BaxMC: a CEGAR approach to Max#SAT

Thomas Vigouroux Cristian Ene David Monniaux
Laurent Mounier Marie-Laure Potet

VERIMAG - UGA

October 20, 2022

fmcad.22

Introduction

Context

In security: the question of *quantitative reachability*.

```
atk = input()
if atk == x and x + y < XMAX:
  win() # trigger / exploit vulnerability
```

Introduction

Context

In security: the question of *quantitative reachability*.

```
atk = input()
if atk == x and x + y < XMAX:
  win() # trigger / exploit vulnerability
```

$$\phi \triangleq atk = x \wedge x + y \leq \texttt{XMAX}$$

Introduction

Context

In security: the question of *quantitative reachability*.

```
atk = input()
if atk == x and x + y < XMAX:
  win() # trigger / exploit vulnerability
```

$$\phi \triangleq atk = x \wedge x + y \leq \texttt{XMAX}$$

Question

Which answer (atk value) maximizes the winning probability ?

Introduction

Context

In security: the question of *quantitative reachability*.

```
atk = input()
if atk == x and x + y < XMAX:
  win() # trigger / exploit vulnerability
```

$$\phi \triangleq atk = x \wedge x + y \leq \texttt{XMAX}$$

Question

Which answer (atk value) maximizes the winning probability ?

Useful applications:

- Comparing bugs quantitatively
- Measure the information leakage of a program

Introduction

Definition of the problem

Definition (Max#SAT)

Given $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$, find a witness x_m that maximizes the following:

$$|\phi(x_m, \mathcal{Y}, \mathcal{Z})| \triangleq |\{y \mid \exists z \,.\, \phi(x_m, y, z)\}|$$

Example

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

Introduction

Definition of the problem

Definition (Max#SAT)

Given $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$, find a witness x_m that maximizes the following:

$$|\phi(x_m, \mathcal{Y}, \mathcal{Z})| \triangleq |\{y \mid \exists z \,.\, \phi(x_m, y, z)\}|$$

Example

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

Model x	Resulting formula	# Models over $\mathcal{Y}$
$a \quad \wedge \quad b$	$c \wedge d$	1

Introduction

Definition of the problem

Definition (Max#SAT)

Given $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$, find a witness x_m that maximizes the following:

$$|\phi(x_m, \mathcal{Y}, \mathcal{Z})| \triangleq |\{y \mid \exists z \,.\, \phi(x_m, y, z)\}|$$

Example

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

Model x	Resulting formula	# Models over $\mathcal{Y}$
$a \wedge b$	$c \wedge d$	1
$a \wedge \neg b$	c	1
$\neg a \wedge b$	$\bot$	0
$\neg a \wedge \neg b$	$\top$	2

Introduction

Definition of the problem

Definition (Max#SAT)

Given $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$, find a witness x_m that maximizes the following:

$$|\phi(x_m, \mathcal{Y}, \mathcal{Z})| \triangleq |\{y \mid \exists z \,.\, \phi(x_m, y, z)\}|$$

Example

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

Model x			Resulting formula	# Models over $\mathcal{Y}$
a	$\wedge$	b	$c \wedge d$	1
a	$\wedge$	$\neg b$	c	1
$\neg a$	$\wedge$	b	$\bot$	0
$\neg a$	$\wedge$	$\neg b$	$\top$	2

A CEGAR-based algorithm

Inspired by CEGAR methods: the formula is refined towards the search of better solutions.

Key Insight

It is possible to quickly overapproximate the maximum model counting of a subspace.

A CEGAR-based algorithm

Inspired by CEGAR methods: the formula is refined towards the search of better solutions.

Key Insight

It is possible to quickly overapproximate the maximum model counting of a subspace.

If this overapproximation is below the current known maximum $\rightarrow$ we can exclude the sub-space from future search.

