Aalto University
School of Science
and Technology

Understanding, improving and parallelizing MUS finding using model rotation

Siert Wieringa

Aalto University
School of Science and Technology
Finland

October 10th, 2012

Minimal Unsatisfiable

- Finding Minimal Unsatisfiable Subsets (MUSes) in unsatisfiable CNF formulas
- An unsatisfiable formula $\mathcal{F}$ is minimal unsatisfiable iff any proper subset of its clauses $\mathcal{F}' \subset \mathcal{F}$ is satisfiable.
- An **assoc** for a clause $c \in \mathcal{F}$ is a complete assignment that satisfies all clauses in $\mathcal{F}$ except c.
- A formula $\mathcal{F}$ is minimal unsatisfiable iff every clause in the formula has at least one assoc.

Classical destructive algorithm

1. $M = \emptyset$
2. `while` $\mathcal{F} \neq M$
3. `pick` a clause $c \in \mathcal{F} \setminus M$
4. `if` $\mathcal{F} \setminus \{c\}$ is satisfiable `then`
5. $M = M \cup \{c\}$
6. `else`
7. $\mathcal{F} = \mathcal{F} \setminus \{c\}$
8. `return` $\mathcal{F}$

Classical destructive algorithm

1. $M = \emptyset$
2. `while` $\mathcal{F} \neq M$
3. `pick` a clause $c \in \mathcal{F} \setminus M$
4. `if` $\mathcal{F} \setminus \{c\}$ is satisfiable `then`
5. $M = M \cup \{c\}$
6. `else`
7. $\mathcal{F} = \mathcal{F}'$ s.t. $\mathcal{F}'$ is unsatisfiable and $\mathcal{F}' \subseteq \mathcal{F} \setminus \{c\}$
8. `return` $\mathcal{F}$

Classical destructive algorithm

1. $M = \emptyset$
2. `while` $\mathcal{F} \neq M$
3. `pick` a clause $c \in \mathcal{F} \setminus M$
4. `if` $\mathcal{F} \setminus \{c\}$ is satisfiable `then`
5. $M = M \cup \{c\}$

$\Rightarrow$ `modelRotate`(c, a) `//(`a`=sat. assign. for` $\mathcal{F} \setminus \{c\}$`)`

6. `else`
7. $\mathcal{F} = \mathcal{F}'$ s.t. $\mathcal{F}'$ is unsatisfiable and $\mathcal{F}' \subseteq \mathcal{F} \setminus \{c\}$
8. `return` $\mathcal{F}$

Model rotation

Marques-Silva and Lynce, SAT2011

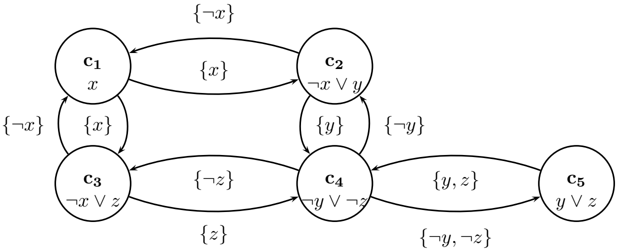
Belov and Marques-Silva, FMCAD2011

function `modelRotate`(`clause` c, `assignment` a)

1. `for all` $l \in c$ `do`
2. $a' = a$ except l is assigned **true** instead of **false**
3. `if` $\left(\begin{array}{l}\text{exactly one clause } c' \in \mathcal{F} \text{ is} \\ \text{not satisfied by } a' \text{ and } c' \notin M\end{array}\right)$ `then`
4. $M = M \cup \{c'\}$
5. `modelRotate`(c', a')

Flip graph

- A vertex for every clause
- Edges are labelled: $L(c_i, c_j) = \{l \mid l \in c_i \text{ and } \neg l \in c_j\}$
- An edge between c_i and c_j iff $L(c_i, c_j) \neq \emptyset$



