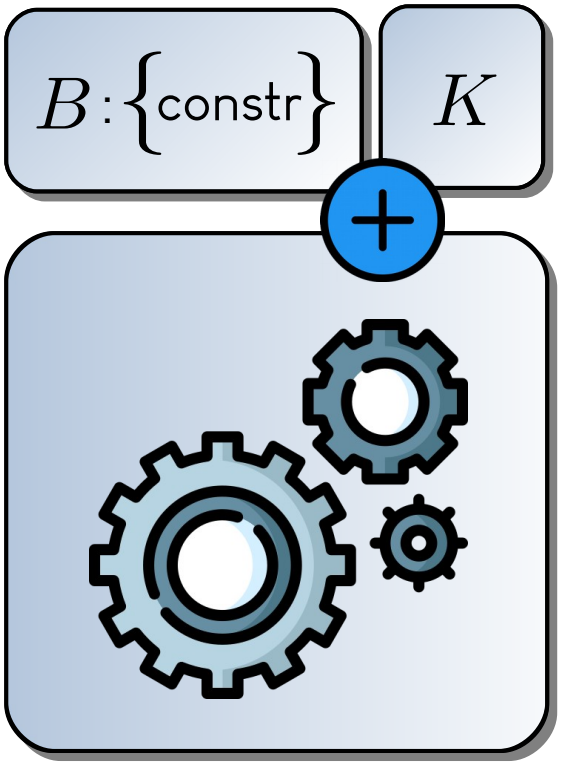


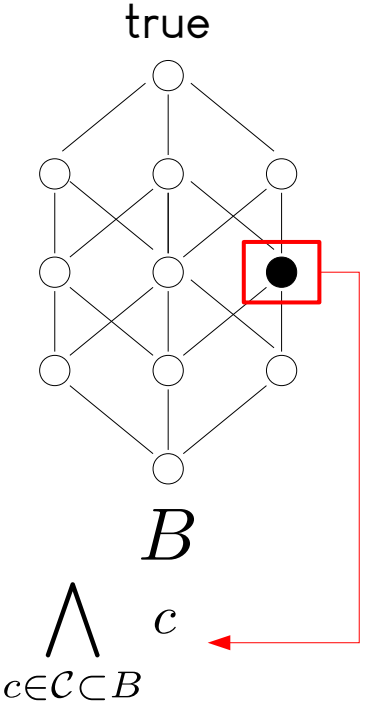
no: Top-down



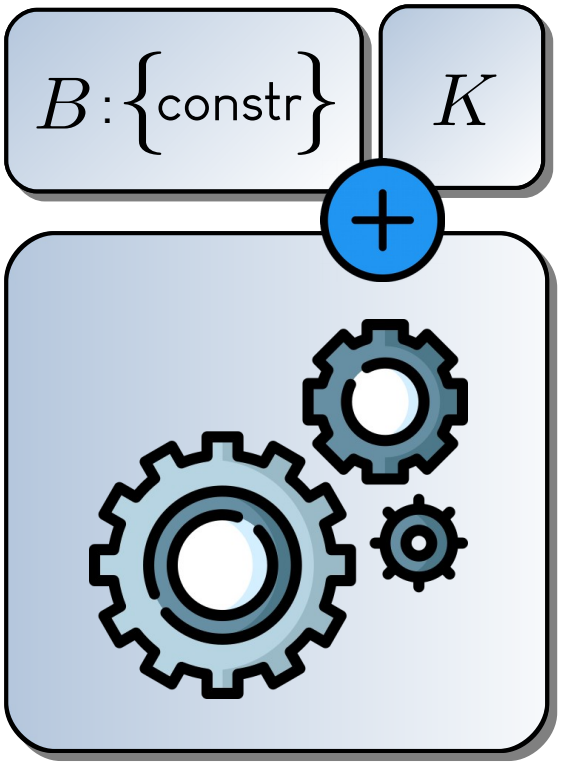
Active Constraint Acquisition

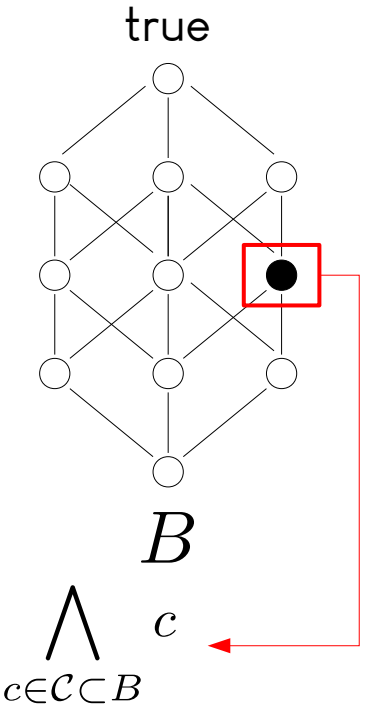






Careful: too many queries





Adapting Constraint Acquisition

Human user → 

Executable under analysis
↳ No limitation on the queries nb.

Query → 
Function inputs (args, global vars)

Constraints → B : Constraints over ptr and int

Background knowledge → K : Background knowledge on pointers


Preprocess (passive mode)
↳ Generates likely to be interesting queries

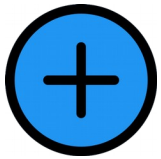
Which constraints ?

Constraints ⟶ B : Constraints over ptr and int

Constraints for memory-related precond.:

$$
\begin{array}{lll}
P & := & C \Rightarrow A \mid A \mid \neg A \\
C & := & C \wedge C \mid A \mid \neg A \\
A & := & valid(p_j) \mid alias(p_j, p_l) \mid deref(p_j, g) \\
 & \mid & i_j = 0 \mid i_j < 0 \mid i_j \leq 0 \mid i_j = i_l \mid i_j < i_l \mid i_j \leq i_l
\end{array}
$$

Method not limited to memory-related precond.



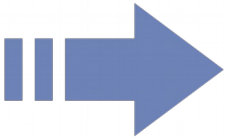
From language to bias B: max. size of horn clauses depending on the function prototype – especially number of integer inputs

Preprocess

Positive (e^+) vs Negative Queries (e^-)