A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

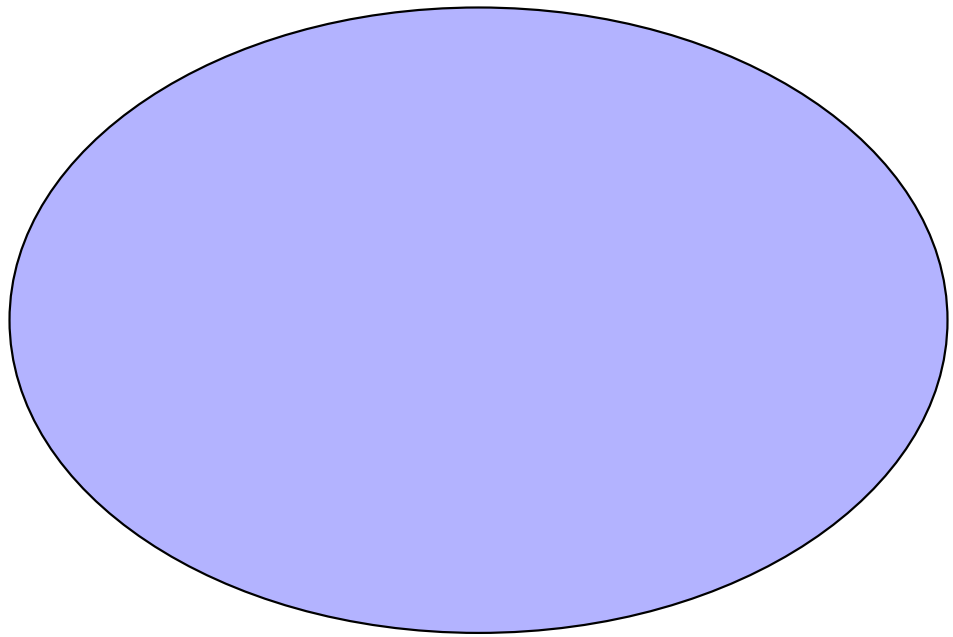
1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

