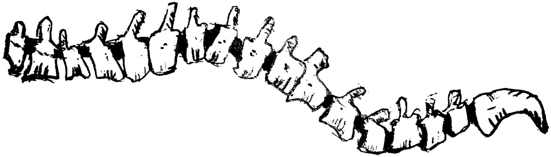
On Computing Backbones of Propositional Theories

Joao Marques-Silva[1] **Mikoláš Janota**[2] Inês Lynce[3]

[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.

$\ldots$	$\mathbf{x_j}$	$\ldots$	$\mathbf{x_k}$	$\ldots$	$\mathbf{x_n}$
$\ldots$	x_j	$\ldots$	$\neg x_k$	$\ldots$	$\neg x_n$

Backbones

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

. . .	$\mathbf{x_j}$	. . .	$\mathbf{x_k}$	. . .	$\mathbf{x_n}$
. . .	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.

$\ldots$	$\mathbf{x_j}$	$\ldots$	$\mathbf{x_k}$	$\ldots$	$\mathbf{x_n}$
$\ldots$	x_j	$\ldots$	$\neg x_k$	$\ldots$	$\neg x_n$
$\ldots$	x_j	$\ldots$	$\neg x_k$	$\ldots$	x_n
$\ldots$	x_j	$\ldots$	$\neg x_k$	$\ldots$	$\neg x_n$
$\ldots$	x_j	$\ldots$	$\neg x_k$	$\ldots$	x_n

Backbones

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

…	$\mathbf{x_j}$	…	$\mathbf{x_k}$	…	$\mathbf{x_n}$
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	$\neg x_n$
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	x_n
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	$\neg x_n$
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	x_n

Backbones

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

…	$\mathbf{x_j}$	…	$\mathbf{x_k}$	…	$\mathbf{x_n}$
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	$\neg x_n$
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	x_n
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	$\neg x_n$
…	$\mathbf{x_j}$	…	$\neg\mathbf{x_k}$	…	x_n

$\phi \models x_j$ $\quad$ $\phi \models \neg x_k$

Motivation

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

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

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

$$\left.\begin{array}{l}\text{gas-engine} \vee \text{electric-engine} \\ \text{electric-engine} \Rightarrow \text{automatic} \\ \neg\text{automatic} \vee \neg\text{manual} \\ \\ \end{array}\right\}$$

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

$$\left.\begin{array}{l} \text{gas-engine} \lor \text{electric-engine} \\ \text{electric-engine} \Rightarrow \text{automatic} \\ \neg\text{automatic} \lor \neg\text{manual} \\ \text{electric-engine} \end{array}\right\}$$

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

$$\left.\begin{array}{l}\text{gas-engine} \vee \text{electric-engine}\\ \text{electric-engine} \Rightarrow \text{automatic}\\ \neg\text{automatic} \vee \neg\text{manual}\\ \text{electric-engine}\end{array}\right\}\ \text{automatic},\ \neg\text{manual}$$

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

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

Armory

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

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

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

Model Enumeration

Input : CNF formula φ
Output: Backbone of φ, ν_R

$\nu_R \leftarrow \{\neg x, x \mid x \in X\}$ // Initial backbone estimate
repeat
  $(\text{outc}, \nu) \leftarrow \texttt{SAT}(\varphi)$ // SAT solver call
  if outc $=$ false **then**
    return ν_R // Terminate if unsatisfiable
  $\nu_R \leftarrow \nu_R \cap \nu$ // Update backbone estimate
  $\omega_B \leftarrow \texttt{BlockClause}(\nu)$ // Block model
  $\varphi \leftarrow \varphi \cup \omega_B$
until $\nu_R = \emptyset$
return $\emptyset$

Iterative SAT Testing

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

Iterative SAT Testing

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

$$\phi \models l$$

Iterative SAT Testing

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

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

Iterative SAT Testing

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

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

$$\phi \models x$$

Iterative SAT Testing

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

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

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

Iterative SAT Testing

Input : CNF formula φ, with variables X
Output: Backbone of φ, ν_R

```
ν_R ← ∅
foreach l ∈ {¬x, x | x ∈ X} do
    (outc, ν) ← SAT(φ ∪ {l̄})
    if outc = false then
        ν_R ← ν_R ∪ {l}          // l is backbone
        φ ← φ ∪ {l}
return ν_R
```

Iterative SAT Testing

Input : CNF formula φ, with variables X
Output: Backbone of φ, ν_R

$\nu_R \leftarrow \emptyset$
foreach $l \in \{\neg x, x \mid x \in X\}$ **do**
  $(\text{outc}, \nu) \leftarrow \texttt{SAT}(\varphi \cup \{\bar{l}\})$
  if outc = false **then**
    $\nu_R \leftarrow \nu_R \cup \{l\}$ // l is backbone
    $\varphi \leftarrow \varphi \cup \{l\}$
return ν_R

- SAT is called twice per variable

Observations I.

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

Observations I.

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

…	$\mathbf{x_i}$	…
…	$\neg x_i$	…
⋮	⋮	⋮

Observations I.

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

...	$\mathbf{x_i}$	...
...	$\neg x_i$	...
⋮	⋮	⋮

$\phi \not\models x_i$

Observations II.

