KU LEUVEN

Constraint Acquisition: Learning Constraint Models from Data

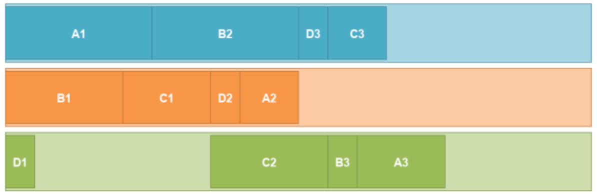
Dimos Tsouros

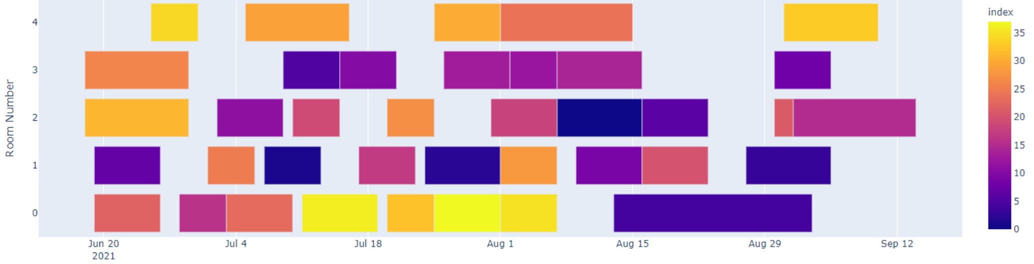
dimos.tsouros@kuleuven.be

14/07/2023

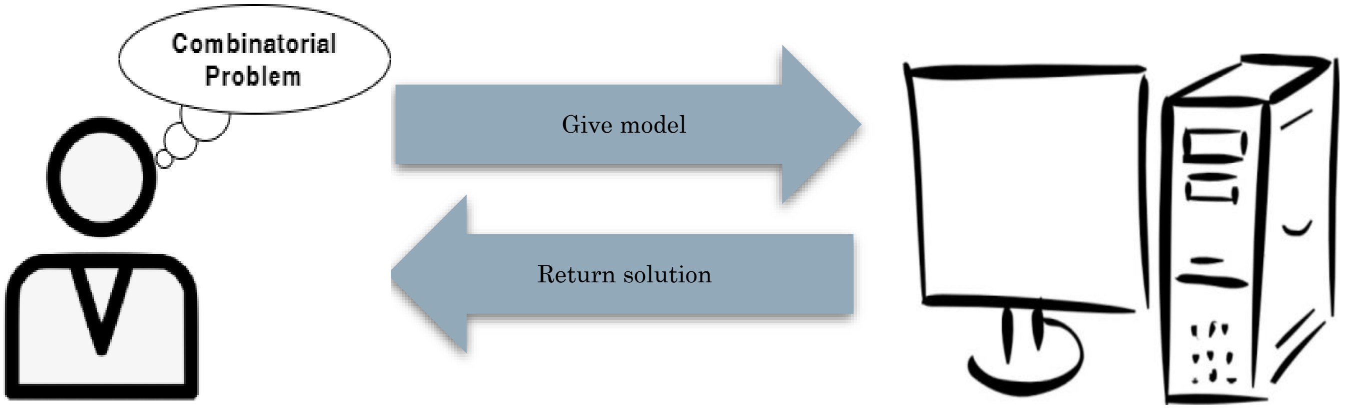
Introduction

- Constraint programming (CP)
  - Solving combinatorial problems in AI

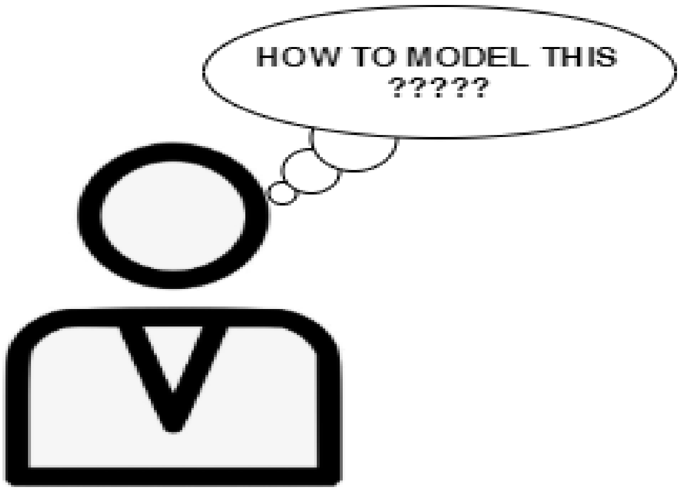




- Model + Solve paradigm



Introduction

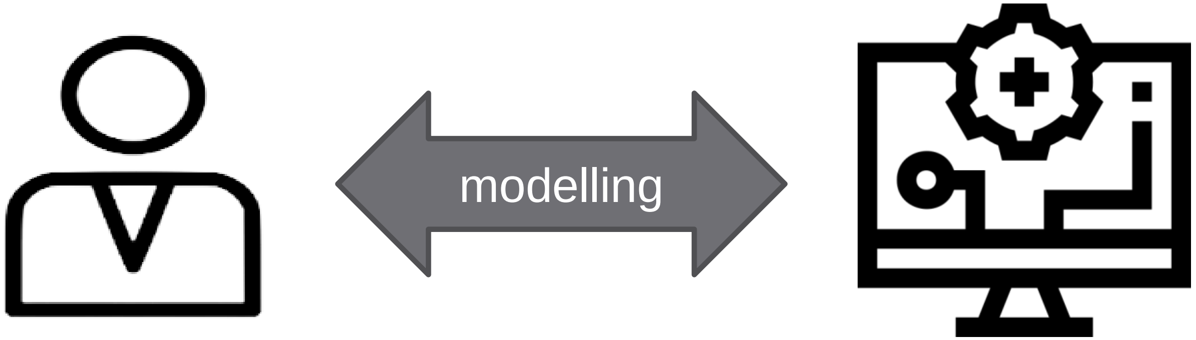


Modelling is not always trivial

- Requires expertise
- Bottleneck for the wider use of CP

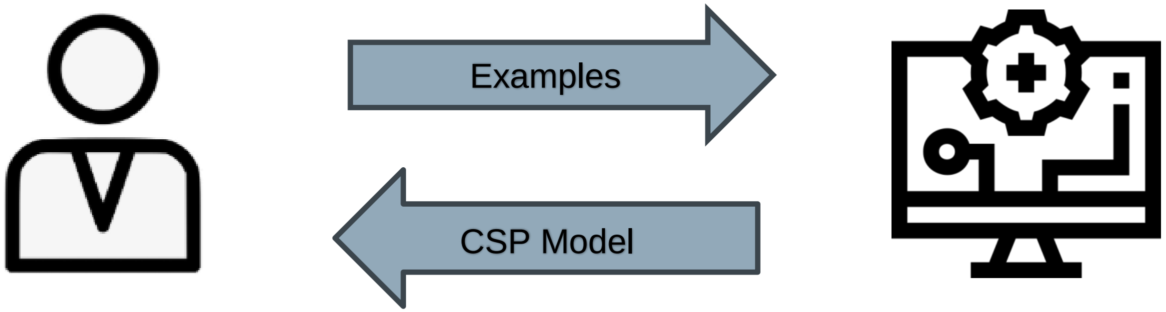
Constraint Acquisition

Structure Learning: learn the constraints of the problem, not parameters

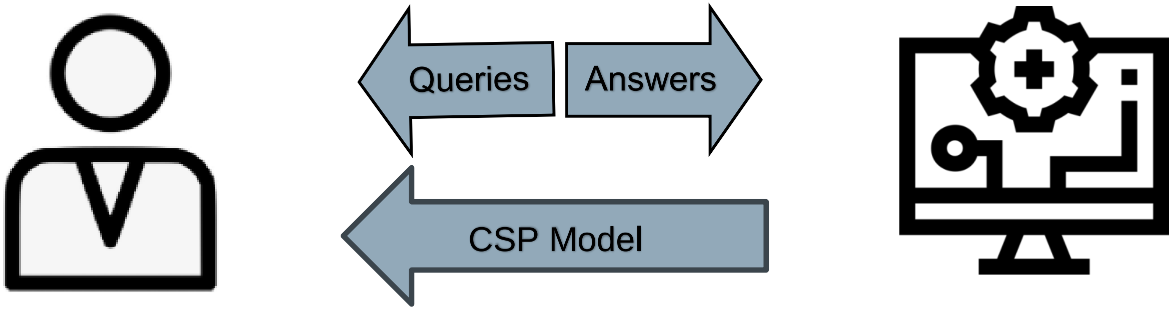


Introduction (4/4)

Passive acquisition: Using existing data

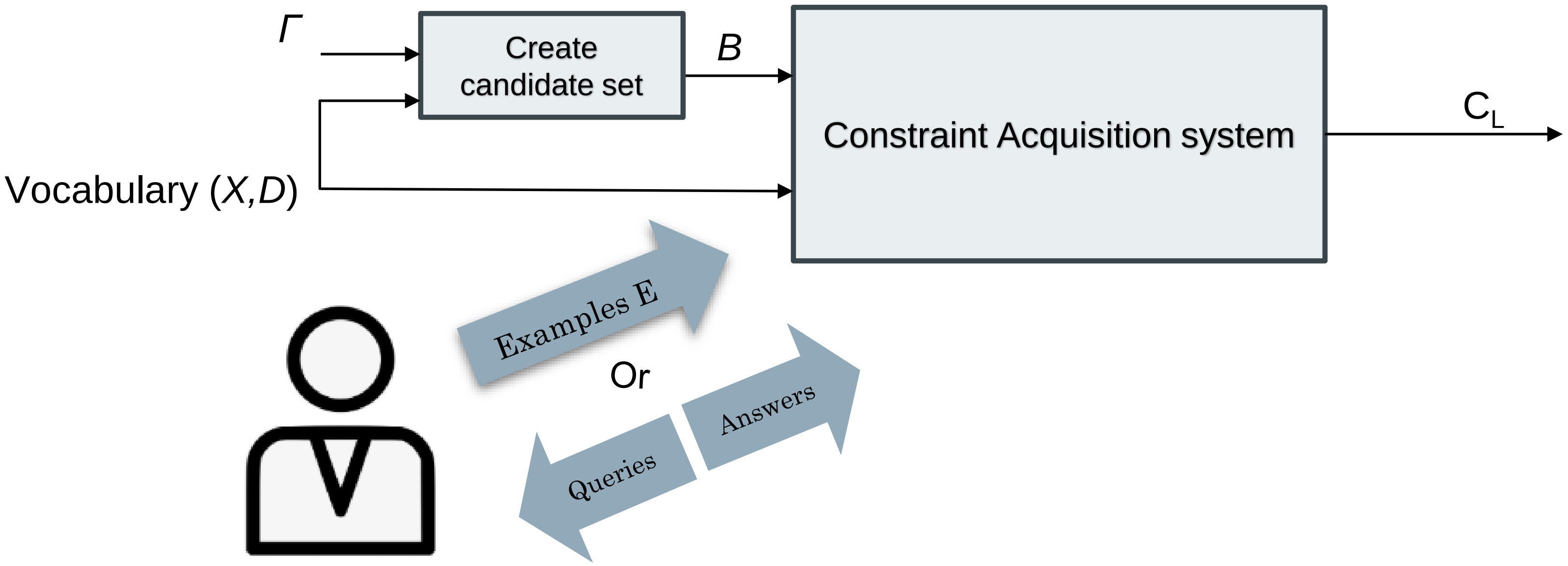


(Inter)active acquisition: Interact with the user



Constraint Acquisition, C. Bessiere et al., AIJ, 2017

Inputs and Outputs



Terminology

X: Variables, $X = \{x_1, x_2, x_3, \ldots, x_{|X|}\}$

D^X: Domains

Γ: Constraint Language, $\Gamma = \{r_1, r_2, r_3, \ldots, r_{|\Gamma|}\}$

B: Set of candidate constraints, $B = \{c_1, c_2, c_3, \ldots, c_{|B|}\}$

C_L: Learned constraint set

E: set of Examples, $E = \{e_1, e_2, e_3, \ldots, e_{|E|}\}$

Goal:
C_L is equivalent to the (unknown) target set C_T

Constraint Acquisition, C. Bessiere et al., AIJ, 2017

Inputs and Outputs

Running Example: *4x4* Sudoku

X: Variables of the problem, $X = \{x_{1,1}, x_{1,2}, x_{1,3}, \ldots, x_{4,4}\}$

D^X: Domains of the variables, $D^{X_i} = \{1, 2, 3, 4\}$

Γ: Constraint Language, $\Gamma = \{=, \neq, \leq, \geq\}$

The user may not know what constraint relations appear in the model

B: Set of candidate constraints, $B = \{x_{1,1} \neq x_{1,2}, x_{1,1} = x_{1,2}, \ldots\}$

C$_L$: learned constraints**,** ideally all the ≠ *constraints in rows, cols and blocks*

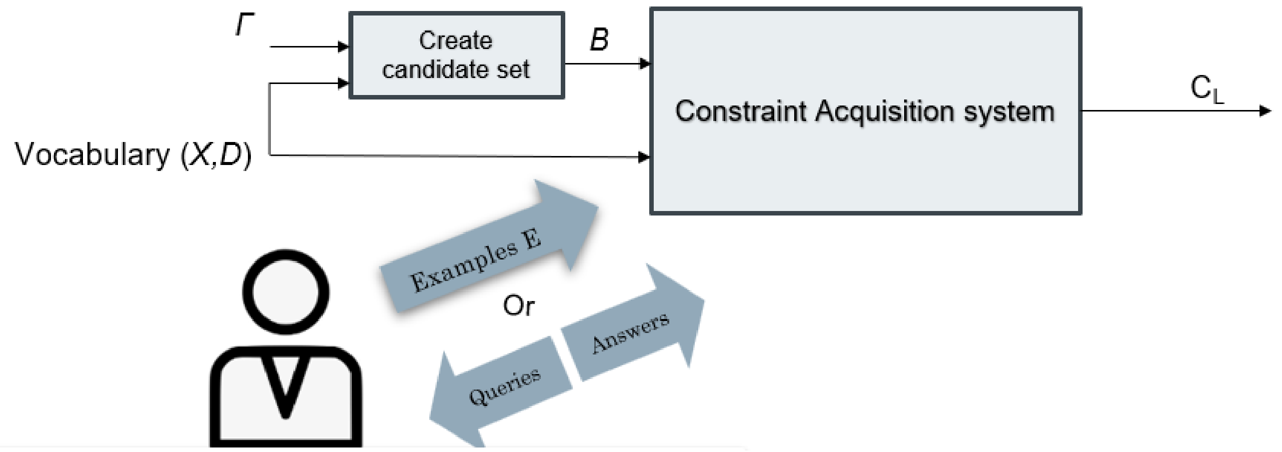
E: set of Examples, $E = \{e_1, e_2, e_3, \ldots, e_{|E|}\}$

Problem Formulation

$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{1,4}$
$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{2,4}$
$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{3,4}$
$X_{4,1}$	$X_{4,2}$	$X_{4,3}$	$X_{4,4}$

Example e_1

1	1	3	4
3	2	1	1
2	2	3	1
2	3	4	3



Inputs and Outputs

Alternative: Model with global constraints

X: Variables of the problem, $X = \{x_{1,1}, x_{1,2}, x_{1,3}, \ldots, x_{4,4}\}$

***D*^{X}**: Domains of the variables, $D^{X_i} = \{1, 2, 3, 4\}$

Γ: Constraint Language, $\Gamma = \{AllDifferent\}$

B: Set of candidate constraints, $B = \{AllDifferent(x_{1,1}, x_{1,2}, \ldots), AllDifferent(x_{1,2}, x_{3,5}, \ldots), \}$

***C*$_L$**: learned constraints, ideally all the *AllDifferent constraints in rows, cols and blocks*

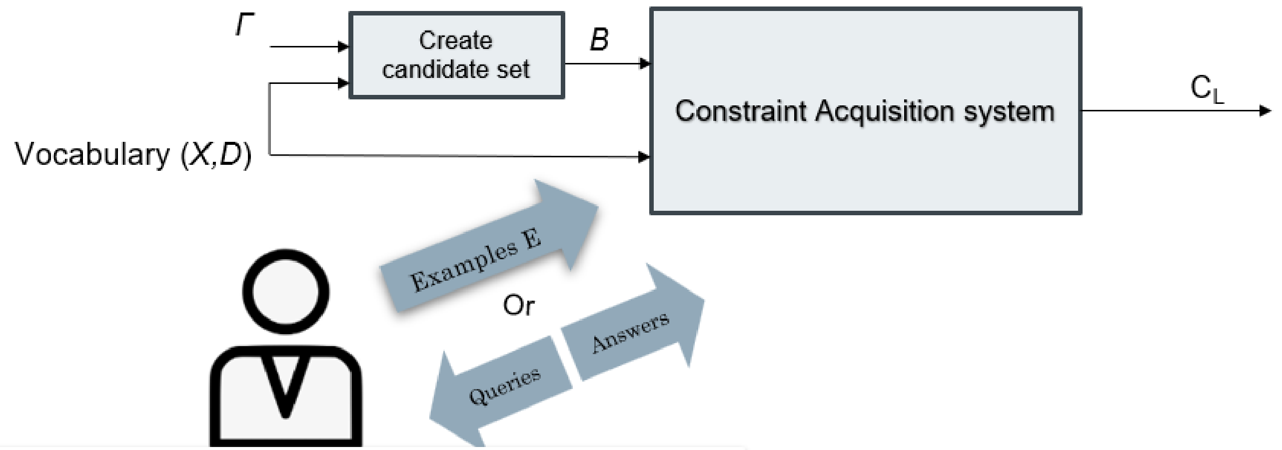
E: set of Examples, $E = \{e_1, e_2, e_3, \ldots, e_{|E|}\}$

