# Active Disjunctive Constraint Acquisition


**Grégoire Menguy, CEA LIST, France**

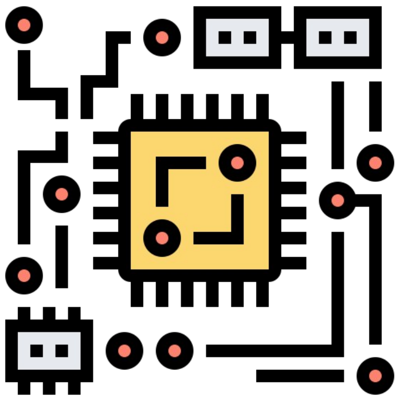
Sébastien Bardin, CEA LIST, France

Nadjib Lazaar, Unversity of Montpellier, France

Arnaud Gotlieb, Simula, Norway

# Solving problems with constraints

The problem



The problem model

$U$ is the voltage
$I$ is the current
$R$ is the resistance

$$U = R \times I$$

...

Find a solution

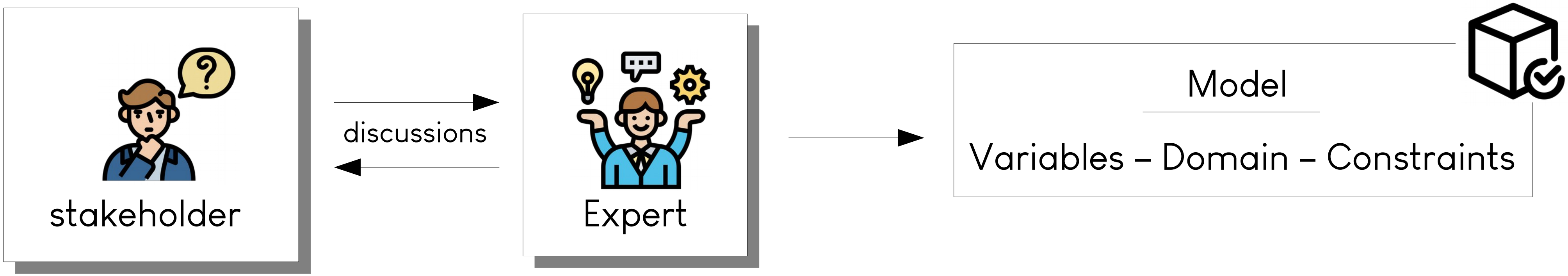
Find an optimal solution

Prove the model

...

But where does the model come from ?

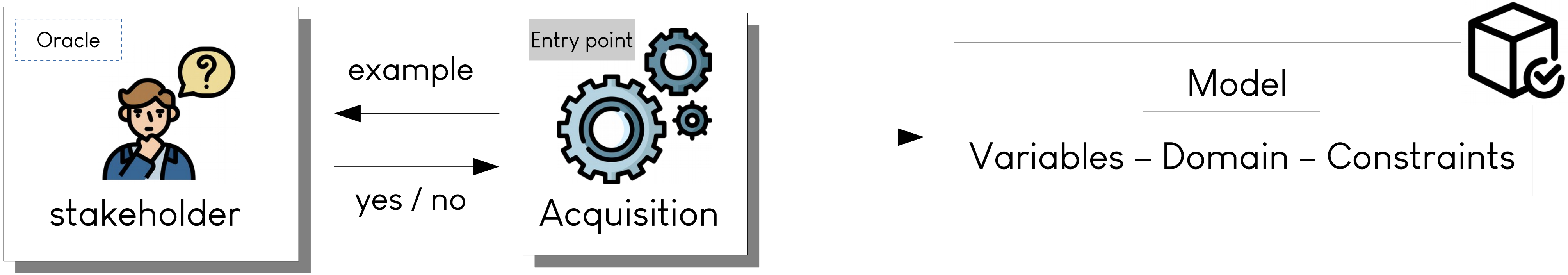
# Modeling is hard




Modeling needs expertise

How to assist the stakeholder ?

