Abstraction-Based Algorithm for 2QBF

Mikoláš Janota[1]   Joao Marques-Silva[1,2]

[1] INESC-ID/IST, Lisbon, Portugal
[2] CASL/CSI, University College Dublin, Ireland

Definition

Given: $\exists X \forall Y.\phi$, where ϕ is a propositional formula
Question: Is there value vector ν such that $\forall Y.\phi[X/\nu]$?

Definition

Given: $\exists X \forall Y.\phi$, where ϕ is a propositional formula
Question: Is there value vector ν such that $\forall Y.\phi[X/\nu]$?

Note that ϕ is an arbitrary Boolean forumula, and hence, $\forall X \exists Y.\phi$ is solved by negating: $\neg\forall X \exists Y.\phi = \exists X \forall Y.\neg\phi$

Definition

Given: $\exists X \forall Y.\phi$, where ϕ is a propositional formula
Question: Is there value vector ν such that $\forall Y.\phi[X/\nu]$?

Note that ϕ is an arbitrary Boolean forumula, and hence, $\forall X \exists Y.\phi$ is solved by negating: $\neg\forall X \exists Y.\phi = \exists X \forall Y.\neg\phi$

Example

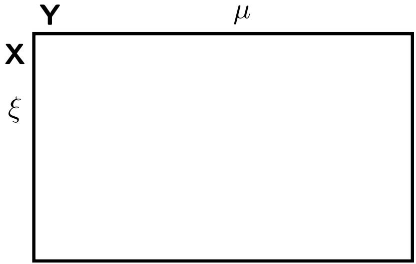
$$\exists x_1, x_2\ \forall y_1, y_2.\ (x_1 \vee x_2) \Rightarrow (y_1 \wedge y_2)$$

solution: $x_1 = 0, x_2 = 0$

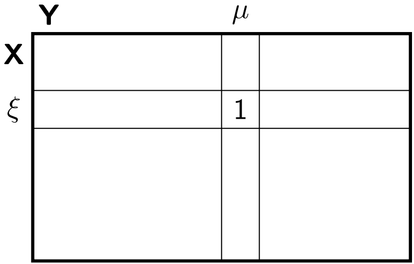
Motivation

- Σ_2^P, Π_2^P complete
- interesting problems in this class, e.g. propositional circumscription [Janota et al., 2010],
  AI [Remshagen and Truemper, 2005],
  LTS diameter [Mneimneh and Sakallah, 2003],
  MUS-membership [Janota and Marques-Silva, 2011]
- separate track at QBF Eval

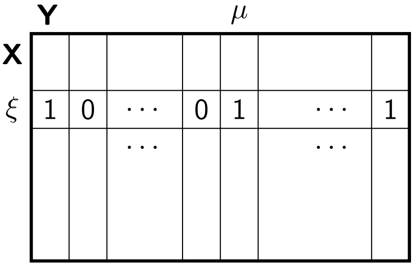
Looking at Valuations



Looking at Valuations



Looking at Valuations



Looking at Valuations

	Y				μ		
X							
ξ	1	0	$\cdots$	0	1	$\cdots$	1
			$\cdots$			$\cdots$	
ν	1	1	$\cdots$	1	1	$\cdots$	1

Looking at Valuations

X	Y				μ		
ξ	1	0	$\cdots$	0	1	$\cdots$	1
			$\cdots$			$\cdots$	
ν	1	1	$\cdots$	1	1	$\cdots$	1

$\phi[Y/\mu]$

Expanding into SAT

$$\exists X \forall Y.\ \phi \ \longrightarrow\ \mathsf{SAT}\left(\bigwedge_{\mu \in \mathcal{B}^{|Y|}} \phi[Y/\mu]\right)$$

Expanding into SAT

$$\exists X \forall Y.\ \phi \ \longrightarrow\ \mathsf{SAT}\left(\bigwedge_{\mu \in \mathcal{B}^{|Y|}} \phi[Y/\mu]\right)$$

Example

$$\exists x_1, x_2\ \forall y_1, y_2.\ (x_1 \vee x_2) \Rightarrow (y_1 \wedge y_2)$$

$$\begin{array}{ll} & (x_1 \vee x_2) \Rightarrow (0 \wedge 0) \\ \wedge & (x_1 \vee x_2) \Rightarrow (0 \wedge 1) \\ \wedge & (x_1 \vee x_2) \Rightarrow (1 \wedge 0) \\ \wedge & (x_1 \vee x_2) \Rightarrow (1 \wedge 1) \end{array}$$

