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

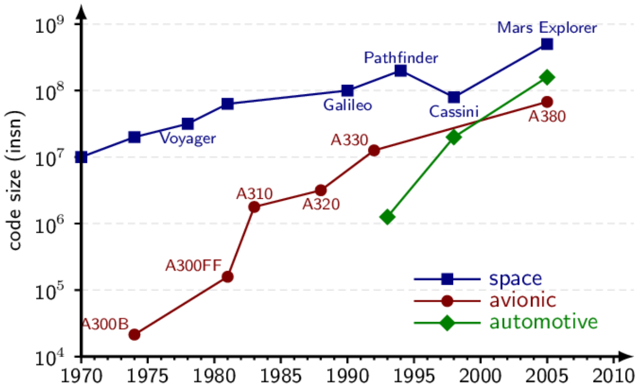
Black-box Precondition Inference through Constraint Acquisition

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

More Complex Software Everywhere

- Software in all domains

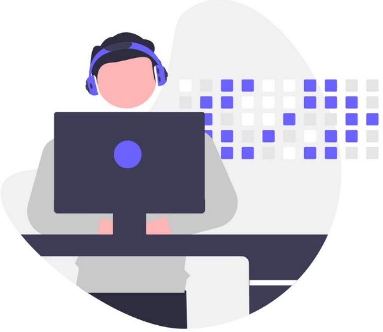
- Bigger and more complex



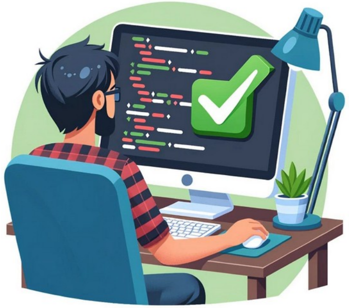
More Complex Software Everywhere

– Softw

– Bigge



Coding is hard



Checking is even harder

Secure Code by Automated Analysis

Help Analysis and Improve Confidence in Software

- Testing

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

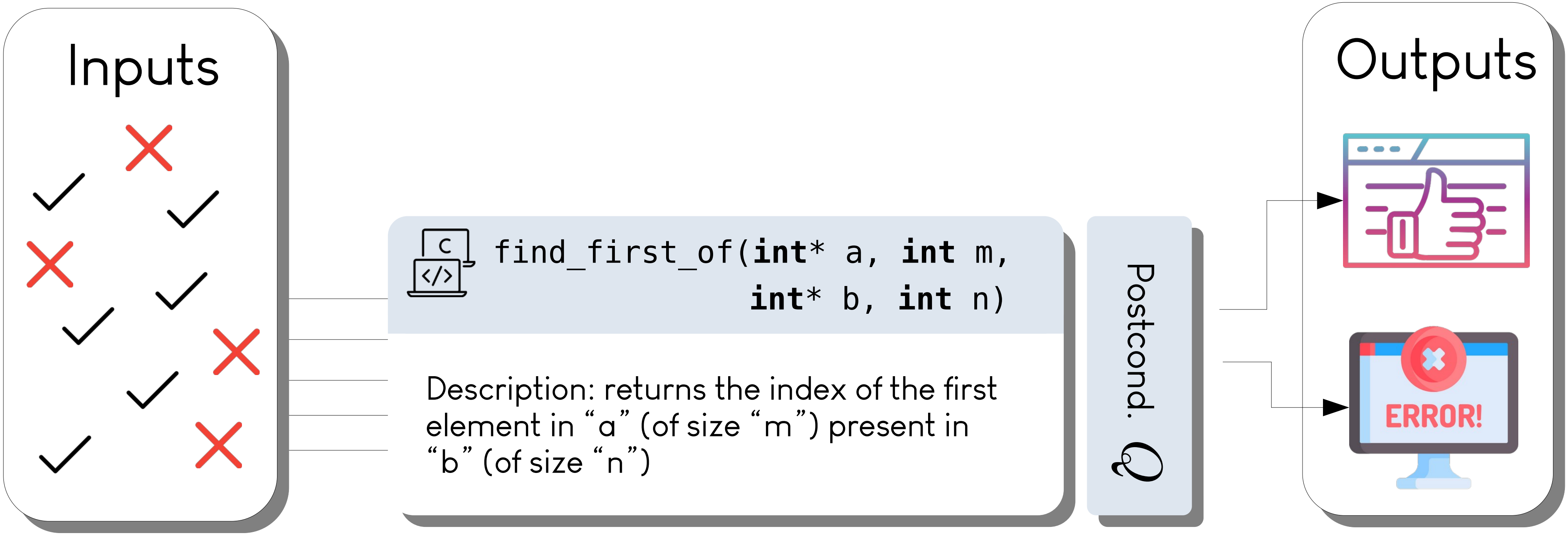
frama C Software Analyzers

ASTRÉE

BINSEC

CPA

Dream: Infer Preconditions



Dream: Infer Preconditions

Inputs

$$valid(a) \land valid(b)$$

```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”)

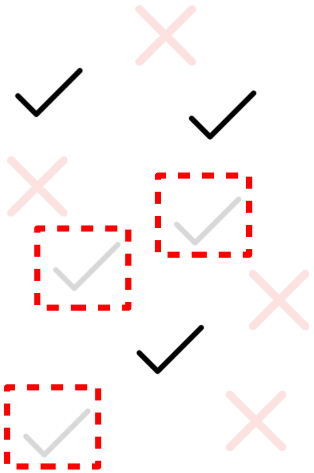
$Q = true$

Outputs

Undecidable problem: Rice theorem (1953)

Dream: Infer Preconditions

Inputs



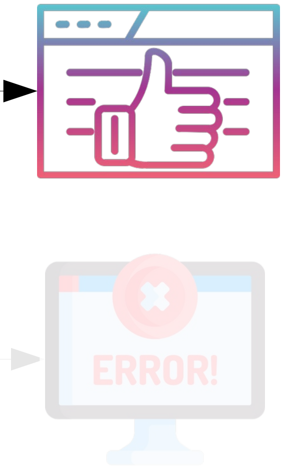
$$valid(a) \land valid(b)$$

```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)

Dream: Infer The Weakest Precond.