# Constraints Acquisition




Different approaches:

– Inductive Logic Programming based [Lallouet et.al., ICTAI10]

– ModelSeeker [Beldiceanu and Simonis et.al., CP12]

– **Conacq based on Version space learning [Bessiere et al. 2017]**

↳ Formal foundations ↳ Passive and active learning

Conacq guarantees:

↳ Termination

↳ Queries are informative

↳ Correctness

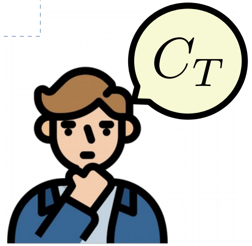
# Problem: CA is conjunctive only ...

Hypothesis:
The target concept $C_T \subset B$

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

Conjunctions of constraints from B

Oracle

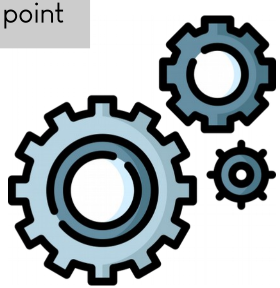



stakeholder

example

yes / no

Entry point



Acquisition

$\bigwedge_i c_i$

# But Most Problems Need Disjunctions

## For example

– Program analysis applications like Precondition acquisition (Menguy et al. 2022)

```c
int sum(int *arr, uint size) {
     int res = 0;
     for (uint i = 0; i < size; i++)
          res = res + arr[i];
     return res;
}
```

→

To execute without crashing the function input should be s.t.,

$$size = 0 \lor valid(arr)$$

– Biology application with the ultra-metric constraint (Moore and Prosser 2008)

$$(X > Y = Z) \lor (Y > X = Z) \lor (Z > X = Y) \lor (X = Y = Z)$$

– and many more

# So how to handle it?

We need
$size = 0 \vee valid(arr)$

– Rely on expert knowledge

Adds exactly the good disjunctions

But we do not have such expertise – otherwise, why using CA ?

```
int sum(int *arr, uint size) {
      int res = 0;
      for (uint i = 0; i < size; i++)
            res = res + arr[i];
      return res;
}
```

– Add all disjunctions with size up to some threshold

Which threshold ?

Combinatorial explosion leads to the bias size explosion

# So how to handle it?

– Rely on exp

Adds ex

But we do n
why using C

– Add all disj

Which t

Combin

```
ze) {
< size; i++)
r[i];
```

## Our key ingredients

– Maximally Satisfiable Subset* (MSS)

- No threshold i.e., no need for an expert
- No bias size explosion
- MSS enumeration is well established with well studied algorithms

* will be defined later

# Contributions

– First extension of CA to the disjunctive case

↳ Extension of representabilty to $\bigvee$-representability

– Show how MSS can be used for concept learning

– Propose DCA and show its guarantees

↳ Same as usual CA: Termination, informativeness of the queries, correctness

– Evaluation DCA

↳ Outperforms CA based inference (faster and handle more complex concepts)

↳ Applied on program analysis – handle complex functions like mbedtls

# New Expressiveness Hypothesis

NEW

$C_T$ 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.,

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

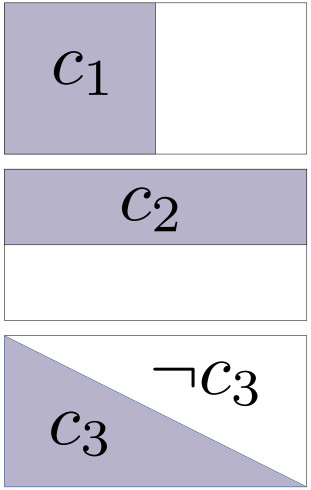
$$d_i = \bigvee_j a_j \quad \text{with} \quad a_j \in B$$

