

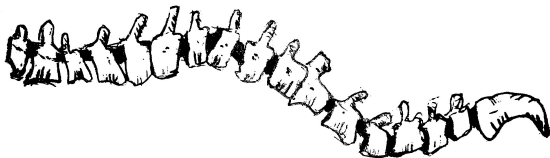
On Computing Backbones of Propositional Theories

Joao Marques-Silva¹ **Mikoláš Janota**² Inês Lynce³

¹ CASL/CSI, University College Dublin, Ireland

² INESC-ID, Lisbon, Portugal

³ INESC-ID/IST, Lisbon, Portugal



Backbones

- **Backbones** of propositional theories are literals that are true in every model.

Backbones

- **Backbones** of propositional theories are literals that are true in every model.

...	x_j	...	x_k	...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$

Backbones

- **Backbones** of propositional theories are literals that are true in every model.

...	x_j	...	x_k	...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n

Backbones

- **Backbones** of propositional theories are literals that are true in every model.

...	x_j	...	x_k	...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n

Backbones

- **Backbones** of propositional theories are literals that are true in every model.

...	x_j	...	x_k	...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n

Backbones

- **Backbones** of propositional theories are literals that are true in every model.

...	x_j	...	x_k	...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n
...	x_j	...	$\neg x_k$...	$\neg x_n$
...	x_j	...	$\neg x_k$...	x_n

$$\phi \models x_j$$

$$\phi \models \neg x_k$$

Motivation

- backbones tell us more about the formula, e.g.
 - upper bound for number of models

2^{n-k} , where n #variables and k #backbones


Motivation

- backbones tell us more about the formula, e.g.
 - upper bound for number of models

2^{n-k} , where n #variables and k #backbones

- product configuration

gas-engine \vee electric-engine
electric-engine \Rightarrow automatic
 \neg automatic \vee \neg manual



Motivation

- backbones tell us more about the formula, e.g.
 - upper bound for number of models

2^{n-k} , where n #variables and k #backbones

- product configuration

gas-engine \vee electric-engine
electric-engine \Rightarrow automatic
 \neg automatic \vee \neg manual
electric-engine

}

Motivation

- backbones tell us more about the formula, e.g.
 - upper bound for number of models

2^{n-k} , where n #variables and k #backbones

- product configuration

gas-engine \vee electric-engine	}	automatic, \neg manual
electric-engine \Rightarrow automatic		
\neg automatic \vee \neg manual		
electric-engine		

Motivation

- Can we compute backbones for large instances?
- How many backbone literals do real-world instances have?

Armory

- We use a satisfiability (SAT) solver as a blackbox

Armory

- We use a satisfiability (SAT) solver as a blackbox

$$\text{SAT}(x \vee y) = (\text{true}, \{x, \neg y\})$$

Armory

- We use a satisfiability (SAT) solver as a blackbox

$$\text{SAT}(x \vee y) = (\text{true}, \{x, \neg y\})$$

$$\text{SAT}(x \wedge \neg x) = (\text{false}, -)$$

Model Enumeration

Input : CNF formula φ

Output: Backbone of φ , ν_R

```
1  $\nu_R \leftarrow \{\neg x, x \mid x \in X\}$            // Initial backbone estimate
2 repeat
3    $(\text{outc}, \nu) \leftarrow \text{SAT}(\varphi)$        // SAT solver call
4   if outc = false then
5     return  $\nu_R$                           // Terminate if unsatisfiable
6    $\nu_R \leftarrow \nu_R \cap \nu$            // Update backbone estimate
7    $\omega_B \leftarrow \text{BlockClause}(\nu)$     // Block model
8    $\varphi \leftarrow \varphi \cup \omega_B$ 
9 until  $\nu_R = \emptyset$ 
10 return  $\emptyset$ 
```


Iterative SAT Testing

- Can we decide whether I is a backbone using a SAT solver?

Iterative SAT Testing

- Can we decide whether I is a backbone using a SAT solver?

$$\phi \models I$$

Iterative SAT Testing

- Can we decide whether I is a backbone using a SAT solver?

$$\phi \models I \quad \text{iff} \quad \text{UNSAT}(\phi \wedge \bar{I})$$

Iterative SAT Testing

- Can we decide whether I is a backbone using a SAT solver?

$$\phi \models I \quad \text{iff} \quad \text{UNSAT}(\phi \wedge \bar{I})$$

$$\phi \models x$$

Iterative SAT Testing

- Can we decide whether I is a backbone using a SAT solver?