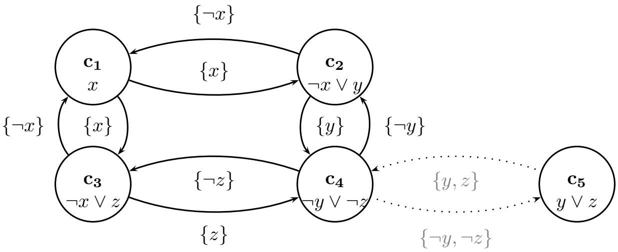
Flip graph

- Lemma 1:

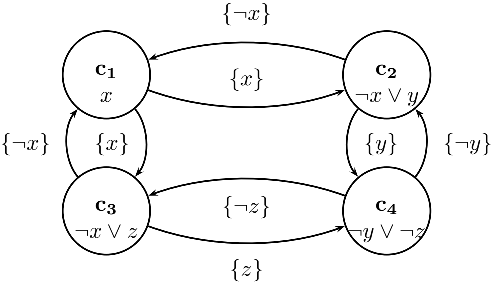
  *Rotating literal l in an assoc for a clause c_i can **not** result in an assoc for clause c_j if $L(c_i, c_j) \neq \{\neg l\}$*

- Possible rotation edges:

  All edges (c_i, c_j) for which $|L(c_i, c_j)| = 1$

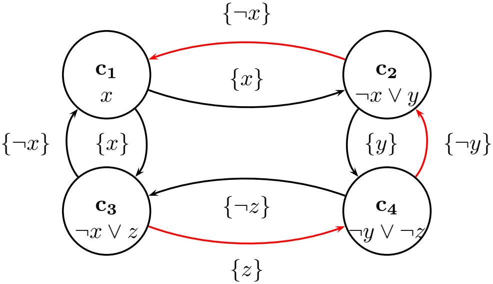


Rotation example



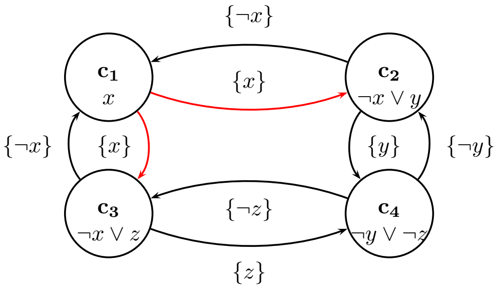
assoc for: c_3 $x =$ **true** $y =$ **true** $z =$ **false**

Rotation example



assoc for: c_3	$x =$ **true**	$y =$ **true**	$z =$ **false**
assoc for: c_4	$x =$ **true**	$y =$ **true**	$z =$ **true**
assoc for: c_2	$x =$ **true**	$y =$ **false**	$z =$ **true**
assoc for: c_1	$x =$ **false**	$y =$ **false**	$z =$ **true**

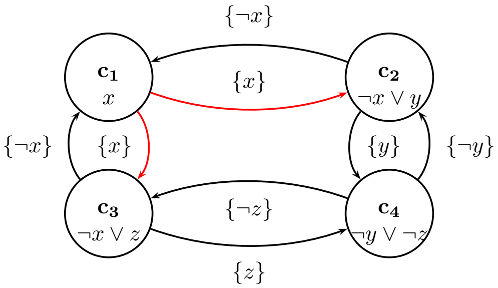
Rotation example



There are 3 possible assocs for c_1.

x=**false** y=**false** z=**true**
x=**false** y=**true** z=**false**
x=**false** y=**false** z=**false**

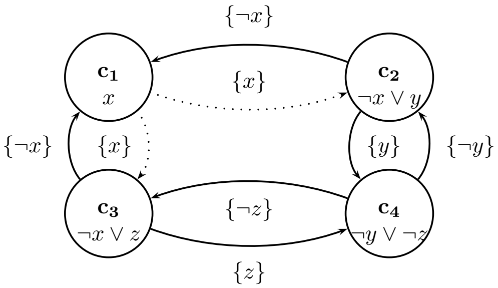
Rotation example



There are 3 possible assocs for c_1. Rotation gives:

x=**true**	y=**false**	z=**true**	$\rightarrow$	assoc for c_2
x=**true**	y=**true**	z=**false**	$\rightarrow$	assoc for c_3
x=**true**	y=**false**	z=**false**	$\rightarrow$	does not satisfy c_2 and c_3!