Expanding into SAT

$$\exists X \forall Y.\ \phi \ \longrightarrow\ \mathsf{SAT}\left(\bigwedge_{\mu \in \mathcal{B}^{|Y|}} \phi[Y/\mu]\right)$$

Example

$$\exists x_1, x_2\ \forall y_1, y_2.\ (x_1 \vee x_2) \Rightarrow (y_1 \wedge y_2)$$

$$\begin{array}{ll} & \mathbf{(x_1 \vee x_2) \Rightarrow (0 \wedge 0)} \\ \wedge & (x_1 \vee x_2) \Rightarrow (0 \wedge 1) \\ \wedge & (x_1 \vee x_2) \Rightarrow (1 \wedge 0) \\ \wedge & (x_1 \vee x_2) \Rightarrow (1 \wedge 1) \end{array}$$

Abstraction

- Consider only some set of valuations $W \subseteq \mathcal{B}^{|Y|}$

$$\bigwedge_{\mu \in W} \phi[Y/\mu]$$

Abstraction

- Consider only some set of valuations $W \subseteq \mathcal{B}^{|Y|}$

$$\bigwedge_{\mu \in W} \phi[Y/\mu]$$

- Any solution to the problem is a solution to the abstraction

$$\bigwedge_{\mu \in \mathcal{B}^{|Y|}} \phi[Y/\mu] \;\Rightarrow\; \bigwedge_{\mu \in W} \phi[Y/\mu]$$

Abstraction

- Consider only some set of valuations $W \subseteq \mathcal{B}^{|Y|}$

$$\bigwedge_{\mu \in W} \phi[Y/\mu]$$

- Any solution to the problem is a solution to the abstraction

$$\bigwedge_{\mu \in \mathcal{B}^{|Y|}} \phi[Y/\mu] \;\Rightarrow\; \bigwedge_{\mu \in W} \phi[Y/\mu]$$

- But not the other way around, a solution to an abstraction is not necessarily a solution to the original problem.

CEGAR Loop

input : $\exists X \forall Y . \phi$
output: (**true**, ν) if there exists ν s.t. $\forall Y \phi[X/\nu]$,
(**false**, –) otherwise

```
W ← ∅
while true do
    (outc1, ν) ← SAT(⋀_{μ∈W} φ[Y/μ])          // find a candidate
    if outc1 = false then
        return (false,–)                        // no candidate found
    end
    if ν is a solution                          // solution check
    then
        return (true, ν)
    else
        Grow W                                  // refinement
    end
```

CEGAR Loop

input : $\exists X \forall Y . \phi$
output: (**true**, ν) if there exists ν s.t. $\forall Y \phi[X/\nu]$,
(**false**, –) otherwise

```
W ← ∅
while true do
    (outc1, ν) ← SAT(⋀_{μ∈W} φ[Y/μ])          // find a candidate
    if outc1 = false then
        return (false,–)                        // no candidate found
    end
    if ν is a solution                          // solution check
    then
        return (true, ν)
    else
        Grow W                                  // refinement
    end
```

Testing for Solution

A value ν is a solution to $\exists X \forall Y.\phi$ iff

$$\forall Y.\phi[X/\nu] \quad \textit{iff} \quad \text{UNSAT}(\neg\phi[X/\nu])$$

Testing for Solution

A value ν is a solution to $\exists X \forall Y.\phi$ iff

$$\forall Y.\phi[X/\nu] \quad \textit{iff} \quad \mathsf{UNSAT}(\neg\phi[X/\nu])$$

If $\mathsf{SAT}(\neg\phi[X/\nu])$ by some μ, then μ is a counterexample to ν

Testing for Solution

A value ν is a solution to $\exists X \forall Y.\phi$ iff

$$\forall Y.\phi[X/\nu] \;\; \textit{iff} \;\; \mathsf{UNSAT}(\neg\phi[X/\nu])$$

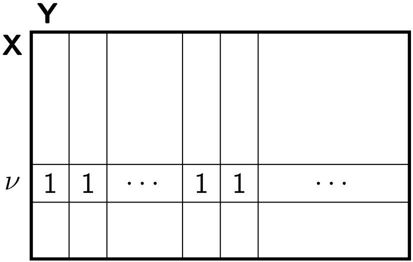
If $\mathsf{SAT}(\neg\phi[X/\nu])$ by some μ, then μ is a counterexample to ν