↳ **All constraints** incoherent with e^+ are not in the solution 

↳ **At least one constraint** incoherent with e^- is in the solution 

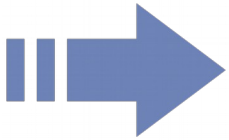


Generate positive queries first to remove a lot of constraints

↳ How to generate positive queries ?

Preprocess

- Goal of developers : software should work
  - Usually they handle well usual cases
- Generate queries where code likely behave correctly



Generate first queries with ≤ 1 one non valid, aliasing or deref pointers

PreCA

NEW

```
Call the preprocess

while true do
    Generate an informative query
    if no-query then «we converged»
    Submit query to the oracle(F, Q)
    if answer is yes then
        Bottom-up-inference()
    else
        Top-down-inference()
```

How Oracle answers queries ?

- Run function F under query
- If $ret \not\models Q$ or 

 ⇢ no
- If  

 ⇢ ukn
- Otherwise ⇢ yes

Back To Our Example

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

Description: returns the index of the first element in “a” present in “b”

Postcondition: $Q = true$

↳ Variables : a, m, b, n

↳ Heuristics : max. Horn clause size = 3

Back To Our Example

```
Fun: find_first_of
In : int* a, int m,
     int* b, int n
```

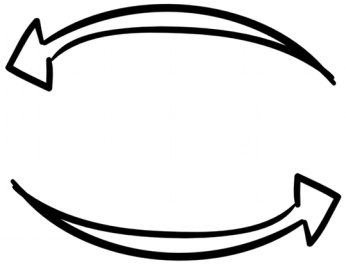
Description: returns the index of the first element in “a” present in “b”