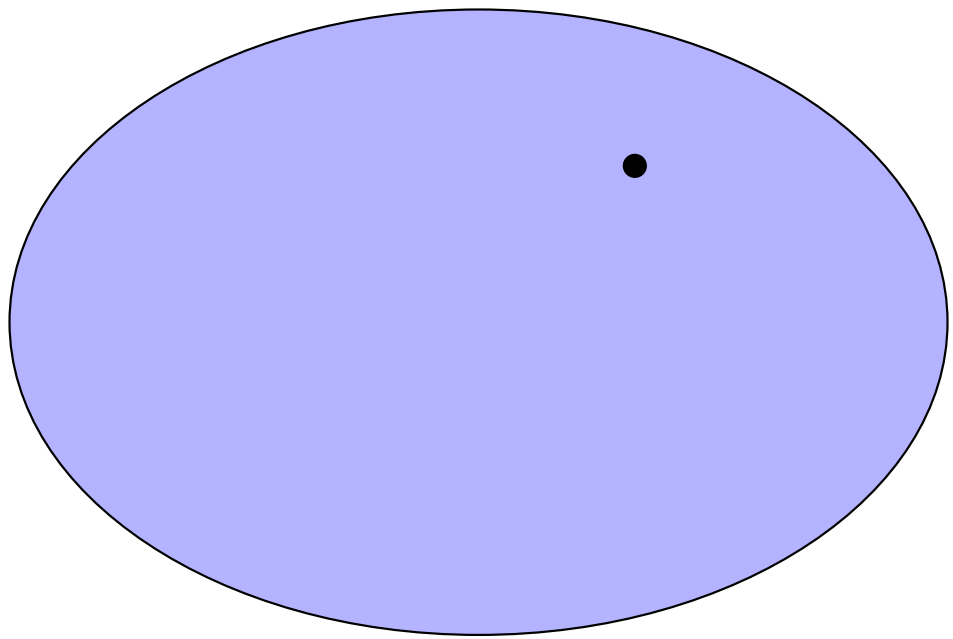


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

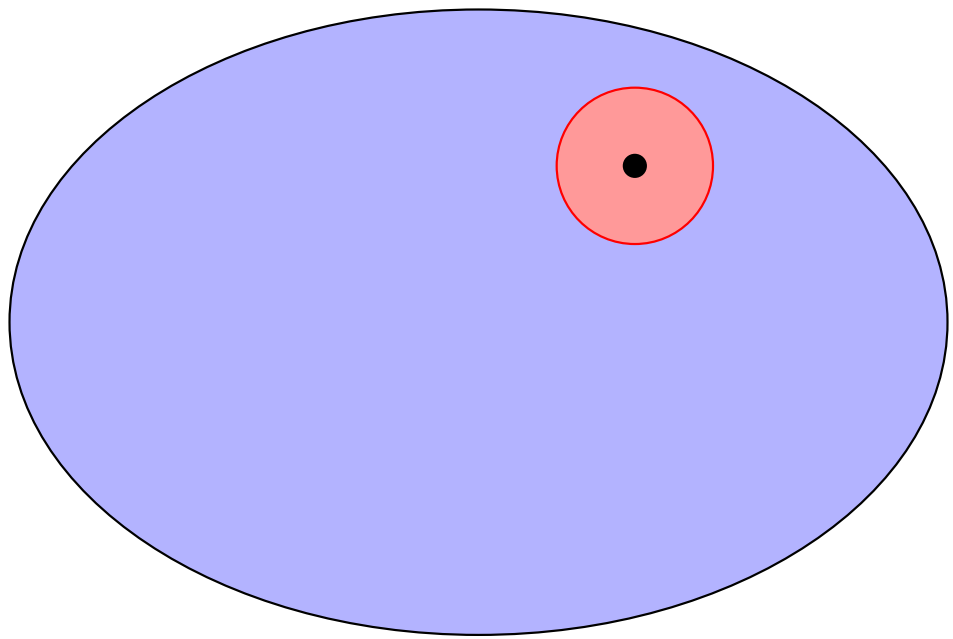


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

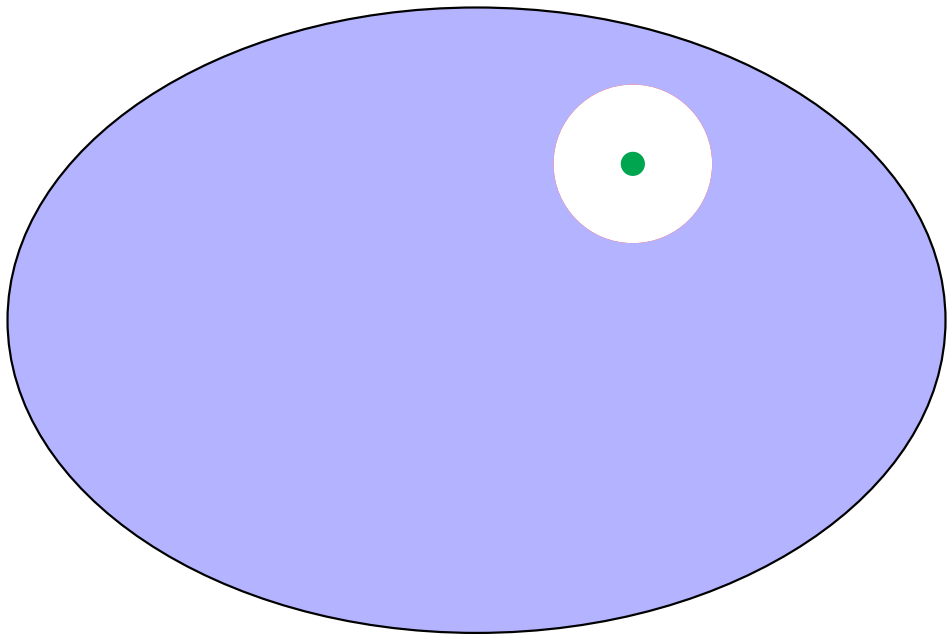