Example

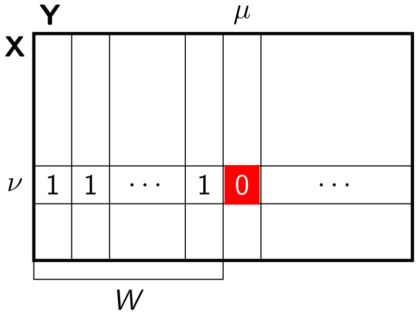
$\exists x_1, x_2 \; \forall y_1, y_2. \; (x_1 \vee x_2) \Rightarrow (y_1 \wedge y_2)$

- candidate: $x_1 = 1, x_2 = 0$
- counterexamples: $y_1 = 0, y_2 = 0$
  $y_1 = 0, y_2 = 1$
  $y_1 = 1, y_2 = 0$

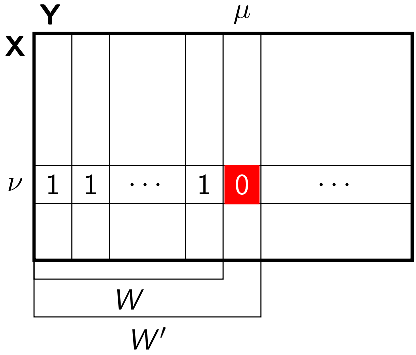
Refinement



Refinement



Refinement

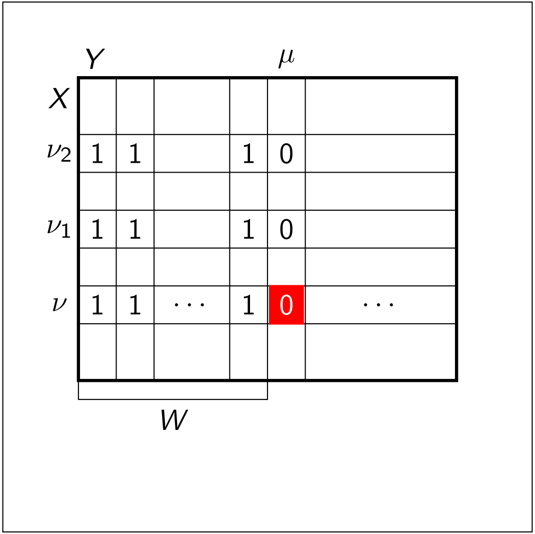


The Algorithm

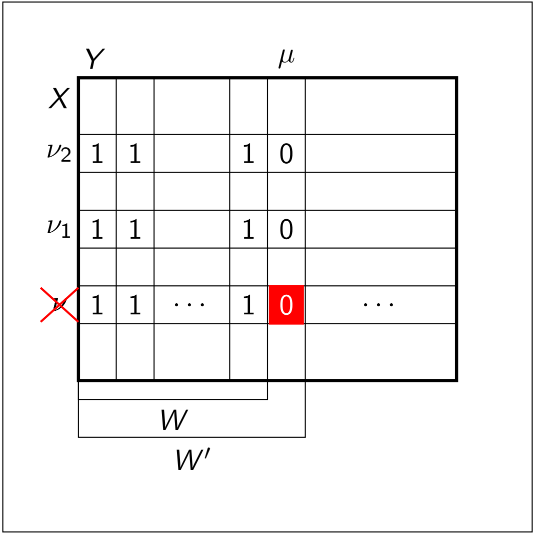
input : $\exists X \forall Y . \phi$
output: $(\textbf{true}, \nu)$ if there exists ν s.t. $\forall Y \phi[X/\nu]$,
$(\textbf{false}, -)$ otherwise

```
ω ← 1
while true do
    (outc₁, ν) ← SAT(ω)              // find a candidate solution
    if outc₁ = false then
        return (false,–)               // no candidate found
    end
    (outc₂, μ) ← SAT(¬φ[X/ν])        // find a counterexample
    if outc₂ = false then
        return (true, ν)               // candidate is a solution
    end
    ω ← ω ∧ φ[Y/μ]                     // refine
end
```

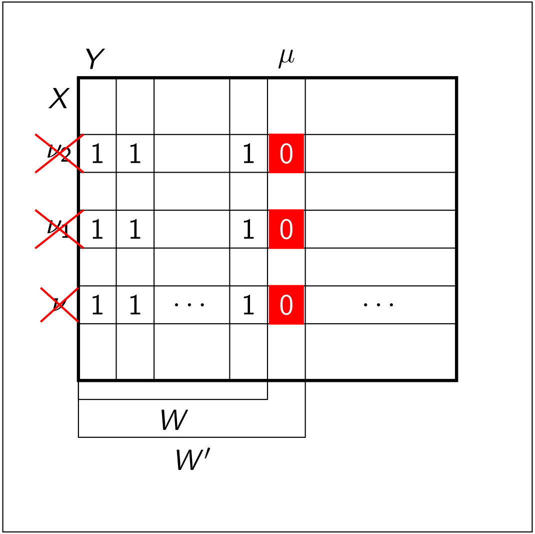
Properties of Refinement



Properties of Refinement



Properties of Refinement



Consequences of Refinement

- No candidate for counterexample appears more than once, therefore the upper bound on the number of iterations is:

$$\min(2^{|X|}, 2^{|Y|})$$

Consequences of Refinement

- No candidate for counterexample appears more than once, therefore the upper bound on the number of iterations is:

$$\min(2^{|X|}, 2^{|Y|})$$

- Heuristic: look for such counterexamples that are also counterexamples to many other candidates, look for μ s.t.

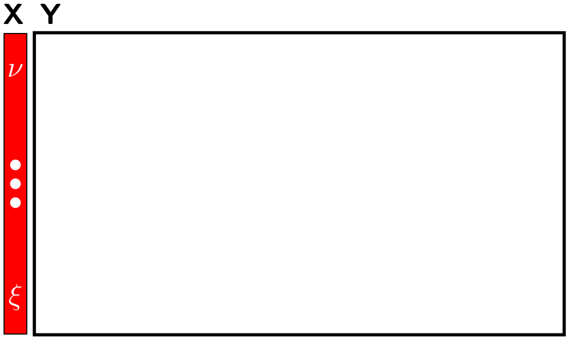
$$\neg\phi[X/\nu] \wedge \max\left(|\{\nu' \mid \neg\phi[X/\nu', Y/\mu]\}|\right)$$

Why the Choice of Counterexamples Matters?

- Consider an invalid QBF and nightmare vs. jackpot scenarios.

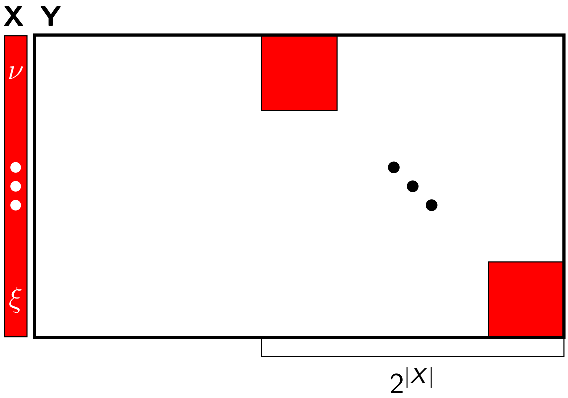
Why the Choice of Counterexamples Matters?

- Consider an invalid QBF and nightmare vs. jackpot scenarios.



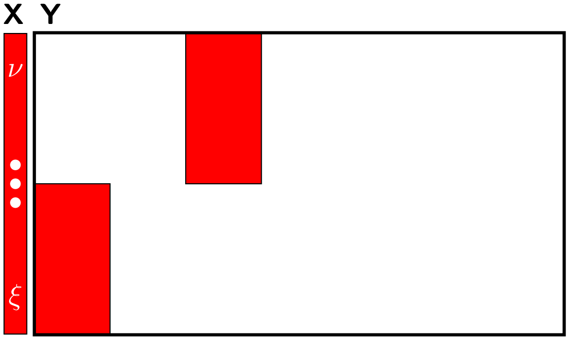
Why the Choice of Counterexamples Matters?

- Consider an invalid QBF and nightmare vs. jackpot scenarios.



Why the Choice of Counterexamples Matters?

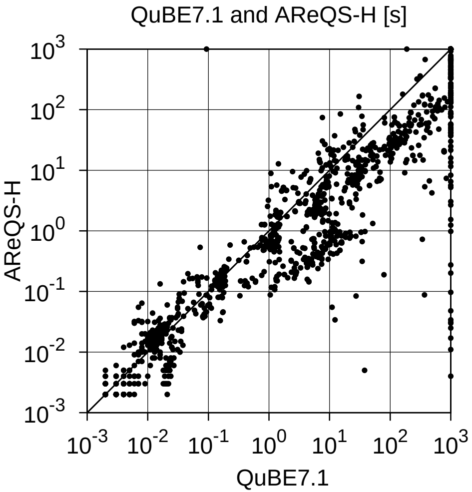
- Consider an invalid QBF and nightmare vs. jackpot scenarios.