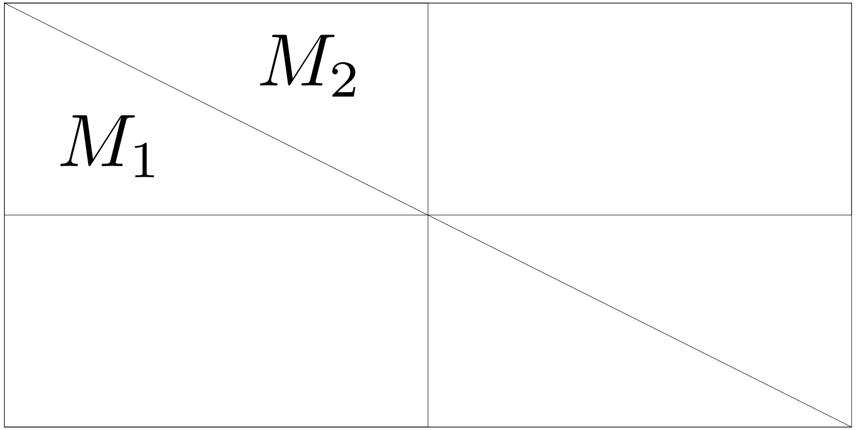
We say that $C_T$ is $\bigvee$-representable by B

# Maximal Satisfiable Set (MSS)

**Definition** : Let $B$ an unsat bias, then $M \subset B$ is an MSS of $B$ iff $M$ is sat but for all $c \in B \setminus M$ , $M \cup \{c\}$ is unsat.

**Example:**



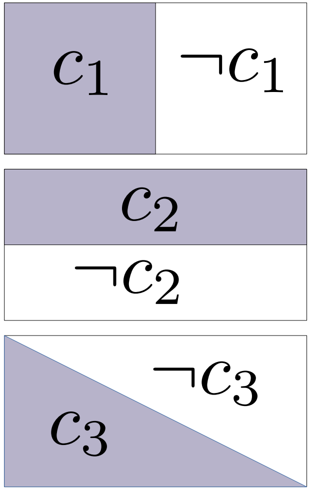




$$M_1 = \{c_1, c_2, c_3\}$$
$$M_2 = \{c_1, c_2, \neg c_3\}$$

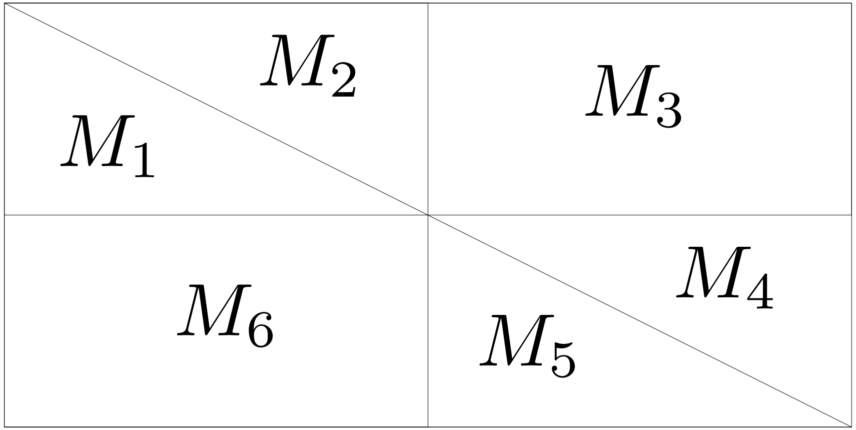
# Maximal Satisfiable Set (MSS)

**Definition** : Let $B$ an unsat bias, then $M \subset B$ is an MSS of $B$ iff $M$ is sat but for all $c \in B \setminus M$ , $M \cup \{c\}$ is unsat.

**Example:**




We have a partition of the domain




$$M_1 = \{c_1, c_2, c_3\}$$

$$M_2 = \{c_1, c_2, \neg c_3\}$$

$$\ldots$$

$$M_6 = \{\neg c_1, \neg c_2, c_3\}$$

# Maximal Satisfiable Set (MSS)

**Definition** : Let $B$ an unsat bias, then $M \subset B$ is an MSS of $B$ iff $M$ is sat but for all $c \in B \setminus M$, $M \cup \{c\}$ is unsat.

**Example:**


**Property**

Let B be a <u>complete</u> bias.

Then the set of MSS induces a partition of the domain


e domain

$c_1, c_2, c_3\}$

$c_1, c_2, \neg c_3\}$

...