Postcondition: $Q = true$

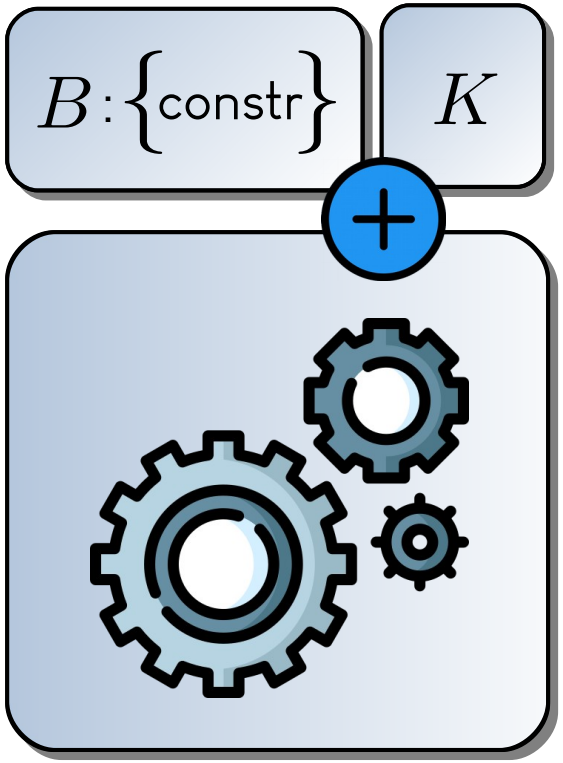
Query

a = NULL, m = 1

b = NULL, n = 3



no



Back To Our Example

```
Fun: find_first_of
In : int* a, int m,
     int* b, int n
```

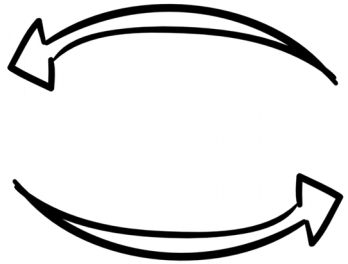
Description: returns the index of the first element in “a” present in “b”

Postcondition: $Q = true$

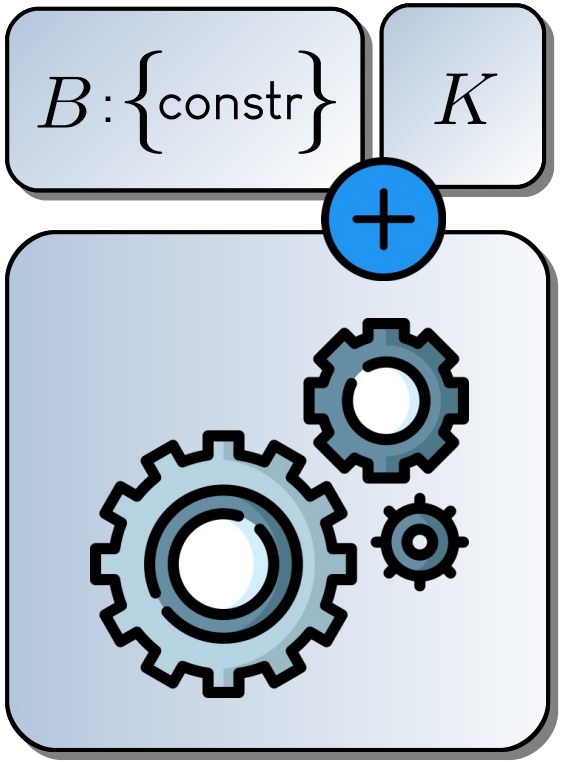
Query

a = NULL, m = 0

b = NULL, n = 0



yes



Back To Our Example

```
Fun: find_first_of
In : int* a, int m,
     int* b, int n
```

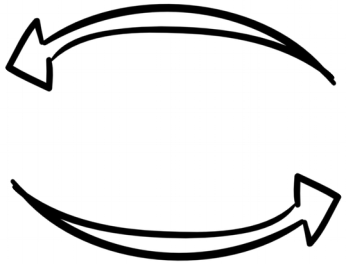
Description: returns the index of the first element in “a” present in “b”