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

Inputs

```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)

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.

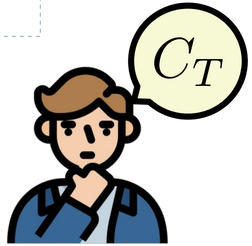
Conacq: Active Constraint Acquisition

Hypothesis:
The target concept $C_T \subset B$

$B = \{c_1, ..., c_k\}$

Conjunctions of constraints from B

Oracle

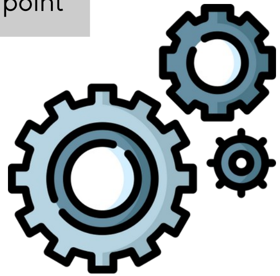


stakeholder

query

yes / no

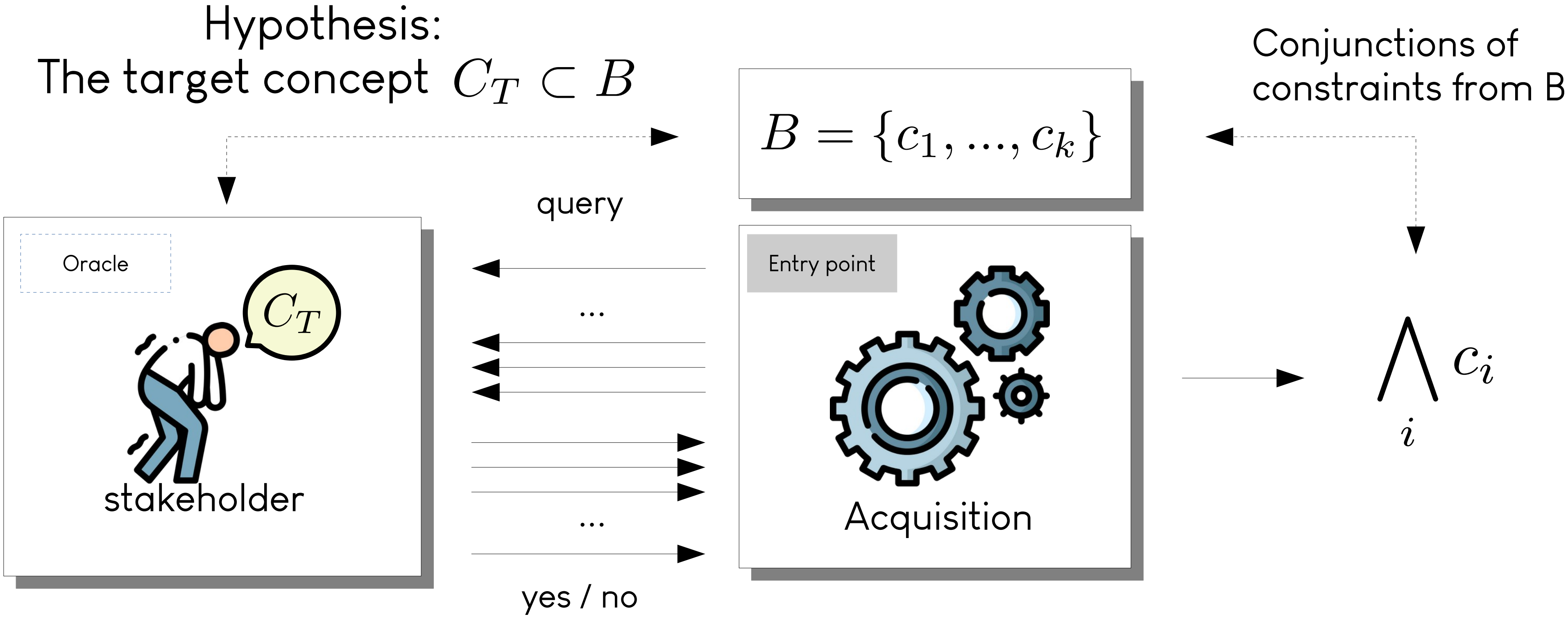
Entry point



Acquisition

$\bigwedge_i c_i$

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

Preprocess (passive mode)

↳ Generates likely to be interesting queries

The Constraint Language

New constraints

Overlap(str1, str2)

strlen(str) > 0

Constraints for memory-related precond.:

Method not limited to memory-related precond.

$$
\begin{array}{lll}
C & := & C \vee C \mid A \mid \neg A \\
A & := & valid(p) \mid alias(p, q) \mid deref(p, g) \\
 & \mid & i = 0 \mid i < 0 \mid i \leq 0 \mid i = j \mid i < j \mid i \leq j
\end{array}
$$

$$valid(p) \equiv p \neq NULL$$

$$alias(p, q) \equiv p = q$$

$$deref(p, g) \equiv p = \&g$$

p, q : pointer variables

i, j : integer (signed and unsigned) variables

g : global variables (of any type)

The Bias

Constraints for memory-related precond.:

$$
\begin{array}{lcl}
C & := & C \vee C \mid A \mid \neg A \\
A & := & valid(p) \mid alias(p,q) \mid deref(p,g) \\
 & \mid & i = 0 \mid i < 0 \mid i \leq 0 \mid i = j \mid i < j \mid i \leq j
\end{array}
$$

Method not limited to memory-related precond.

From language to bias B:

↳ The bias is a finite set → Which disjunctions to include ?

↳ Proposal: max. size of disjunctions depending on the function prototype – especially number of integer inputs

Preprocess

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

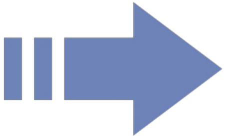
- **All constraints** incoherent with e^+ are not in the solution
- **At least one constraint** incoherent with e^- is in the solution

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

Theoretical Analysis

PreCA guarantees

↳ If B is expressive enough → ∅ or Precond.

↳ ⊕ If oracle never answers "unk" → Weakest precond.

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

Problem

What happens if horn clauses heuristics fails?

B is not expressive enough

~~If B is expressive enough~~ $\rightarrow$ ~~$\varnothing$ or Precond.~~

~~$\oplus$ If oracle never answers “unk”~~ $\rightarrow$ ~~Weakest precond.~~

Better handling of disjunctions

C_T is a conjunctions of constraints from B, i.e.,

$$C_T \subset B$$

$\longrightarrow$

C_T is a conjunction of disjunctions of B's constraints, i.e.,

$$C_T = \{d_i\}_{i \in I}$$

$$d_i = \bigvee_j a_j \quad \text{with} \quad a_j \in B$$

⇒ We say that C_T is $\bigvee$-representable by B

Disjunctive Constraint Acquisition

NEW

Key points :

- MSSes induce a partition
- Check classification of one element per MSS

Deduce the classification of all the domain

Algorithm 1: DCA

In : A nonempty complete bias B