$M_6 = \{\neg c_1, \neg c_2, c_3\}$

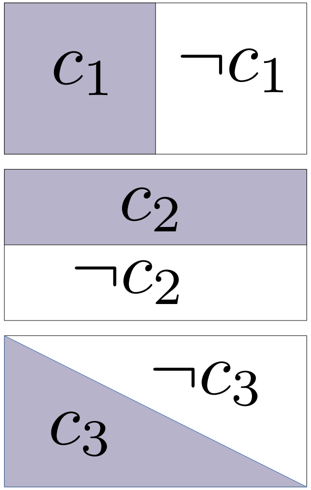
# Classification of MSS

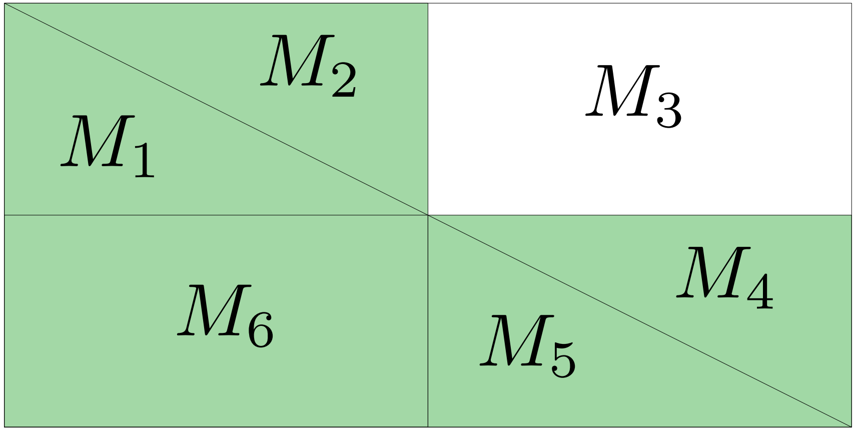
**Property** : If $C_T$ is $\bigvee$-representable by the complete bias $B$
Then <u>all the solutions of an MSS share the same classification</u>

# Classification of MSS

**Property** : If $C_T$ is $\bigvee$-representable by the complete bias $B$
Then <u>all the solutions of an MSS share the same classification</u>

**Example:** Target concept is $c_1 \vee \neg c_2$



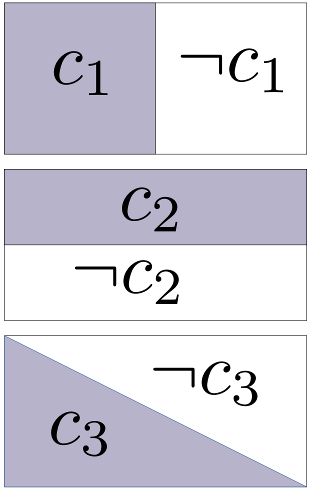



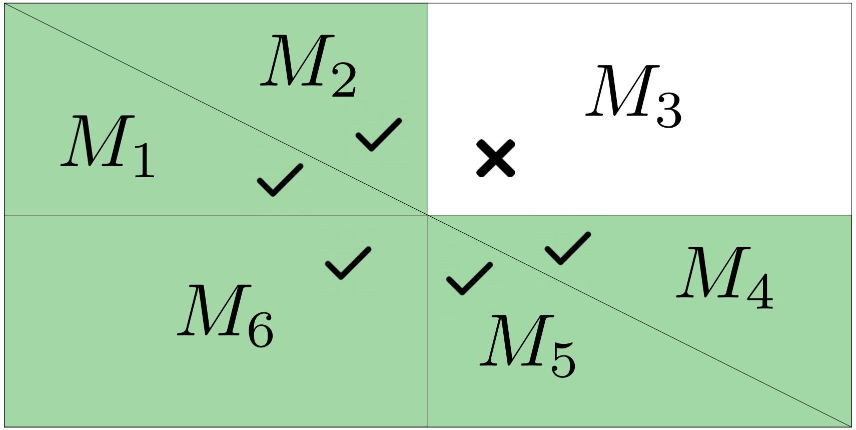

$$M_1 = \{c_1, c_2, c_3\}$$
$$M_2 = \{c_1, c_2, \neg c_3\}$$
$$\dots$$
$$M_6 = \{\neg c_1, \neg c_2, c_3\}$$