$$\phi \models I \quad \text{iff} \quad \text{UNSAT}(\phi \wedge \bar{I})$$

$$\phi \models x \quad \text{iff} \quad \text{UNSAT}(\phi \wedge \neg x)$$

Iterative SAT Testing

Input : CNF formula φ , with variables X

Output: Backbone of φ , ν_R

```
1  $\nu_R \leftarrow \emptyset$ 
2 foreach  $I \in \{\neg x, x \mid x \in X\}$  do
3    $(\text{outc}, \nu) \leftarrow \text{SAT}(\varphi \cup \{\bar{I}\})$ 
4   if  $\text{outc} = \text{false}$  then
5      $\nu_R \leftarrow \nu_R \cup \{I\}$  //  $I$  is backbone
6      $\varphi \leftarrow \varphi \cup \{I\}$ 
7 return  $\nu_R$ 
```

Iterative SAT Testing

Input : CNF formula φ , with variables X

Output: Backbone of φ , ν_R

```
1  $\nu_R \leftarrow \emptyset$ 
2 foreach  $l \in \{\neg x, x \mid x \in X\}$  do
3    $(\text{outc}, \nu) \leftarrow \text{SAT}(\varphi \cup \{\bar{l}\})$ 
4   if  $\text{outc} = \text{false}$  then
5      $\nu_R \leftarrow \nu_R \cup \{l\}$  //  $l$  is backbone
6      $\varphi \leftarrow \varphi \cup \{l\}$ 
7 return  $\nu_R$ 
```

- SAT is called twice per variable

Observations I.

- if ν is a model of ϕ and $\nu \models I$ then \bar{I} is **not** a backbone

Observations I.

- if ν is a model of ϕ and $\nu \models I$ then \bar{I} is **not** a backbone

...	x_i	...
...	$\neg x_i$...
\vdots	\vdots	\vdots

Observations I.

- if ν is a model of ϕ and $\nu \models l$ then \bar{l} is **not** a backbone

...	x_i	...	$\phi \not\models x_i$
...	$\neg x_i$...	
\vdots	\vdots	\vdots	

Observations II.

- **Implicant** $\nu = l_1 \wedge \dots \wedge l_k$ is a conjunction of literals such that

$$\nu \Rightarrow \phi$$

Observations II.

- **Implicant** $\nu = l_1 \wedge \dots \wedge l_k$ is a conjunction of literals such that

$$\nu \Rightarrow \phi$$

- Any literal **not** appearing in ν is **not** a backbone.

Observations II.

- **Implicant** $\nu = l_1 \wedge \dots \wedge l_k$ is a conjunction of literals such that

$$\nu \Rightarrow \phi$$

- Any literal **not** appearing in ν is **not** a backbone.

$$(x \vee y) \wedge (x \vee \neg y)$$

Observations II.

- **Implicant** $\nu = l_1 \wedge \dots \wedge l_k$ is a conjunction of literals such that

$$\nu \Rightarrow \phi$$

- Any literal **not** appearing in ν is **not** a backbone.

$$x \Rightarrow ((x \vee y) \wedge (x \vee \neg y))$$

Observations II.

- **Implicant** $\nu = l_1 \wedge \dots \wedge l_k$ is a conjunction of literals such that

$$\nu \Rightarrow \phi$$

- Any literal **not** appearing in ν is **not** a backbone.

$$x \Rightarrow ((x \vee y) \wedge (x \vee \neg y))$$

x	y
x	$\neg y$
x	y
\vdots	\vdots

Observations II.

- **Implicant** $\nu = l_1 \wedge \dots \wedge l_k$ is a conjunction of literals such that

$$\nu \Rightarrow \phi$$

- Any literal **not** appearing in ν is **not** a backbone.

$$x \Rightarrow ((x \vee y) \wedge (x \vee \neg y))$$

x	y
x	$\neg y$
x	y
\vdots	\vdots

- Implicants are like “models with wild cards”

$$\nu \quad * \quad * \quad * \quad *$$

Implicants and CNF

- Any model of a formula is an implicant

$$x \wedge y \wedge \neg z \Rightarrow \begin{array}{l} x \vee y \vee z \\ x \vee \neg y \vee \neg z \\ x \vee y \vee \neg z \end{array}$$

Implicants and CNF

- Any model of a formula is an implicant

$$x \wedge y \wedge \neg z \Rightarrow \begin{array}{l} \boxed{x} \vee \boxed{y} \vee z \\ \boxed{x} \vee \neg y \vee \boxed{\neg z} \\ \boxed{x} \vee \boxed{y} \vee \boxed{\neg z} \end{array}$$

- A model can be **reduced**

$$y \wedge \neg z \Rightarrow \begin{array}{l} x \vee \boxed{y} \vee z \\ x \vee \neg y \vee \boxed{\neg z} \\ x \vee \boxed{y} \vee \boxed{\neg z} \end{array}$$

Implicants and CNF

- Any model of a formula is an implicant

$$x \wedge y \wedge \neg z \Rightarrow \begin{array}{l} x \vee y \vee z \\ x \vee \neg y \vee \neg z \\ x \vee y \vee \neg z \end{array}$$