Problem Formulation

$x_{1,1}$	$x_{1,2}$	$x_{1,3}$	$x_{1,4}$
$x_{2,1}$	$x_{2,2}$	$x_{2,3}$	$x_{2,4}$
$x_{3,1}$	$x_{3,2}$	$x_{3,3}$	$x_{3,4}$
$x_{4,1}$	$x_{4,2}$	$x_{4,3}$	$x_{4,4}$

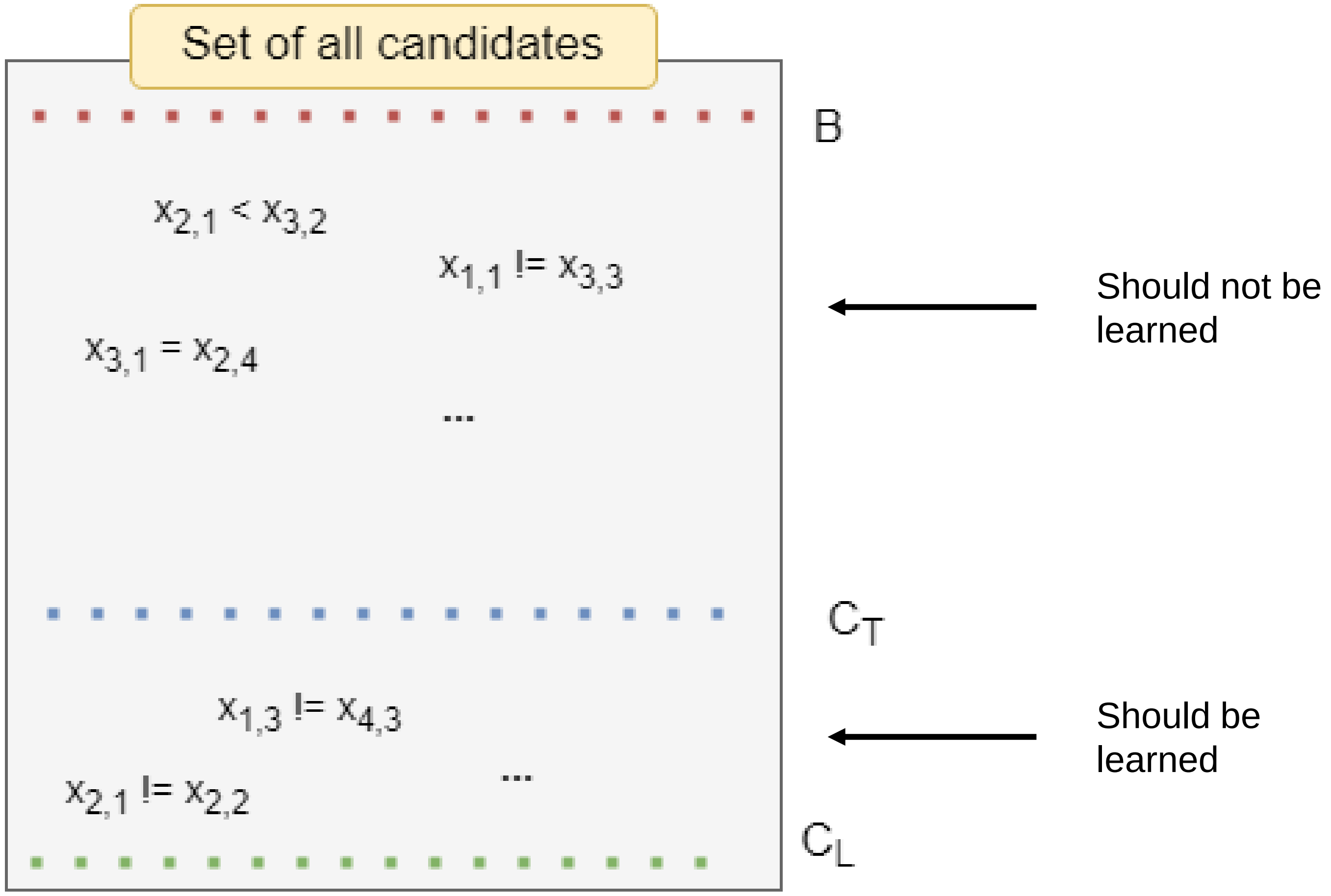
Example e_1

1	1	3	4
3	2	1	1
2	2	3	1
2	3	4	3



Adapting Candidate Elimination

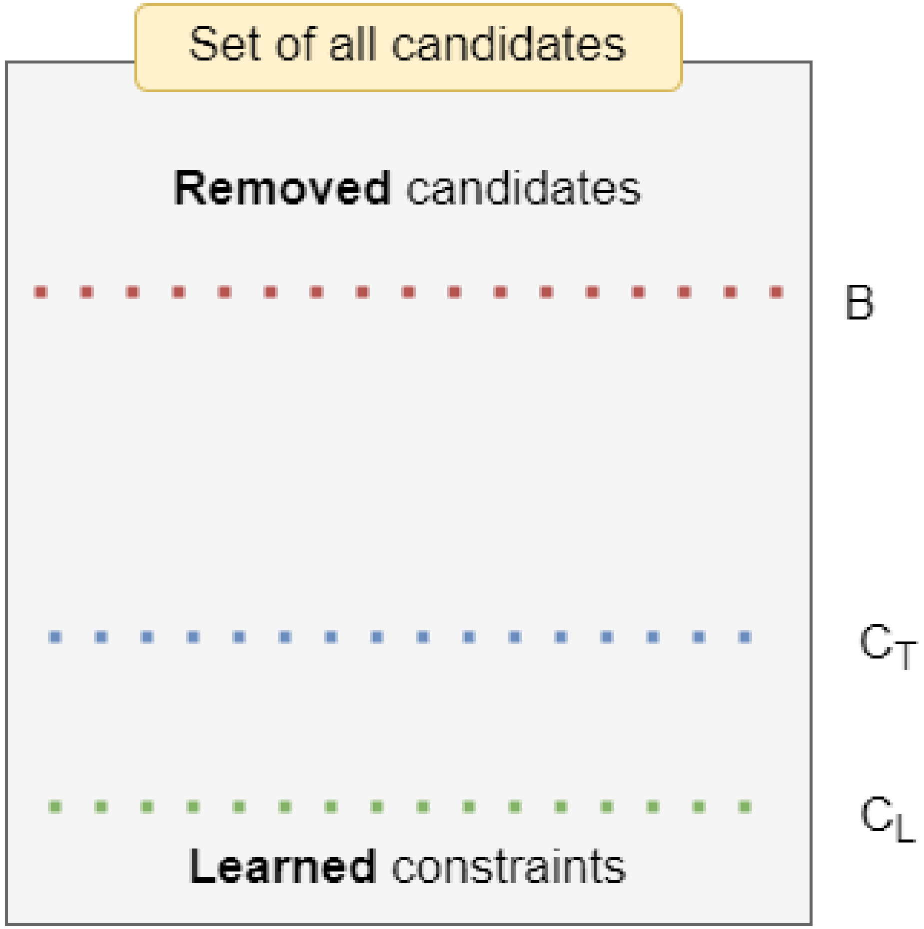
- **B**: set of (remaining) candidate constraints
- **C_T**: target set of constraints
- **C_L**: learned set of constraints



Adapting Candidate Elimination

During the learning process:

- Constraints are removed from B
- Constraints are added to C_L

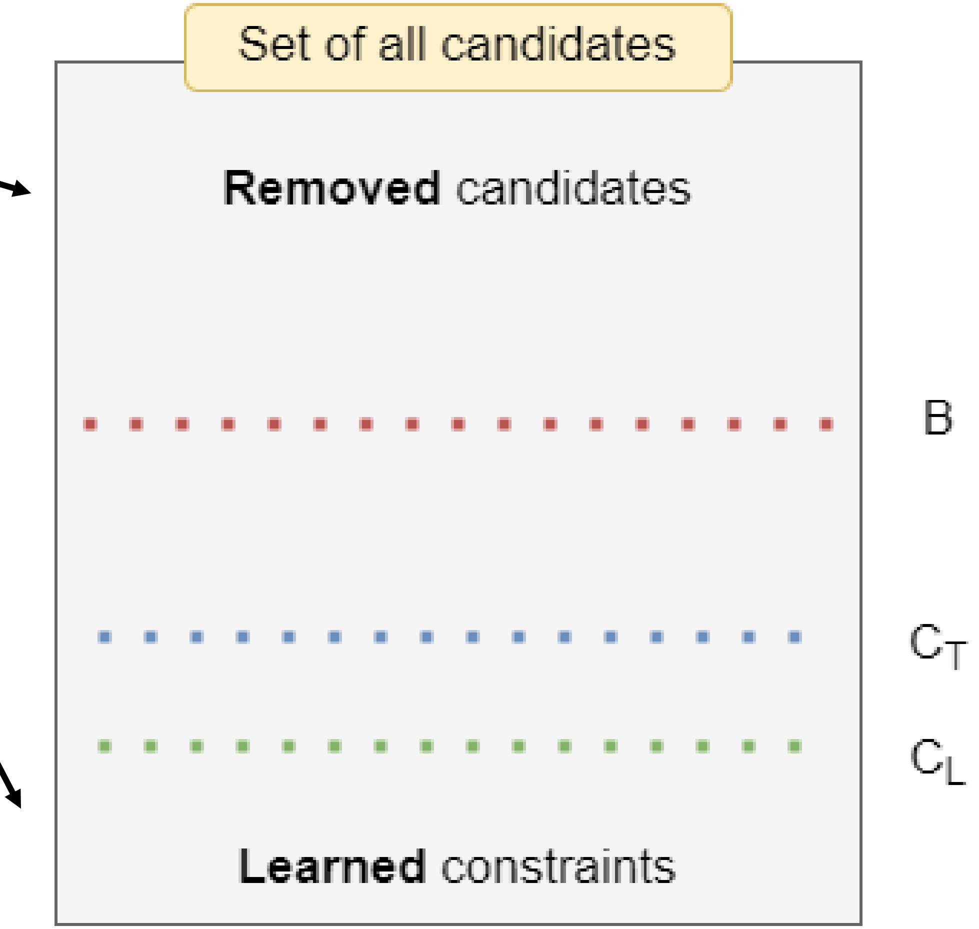


- **B**: set of (remaining) candidate constraints
- **C_T**: target set of constraints
- **C_L**: learned set of constraints

Adapting Candidate Elimination

During the learning process:

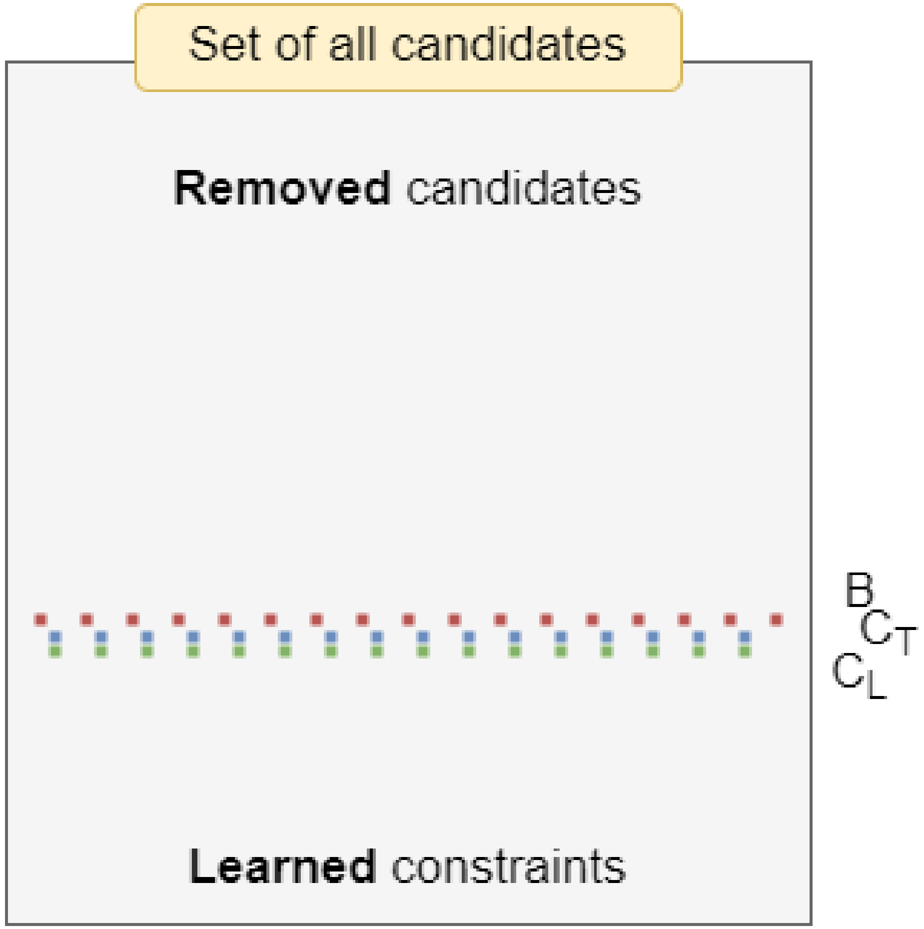
- Constraints are removed from *B*
- Constraints are added to C_L



- **B**: set of (remaining) candidate constraints
- **C_T**: target set of constraints
- **C_L**: learned set of constraints

Adapting Candidate Elimination

If C_T is representable by B the CA system will eventually ***converge*** to a C_L equivalent to C_T



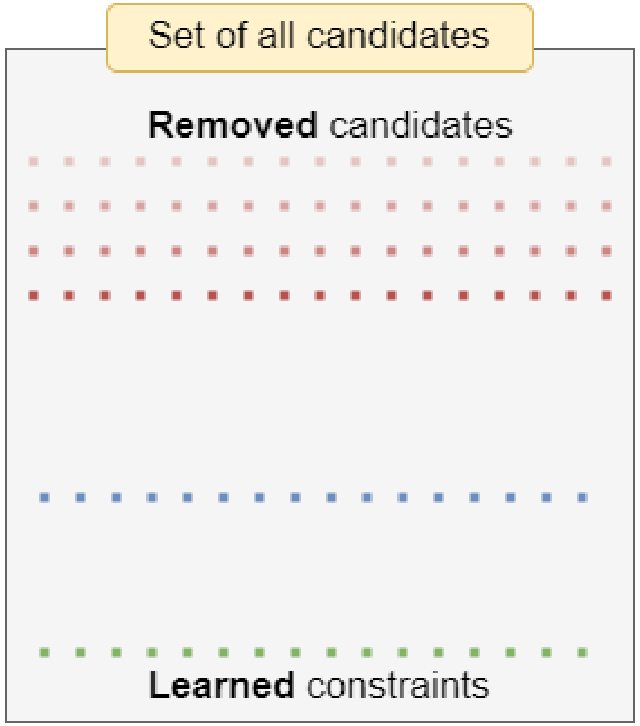
Adapting Candidate Elimination

- Examples: Assignments to the variables of the problem

- Learning from ***positive*** examples (Solutions):

- Violated constraints cannot be part of the model
- Otherwise, it could not be a solution

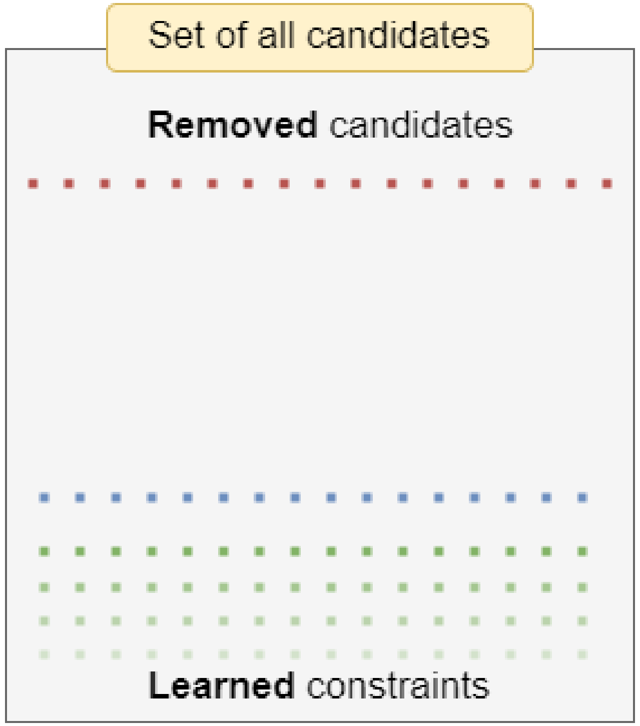
→ Shrinking the bias



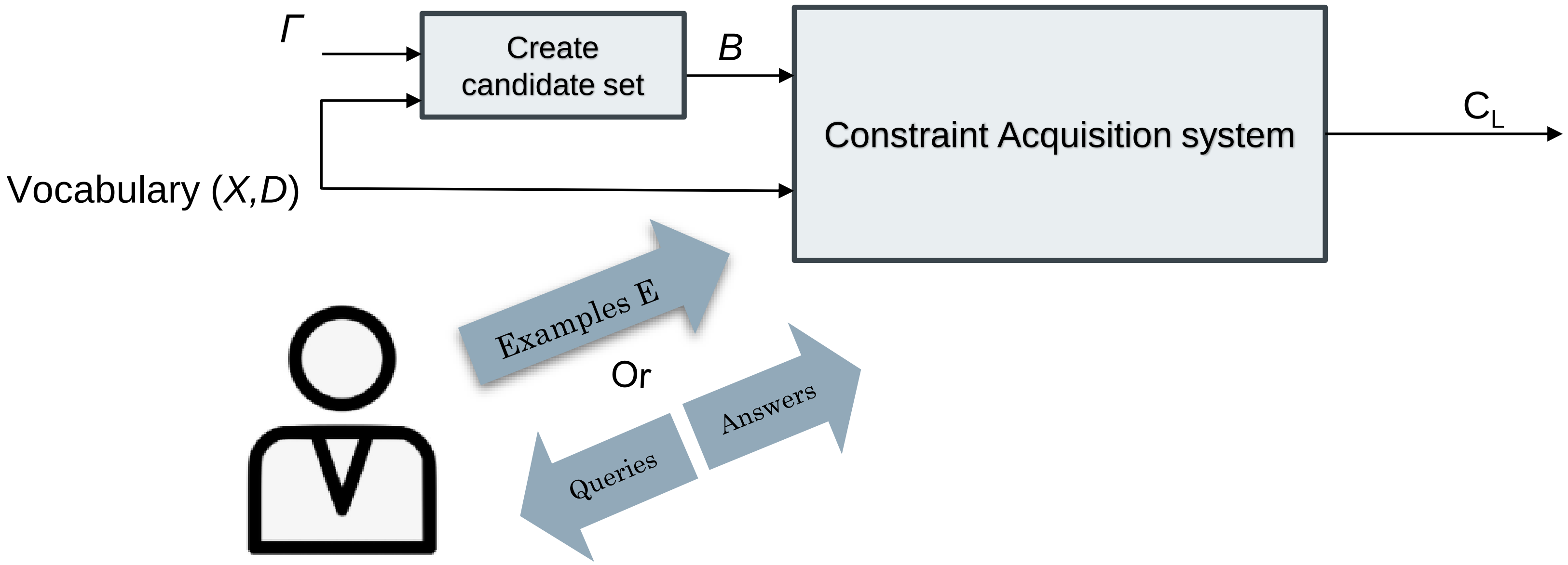
- Learning from ***negative*** examples (Non-solutions):