# Classification of MSS

**Property** : If $C_T$ is $\bigvee$-representable by the complete bias $B$
Then <u>all the solutions of an MSS share the same classification</u>

**Example:** Target concept is $c_1 \vee \neg c_2$




$$M_1 = \{c_1, c_2, c_3\}$$
$$M_2 = \{c_1, c_2, \neg c_3\}$$
$$\dots$$
$$M_6 = \{\neg c_1, \neg c_2, c_3\}$$

# 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 ← ⊤
    foreach M ∈ MSS_B do
        pick e ∈ sol(M)
        if ask(e) ≠ yes then
            L ← L ∧ ¬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 constraint network $L$ s.t., $L \equiv C_T$

⊕ Terminates, result agrees with all tested queries

# Theoretical analysis

**Proposition** : DCA generates informative queries only

Remark: Informativ… …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

# Implementation

↳ Extension of the Conacq.2 framework

lirmm / **ConstraintAcquisition** Public

<> **Code** ⊙ Issues ⇅ Pull requests ⊙ Actions

Constraint acquisition ☆

Christian Bessiere [a,*], Frédéric Koriche [b], Nadjib Lazaar [a], Barry O'Sullivan [c]

↳ MSS enumeration algorithm: DAA vs MARCO

**Enumerating Infeasibility: Finding Multiple MUSes Quickly**

Mark H. Liffiton and Ammar Malik

# Evaluation: Datasets

## Benchmark 1

– Random – e.g., $(X1 = X2 \vee X0 = X2) \wedge (X1 \leq X2 \vee X0 > X2) \wedge (X1 > X2 \vee X0 \geq X2)$

– Domain – e.g., $(X = 1 \wedge X_1 = 1 \wedge X_2 = 0 \wedge X_3 = 0)$
$\vee(X = 2 \wedge X_1 = 0 \wedge X_2 = 1 \wedge X_3 = 0) \vee (X = 3 \wedge X_1 = 0 \wedge X_2 = 0 \wedge X_3 = 1)$

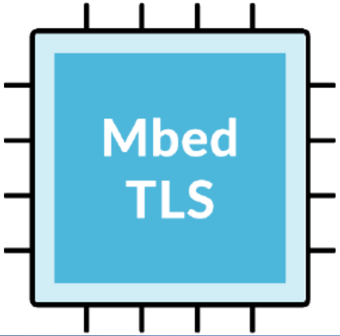
– Ultra-metric – e.g., $X > Y = Z \vee Y > X = Z \vee Z > X = Y \vee X = Y = Z$

## Benchmarck 2

**Automated Program Analysis:**
**Revisiting Precondition Inference through Constraint Acquisition**

**Grégoire Menguy**[1], **Sébastien Bardin**[1], **Nadjib Lazaar**[2] and **Arnaud Gotlieb**[3]
[1]Université Paris-Saclay, CEA, List, Palaiseau, France
[2]LIRMM, University of Montpellier, CNRS, Montpellier, France
[3]Simula Research Laboratory, Oslo, Norway
{gregoire.menguy, sebastien.bardin}@cea.fr, nadjib.lazaar@lirmm.fr, arnaud@simula.no

$\oplus$




More complex
&
More disj.

# Alternative approaches

## Benchmark 1

- Conacq: includes in B all disjunctions of size up to the maximal needed
- $\text{Conacq}_{\text{Omniscient}}$: includes only disjunction size needed

## Benchmark 2 (precondition inference)

- PreCA: as presented in the paper
- PreCA $|disj| \leq t$ : includes disj. of size up to a threshold “t”
- $\text{PreCA}_{\text{Omniscient}}$: includes only disjunction size needed

Remark: Queries are answered using an oracle (the code under analysis itself for precond. inference)

# Results Bench 1

DCA is faster for Disj > 1

DCA is not impacted by disjunction size

– But Conacq bias size explodes

DCA handles more complex cases (e.g., $UM_5$)

Notations:

- $|B|$ is the number of constraints in the bias (for Conacq it includes disjunctions but not for DCA)
- $|E|$ is the number of queries submitted

| | *Disj* | CONACQ | | | CONACQ$_{omiscient}$ | | | DCA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\|B\|$ | $\|E\|$ | Time | $\|B\|$ | $\|E\|$ | Time | $\|B\|$ | $\|E\|$ | Time |
| $RAND_{2,1}$ | 1 | 6 | 3 | **0.2s** | 6 | 3 | 0.3s | 6 | 3 | 0.2s |
| $RAND_{2,2}$ | 2 | 18 | 3 | 0.4s | 18 | 3 | 0.3s | 6 | 3 | **0.2s** |
| $RAND_{2,3}$ | 3 | 26 | 3 | 0.4s | 14 | 3 | 0.2s | 6 | 3 | **0.2s** |
| $RAND_{2,4}$ | 4 | 26 | 3 | 0.4s | 6 | 3 | 0.2s | 6 | 3 | **0.2s** |
| $RAND_{3,1}$ | 1 | 18 | 7 | **0.5s** | 18 | 6 | 0.3s | 18 | 13 | 0.9s |
| $RAND_{3,2}$ | 2 | 162 | 13 | 5s | 162 | 13 | 3.7s | 18 | 13 | **0.9s** |
| $RAND_{3,3}$ | 3 | 834 | 13 | 154s | 690 | 13 | 43s | 18 | 13 | **1s** |
| $RAND_{3,4}$ | 4 | 2850 | 13 | 817s | 2034 | 13 | 140s | 18 | 13 | **0.9s** |
| $RAND_{4,1}$ | 1 | 36 | 14 | **1s** | 36 | 15 | 1s | 36 | 75 | 38s |
| $RAND_{4,2}$ | 2 | 648 | 54 | 286s | 648 | 37 | 47s | 36 | 75 | **39s** |
| $RAND_{4,3}$ | 3 | 7176 | - | TO | 6564 | - | TO | 36 | 75 | **36s** |
| $RAND_{4,4}$ | 4 | 56136 | - | TO | 48996 | - | TO | 36 | 75 | **37s** |
| $DOM_3$ | 3 | 834 | 24 | 297s | 690 | 24 | 217s | 18 | 24 | **0.6s** |
| $DOM_4$ | 4 | 9968 | - | TO | 7944 | - | TO | 24 | 64 | **1.2s** |
| $DOM_5$ | 5 | 122026 | - | TO | 96126 | - | TO | 30 | 160 | **3.3s** |
| $DOM_6$ | 6 | - | - | TO | - | - | TO | 36 | 384 | **9.7s** |
| $DOM_7$ | 7 | - | - | ME | - | - | ME | 42 | 896 | **44s** |
| $DOM_8$ | 8 | - | - | ME | - | - | ME | 48 | 2048 | **233s** |
| $DOM_9$ | 9 | - | - | ME | - | - | ME | 54 | 4608 | **1690s** |
| $DOM_{10}$ | 10 | - | - | ME | - | - | ME | 60 | - | TO |
| $UM_3$ | 4 | 472 | 13 | 50s | 252 | 13 | 13s | 12 | 13 | **0.5s** |
| $UM_4$ | 4 | 9968 | - | TO | 7944 | - | TO | 24 | 75 | **3s** |
| $UM_5$ | 4 | 87440 | - | TO | 77560 | - | TO | 40 | 541 | **200s** |
| $UM_6$ | 4 | - | - | TO | - | - | TO | 60 | - | TO |

# Results Bench 1

- DCA is faster for Disj > 1
- DCA is not impacted by disjunction size
  - But Conacq bias size explodes
- DCA handles more complex cases (e.g., $UM_5$)