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

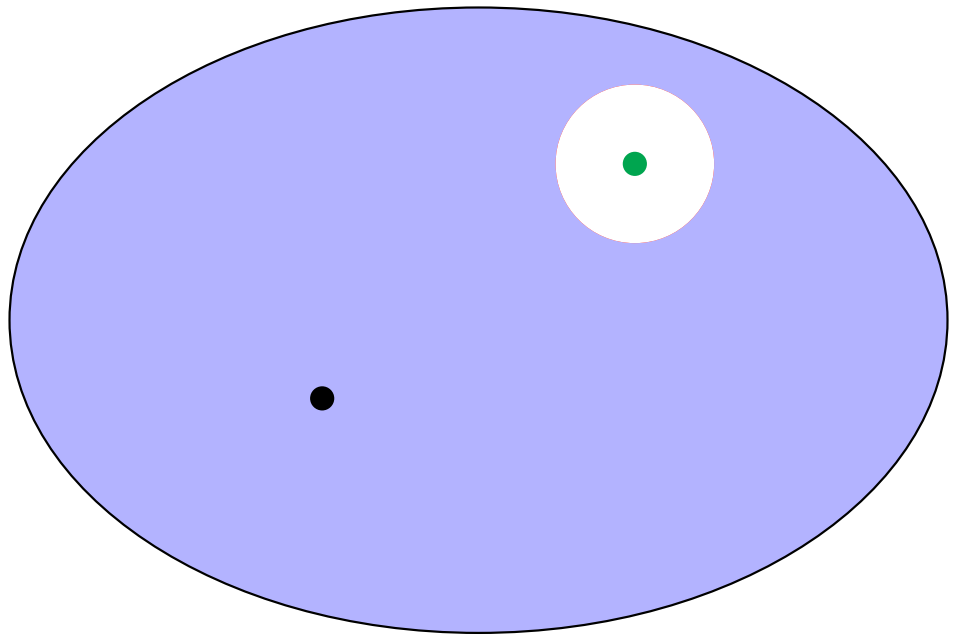


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

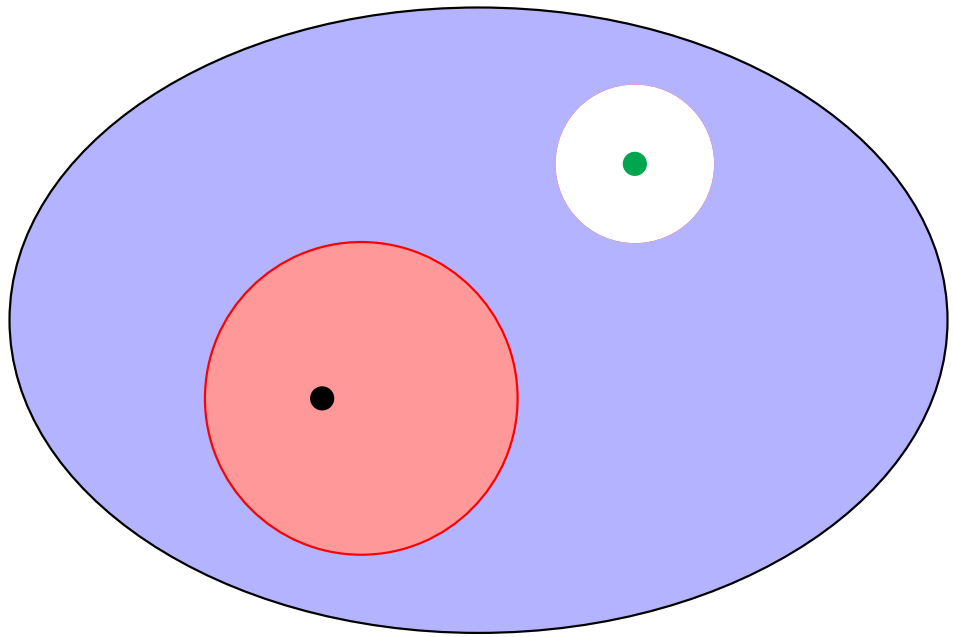


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