- One (or more) violated constraint is a constraint of the problem
- Otherwise, it would be a solution

→ Learning Constraints



Small Summary



We discussed:

- Why Constraint Acquisition
- Passive and active acquisition
- Inputs and outputs of CA
- Candidate Elimination

Passive Acquisition

Assignment to all variables of the problem, labelled as:

- a solution
- or a non-solution

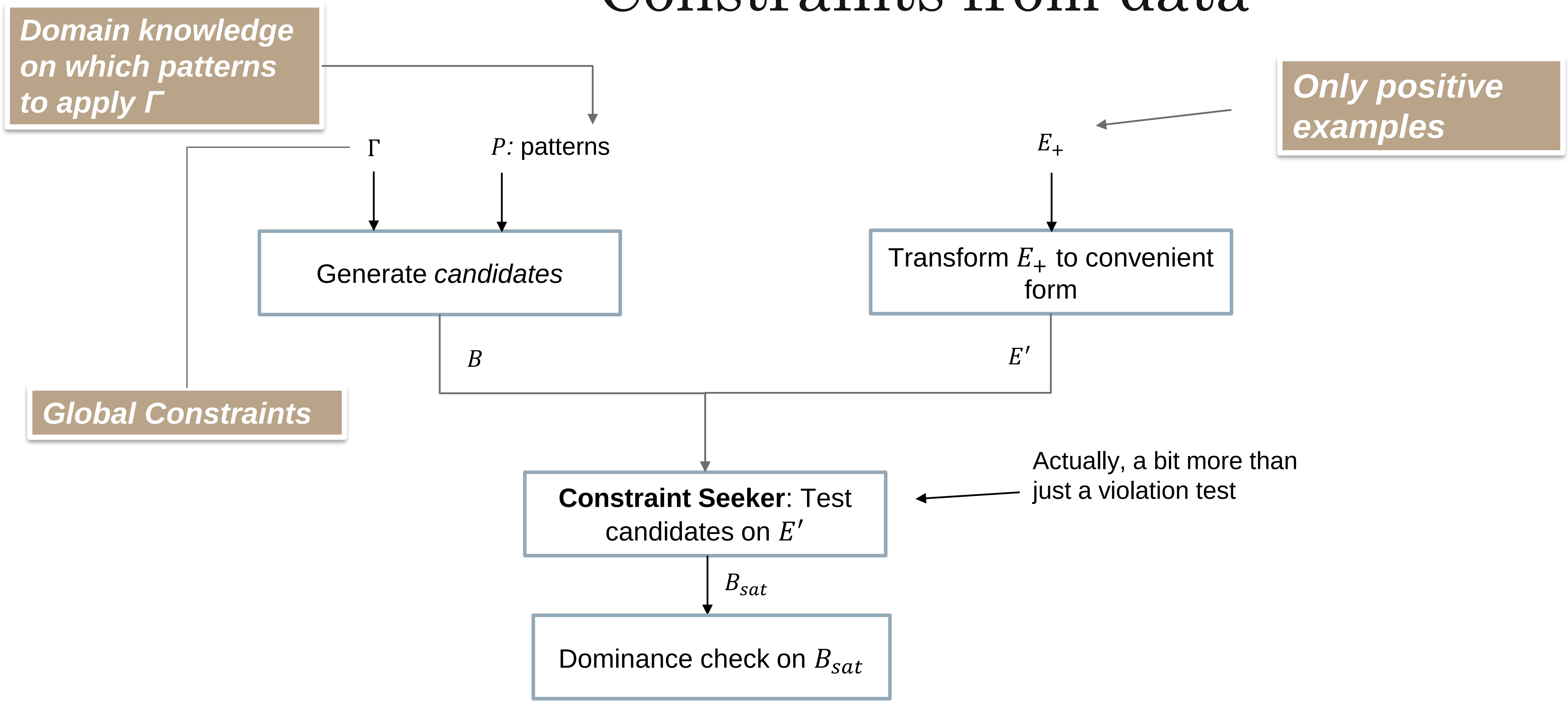
Constraint Acquisition, C. Bessiere et al., AIJ, 2017

Passive Acquisition

Approaches

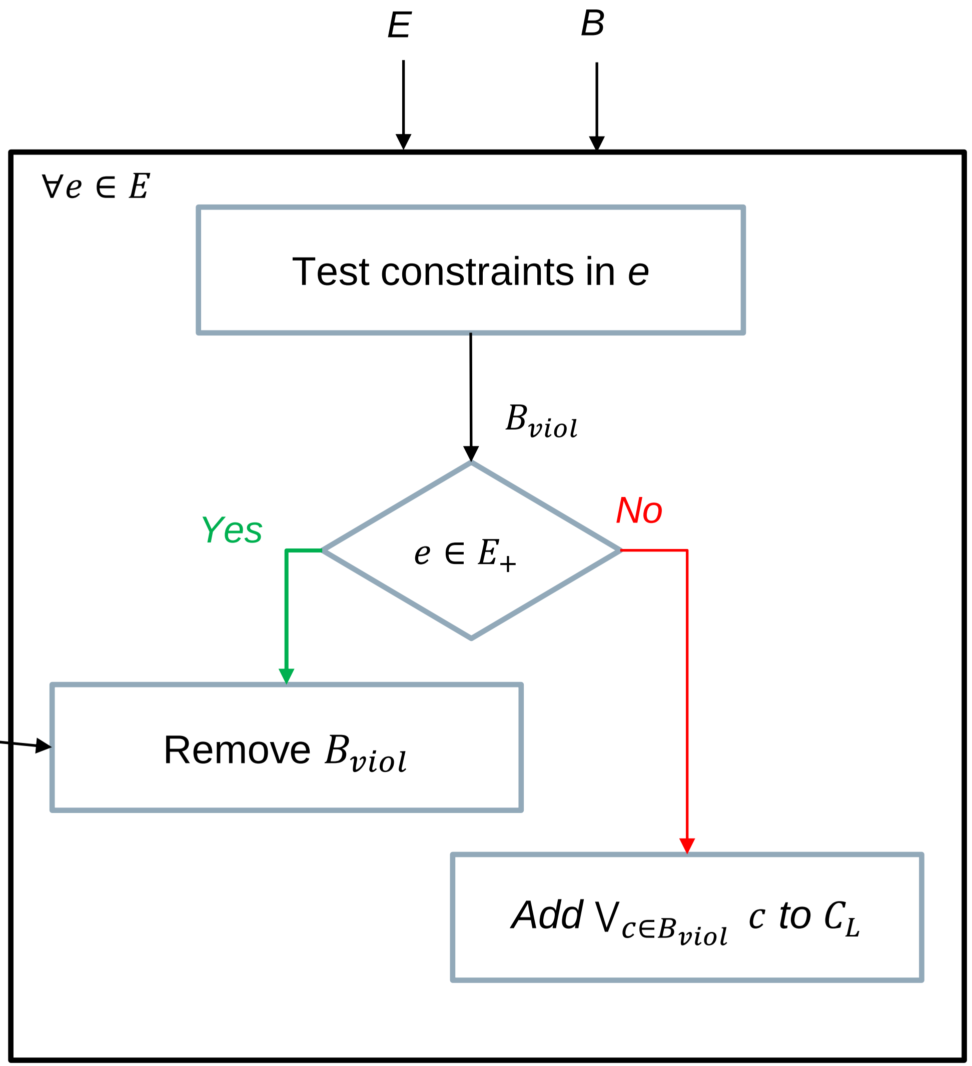
- **Version Space Learning**
  - ConAcq *[C. Bessiere et al., AIJ, 2017]*
- **Generate and Aggregate**
  - COUNT-CP *[M. Kumar et al., CP, 2022]*
- **Data Mining**
  - MineAcq *[S. Prestwich., CP, 2021]*
- **Training Classifiers**
  - BAYESACQ *[S. Prestwich et al., AMAI, 2021]*
  - EML *[M. Lombardi et al., 2017]*
- **Statistical Approaches**
  - SEQACQ *[S. Prestwich., ECAI, 2020]*
  - BAYESACQ *[S. Prestwich et al., AMAI, 2021]*
- **Other**
  - ModelSeeker *[N. Beldiceanu et al., CP, 2012]*
  - CABSC *[C. Coulombe et al., CP, 2012]*

ModelSeeker – Learning Global Constraints from data



A Model Seeker: Extracting Global Constraint Models from Positive Examples, N. Beldiceanu et al., CP, 2012

ConAcq – Sat-based version space approach



Also from disjunctions in C_L

Constraint Acquisition, C. Bessiere et al., AIJ, 2017

Statistical approaches – BayesAcq

Robust approach: Can handle noise in training set

E B

Create E_c,

Train Bernoulli NB in E_c

Derive P($e \notin sol(c) \mid E_-$) and P($e \notin sol(c) \mid E_+$) from NB $\forall c \in B$

$$C_L = \left\{ c \mid \frac{P(e \notin sol(c) \mid E_-)}{P(e \notin sol(c) \mid E_+)} > threshold, \forall c \in B \right\}$$

transforming *E* using candidate violations as Boolean features,

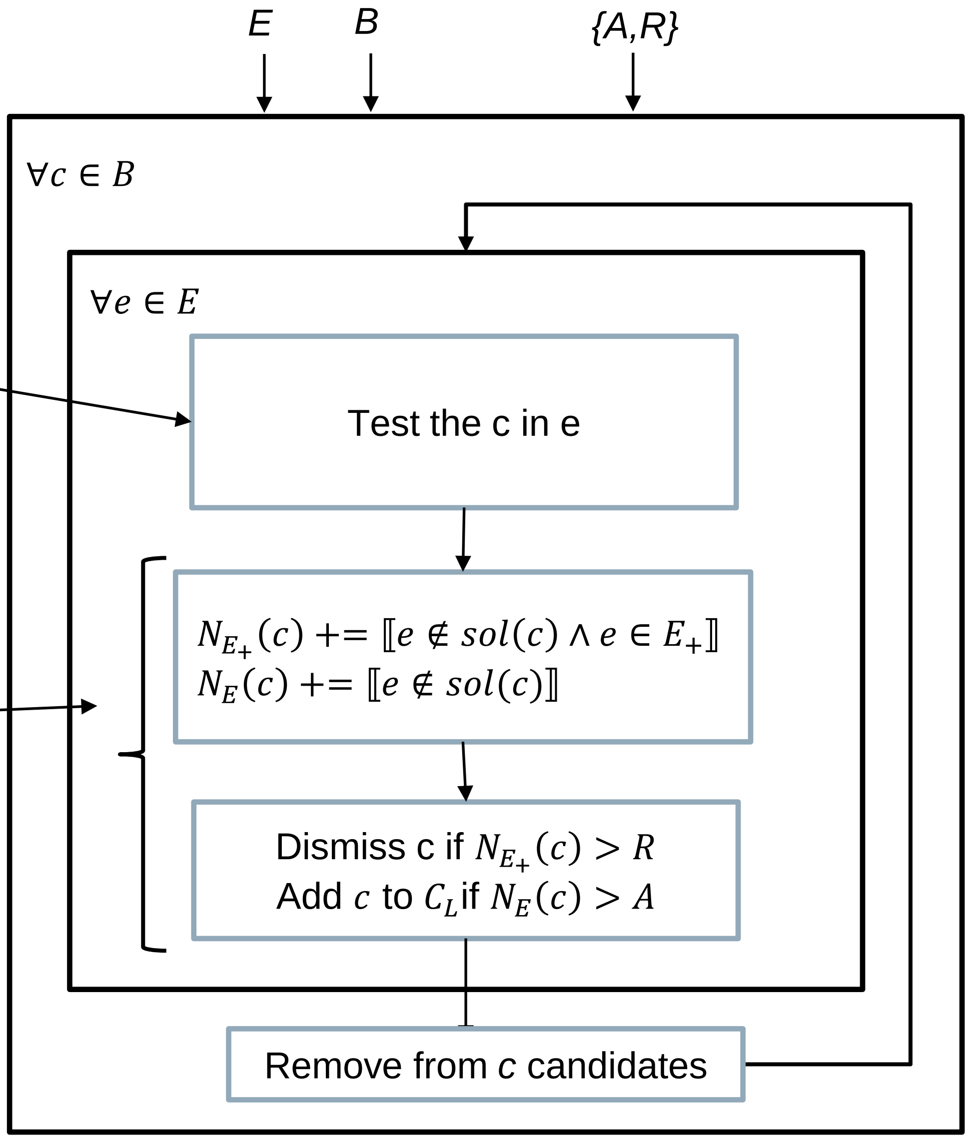
$F_e = \{ e \in sol(c) \mid \forall c \in B \}, \forall e \in E$

Learn constraints with probability higher than a threshold

Classifier-based constraint acquisition, S. Prestwich et al., AMAI, 2021

Statistical approaches – SeqAcq

Robust approach: Can handle noise in training set



Not representing the whole set of candidates at the same time

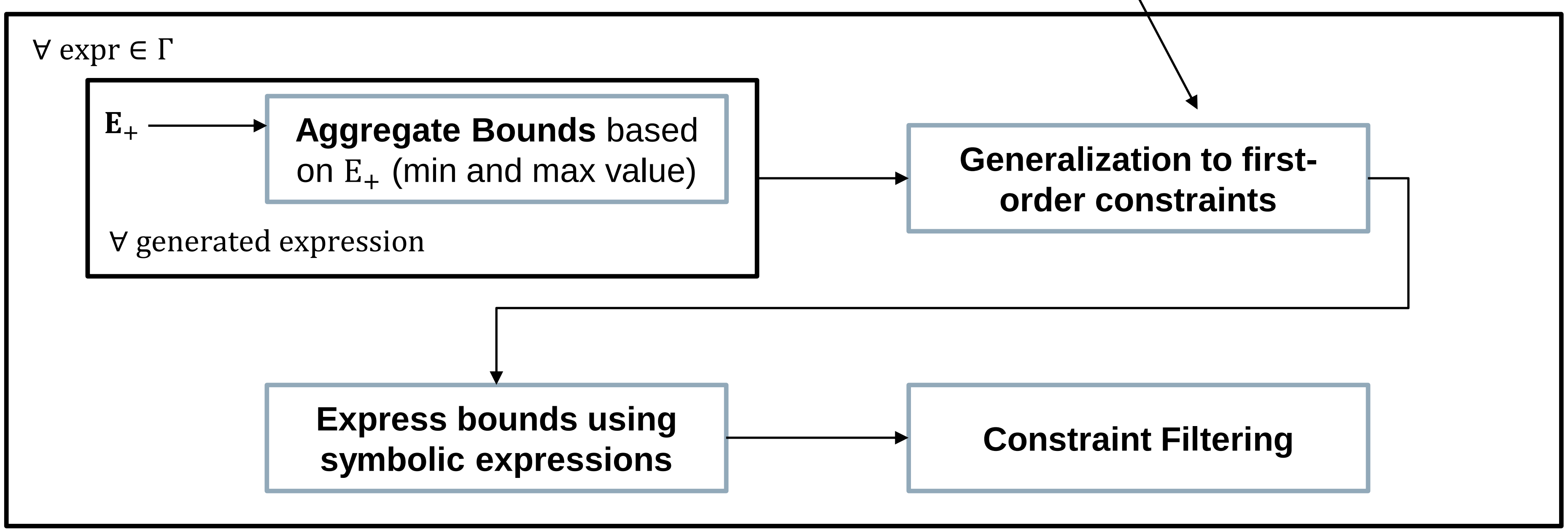
Use of a Sequential Probability Ratio Test (SPRT) based approach in E, $\forall c \in B$

Robust Constraint Acquisition by Sequential Analysis, S. Prestwich., ECAI, 2020

COUNT-CP: Learn using generate and aggregate

- Learns constraints of the following form: $lb <= expr <= ub$
- Aggregates the bounds of the expressions based on E_+
- Language Γ contains **numerical** expressions and not constraints e.g. $x_{i,j} - x_{k,l}$, $x_{i,j} + x_{k,l}$ etc.