Out : A conjunction of disjunctions of constraints from B

begin
  $L \leftarrow \top$
  foreach $M \in \textsc{Mss}_B$ **do**
    pick $e \in sol(M)$
    if $\texttt{ask}(e) \neq yes$ **then**
      $L \leftarrow L \wedge \neg M$
  return L

Theoretical analysis

Proposition : DCA generates informative queries only

Remark: Informativeness must be extended because of disjunctive behaviors. Some queries which were not informative in CA is informative now.

Theorem : If C_T is $\bigvee$-representable by B then DCA infers a concept L s.t., $L \equiv C_T$

⊕ Terminates, result agrees with all tested queries

Theoretical analysis

Proposition : DCA generates informative queries only

Remark: Informati… …iors. Some
queries which were…

Theorem : If … …A infers a
constraint ne…

DCA have the same good properties as CA

Terminates, result agrees with all tested queries

DCA for Precondition Inference

	Min bias				Avg bias				Max bias			
	1s	5s	5 mins	1h	1s	5s	5 mins	1h	1s	5s	5 mins	1h
PreCA	34/60	45/60	48/60	48/60	32/60	44/60	46/60	46/60	24/60	36/60	44/60	45/60
↳ No disj	21/60	21/60	21/60	21/60	21/60	21/60	21/60	21/60	20/60	21/60	21/60	21/60
↳ $\|disj\| \leq 2$	38/60	43/60	44/60	44/60	35/60	42/60	44/60	44/60	21/60	38/60	44/60	44/60
↳ $\|disj\| \leq 3$	30/60	44/60	48/60	48/60	26/60	43/60	46/60	46/60	18/60	31/60	42/60	44/60
↳ $\|disj\| \leq 4$	30/60	43/60	48/60	48/60	26/60	42/60	45/60	46/60	18/60	29/60	35/60	40/60
↳ $\|disj\| \leq 7$	30/60	43/60	48/60	48/60	27/60	42/60	45/60	45/60	18/60	28/60	35/60	35/60
↳ $\|disj\| \leq 10$	30/60	43/60	48/60	48/60	27/60	42/60	45/60	45/60	17/60	27/60	35/60	35/60
↳ *Omniscient*	38/60	45/60	48/60	48/60	34/60	44/60	46/60	46/60	26/60	40/60	43/60	45/60
DCA	40/60	45/60	51/60	54/60	38/60	45/60	49/60	51/60	31/60	42/60	47/60	51/60

↳ Over each bias DCA infers in 5mins more preconditions than PreCA in 1h

↳ DCA is even more efficient than $\text{PreCA}_{\text{Omniscient}}$

Results Bench 2 (precond. inference)

	Min bias				Avg bias				Max bias			
	1s	5s	5 mins	1h	1s	5s	5 mins	1h	1s	5s	5 mins	1h
PreCA	34/60	45/60	48/60	48/60	32/60	44/60	46/60	46/60	24/60	36/60	44/60	45/60
↳ No disj	21/60	21/60	21/60	21/60	21/60	21/60	21/60	21/60	20/60	21/60	21/60	21/60
↳ $\lvert disj \rvert \leq 2$	38/60	43/60	44/60	44/60	35/60	42/60	44/60	44/60	21/60	38/60	44/60	44/60
↳ $\lvert disj \rvert \leq 3$	30/60	44/60	48/60	48/60	26/60	43/60	46/60	46/60	18/60	31/60	42/60	44/60
↳ $\lvert disj \rvert \leq 4$	30/60	43/60	48/60	48/60	26/60	42/60	45/60	46/60	18/60	29/60	35/60	40/60
↳ $\lvert disj \rvert \leq 7$	30/60	43/60	48/60	48/60	27/60	42/60	45/60	45/60	18/60	28/60	35/60	35/60
↳ $\lvert disj \rvert \leq 10$	30/60	43/60	48/60	48/60	27/60	42/60	45/60	45/60	17/60	27/60	35/60	35/60
↳ *Omniscient*	38/60	45/60	48/60	48/60	34/60	44/60	46/60	46/60	26/60	40/60	43/60	45/60
DCA	40/60	45/60	51/60	54/60	38/60	45/60	49/60	51/60	31/60	42/60	47/60	51/60

↳ Over each bias DCA infers in 5mins more preconditions than PreCA in 1h

↳ DCA is even more efficient than $\text{PreCA}_{\text{Omniscient}}$

An example from MbedTLS

PreCA with Conacq cannot do it in 1h