| | *Disj* | CONACQ $\|B\|$ | CONACQ $\|E\|$ | CONACQ Time | CONACQ$_{omiscient}$ $\|B\|$ | CONACQ$_{omiscient}$ $\|E\|$ | CONACQ$_{omiscient}$ Time | DCA $\|B\|$ | DCA $\|E\|$ | DCA Time |
|---|---|---|---|---|---|---|---|---|---|---|
| $RAND_{2,1}$ | 1 | 6 | 3 | **0.2s** | 6 | 3 | 0.3s | 6 | 3 | 0.2s |
| $RAND_{2,2}$ | 2 | 18 | 3 | 0.4s | 18 | 3 | 0.3s | 6 | 3 | **0.2s** |
| $RAND_{2,3}$ | 3 | 26 | 3 | 0.4s | 14 | 3 | 0.2s | 6 | 3 | **0.2s** |
| $RAND_{2,4}$ | 4 | 26 | 3 | 0.4s | 6 | 3 | 0.2s | 6 | 3 | **0.2s** |
| $RAND_{3,1}$ | 1 | 18 | 7 | **0.5s** | 18 | 6 | 0.3s | 18 | 13 | 0.9s |
| $RAND_{3,2}$ | 2 | 162 | 13 | 5s | 162 | 13 | 3.7s | 18 | 13 | **0.9s** |
| $RAND_{3,3}$ | 3 | 834 | 13 | 154s | 690 | 13 | 43s | 18 | 13 | **1s** |
| $RAND_{3,4}$ | 4 | 2850 | 13 | 817s | 2034 | 13 | 140s | 18 | 13 | **0.9s** |
| $RAND_{4,1}$ | 1 | 36 | 14 | **1s** | 36 | 15 | 1s | 36 | 75 | 38s |
| $RAND_{4,2}$ | 2 | 648 | 54 | 286s | 648 | 37 | 47s | 36 | 75 | **39s** |
| $RAND_{4,3}$ | 3 | 7176 | - | TO | 6564 | - | TO | 36 | 75 | **36s** |
| $RAND_{4,4}$ | 4 | 56136 | - | TO | 48996 | - | TO | 36 | 75 | **37s** |
| $DOM_3$ | 3 | 834 | 24 | 297s | 690 | 24 | 217s | 18 | 24 | **0.6s** |
| $DOM_4$ | 4 | 9968 | - | TO | 7944 | - | TO | 24 | 64 | **1.2s** |
| $DOM_5$ | 5 | 122026 | - | TO | 96126 | - | TO | 30 | 160 | **3.3s** |
| $DOM_6$ | 6 | - | - | TO | - | - | TO | 36 | 384 | **9.7s** |
| $DOM_7$ | 7 | - | - | ME | - | - | ME | 42 | 896 | **44s** |
| $DOM_8$ | 8 | - | - | ME | - | - | ME | 48 | 2048 | **233s** |
| $DOM_9$ | 9 | - | - | ME | - | - | ME | 54 | 4608 | **1690s** |
| $DOM_{10}$ | 10 | - | - | ME | - | - | ME | 60 | - | TO |
| $UM_3$ | 4 | 472 | 13 | 50s | 252 | 13 | 13s | 12 | 13 | **0.5s** |
| $UM_4$ | 4 | 9968 | - | TO | 7944 | - | TO | 24 | 75 | **3s** |
| $UM_5$ | 4 | 87440 | - | TO | 77560 | - | TO | 40 | 541 | **200s** |
| $UM_6$ | 4 | - | - | TO | - | - | TO | 60 | - | TO |

Notations:
- $|B|$ is the number of constraints in the bias (for Conacq it includes disjunctions but not for DCA)
- $|E|$ is the number of queries submitted

# Results Bench 1

- DCA is faster for Disj > 1
- DCA is not impacted by disjunction size
  – But Conacq bias size explodes
- DCA handles more complex cases (e.g., $UM_5$)

Notations:

- $|B|$ is the number of constraints in the bias (for Conacq it includes disjunctions but not for DCA)
- $|E|$ is the number of queries submitted