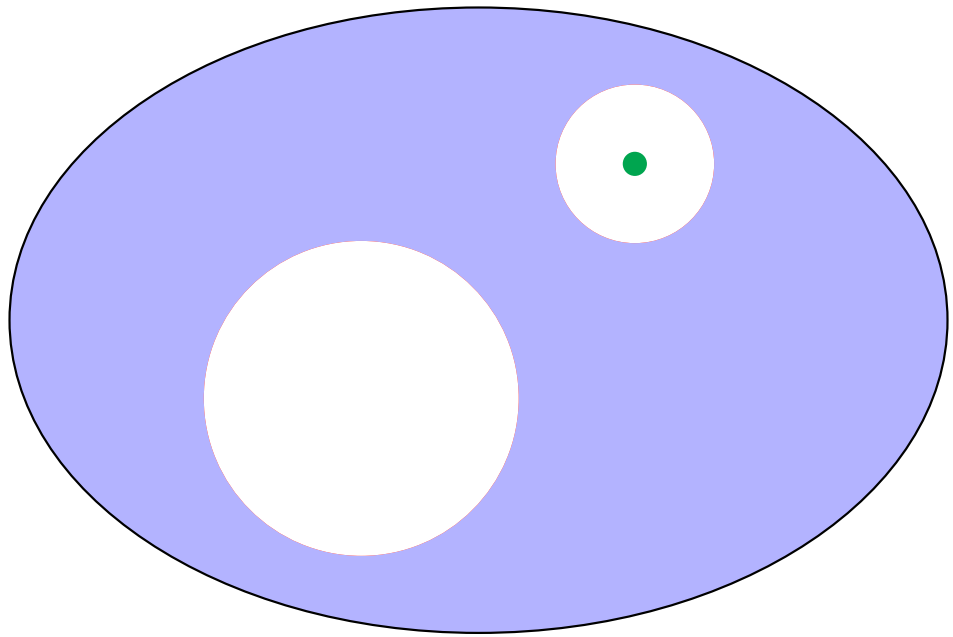


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

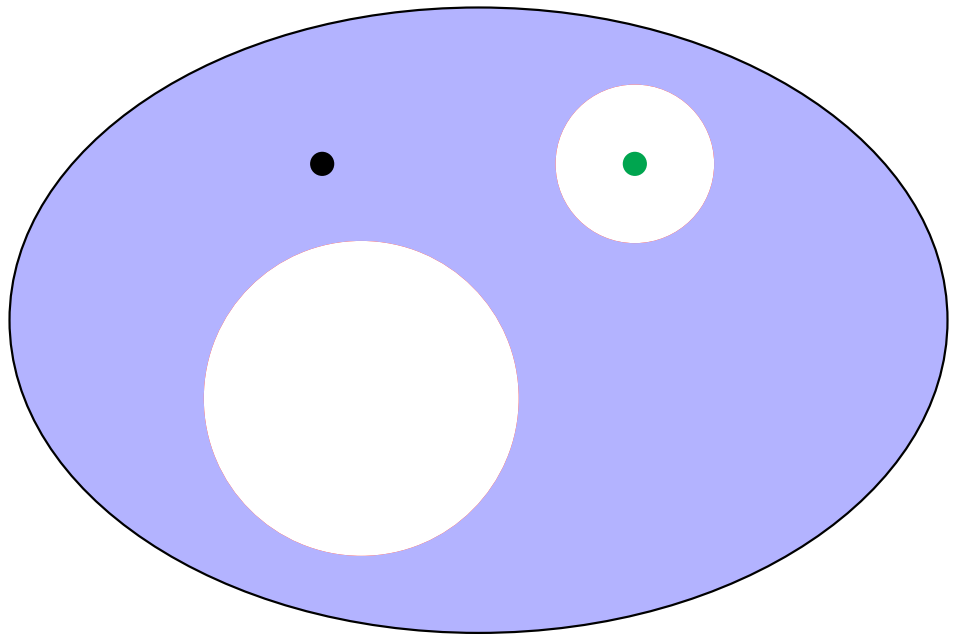


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

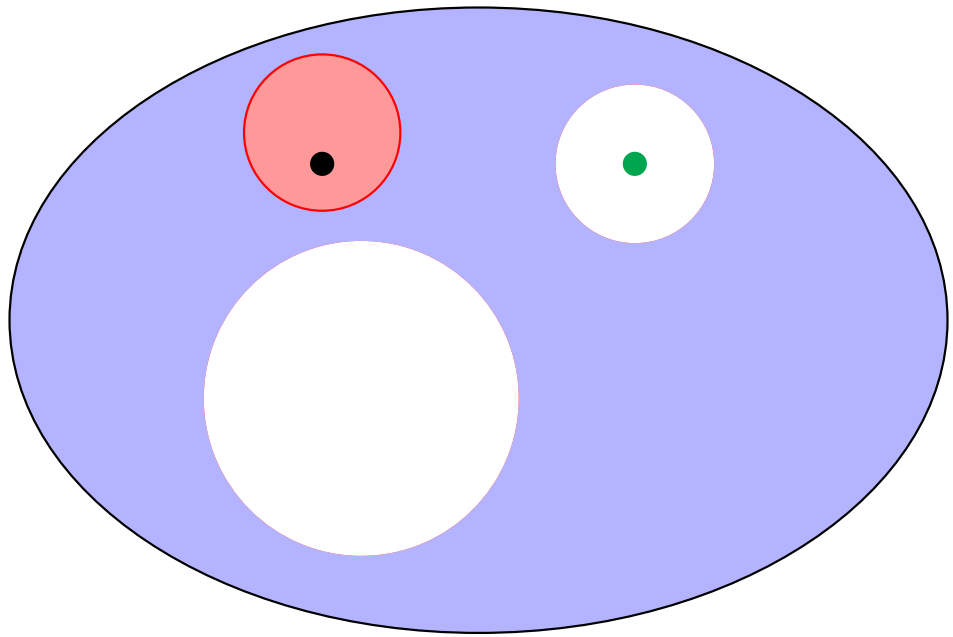