Pattern-based grouping like in ModelSeeker



Learning Constraint Programming Models from Data using Generate-and-Aggregate, M. Kumar et al., CP, 2022

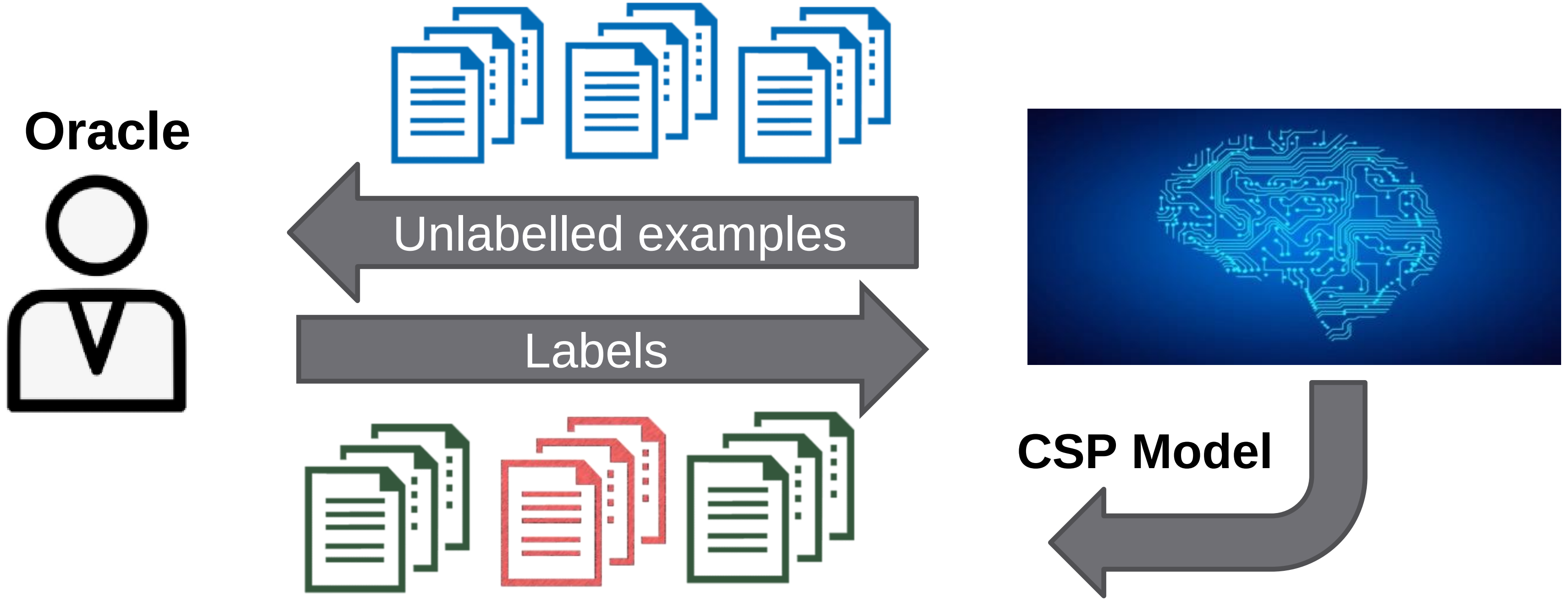
Passive Constraint Acquisition

Small Summary

We discussed:

- Passive Constraint Acquisition
- Different approaches
- Learning global and fixed arity constraints
- Learning first-order constraints

Interactive Constraint Acquisition



Membership query

1	1	3	4
3	2	1	1
2	2	3	1
2	3	4	3

Answer: Negative in both of them (a constraint is violated)

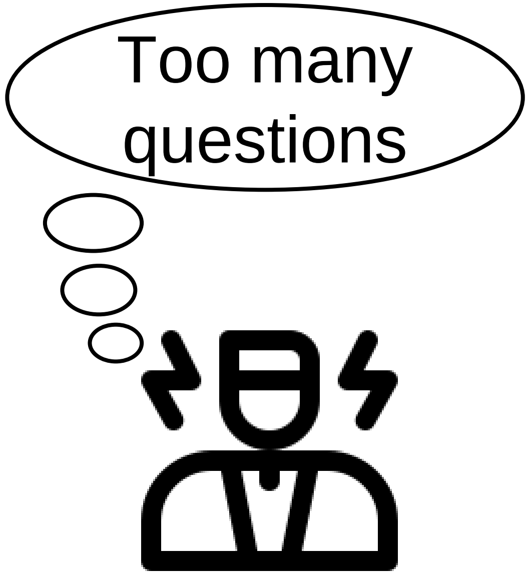
Partial query

1	1	-	4
3	-	1	-
-	-	-	-
2	-	-	-

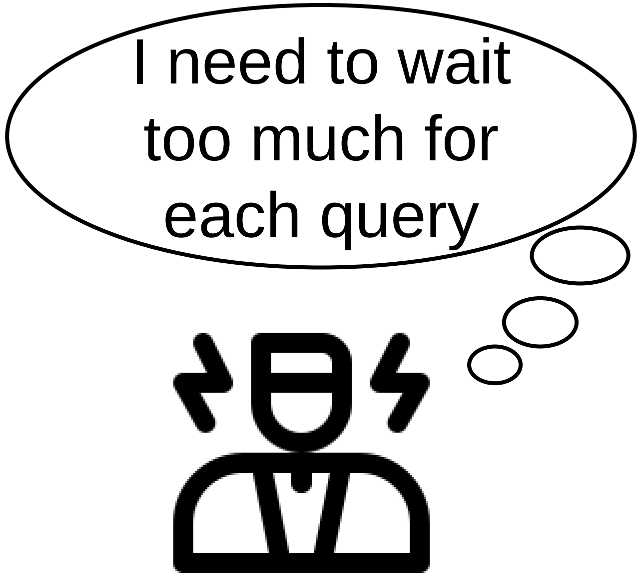
Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Challenges for interactive CA

Minimum number of queries

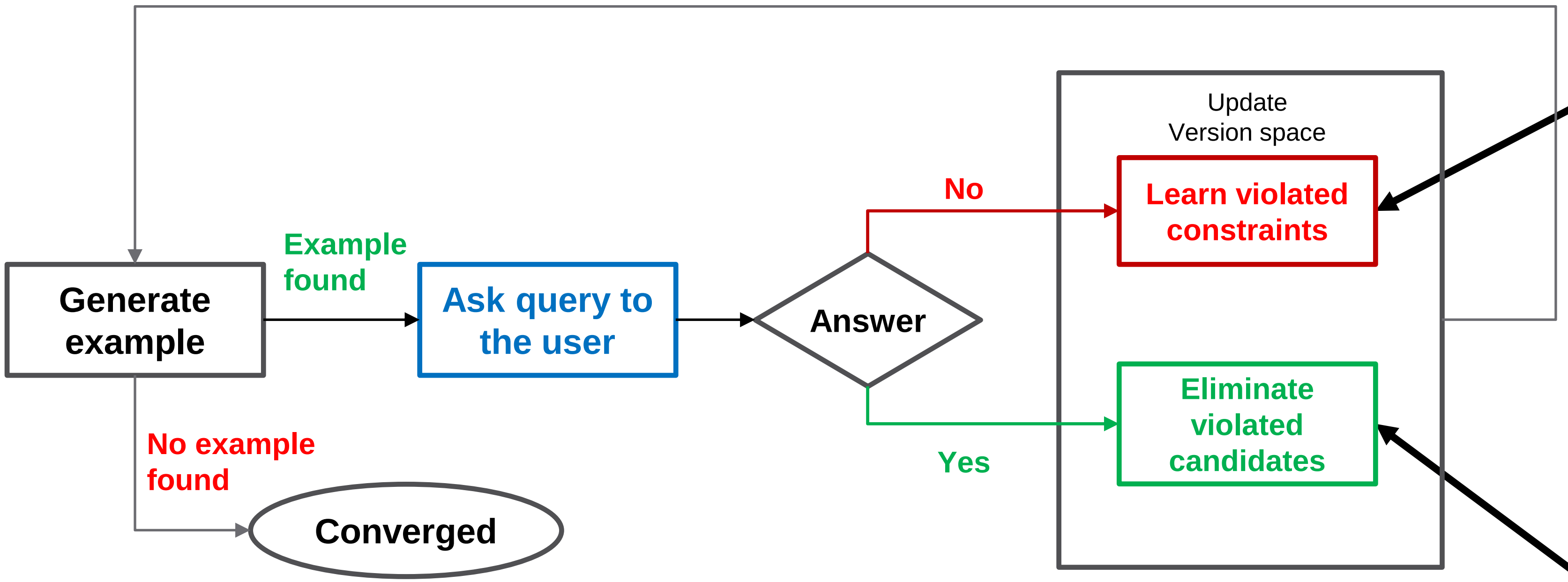


Minimum waiting time for the user



Learning constraint models from data, D Tsouros et al., 2023

Interactive Constraint Acquisition

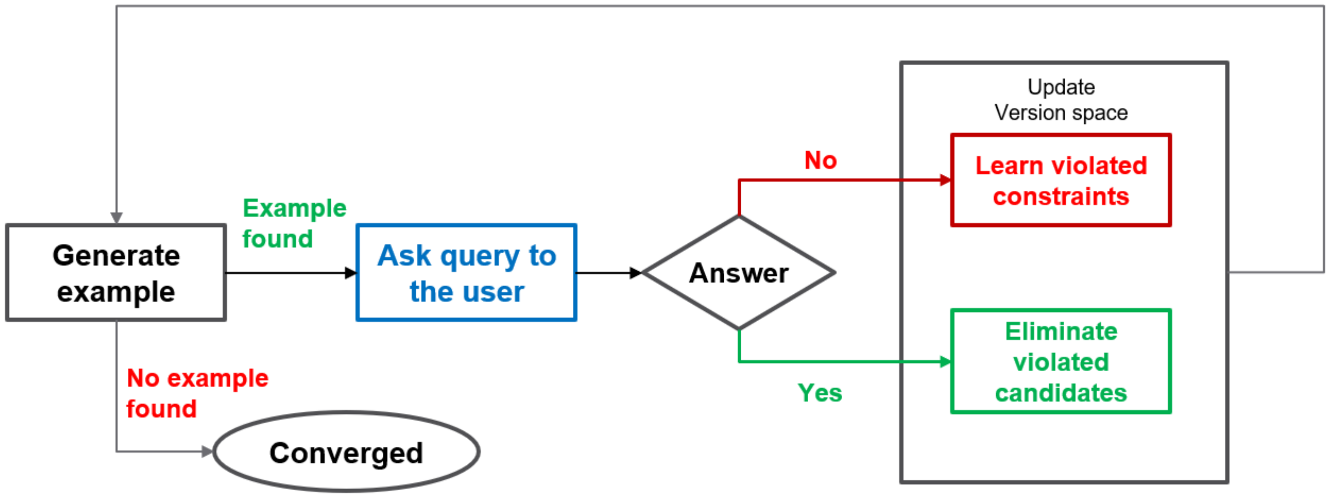


Learn constraints: zoom in violated constraints' scopes using partial queries and add them to C_L

Shrink version space: remove violated constraints from B

Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Interactive Constraint Acquisition



Positive answer: Eliminate candidates

```
Query: is this a solution?
[[3 4 1 2]
 [1 2 3 4]
 [4 1 2 3]
 [2 3 4 1]]
Answer:  Yes
```

Negative answer: Learn constraints

```
Query 0: is this a solution?
[[1 3 2 2]
 [3 2 3 2]
 [1 1 1 1]
 [2 4 3 2]]
Answer:  No
```

Query generation

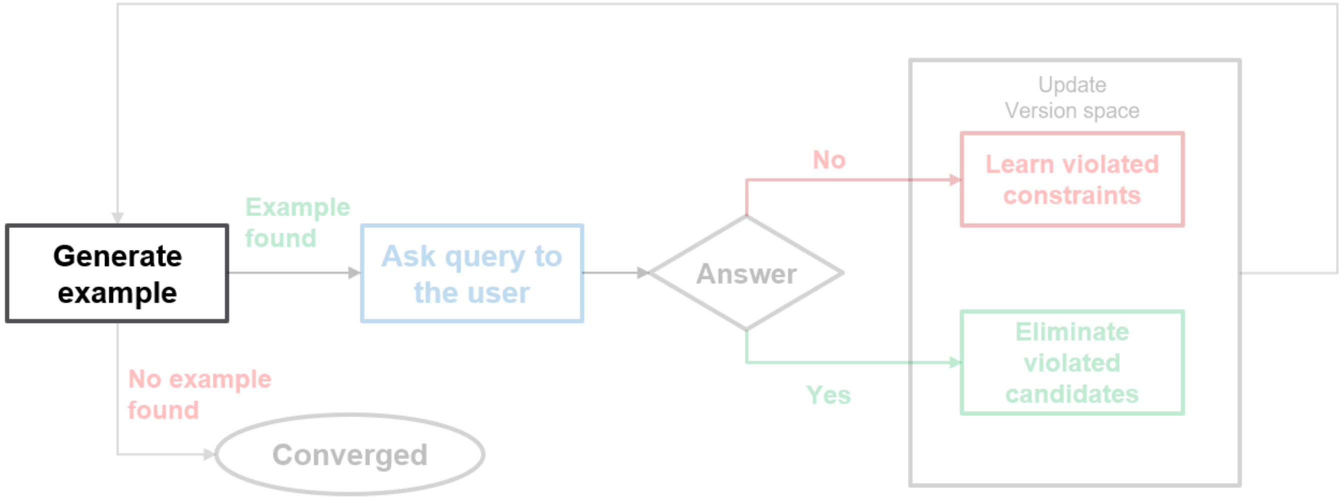
Informative query

- Generate "informative" examples

Quality of query

- Get the maximum amount of information

Convergence



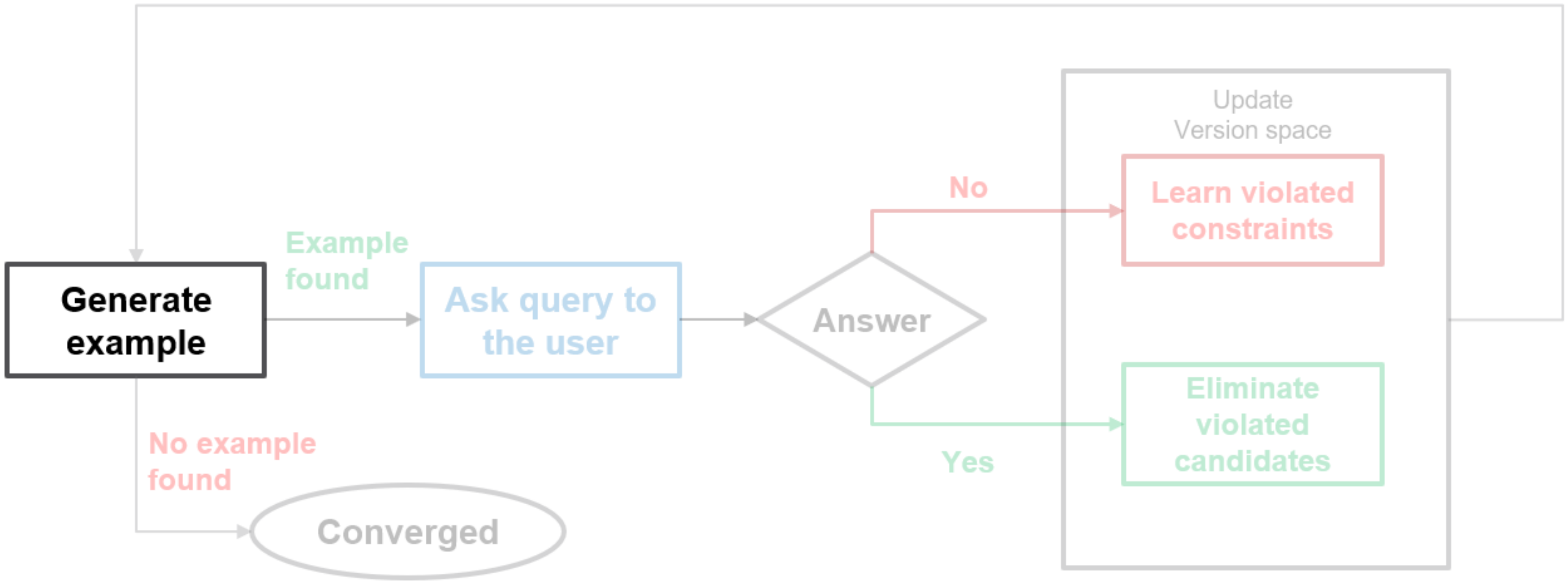
Query generation

Find an Informative ("irredundant") query

- Not violating any learned constraint in C_L
- Violating at least one constraint from *B*

Why??

