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

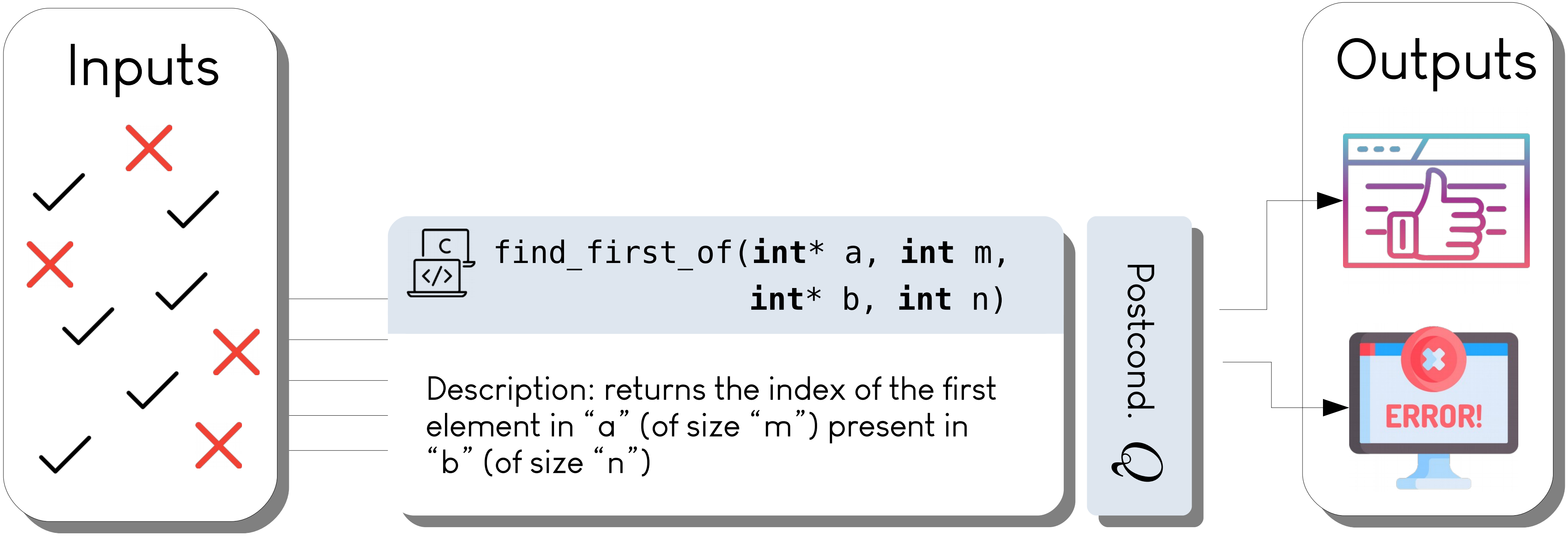
IJCAI ECAI VIENNA 22

On the Way to Secure Code

Improve Confidence in Software

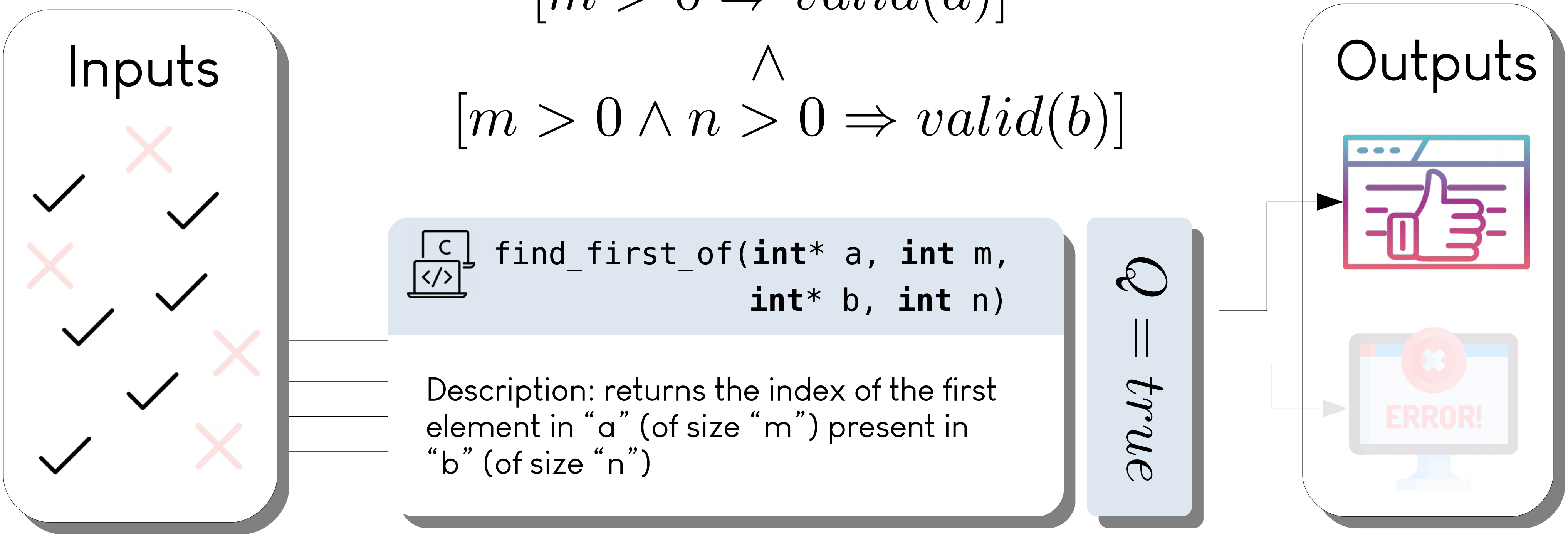
- Testing
- Formal Verification
  - E.g., Precondition / postconditon
    - Enable to scale to big code
    - Almost never given in practice