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

$$\nu \Rightarrow \phi$$

Observations II.

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

$$\nu \Rightarrow \phi$$

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

Observations II.

- Implicant $\nu = l_1 \wedge \ldots \wedge l_k$ 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 $\nu = l_1 \wedge \ldots \wedge l_k$ 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 $\nu = l_1 \wedge \ldots \wedge l_k$ 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 $\nu = l_1 \wedge \ldots \wedge l_k$ 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

$$x \wedge y \wedge \neg z \quad \Rightarrow \quad \begin{array}{ccccc} \mathbf{x} & \vee & \mathbf{y} & \vee & z \\ \mathbf{x} & \vee & \neg y & \vee & \neg \mathbf{z} \\ \mathbf{x} & \vee & \mathbf{y} & \vee & \neg \mathbf{z} \end{array}$$

Implicants and CNF

- Any model of a formula is an implicant

$$x \wedge y \wedge \neg z \quad \Rightarrow \quad \begin{array}{ccccc} \mathbf{x} & \vee & \mathbf{y} & \vee & z \\ \mathbf{x} & \vee & \neg y & \vee & \mathbf{\neg z} \\ \mathbf{x} & \vee & \mathbf{y} & \vee & \mathbf{\neg z} \end{array}$$

- A model can be reduced

$$y \wedge \neg z \quad \Rightarrow \quad \begin{array}{ccccc} x & \vee & \mathbf{y} & \vee & z \\ x & \vee & \neg y & \vee & \mathbf{\neg z} \\ x & \vee & \mathbf{y} & \vee & \mathbf{\neg z} \end{array}$$

Implicants and CNF

- Any model of a formula is an implicant

$$x \wedge y \wedge \neg z \;\Rightarrow\; \begin{array}{ccccc} \mathbf{x} & \vee & \mathbf{y} & \vee & z \\ \mathbf{x} & \vee & \neg y & \vee & \mathbf{\neg z} \\ \mathbf{x} & \vee & \mathbf{y} & \vee & \mathbf{\neg z} \end{array}$$

- A model can be reduced

$$y \wedge \neg z \;\Rightarrow\; \begin{array}{ccccc} x & \vee & \mathbf{y} & \vee & z \\ x & \vee & \neg y & \vee & \mathbf{\neg z} \\ x & \vee & \mathbf{y} & \vee & \mathbf{\neg z} \end{array}$$

- Multiple reductions may exist

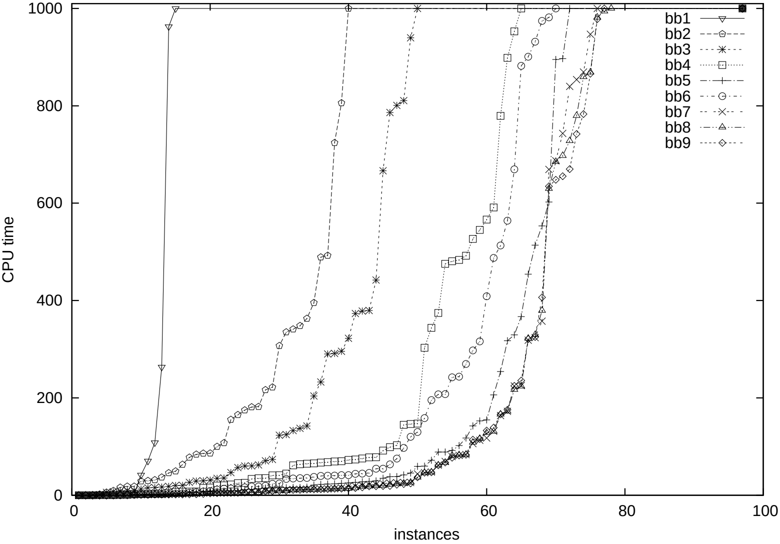
$$x \;\Rightarrow\; \begin{array}{ccccc} \mathbf{x} & \vee & y & \vee & z \\ \mathbf{x} & \vee & \neg y & \vee & \neg z \\ \mathbf{x} & \vee & y & \vee & \neg z \end{array}$$

Improving Iterative Testing

Input : CNF formula φ, with variables X
Output: Backbone of φ, ν_R

$\Lambda \leftarrow \{x, \neg x \mid x \in X\}$ `// candidates for backbone`
$\nu_R \leftarrow \emptyset$ `// initial backbone estimate`
foreach $l \in \Lambda$ **do**
  $(\text{outc}, \nu) \leftarrow \texttt{SAT}(\varphi \cup \{\bar{l}\})$
  if outc = false **then**
    $\nu_R \leftarrow \nu_R \cup \{l\}$ `// Backbone identified`
    $\varphi \leftarrow \varphi \cup \{l\}$
  else
    $\nu \leftarrow \texttt{ReduceModel}(\nu)$ `// Simplify model`
    $\Lambda \leftarrow \Lambda \cap \nu$
return ν_R

Results



CPU time
instances
bb1
bb2
bb3
bb4
bb5
bb6
bb7
bb8
bb9
0
200
400
600
800
1000
0
20
40
60
80
100

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-vmpc-s05-25	625	100.0	92.1
grieu-vmpc-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 off backbones appeared in real-world examples.