Find $e \in sol(C_L \wedge \bigvee_{c \in B} \sim c)$



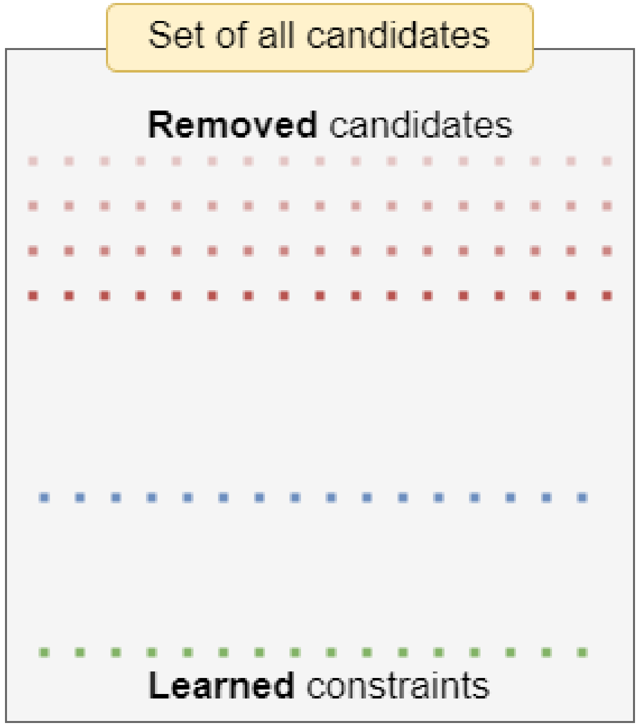
Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

What we can learn from examples

- Learning from ***positive*** examples (Solutions):

- Violated constraints cannot be part of the model
- Otherwise, it could not be a solution

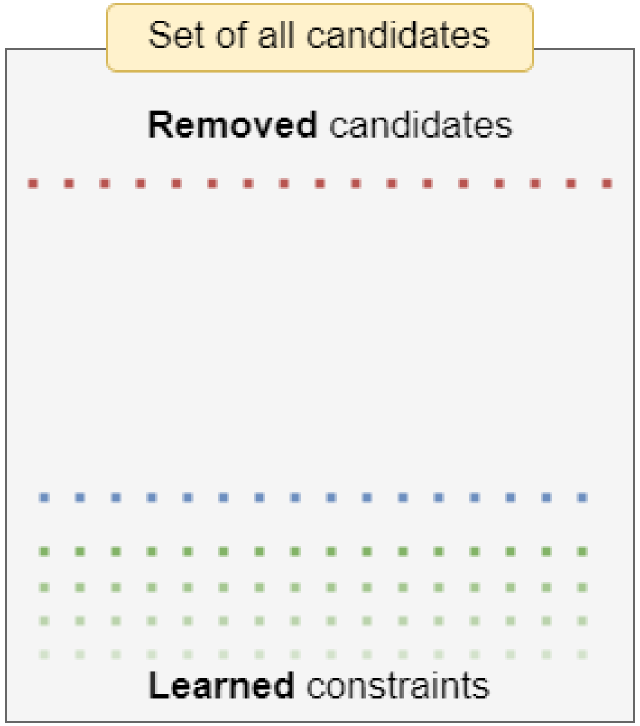
Shrinking the bias



- Learning from ***negative*** examples (Non-solutions):

- One (or more) violated constraint is a constraint of the problem
- Otherwise, it would be a solution

Learning Constraints



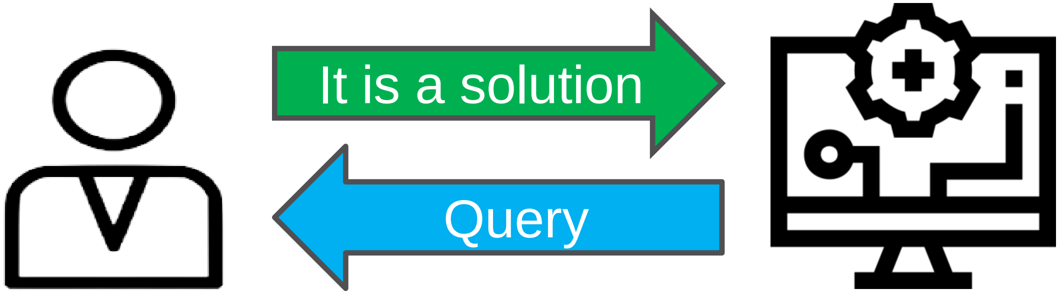
- In both cases, we get information only about the violated candidates and not about the satisfied ones!

Query generation

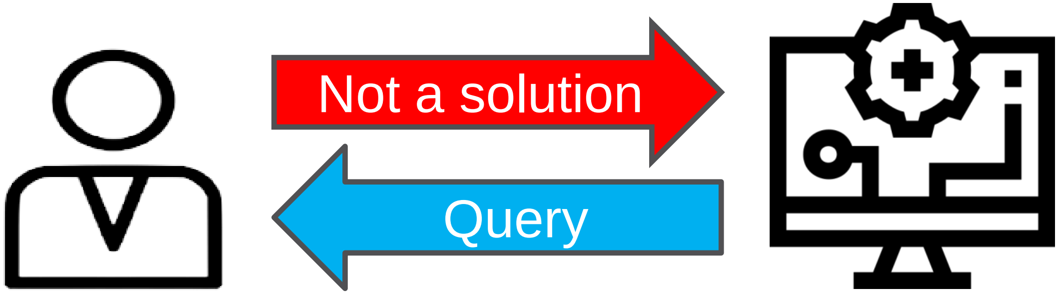
Find an Informative (“irredundant”) query

- Not violating any learned constraint in C_L
- Violating at least one constraint from *B*

Find $e \in sol(C_L \wedge \bigvee_{c \in B} {\sim}c)$



- We know that the violated constraints cannot be part of the model
- Otherwise, it could not be a solution



- We know that (at least) one of the violated constraints is a constraint of the problem
- Otherwise, it would be a solution

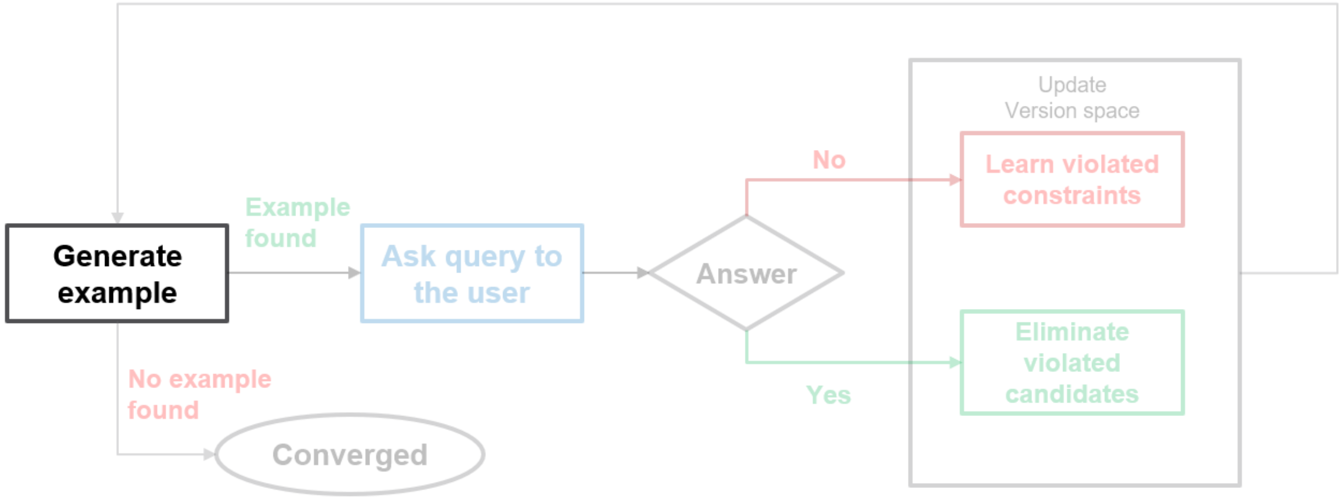
Query generation

Informative query

Quality of query

- Get the maximum amount of information

Convergence



Query generation

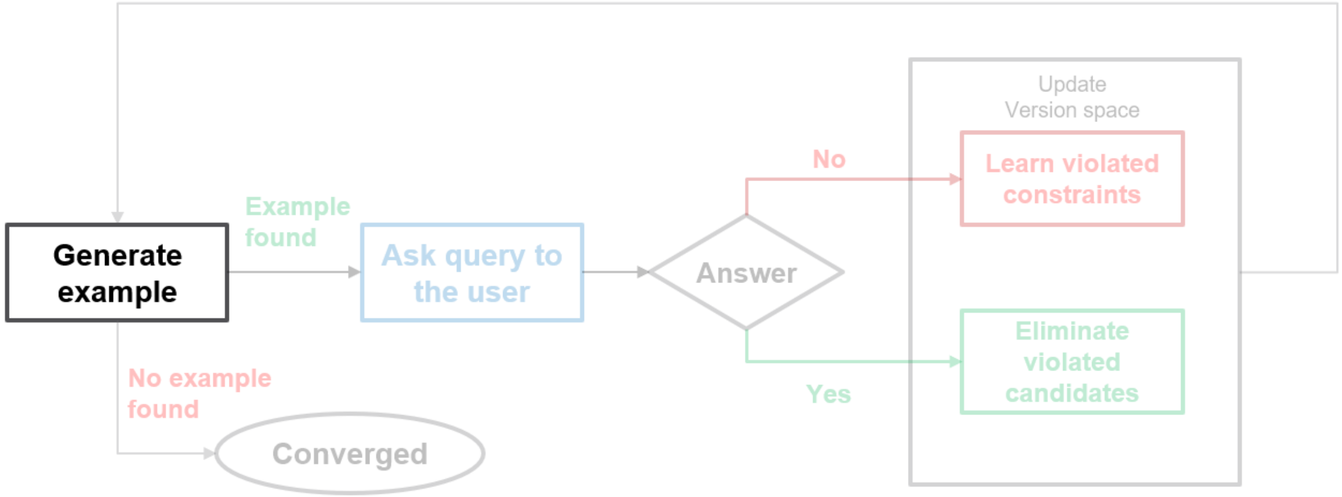
Quality of query

- *Better* generated examples lead to faster convergence
  - More information per query -> less queries needed

Typically: maximize candidate violations

$$\max \sum_{c \in B} \sim c$$

Not fully aligning with the goal!!
We will discuss this later



Constraint acquisition via partial queries, C. Bessiere et al., IJCAI, 2013

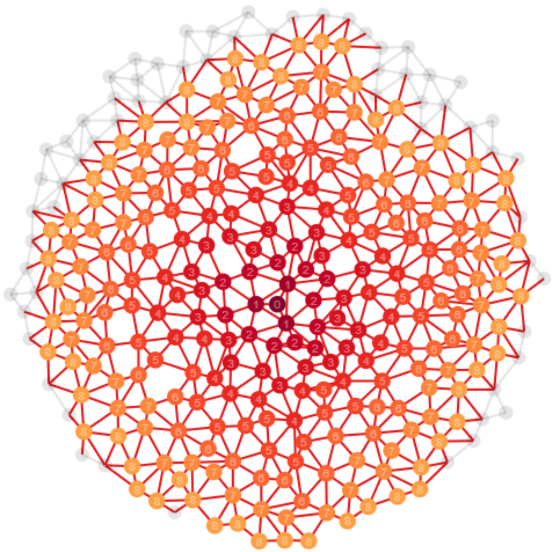
KU LEUVEN

Query generation

Finding an Irredundant query

- Finding an informative query is not always easy…

B can be huge!!



B can contain indirectly implied constraints

Assume a simple 9x9 Sudoku puzzle.
- Combinations of ≠ constraints imply others
- 648 of them imply the rest 162

When the 648 constraints have been learned and must be satisfied, the rest cannot be violated!

Indirect implications are not detected with simple propagation!!

Query generation

Custom solvers

- Custom solvers are often employed to deal with this
  - Looking for $e_Y \in sol(C_L[Y] \wedge \bigvee_{c \in B[Y]} \sim c)$
  - Arbitrary $Y \subseteq X$

Projection-based query generation

- Project down to the relevant variables

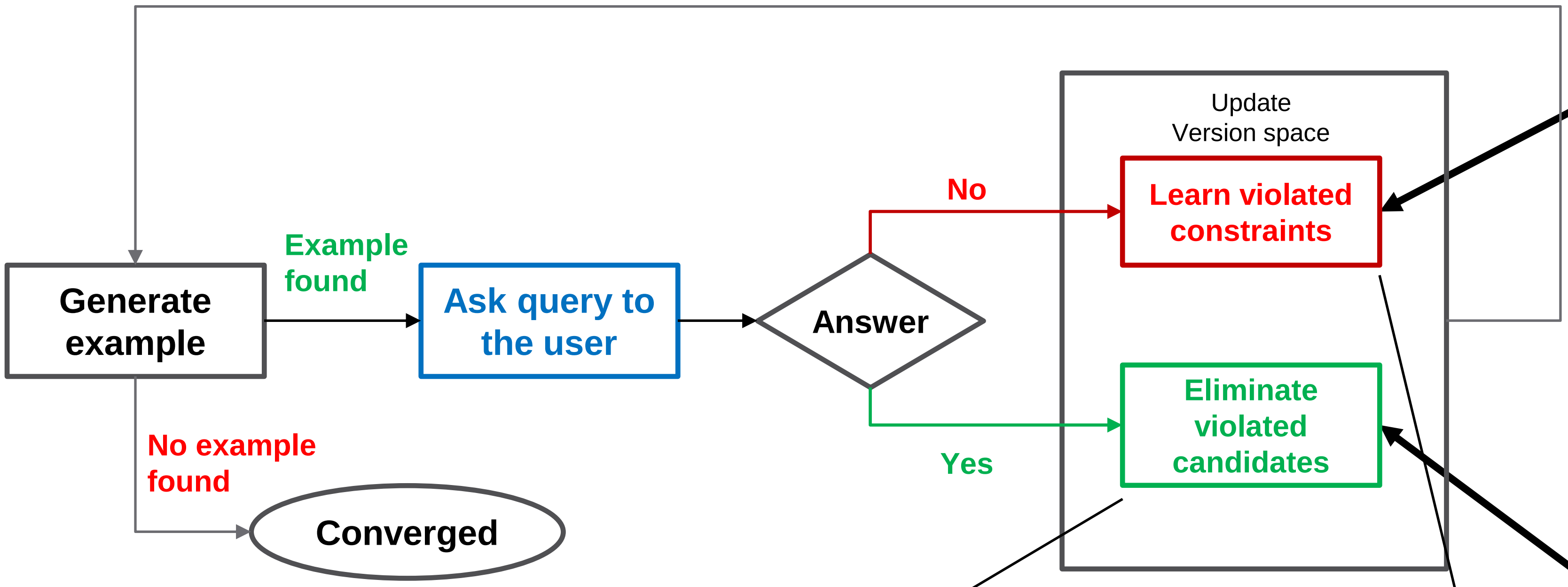
$$Y = \bigcup_{c \in B} var(c)$$

  - We can only get information on variables of constraints in B
  - Avoiding indirect implications
- Find $e_Y \in sol(C_L[Y] \wedge \bigvee_{c \in B[Y]} \sim c)$

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

Interactive Constraint Acquisition

What happens after query generation?



Learn constraints: zoom in violated constraints' scopes using partial queries and add them to C_L

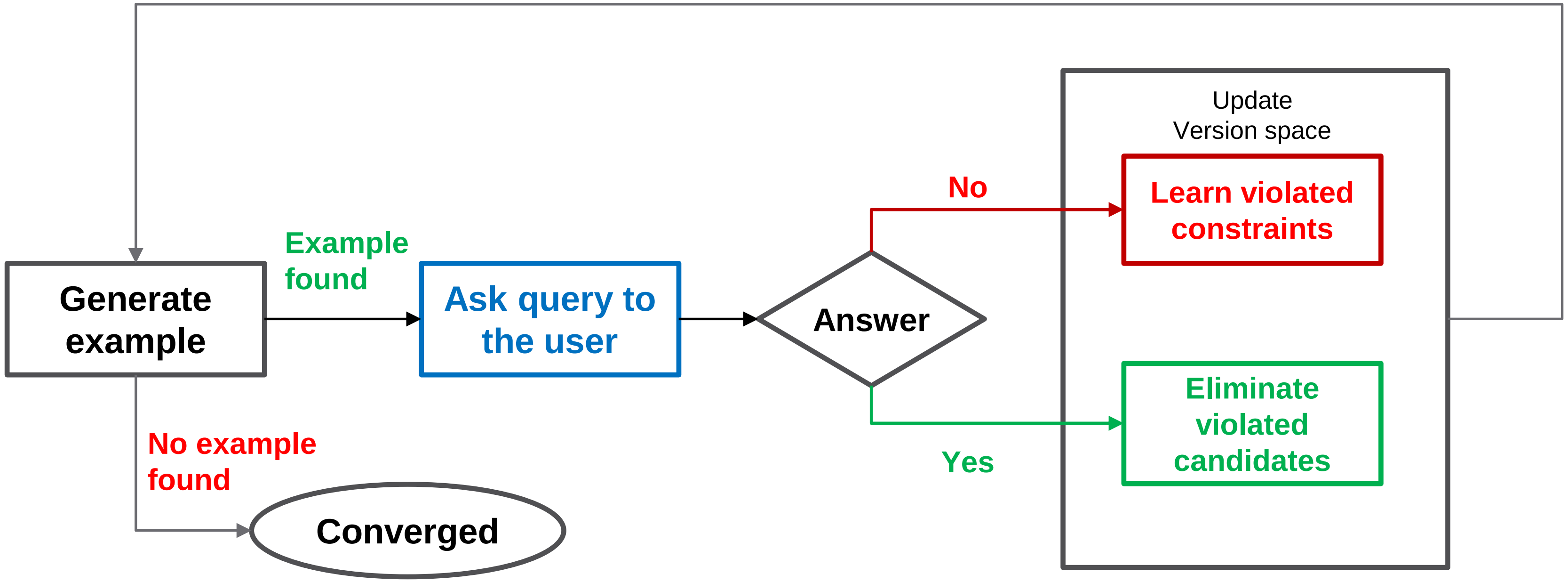
Shrink version space: remove violated constraints from B

e: example generated
$\kappa B \leftarrow \{c \mid c \notin sol(e)\}$

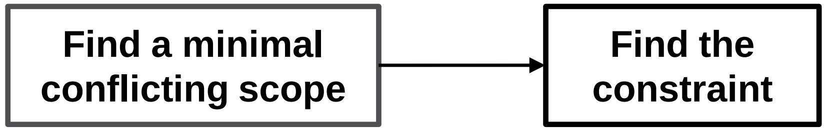
$B \leftarrow B \setminus \kappa B(e)$

How to find which of the violated constraint(s) to learn?