Dream: Infer Preconditions



Dream: Infer Preconditions

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



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

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

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

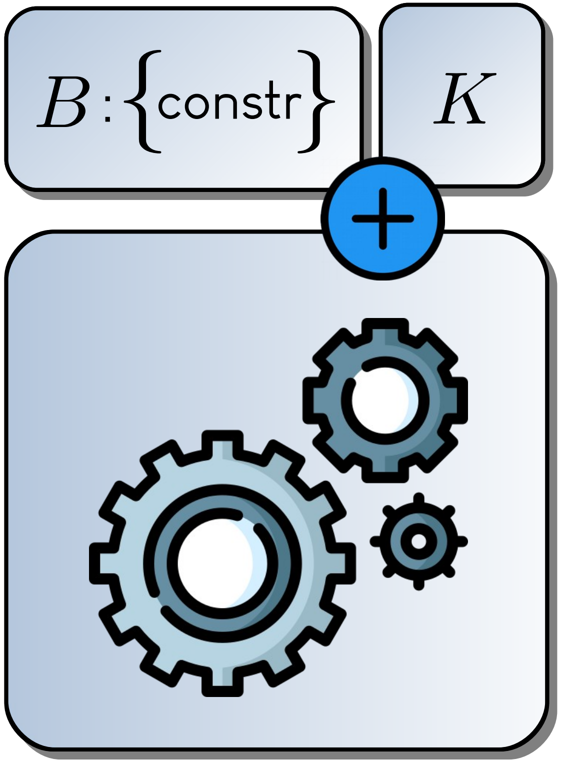
↳ Hard to design models

Constraint Acquisition

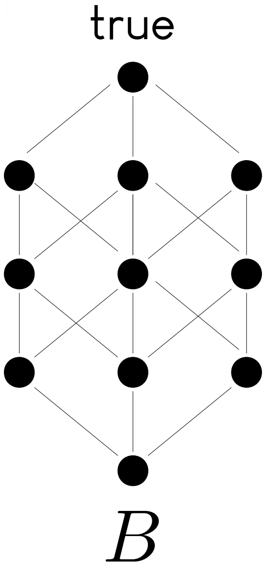
↳ Version Space Learning (Mitchell, 82)