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop

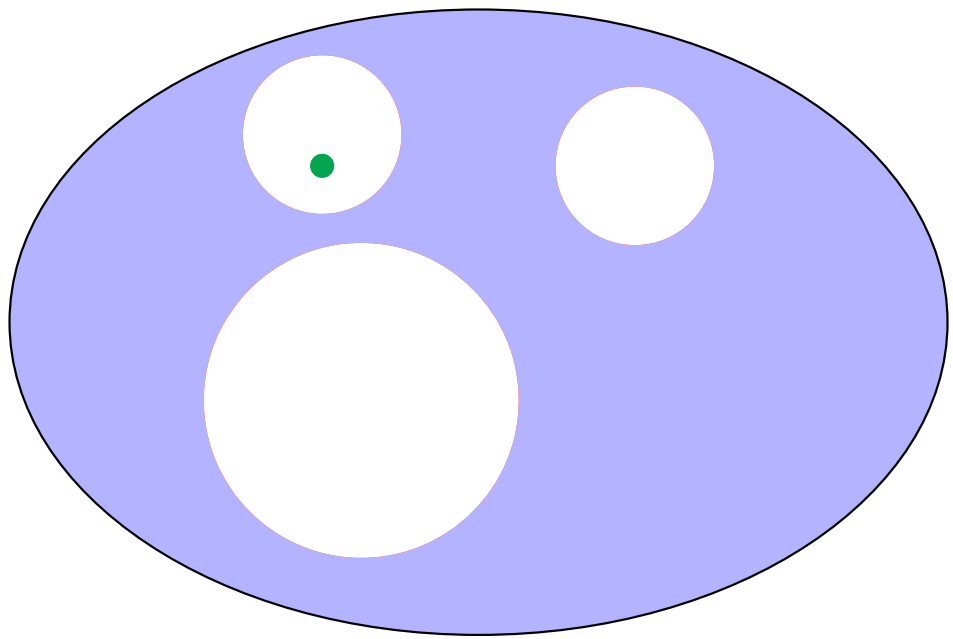


A CEGAR-based algorithm

Main algorithm

Takes $\phi(\mathcal{X}, \mathcal{Y}, \mathcal{Z})$ as input and returns x_m.