Learning a constraint



- 2-step process:



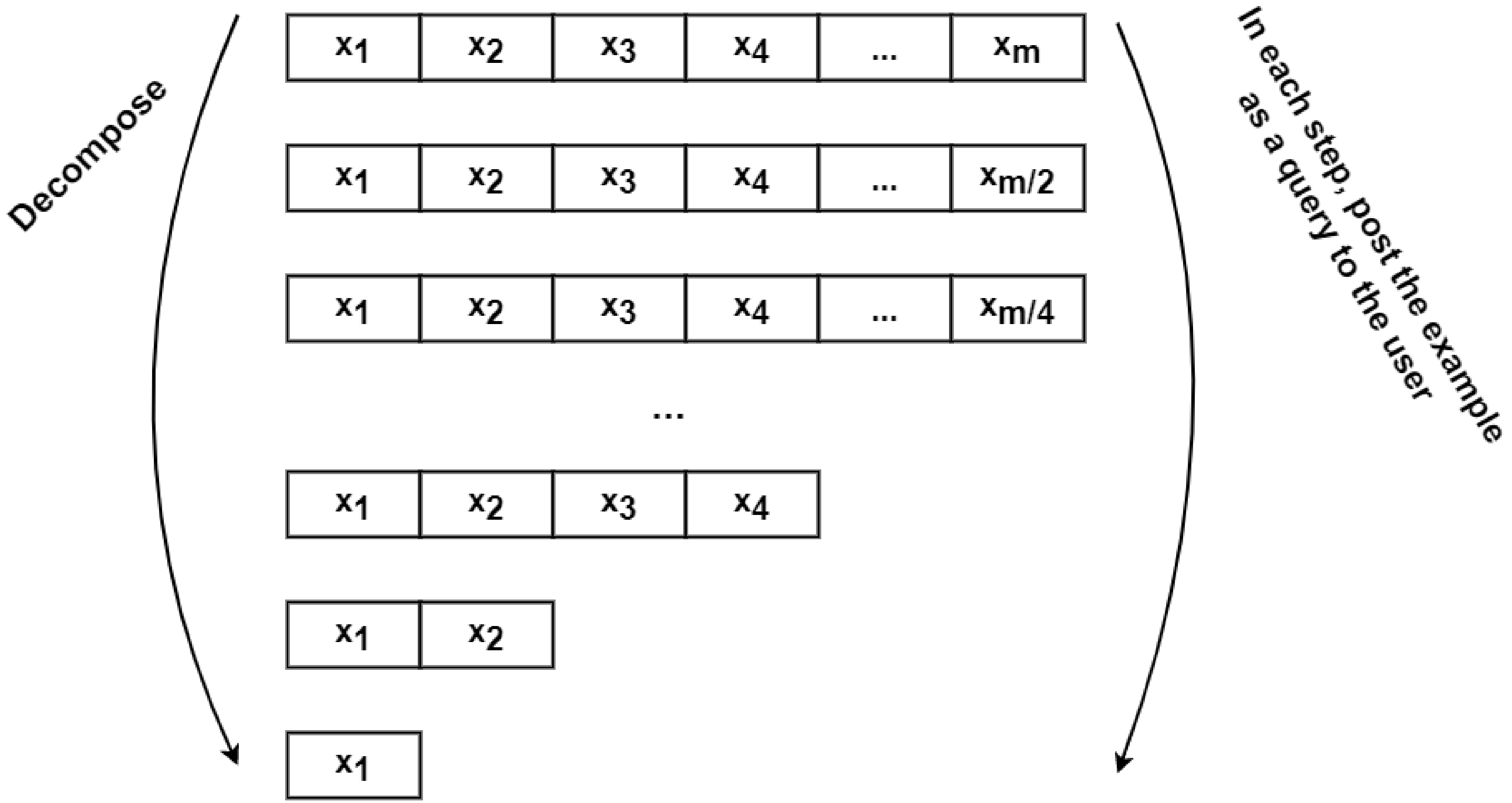
1. ***FindScope***: exploit partial (sub)queries to find the problematic part of the assignment
2. ***FindC***: Try different assignments to find the specific constraint in the scope

Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Finding the scope of a constraint

Exploit partial (sub)queries to find the conflicting part

Find a minimal conflicting scope → Find the constraint

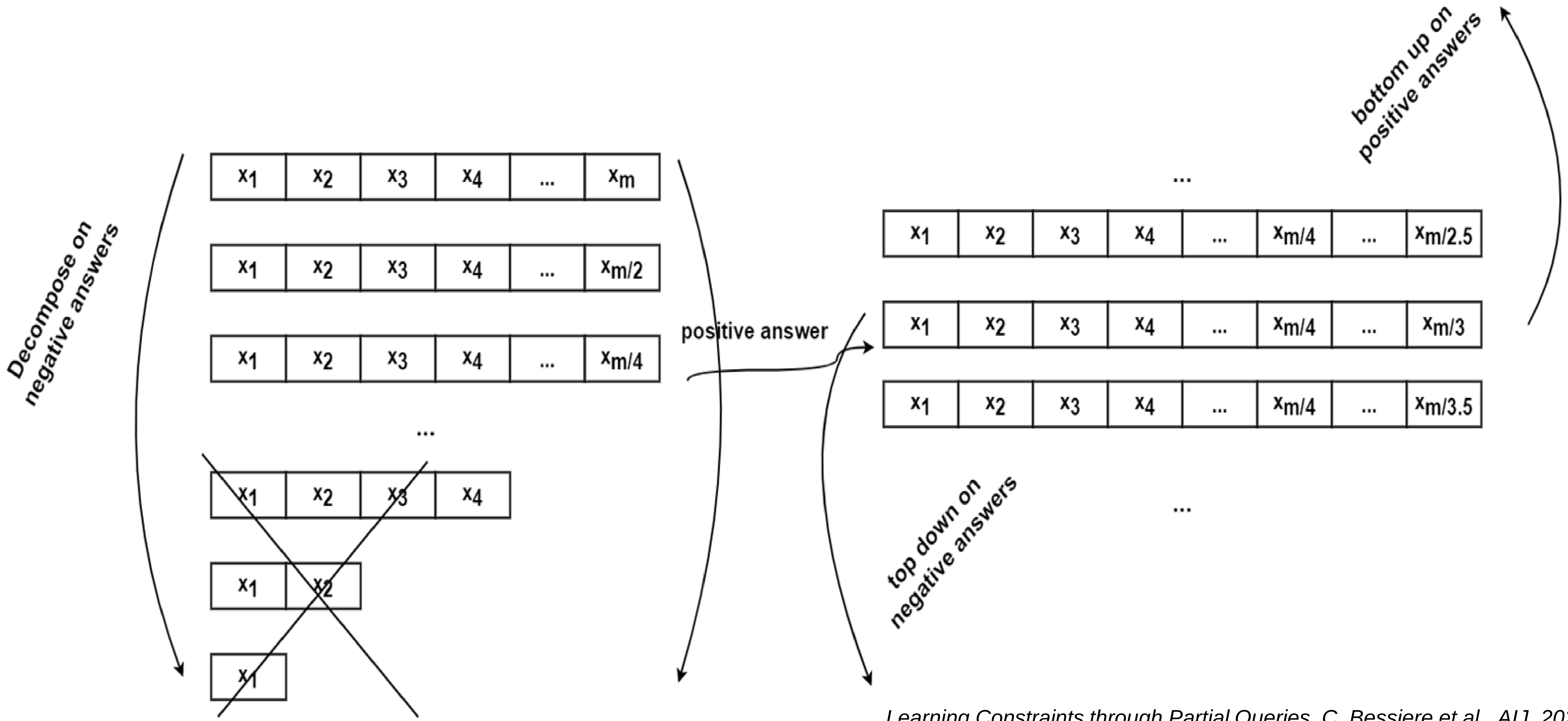


Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Finding the scope of a constraint

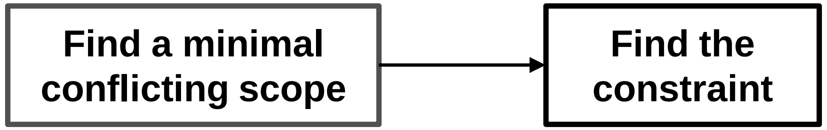
Exploit partial (sub)queries to find the conflicting part

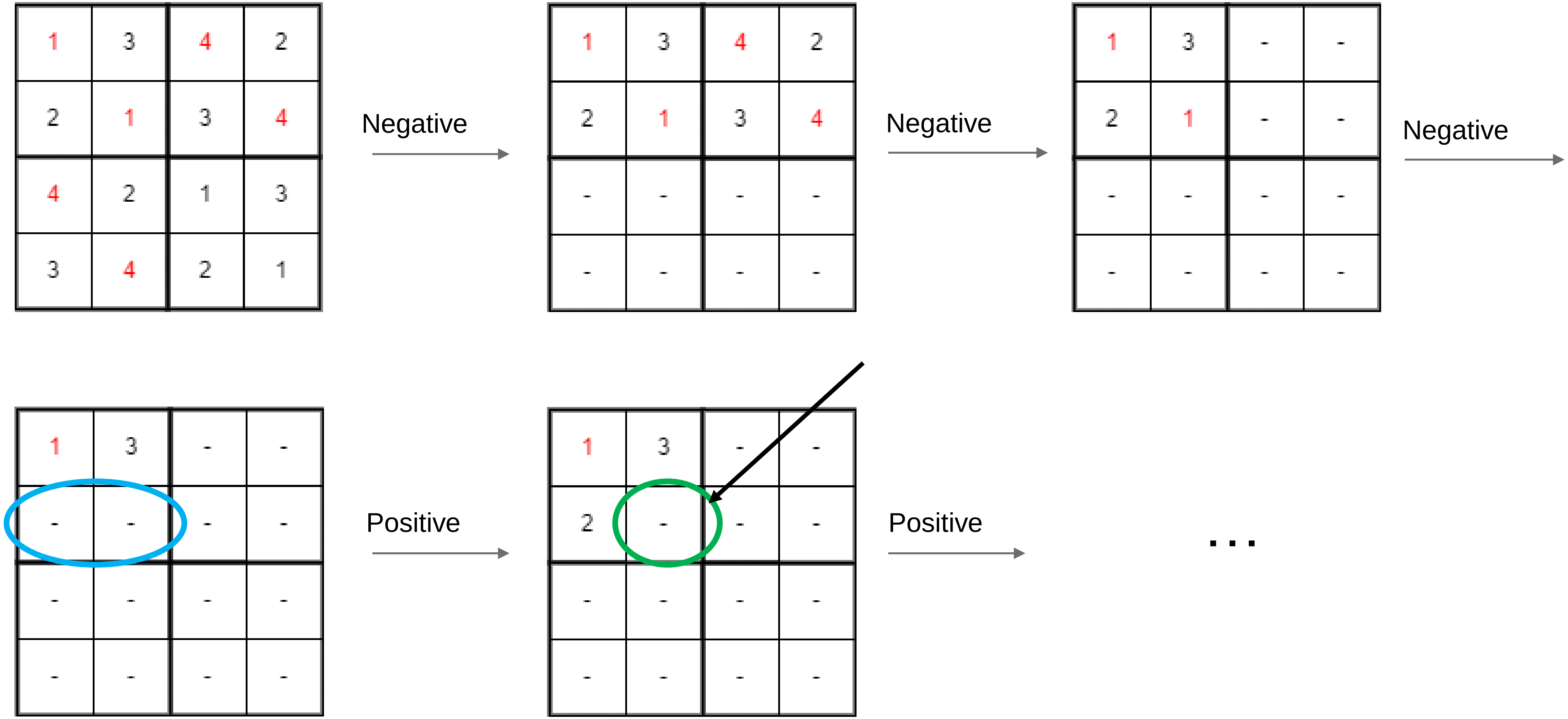
Find a minimal conflicting scope → **Find the constraint**



Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Finding the scope of a constraint





Finding the relation of a constraint

Find a minimal conflicting scope → **Find the constraint**

1	3	4	2
2	1	3	4
4	2	1	3
3	4	2	1

We have found the scope: $\{x_{1,1}, x_{2,2}\}$
Assume that the candidate constraints for this scope are:

$$\{x_{1,1} \neq x_{2,2}, x_{1,1} > x_{2,2}, x_{1,1} < x_{2,2}\}$$

What is the real conflict?

Try different assignments to find the specific constraint in the scope

1	-	-	-
-	2	-	-
-	-	-	-
-	-	-	-

Positive →

Remove $x_{1,1} > x_{2,2}$

2	-	-	-
-	1	-	-
-	-	-	-
-	-	-	-

Positive →

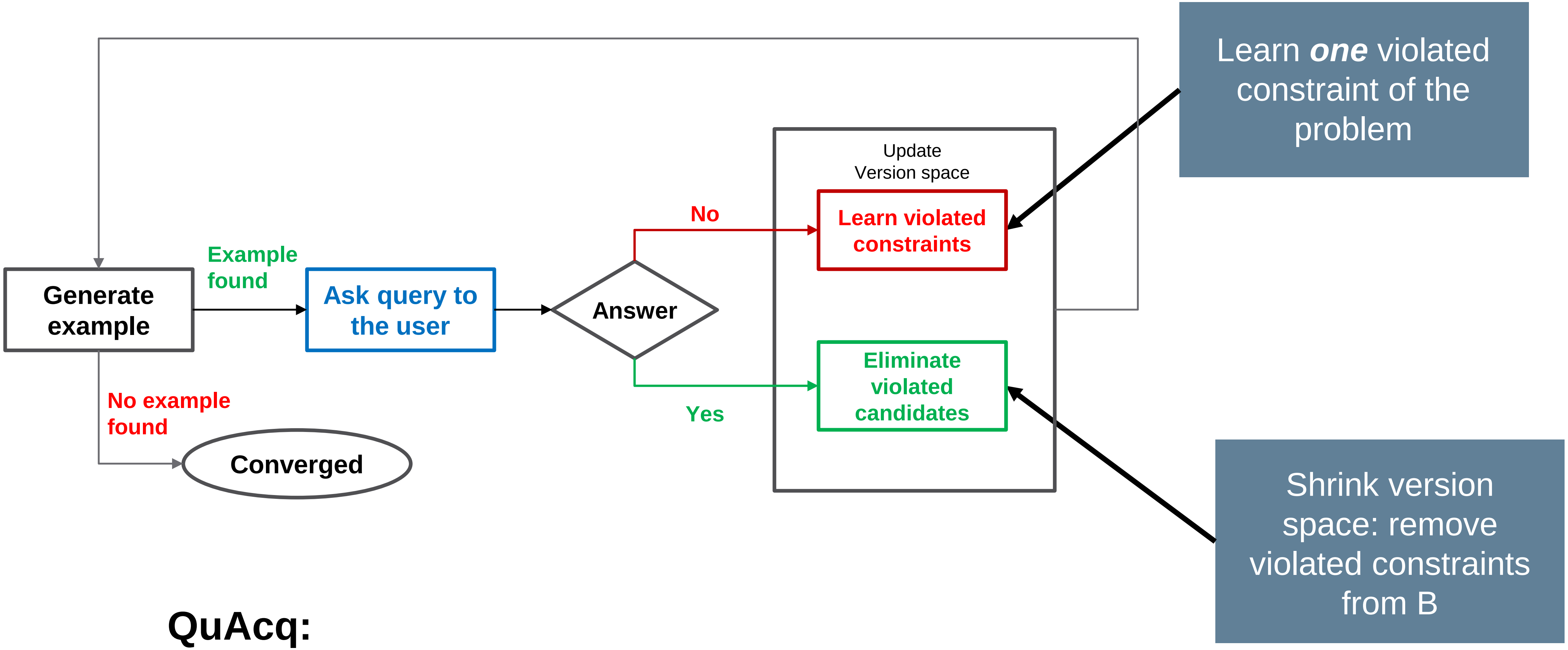
Remove $x_{1,1} < x_{2,2}$

Only $x_{1,1} \neq x_{2,2}$ left as candidate: Learn it
$C_L \leftarrow C_L \cup \{x_{1,1} \neq x_{2,2}\}$

Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

KU LEUVEN

Interactive Constraint Acquisition
QuAcq

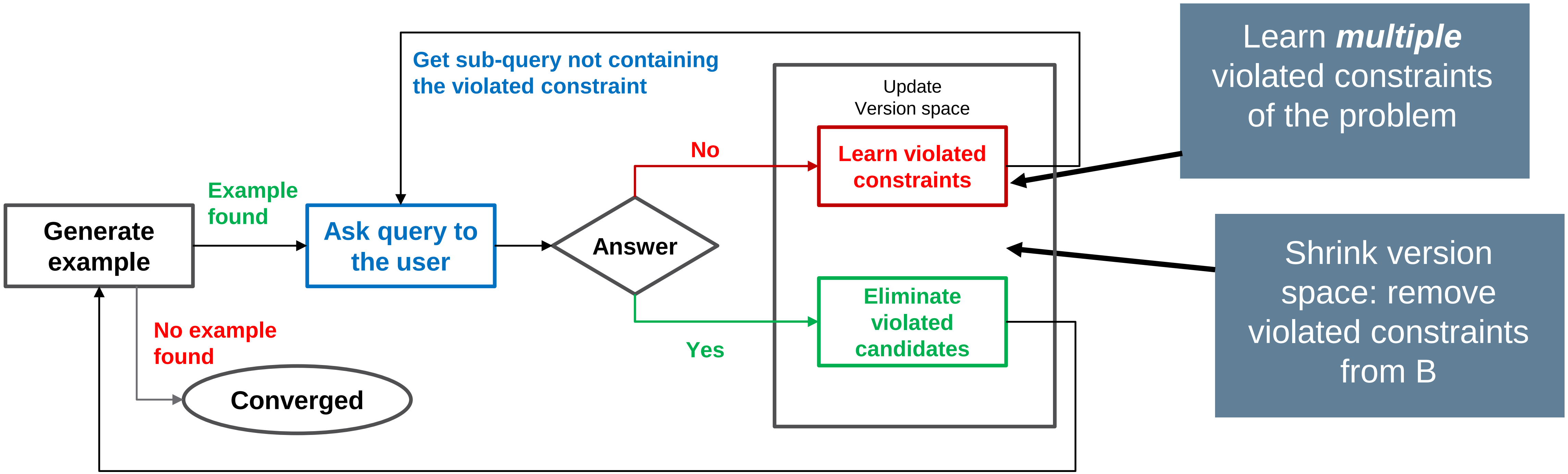


QuAcq:
- Learning one violated constraint per generated example
- Logarithmic number of queries for each constraint

Learning Constraints through Partial Queries, C. Bessiere et al., AIJ, 2023

Interactive Constraint Acquisition

Multiple Acquisition



Multiple Acquisition:

- Learn multiple constraints in each loop instance
- Don't generate a new example when a constraint is learnt
  - Instead, get an example in a subset of variables not violating the constraint found