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

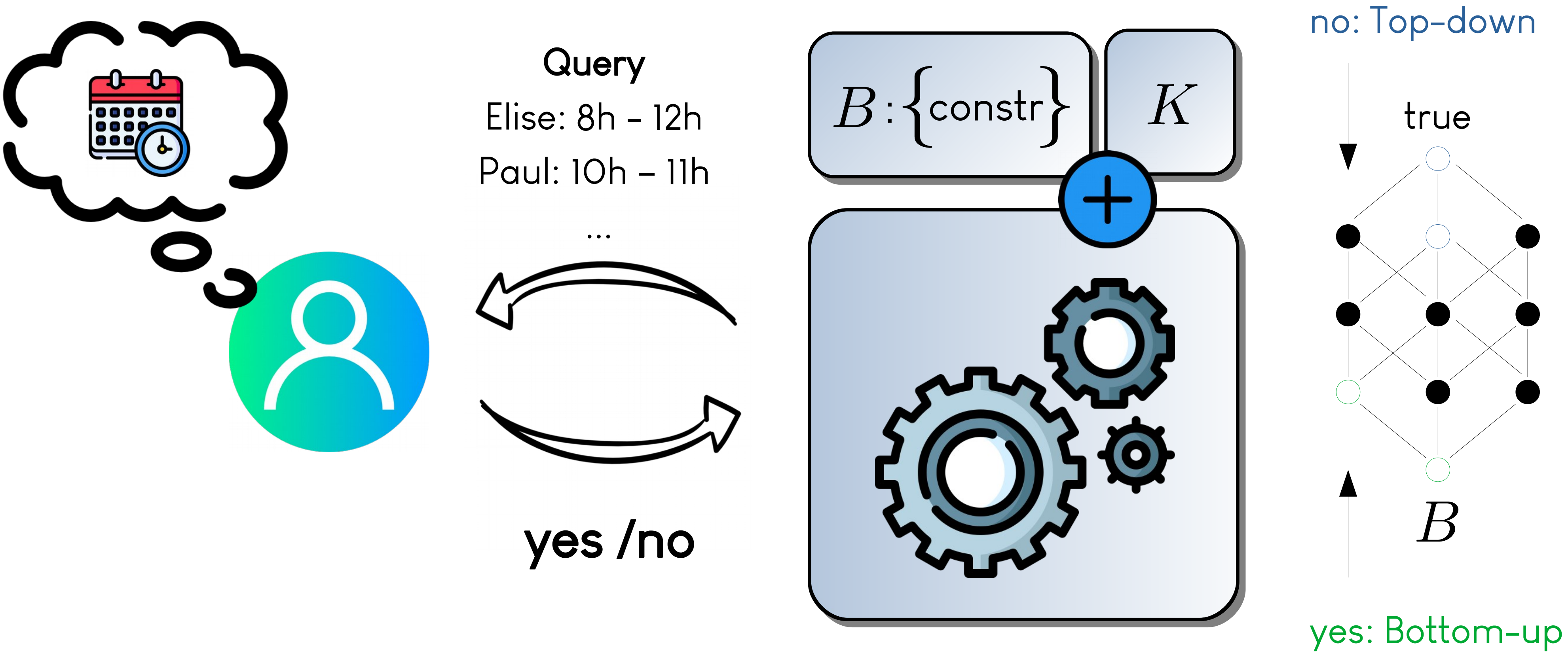
Active Constraint Acquisition



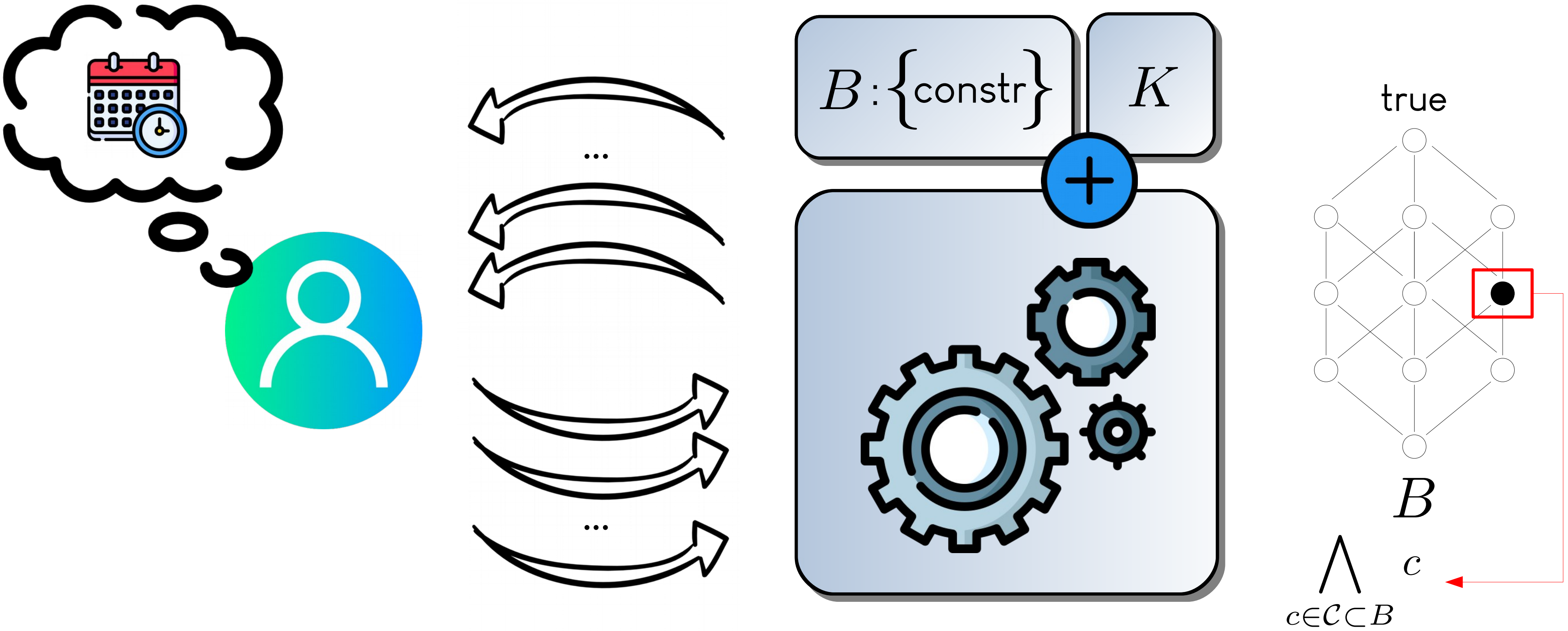
Background knowledge:
rules to speed up learning