1. Find an x candidate
2. *Generalize* it to a partial witness $x|_{\mathcal{E}}$ such that $|\phi(x|_{\mathcal{E}}, \mathcal{Y}, \mathcal{Z})|$ is close (from below) to the current maximum
3. Exclude the sub-space $x|_{\mathcal{E}}$ from the future search
4. Loop



A CEGAR-based algorithm

Main algorithm

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

x	Generalization	Removed subspaces
$\neg a \wedge b$	$\neg a \wedge b$	$\{\neg a \wedge b\}$

A CEGAR-based algorithm

Main algorithm

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

x	Generalization	Removed subspaces
$\neg a \wedge b$	$\neg a \wedge b$	$\{\neg a \wedge b\}$
$a \wedge b$	a	$\{\neg a \wedge b, a\}$

A CEGAR-based algorithm

Main algorithm

With $\mathcal{X} = \{a, b\}$, $\mathcal{Y} = \{c\}$, $\mathcal{Z} = \{d\}$:

$$\phi = (\neg a \vee c) \wedge (a \vee \neg b) \wedge (\neg b \vee d)$$

x	Generalization	Removed subspaces
$\neg a \wedge b$	$\neg a \wedge b$	$\{\neg a \wedge b\}$
$a \wedge b$	a	$\{\neg a \wedge b, a\}$
$\neg a \wedge \neg b$	$\neg a \wedge \neg b$	$\{\neg a \wedge b, a, \neg a \wedge \neg b\}$

A CEGAR-based algorithm

Generalizing candidates

Key insight

Relaxing variables in a solution can only increase its model counting

A CEGAR-based algorithm

Generalizing candidates

Key insight

Relaxing variables in a solution can only increase its model counting

Trial and error generalization:

- Try to release one variable from the solution: $x|_{\mathcal{E}'}$
- If $|\phi(x|_{\mathcal{E}'}, \mathcal{Y}, \mathcal{Z})|$ stays below the current maximum: use $x|_{\mathcal{E}'}$ as the generalization from now on.

A CEGAR-based algorithm

Generalizing candidates

Generalization is an instance of a broader problem: *Minimal Subset to a Monotonous Predicate*.

Generic solvers exist:

- QuickXPlain[1]: recursively split the solution
- MergeXPlain[2]: extend QuickXPlain to return multiple solutions

[1] Junker, "QuickXplain: Conflict detection for arbitrary constraint propagation algorithms", 2001

[2] Shchekotykhin, Jannach, and Schmitz, "MergeXplain: Fast computation of multiple conflicts for diagnosis", 2015

A CEGAR-based algorithm

Generalizing candidates

Generalization is an instance of a broader problem: *Minimal Subset to a Monotonous Predicate*.

Generic solvers exist:

- QuickXPlain[1]: recursively split the solution
- MergeXPlain[2]: extend QuickXPlain to return multiple solutions

In our case, trial and error seems to be better.

[1] Junker, "QuickXplain: Conflict detection for arbitrary constraint propagation algorithms", 2001

[2] Shchekotykhin, Jannach, and Schmitz, "MergeXplain: Fast computation of multiple conflicts for diagnosis", 2015

A CEGAR-based algorithm

Theorems

Theorem (Exact #SAT oracle)

BaxMC *is correct, i.e. always returns the actual maximum.*

Theorem (Approximate #SAT oracle)

BaxMC *is probably approximately correct, i.e. returns the correct answer within a margin of* ϵ*, with a probability of* δ*.*
Assuming that the $\#SAT$ *oracle is correct within a margin of* ϵ' *with a probability of* δ'*.*

Related works

MaxCount[1]: amplification then sampling

`d4max`[2]: knowledge compilation to speed-up model counting

`erssat`[3]: more general problem, takes inspiration from *quantified boolean formulas*