Efficient Multiple Constraint Acquisition, D. Tsouros et al., Constraints, 2020

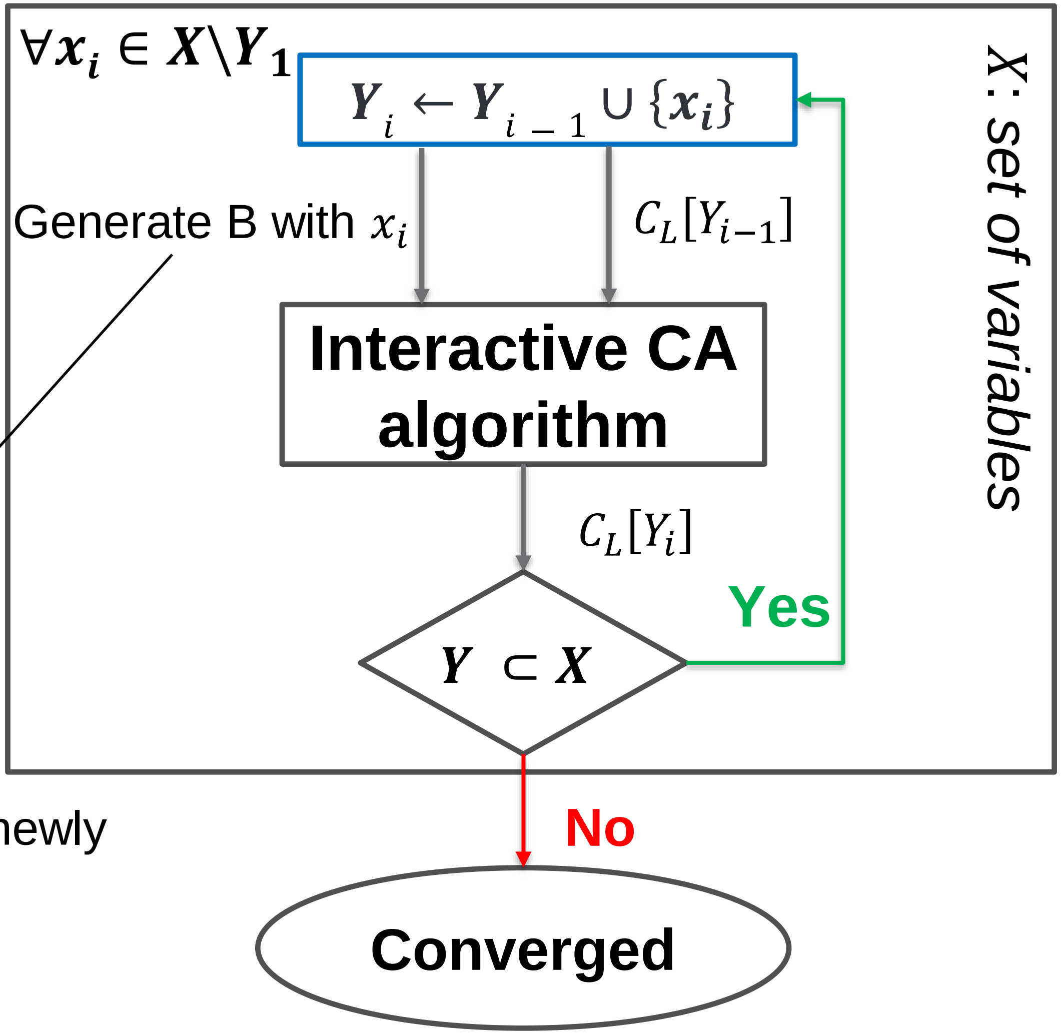
What if the set of candidates is too large??

- Can’t store all of it at the same time??
- Too slow query generation??

GrowAcq: Growing Acquisition

Start with $Y_1 \leftarrow \emptyset$, or a small subset of X

Bottom-up approach: Using Interactive CA algorithms, on an (incrementally growing) subset of variables

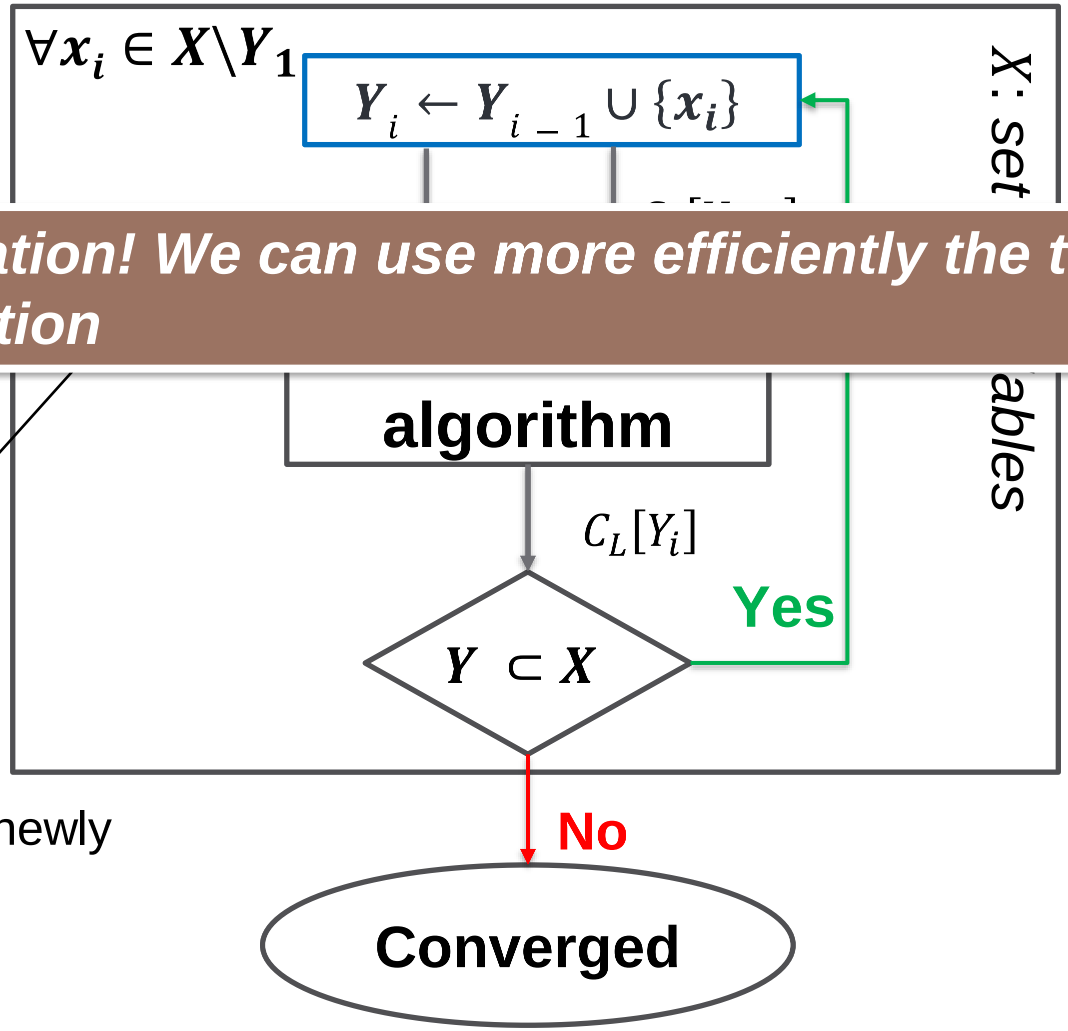


Need only constraints that newly added variable participates

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

GrowAcq: Growing Acquisition

Start with $Y_1 \leftarrow \emptyset$, or a small subset of X



Saves time during query generation! We can use more efficiently the time to guide better Constraint Acquisition

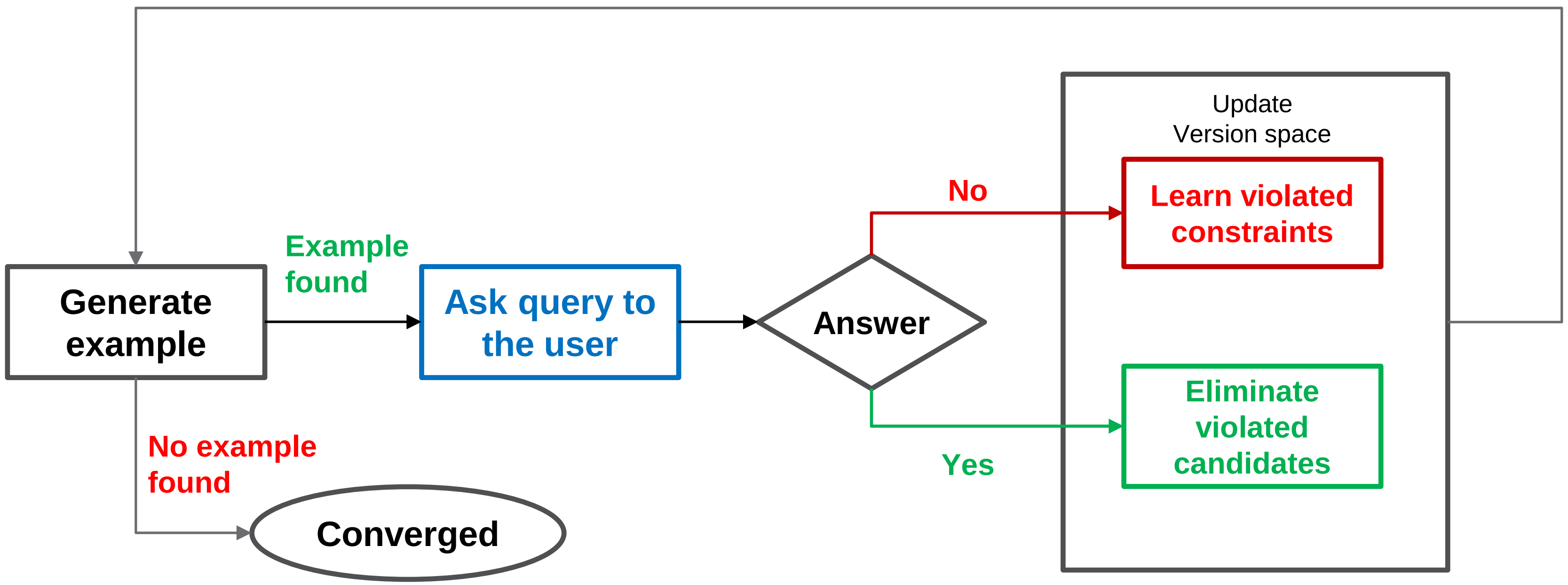
Bot

Inte... CA algorithms, on an (incrementally growing) subset of variables

Need only constraints that newly added variable participates

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

Interactive Constraint Acquisition
Small summary



We discussed:

- How Interactive CA works
- Query generation
- Finding the scope of problem constraints
- Finding the relation of the constraints
- Growing Acquisition

Guiding Interactive Constraint Acquisition

Guiding Query Generation

Quality of query

- *Better* generated examples lead to faster convergence
  - More information per query -> less queries needed

Typically: maximize candidate violations

$$\max \sum_{c \in B} \sim c$$

Not fully aligning with the goal!!

Guiding Query Generation

Better generated examples lead to faster convergence

Positive answers: shrink *B* fast

- Negative answers: Find the conflict fast

The more we have violated the faster B will shrink

The less candidates we have violated, the less queries we need to find the constraint(s)

$$\max \sum_{c \in B} [\![e \notin sol(c)]\!]$$

Opposite objectives based on the (future) answer

$$\min \sum_{c \in B} [\![e \notin sol(c)]\!]$$

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

KU LEUVEN

Guiding Query Generation

Better generated examples lead to faster convergence

Positive answers: shrink *B* fast

- Negative answers: Find the conflict fast

We cannot know the answer of the user before we ask the query → max violations

The ... faster B will shrink

The less candidates we have violated, the ... (and relation) of the constraint(s)

But what if we can predict if a candidate is a constraint of the problem or not?

$$\boldsymbol{max} \sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$$

Opposite objectives based on the (future) answer

$$\boldsymbol{min} \sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$$

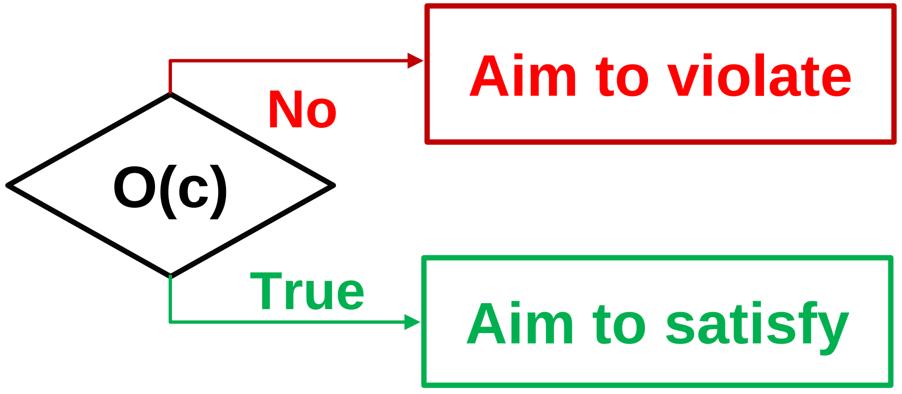
Guiding Query Generation

- Positive answers: shrink *B* fast → $\boldsymbol{max}\sum_{c\in B}[\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$
- Negative answers: Find the conflict fast → $\boldsymbol{min}\sum_{c\in B}[\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$

What if we can predict if a candidate is a constraint of the problem or not?

Use of *Oracle* $\mathrm{O(c)} = (\mathrm{c} \in CT)$, to guide query generation based on the *prediction of the constraint*

$$e = \underset{e\,\in Sol(C_L\ \wedge B)}{\operatorname{argmax}} \sum_{c\in B} [\![e \notin sol(c)]\!] \cdot (1 - |\Gamma| \cdot [\![O(c)]\!])$$



Why?

1. Aim for positive answers first: $\boldsymbol{max}\sum_{c\in B}[\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$
2. When a (probably true) constraint has to be violated, leading to a *negative answer* $\boldsymbol{min}\sum_{c\in B}[\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

Guiding Query Generation

- Positive answers: shrink *B* fast → $\boldsymbol{max} \sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$
- Negative answers: Find the conflict fast → $\boldsymbol{min} \sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!]$

What if we can predict if a candidate is a constraint of the problem or not?

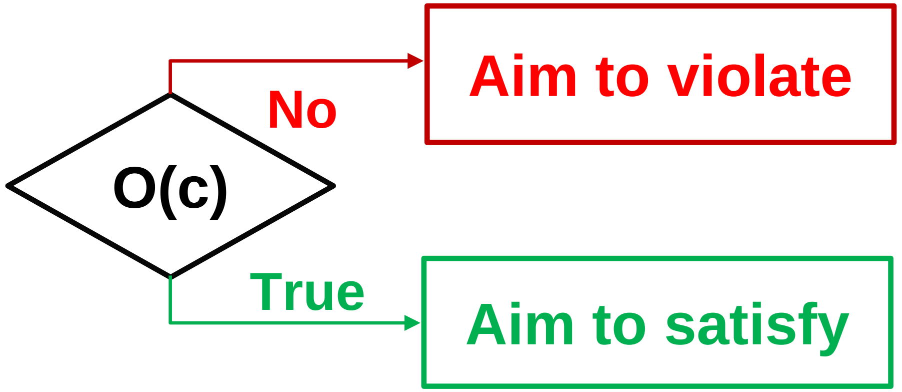
Use of *Oracle* $\mathrm{O(c)} = (\mathrm{c} \in CT)$, to guide query generation based on the *prediction of the constraint*

If the constraint *c* is violated

Increase objective value by 1

If it is a constraint predicted to be true: reduce objective value significantly

$$e = \underset{e \in Sol(C_L \ \wedge B)}{\operatorname{argmax}} \sum_{c \in B} \overbrace{[\![e \notin sol(c)]\!]} \cdot (\overbrace{1} - \overbrace{|\Gamma| \cdot [\![O(c)]\!]})$$



Why?

1. Aim for positive answers first: $\boldsymbol{max}(\sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!])$
2. When a (probably true) constraint has to be violated, leading to a *negative answer* $\boldsymbol{min}(\sum_{c \in B} [\![\boldsymbol{e} \notin \boldsymbol{sol}(\boldsymbol{c})]\!])$

Guided Bottom-up Constraint Acquisition, D. Tsouros et al., CP, 2023

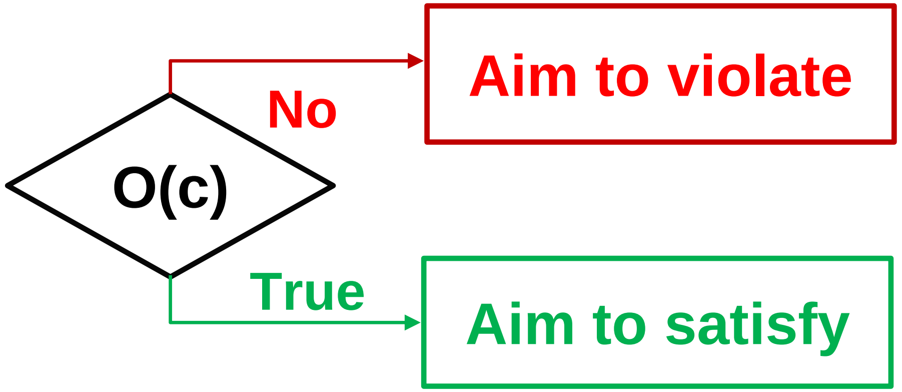
Guiding CA when finding the scope

Exploit partial (sub)queries to find the conflicting part

How are the removed variable assignments decided???