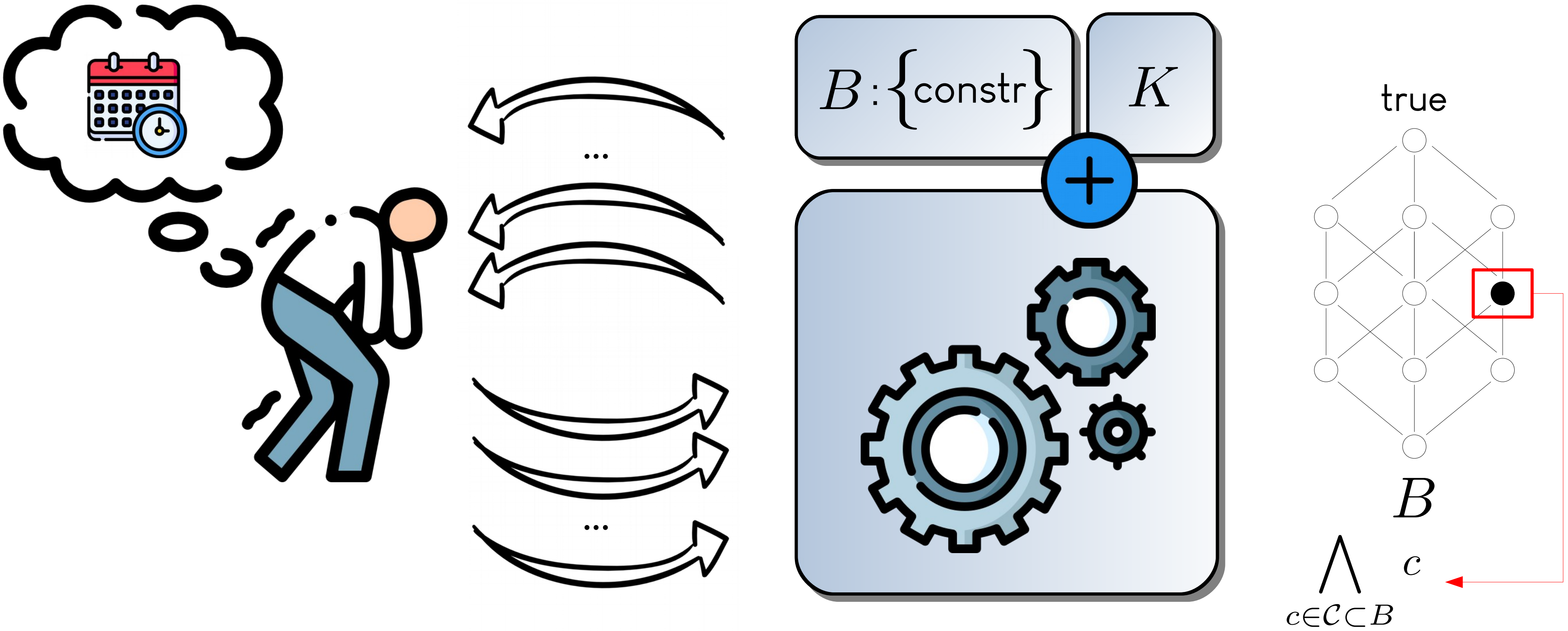
Active Constraint Acquisition



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 random queries

Adapting Constraint Acquisition

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.

Background knowledge → K : Background knowledge on pointers

e.g., $valid(ptr_1) \wedge alias(ptr_1, ptr_2) \Rightarrow valid(ptr_2)$

PreCA

```
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

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

Come see our poster: stand 152, row 5

@grmenguy

https://gregoiremenguy.github.io/