| | | CONACQ | | | CONACQ$_{omiscient}$ | | | DCA | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | *Disj* | $\|B\|$ | $\|E\|$ | Time | $\|B\|$ | $\|E\|$ | Time | $\|B\|$ | $\|E\|$ | Time |
| $RAND_{2,1}$ | 1 | 6 | 3 | **0.2s** | 6 | 3 | 0.3s | 6 | 3 | 0.2s |
| $RAND_{2,2}$ | 2 | 18 | 3 | 0.4s | 18 | 3 | 0.3s | 6 | 3 | **0.2s** |
| $RAND_{2,3}$ | 3 | 26 | 3 | 0.4s | 14 | 3 | 0.2s | 6 | 3 | **0.2s** |
| $RAND_{2,4}$ | 4 | 26 | 3 | 0.4s | 6 | 3 | 0.2s | 6 | 3 | **0.2s** |
| $RAND_{3,1}$ | 1 | 18 | 7 | **0.5s** | 18 | 6 | 0.3s | 18 | 13 | 0.9s |
| $RAND_{3,2}$ | 2 | 162 | 13 | 5s | 162 | 13 | 3.7s | 18 | 13 | **0.9s** |
| $RAND_{3,3}$ | 3 | 834 | 13 | 154s | 690 | 13 | 43s | 18 | 13 | **1s** |
| $RAND_{3,4}$ | 4 | 2850 | 13 | 817s | 2034 | 13 | 140s | 18 | 13 | **0.9s** |
| $RAND_{4,1}$ | 1 | 36 | 14 | **1s** | 36 | 15 | 1s | 36 | 75 | 38s |
| $RAND_{4,2}$ | 2 | 648 | 54 | 286s | 648 | 37 | 47s | 36 | 75 | **39s** |
| $RAND_{4,3}$ | 3 | 7176 | - | TO | 6564 | - | TO | 36 | 75 | **36s** |
| $RAND_{4,4}$ | 4 | 56136 | - | TO | 48996 | - | TO | 36 | 75 | **37s** |
| $DOM_3$ | 3 | 834 | 24 | 297s | 690 | 24 | 217s | 18 | 24 | **0.6s** |
| $DOM_4$ | 4 | 9968 | - | TO | 7944 | - | TO | 24 | 64 | **1.2s** |
| $DOM_5$ | 5 | 122026 | - | TO | 96126 | - | TO | 30 | 160 | **3.3s** |
| $DOM_6$ | 6 | - | - | TO | - | - | TO | 36 | 384 | **9.7s** |
| $DOM_7$ | 7 | - | - | ME | - | - | ME | 42 | 896 | **44s** |
| $DOM_8$ | 8 | - | - | ME | - | - | ME | 48 | 2048 | **233s** |
| $DOM_9$ | 9 | - | - | ME | - | - | ME | 54 | 4608 | **1690s** |
| $DOM_{10}$ | 10 | - | - | ME | - | - | ME | 60 | - | TO |
| $UM_3$ | 4 | 472 | 13 | 50s | 252 | 13 | 13s | 12 | 13 | **0.5s** |
| $UM_4$ | 4 | 9968 | - | TO | 7944 | - | TO | 24 | 75 | **3s** |
| $UM_5$ | 4 | 87440 | - | TO | 77560 | - | TO | 40 | 541 | **200s** |
| $UM_6$ | 4 | - | - | TO | - | - | TO | 60 | - | TO |

# 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 $\text{PreCA}_{\text{Omniscient}}$

# 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 |
| ↳ $\|disj\| \leq 2$ | 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 $\text{PreCA}_{\text{Omniscient}}$

# An example from MbedTLS

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

Description: Delete structures of selected MD

Postcondition: $Q =$ "$ret = 0$"

```c
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: 401s
- Solution (simplified):

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

# Discussion

- Why DCA is faster than Conacq ?
  - DCA only relies on SAT and CP solving
  - Conacq also relies on Pseudo boolean solving
- Including a background knowledge
  - DCA can integrate a Background knowledge as a set of MUS
  - But experiments showed no impact
- DCA returns hard to understand results
  - Needs to simplify the results

# Conclusion

- First extension of constraint acquisition to the disjunctive cases
  - New acquisition hypothesis : extension to $\bigvee$-representability
- MSS enumeration for concept learning
  - MSSes induce a partition
  - A partition share the same classification
- DCA shows promising results on real-world instances and standard academic problems
  - e.g., application to precondition inference in program analysis

**Leads to DCA**

Same guarantees than best $\bigwedge$-approches

- termination
- queries informativeness
- correctness

# End

Thank you for your attention