Postcondition: $Q = true$

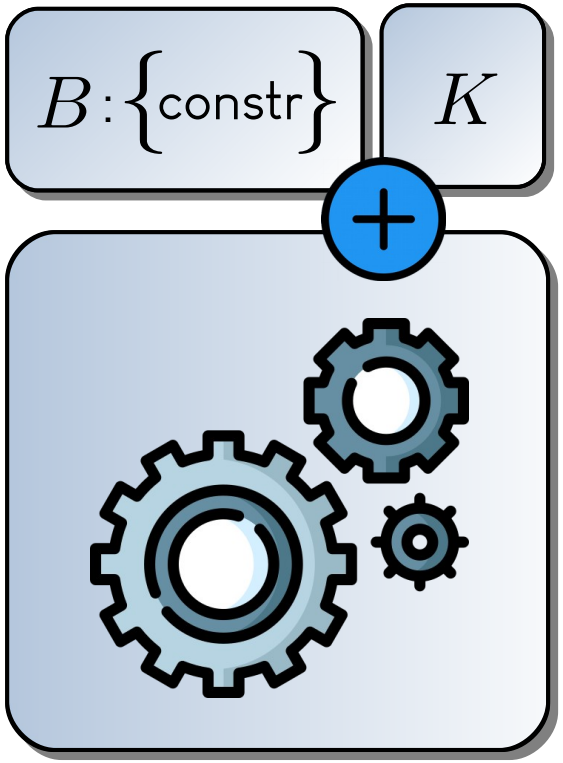
Query

a = @1, m = 3

b = @2, n = 3



yes



Back To Our Example

```
Fun: find_first_of
In : int* a, int m,
     int* b, int n
```

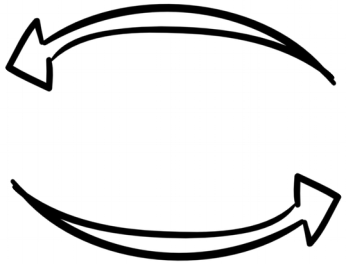
Description: returns the index of the first element in “a” present in “b”

Postcondition: $Q = true$

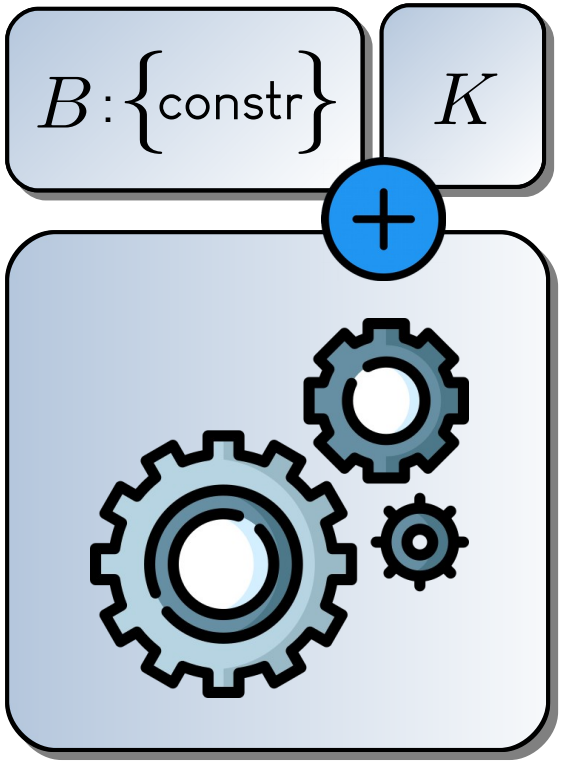
Query

a = @1, m = 3

b = @1, n = 3



yes



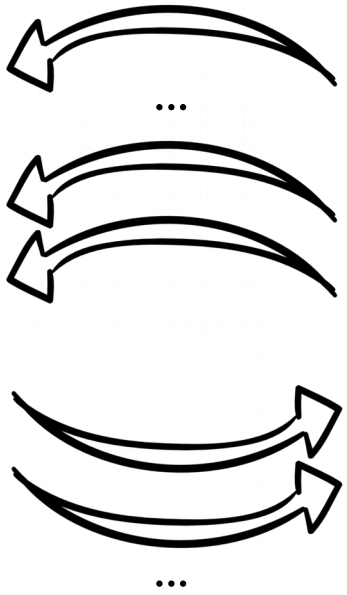
Back To Our Example

```
Fun: find_first_of
In : int* a, int m,
     int* b, int n
```

Description: returns the index of the first element in “a” present in “b”

Postcondition: $Q = true$

45 queries



Result

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

Theoretical Analysis

PreCA guarantees

↳ If B is expressive enough ⇢ ∅ or Precond.

↳ ⊕ If oracle never answers “unk” ⇢ 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:

1 hour

	PreCA		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

AI contributions

- 1st adaptation of CA for prog. analysis
  - new use case for CA
  - no user (no limit for queries nb)
- Translate core concepts :
  - Set of constraints
  - Background knowledge
- Extend CA (ukn, preprocess)

Opens new research directions for CA

Prog. analysis contribs

New efficient precond. inference tool

- Good guarantees
- Outperforms concurrent tools

+

- Does not need the source code

Thank you for your attention

@grmenguy

https://gregoiremenguy.github.io/