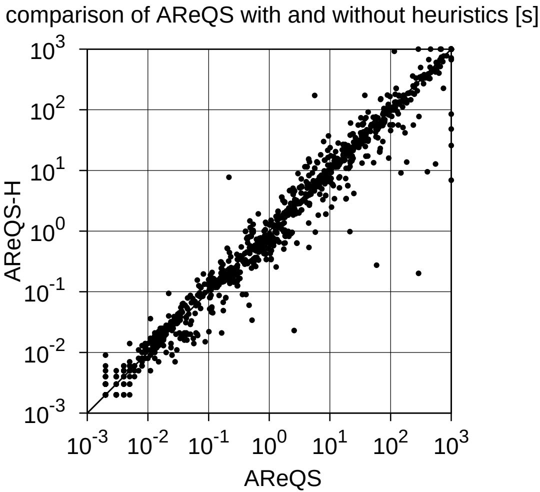
Results

	struqs	QuBE7.1	qbf2circ	AReQS	AReQS-H
2qbf 10 pre (114)	30	93	37	**101**	**101**
circ pre (117)	6	113	117	**117**	**117**
icore pre (140)	30	23	33	**62**	**62**
robots pre (999)	516	921	647	974	**975**
noprepro (232)	15	47	18	51	**55**
total (1602)	597	1197	852	1305	**1310**

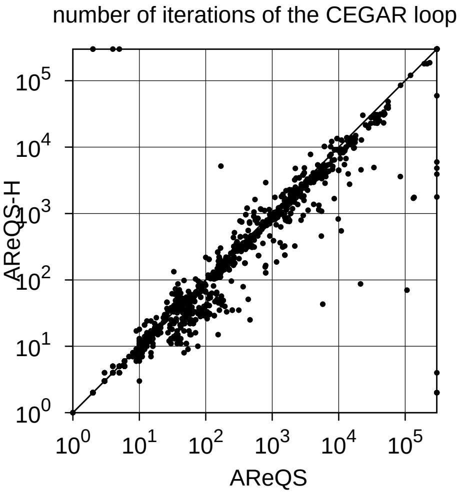
Results QuBE/AReQS-H



Results AReQS/AReQS-H



Results AReQS/AReQS-H Iterations



Conclusions

- We designed an algorithm for solving 2QBF, which is using SAT solver as an oracle.

Conclusions

- We designed an algorithm for solving 2QBF, which is using SAT solver as an oracle.
- The QBF is gradually extended to a SAT formula (exponential size)

Conclusions

- We designed an algorithm for solving 2QBF, which is using SAT solver as an oracle.
- The QBF is gradually extended to a SAT formula (exponential size)
- For a formula $\exists X \forall Y.\ \phi$ no valuation of X or Y repeats.

Conclusions

- We designed an algorithm for solving 2QBF, which is using SAT solver as an oracle.
- The QBF is gradually extended to a SAT formula (exponential size)
- For a formula $\exists X \forall Y.\ \phi$ no valuation of X or Y repeats.
- If $\neg\phi[X/nu, Y/mu]$, we never try ν' s.t. $\neg\phi[X/nu', Y/mu]$,

Conclusions

- We designed an algorithm for solving 2QBF, which is using SAT solver as an oracle.
- The QBF is gradually extended to a SAT formula (exponential size)
- For a formula $\exists X \forall Y.\ \phi$ no valuation of X or Y repeats.
- If $\neg\phi[X/nu, Y/mu]$, we never try ν' s.t. $\neg\phi[X/nu', Y/mu]$,
- It is to be expected that the algorithm will work well for formulas where counterexamples takes out many candidates.

Conclusions

- We designed an algorithm for solving 2QBF, which is using SAT solver as an oracle.
- The QBF is gradually extended to a SAT formula (exponential size)
- For a formula $\exists X \forall Y.\ \phi$ no valuation of X or Y repeats.
- If $\neg\phi[X/nu, Y/mu]$, we never try ν' s.t. $\neg\phi[X/nu', Y/mu]$,
- It is to be expected that the algorithm will work well for formulas where counterexamples takes out many candidates.
- A QCNF implementation of the algorithm consistently outperforms current solvers.

Janota, M., Grigore, R., and Marques-Silva, J. (2010).
Counterexample guided abstraction refinement algorithm for propositional circumscription.
In *JELIA '10*.

Janota, M. and Marques-Silva, J. (2011).
On deciding MUS membership with qbf.
In *CP '11, to appear*.

Mneimneh, M. N. and Sakallah, K. A. (2003).
Computing vertex eccentricity in exponentially large graphs: QBF formulation and solution.
In *SAT '03*.

Remshagen, A. and Truemper, K. (2005).
An effective algorithm for the futile questioning problem.
JAR '05.