- A model can be **reduced**

$$y \wedge \neg z \Rightarrow \begin{array}{l} x \vee y \vee z \\ x \vee \neg y \vee \neg z \\ x \vee y \vee \neg z \end{array}$$

- Multiple reductions may exist

$$x \Rightarrow \begin{array}{l} x \vee y \vee z \\ x \vee \neg y \vee \neg z \\ x \vee y \vee \neg z \end{array}$$

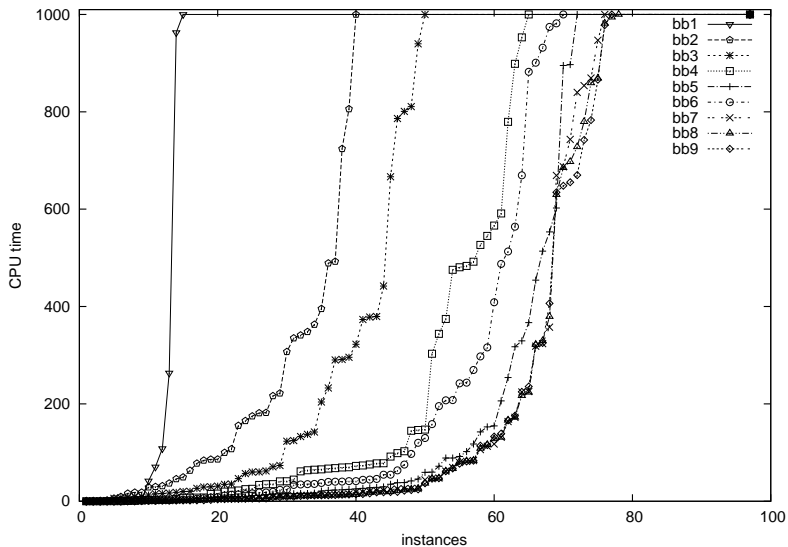
Improving Iterative Testing

Input : CNF formula φ , with variables X

Output: Backbone of φ , ν_R

```
1  $\Lambda \leftarrow \{x, \neg x \mid x \in X\}$  // candidates for backbone
2  $\nu_R \leftarrow \emptyset$  // initial backbone estimate
3 foreach  $I \in \Lambda$  do
4    $(\text{outc}, \nu) \leftarrow \text{SAT}(\varphi \cup \{\bar{I}\})$ 
5   if  $\text{outc} = \text{false}$  then
6      $\nu_R \leftarrow \nu_R \cup \{I\}$  // Backbone identified
7      $\varphi \leftarrow \varphi \cup \{I\}$ 
8   else
9      $\nu \leftarrow \text{ReduceModel}(\nu)$  // Simplify model
10     $\Lambda \leftarrow \Lambda \cap \nu$ 
11 return  $\nu_R$ 
```

Results



Results

Instance	#vars	%bb	best time [s]
narain-sat07-clauses-2	75528	89.3	869.3
2dlx_cc_mc_ex_bp_f2_bug001	4821	36.6	14.8
2dlx_cc_mc_ex_bp_f2_bug005	4824	44.7	17.9
2dlx_cc_mc_ex_bp_f2_bug009	4824	34.8	12.1
grieu-vmcpc-s05-25	625	100.0	92.1
grieu-vmcpc-s05-27	729	92.9	591.2
IBM_FV_03_SAT_dat.k35	34174	59.8	320.8
IBM_FV_04_SAT_dat.k25	27670	78.4	163.6
IBM_FV_06_SAT_dat.k35	42801	50.8	655.4
IBM_FV_1_02_3_SAT_dat.k20	15775	17.4	36.8
AProVE09-03	59231	51.7	743.3
AProVE09-05	14685	76.3	41.7
AProVE09-17	33894	65.4	629.8
AProVE09-24	61164	18.0	648.0

Summary

- We used a SAT solver in a blackbox approach to compute backbones of a propositional formula.

Summary

- We used a SAT solver in a blackbox approach to compute backbones of a propositional formula.
- In the worst case, the algorithm calls the solver as many times as there are variables in the formula. However, this can be reduced.

Summary

- We used a SAT solver in a blackbox approach to compute backbones of a propositional formula.
- In the worst case, the algorithm calls the solver as many times as there are variables in the formula. However, this can be reduced.
- Backbones can be computed for formulas with dozens of thousands of variables.

Summary

- We used a SAT solver in a blackbox approach to compute backbones of a propositional formula.
- In the worst case, the algorithm calls the solver as many times as there are variables in the formula. However, this can be reduced.
- Backbones can be computed for formulas with dozens of thousands of variables.
- Large number of backbones appeared in real-world examples.