[1] Fremont, Rabe, and Seshia, "Maximum model counting", 2017

[2] Audemard, Lagniez, and Miceli, "A New Exact Solver for (Weighted) Max#SAT", 2022

[3] Lee, Wang, and Jiang, "Solving Exist-Random Quantified Stochastic Boolean Satisfiability via Clause Selection", 2018

Some experimental results

Faster for approximate resolution on state-of-the-art benchmarks[1]:

Benchmark name	BaxMC	MaxCount
`backdoor-32-24*`	34.50	231.87
`backdoor-2x16-8*`	60.02	6512.28
`pwd-backdoor`	236.87	TO
`bin-search-16`	1048.43	1490.44
`CVE-2007-2875`	36.14	TO
`CVE-2009-3002`	TO	MO
`reverse`	TO	MO

Yields better results in nearly all cases.

[1] Fremont, Rabe, and Seshia, "Maximum model counting", 2017

Extensions of this algorithm

Implemented optimizations:

- Symmetry breaking
- Equivalent literals handling

Extensions of this algorithm

Implemented optimizations:

- Symmetry breaking
- Equivalent literals handling

Simple heuristic to prioritize search:

1. When overapproximating the count of a subspace, it may be above the current maximum
2. It may contain promising witnesses
3. Focus the future search in these areas

Future works

On BaxMC resolution:

- Incremental
- Parallel
- Extend to solve stochastic SAT[1] instances

[1] Lee, Wang, and Jiang, "Solving Exist-Random Quantified Stochastic Boolean Satisfiability via Clause Selection", 2018

Future works

On BaxMC resolution:

- Incremental
- Parallel
- Extend to solve stochastic SAT[1] instances

On optimizing search:

- Better symmetry breaking
- Leads scheduling

[1] Lee, Wang, and Jiang, "Solving Exist-Random Quantified Stochastic Boolean Satisfiability via Clause Selection", 2018

Future works

On BaxMC resolution:

- Incremental
- Parallel
- Extend to solve stochastic SAT[1] instances

On optimizing search:

- Better symmetry breaking
- Leads scheduling

Actually use this in a security setting.

[1] Lee, Wang, and Jiang, "Solving Exist-Random Quantified Stochastic Boolean Satisfiability via Clause Selection", 2018

Take-home message

- New solver for $\mathrm{Max\#SAT}$
- Two solving modes: exact and approximate
- Better experimental performances

Docker and sources available at:
`https://www-verimag.imag.fr/~vigourth/research/baxmc/`

@VERIMAG (Grenoble):

- Professorship
- Tenured full-time research positions
- Possibilities of post-docs

ask David Monniaux (in audience)

Bootstrapping the search

Idea:

1. Search for a good solution *fast* using a different method
2. Start BaxMC with the values found by the earlier method

Bootstrapping the search

Idea:

1. Search for a good solution *fast* using a different method
2. Start BaxMC with the values found by the earlier method

Consequences:

- The result is at least better
- Guarantees about the optimality of the results

References

- Audemard, Gilles, Jean-Marie Lagniez, and Marie Miceli. "A New Exact Solver for (Weighted) Max#SAT". In: *25th International Conference on Theory and Applications of Satisfiability Testing, SAT 2022, August 2-5, 2022, Haifa, Israel*. Ed. by Kuldeep S. Meel and Ofer Strichman. Vol. 236. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2022, 28:1–28:20. DOI: 10.4230/LIPIcs.SAT.2022.28. URL: https://doi.org/10.4230/LIPIcs.SAT.2022.28.
- Fremont, Daniel, Markus Rabe, and Sanjit Seshia. "Maximum model counting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1. 2017.
- Junker, Ulrich. "QuickXplain: Conflict detection for arbitrary constraint propagation algorithms". In: *IJCAI01 Workshop on Modelling and Solving problems with constraints*. Vol. 4. Citeseer. 2001.
- Lee, Nian-Ze, Yen-Shi Wang, and Jie-Hong R. Jiang. "Solving Exist-Random Quantified Stochastic Boolean Satisfiability via Clause Selection". In: *Proceedings of the Twenty-Seventh*