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
On the Way to Secure Code

Improve Confidence in Software

- Testing
- Formal Verification
  - E.g., Precondition / postcondition
    - Enable to scale to big code
    - Almost never given in practice
Dream: Infer Preconditions

**Inputs**

- [ ]
- [x]
- [ ]
- [x]
- [ ]
- [x]
- [ ]
- [x]
- [ ]
- [x]
- [ ]

**Outputs**

- [ ]
- [x]
- [ ]
- [ ]
- [ ]

```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”).
**Dream: Infer Preconditions**

\[
\begin{align*}
&m > 0 \Rightarrow valid(a) \\
\land \\
&m > 0 \land n > 0 \Rightarrow valid(b)
\end{align*}
\]

**Inputs**
- ✔️
- ✗
- ✔️
- ✗
- ✔️
- ✗
- ✔️
- ✗
- ✔️
- ✗

**Outputs**
- ✔️
- ✗

**Q = true**

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

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
Goal

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

👍 Does not need the source code
👍 Clear guarantees

Code Based:

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

Constraint Acquisition
Based Precond. Inference

Data-Driven Precondition Inference with Learned Features

Sarwat Pudhi
Univ. of California, Los Angeles, USA
pudhi@cs.ucla.edu

Rahul Sharma
Stanford University, USA
sharma@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)
Active Constraint Acquisition

Background knowledge: rules to speed up learning

\[ B : \{ \text{constr} \} \]

true

\[ B \]
Active Constraint Acquisition

Query
Elise: 8h - 12h
Paul: 10h - 11h
...
yes / no

B : \{ constr \}
K
+

yes: Bottom-up
no: Top-down

true
B

IJCAI-ECAI 2022 – G. Menguy et al.
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 $\rightarrow B : \text{Constraints over ptr and int}$

Constraints for memory-related precond.:

\[
\begin{align*}
P & := C \Rightarrow A \mid A \mid \neg A \\
C & := C \land C \mid A \mid \neg A \\
A & := \text{valid}(p_j) \mid \text{alias}(p_j, p_i) \mid \text{deref}(p_j, g) \\
& \mid i_j = 0 \mid i_j < 0 \mid i_j \leq 0 \mid i_j = i_l \mid i_j < i_l \mid i_j \leq i_l
\end{align*}
\]

Method not limited to memory-related precond.

Background knowledge $\rightarrow K : \text{Background knowledge on pointers}$

e.g., $\text{valid}(ptr_1) \land \text{alias}(ptr_1, ptr_2) \Rightarrow \text{valid}(ptr_2)$
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 $\text{ret} \neq Q$ or
- If $\text{no}$
- Otherwise $\text{unknown}$
- Otherwise $\text{yes}$
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 ...*)

<table>
<thead>
<tr>
<th><strong>Evaluation:</strong></th>
<th><strong>1 hour</strong></th>
<th><strong>PreCA</strong></th>
<th><strong>Daikon, PIE, Gehr et al</strong></th>
<th><strong>P-Gen</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>$Q = \text{true}$</td>
<td></td>
<td>92%</td>
<td>At most 52%</td>
<td>74%</td>
</tr>
<tr>
<td>$Q \neq \text{true}$</td>
<td></td>
<td>41%</td>
<td>At most 23%</td>
<td>34%</td>
</tr>
</tbody>
</table>

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/