```
int mbedtls_md_finish(mbedtls_md_context_t *ctx,
                      unsigned char *output);
```

Description: Delete structures of selected MD

Postcondition: $Q = \text{“}ret = 0\text{”}$

```
typedef enum {
    MBEDTLS_MD_NONE=0,    /**< None. */
    MBEDTLS_MD_MD5,       /**< MD5 */
    MBEDTLS_MD_SHA1,      /**< SHA-1 */
    MBEDTLS_MD_SHA224,    /**< SHA-224 */
    MBEDTLS_MD_SHA256,    /**< SHA-256 */
    MBEDTLS_MD_SHA384,    /**< SHA-384 */
    MBEDTLS_MD_SHA512,    /**< SHA-512 */
    MBEDTLS_MD_RIPEMD160, /**< RIPEMD-160 */
} mbedtls_md_type_t;
```

DCA result:

- #queries: 416
- convergence time: 6 min 41s
- Solution (simplified):

$$valid(ctx) \wedge valid(md\_info) \wedge$$
$$\neg alias(ctx, md\_info) \wedge valid(md\_ctx) \wedge$$
$$\left(\begin{array}{l} type = 1 \vee type = 2 \vee type = 3 \vee \\ type = 4 \vee type = 5 \vee type = 6 \vee type = 7 \end{array} \right)$$
$$\wedge \quad valid(output)$$

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
- Extend CA & new algorithm

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/