Follow the same logic
- But the new example is a sub-example of previous one
- Instead of deciding variable assignments, decide which variables to keep in the assignment

$$e_Y = \underset{e_Y \,|\, Y \subseteq var(e)}{\operatorname{argmax}} \sum_{c \in B} [\![e_Y \notin sol(c)]\!] \cdot (1 - |\Gamma| \cdot [\![O(c)]\!])$$



Guiding CA when finding the relation

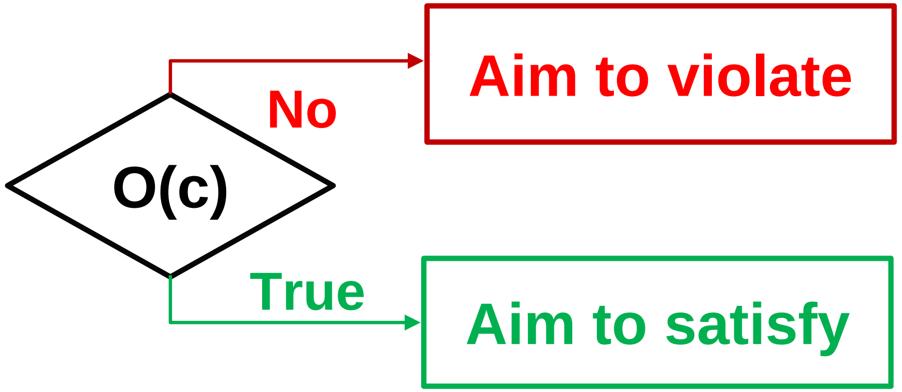
Try different assignments to find the specific constraint in the scope

How are the assignments decided???

Follow the same logic

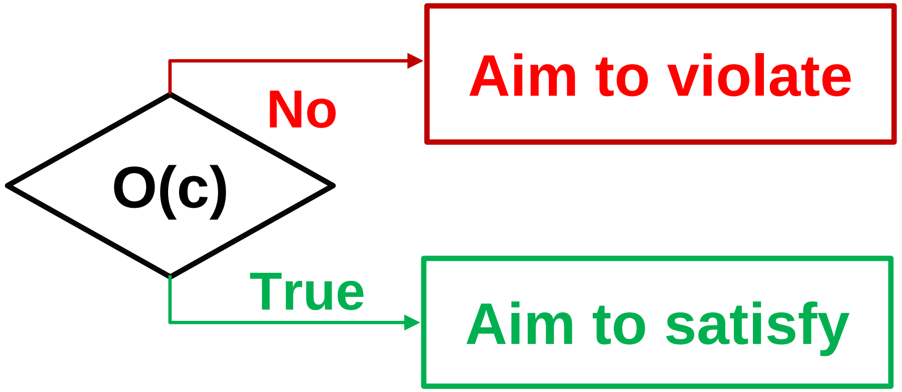
- But the new example has to be an assignment only to the scope S found

$$e_S = \underset{e_S \in Sol(C_L[S] \, \wedge B[S])}{\operatorname{argmax}} \sum_{c \in B} [\![e_S \notin sol(c)]\!] \cdot (1 - |\Gamma| \cdot [\![O(c)]\!])$$



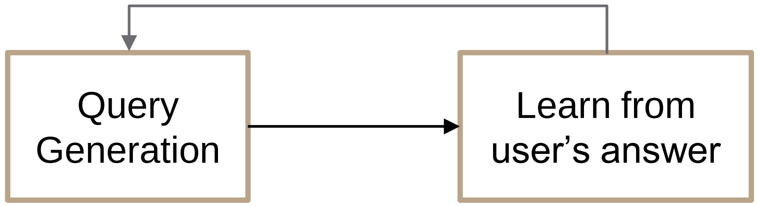
Do we have an oracle $O(c)$ to guide CA

The oracle $O(c)$ "classifies" a candidate as a problem constraint or not



It is a prediction problem

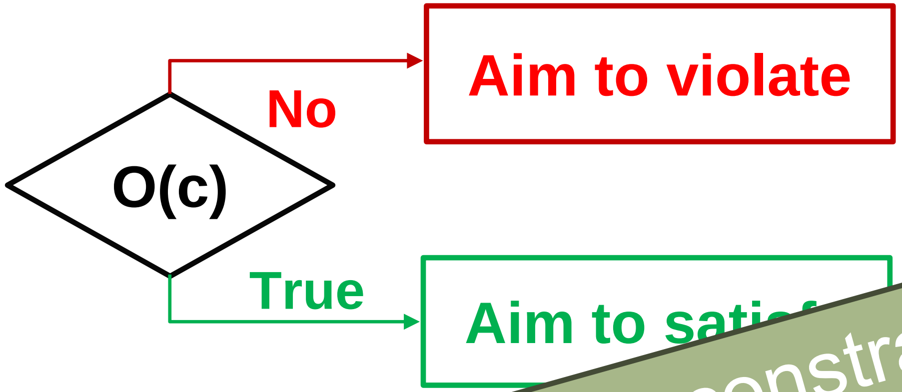
Use Machine Learning!!



KU LEUVEN

Do we have an oracle $O(c)$ to guide CA

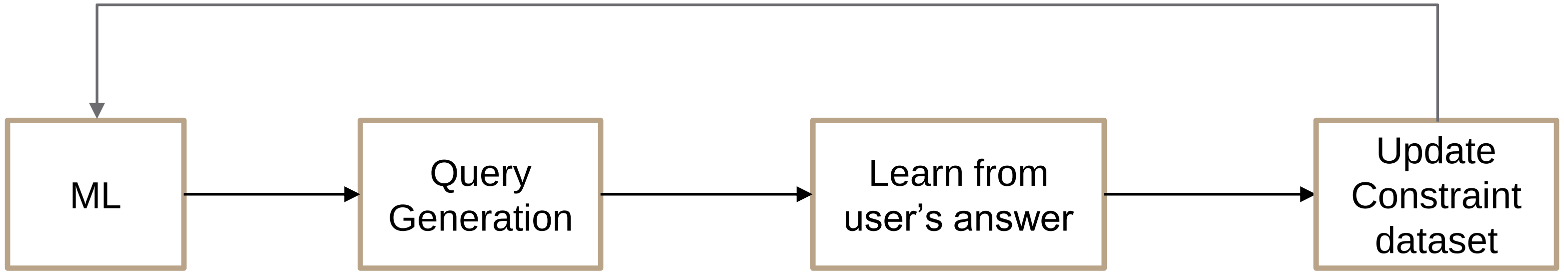
The oracle $O(c)$ "classifies" a candidate as a problem constraint or not



It is a ... problem

Predictions for constraints not for variable assignments!!

Use Machine Learning!!



Using Machine Learning for the prediction

Dataset: Constraint features and class (True or False)
- Constructing during the acquisition process
- Constraints that we know are part of the problem or not
- When a constraint is learned add a positive instance
- When a constraint is removed from *B add a negative instance*
- Use both relation and scope features

Relation-based features	Scope-based features
Relation name (string)	Dim[i] same_val (Bool)
Has constant (Bool)	Dim[i] avg (float)
Constant value (int)	Dim[i] distance (int)
Arity (int)	…

Using Machine Learning for the prediction

Dataset: Constraint features and class (True or False)

- Constructing during the acquisition process
- Constraints that we know are part of the problem or not
- When a constraint is learned add a positive instance
- When a constraint is removed from *B add a negative instance*
- Use both relation and scope features

Example for constraint $x_{1,1} \neq x_{1,2}$ *in Sudoku*

Relation-based features		Scope-based features	
Relation name (string)	≠	Dim[i] same_val (Bool)	True, False
Has constant (Bool)	False	Dim[i] avg (float)	1, 1.5
Constant value (int)	-1	Dim[i] distance (int)	0, 1
Arity (int)	2	…	

For the 2 dimensions

Using Machine Learning in the Oracle

The oracle $O(c)$ “classifies” a candidate as a problem constraint or not

Use of any classification technique to simulate the Oracle

$$O(c) = Class(c)$$ ← classifier.predict()

Use of probabilities?

$$O(c) = \frac{1}{\log(|Y|)} \leq P(c)$$ ← classifier.predict_proba()

Minimize the *expected* number of queries
$|Y|$: size of the example
$\log(|Y|)$: number of queries for each constraint when not guided
$\frac{1}{\log(|Y|)}$: Percentage of queries resulting on a constraint learnt

KU LEUVEN

Using Machine Learning in the Oracle

The oracle $O(c)$ "classifies" a candidate as a problem constraint or not

Use of any classification technique to simulate the Oracle

$$O(c) = Class(c)$$

classifier.predict()

Use of probabilities?

$$O(c) = \frac{1}{\log(|Y|)} \leq P(c)$$

classifier.predict_proba()

- Using predicted probabilities instead of class works way better!!
- Assuming that a candidate is a constraint of the problem even if the probability is less than 50% (but above the threshold)
- Threshold defined to minimize number of queries!

Minimize the *expected* n

$|Y|$: size of the example

$\log(|Y|)$: number of queries for each constraint when not guided

$\frac{1}{\log(|Y|)}$: Percentage of queries resulting on a constraint learnt

Open challenges

Challenges

Number of queries

- Number of queries needed to converge is still large.
- Use also more expressive types of queries
- Generalize!

Challenges

Specific classes of constraints

- **Global constraints:** Exploding the set of candidate constraints
- **Linear inequalities** with constants: Need to consider all possible constants -> Exploding the set of candidate constraints

COUNT

ALLDIFFERENT

CUMULATIVE

SUM

CIRCUIT

$x_1 + 5 < x_2$

$|x_1 + 12| > x_4$

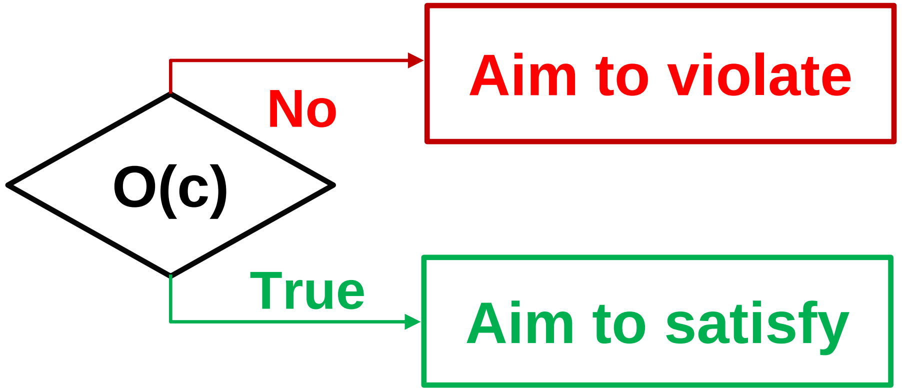
$x_1 - x_2 \mathrel{!=} 238$

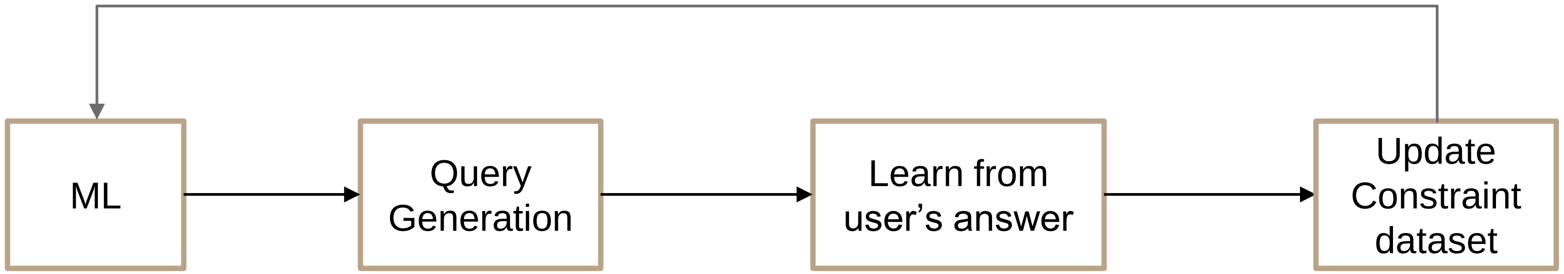
Challenges

Noisy data

- Unlike in machine learning, most constraint acquisition techniques still assume the user always (knows how to) answer correctly
- Tighter integration with modern machine learning techniques

Conclusions





We discussed:

- How to guide constraint acquisition using probabilities for the candidates
- Guiding all layers of constraint acquisition
- How to use Machine Learning for guiding
- Open challenges

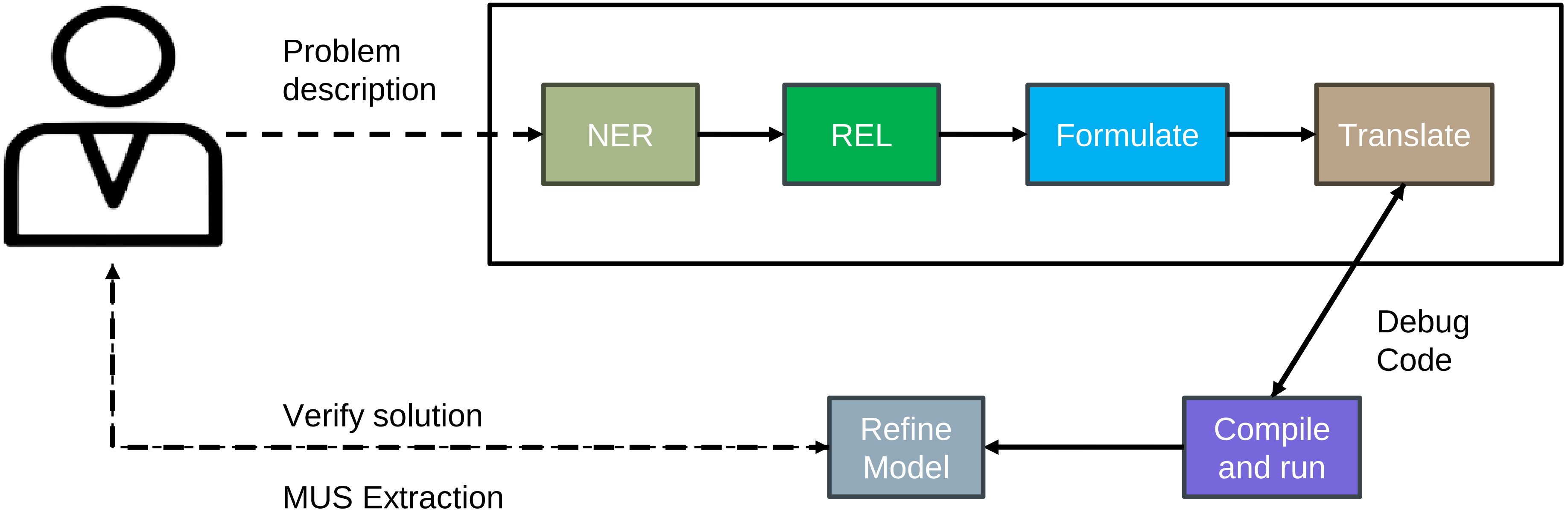
Holy grail 2.0

Acquire the model from text description!

Recent advances in LLMs and NL4OPT

Can we use this technology to acquire models directly from text description??

Holy grail 2.0



KU LEUVEN

Holy grail 2.0

I have only 10 hours to spend in Leuven. So I can only see a small number of attractions in the city. Visiting each attraction takes different amounts of time independent of the order in which it is visited, and they will give me different levels of happiness. Is it possible to come up with a list of attractions to visit to get maximum happiness in these 10 hours?

Parameters:

- ‘n‘: the number of attractions
- ‘t‘: a list of ‘n‘ integers representing the time it takes to visit each attraction
- ‘h‘: a list of ‘n‘ integers representing the happiness level of each attraction
- ‘max_time‘: the maximum time available to visit the attractions (in this case, 10 hours)

Variables:

- ‘x‘: a list of ‘n‘ binary variables, where ‘x[i] = 1‘ if attraction ‘i‘ is visited, and ‘x[i] = 0‘ otherwise.

Constraints:

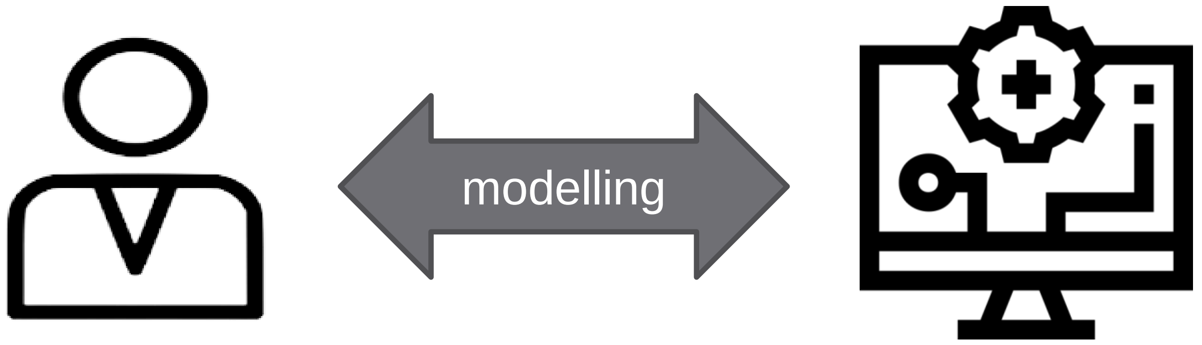
- The total time spent visiting the attractions cannot exceed ‘max_time‘: $\sum_{i=1}^{n} t_i x_i \leq max\_time$
- We can only visit each attraction at most once: $\sum_{i=1}^{n} x_i \leq 1$

Objective function:

- Maximize the total happiness level of the visited attractions: $\max \sum_{i=1}^{n} h_i x_i$

```
------------- Model in CPMpy: ---------------

from cpmpy import *

# Parameters
n = 5
t = [2, 3, 1, 4, 2]
h = [5, 7, 3, 8, 4]
max_time = 10

# Variables
x = boolvar(shape=n)

# Constraints
constraints = [
    # Total time spent visiting the attractions cannot exceed max_time
    sum(t[i]*x[i] for i in range(n)) <= max_time,
    # We can only visit each attraction at most once
    sum(x[i] for i in range(n)) <= 1
]

# Objective function
objective = maximize(sum(h[i]*x[i] for i in range(n)))

# Solve the problem
model = Model(constraints + [objective])
stats = model.solve()
```

Conclusions



We discussed:

- Background on constraint acquisition
- Passive and interactive constraint acquisition
- Approaches for passive acquisition
- How interactive constraint acquisition works
- Using Machine Learning to guide interactive constraint acquisition
- Open challenges

Thank you for your attention