Rotation example



- Guaranteed rotation edges:

  All possible rotation edges (c_i, c_j) *s.t. for all* $c_k \neq c_j$ *it holds that* $L(c_i, c_j) \neq L(c_i, c_k)$

Rotation theory

- Theorem 1:

  If c_i has an assoc and a guaranteed rotation edge (c_i, c_j) exist then c_j has an assoc

- If c_i has an assoc and a path over guaranteed rotation edges from c_i to c_j exist then c_j has an assoc

- Corollary 1:

  If a path over guaranteed rotation edges between c_i and c_j exists in both directions then c_i has an assoc iff c_j has an assoc

Strongly Connected Components

- A directed graph is *strongly connected* if there exists a path between any two of its vertices
- The *strongly connected components* (SCCs) of a directed graph are its maximal strongly connected subgraphs

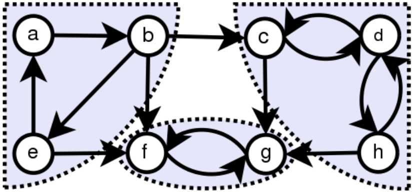


Figure source: `http://en.wikipedia.org/wiki/Strongly_connected_component`

Rotation theory continued

- Let $\mathcal{F}$ be an unsatisfiable formula
- Let E_G be the set of guaranteed rotation edges $\mathcal{F}$ induces
- Consider the SCCs of the graph $G = (\mathcal{F}, E_G)$
- Corollary 2:

  A MUS $\mathcal{F}' \subseteq \mathcal{F}$ can be found using no more SAT solver calls than there are SCCs in $G = (\mathcal{F}, E_G)$

Rotation theory continued

- Let a *root SCC* be an SCC that contains no vertices with incoming edges originating outside the SCC
- Let $\mathcal{F}'$ be an *minimal unsatisfiable* formula, and E'_G the set of guaranteed rotation edges it induces
- Corollary 3:

  An assoc can be found for every clause in $\mathcal{F}'$ using no more SAT solver calls than there are root SCCs in $G' = (\mathcal{F}', E'_G)$

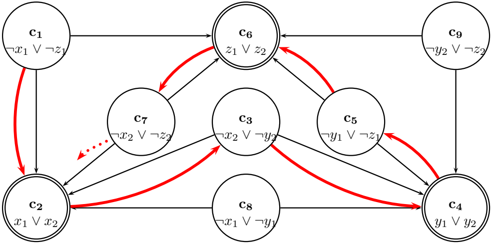
Statistics - benchmarks from Marques-Silva et.al.

	original		MUSes	
# benchmarks	500		491	
# with single root SCC	148		148	
avg. # clauses	6874.4		6204.2	
avg. # SCCs	3000.4	(44%)	2484.1	(40%)
avg. # root SCCs	2124.7		1680.5	(27%)
avg. clause length	2.32		2.35	
avg. out-degree E_P	12.30		11.24	
avg. out-degree E_G	1.60		1.63	

Statistics - benchmarks from SAT11 competition

	original		MUSes	
# benchmarks	298		262	
# with single root SCC	0		51	
avg. # clauses	404574		8162.7	
avg. # SCCs	327815	(81%)	3355.6	(41%)
avg. # root SCCs	258350		1891.1	(23%)
avg. clause length	2.53		2.42	
avg. out-degree E_P	86.84		14.16	
avg. out-degree E_G	0.89		1.59	

Improving & Parallelizing



- The suggested improvement for model rotation concerns cyclic paths.
- The parallelization uses the *Tarmo* parallel incremental SAT solver.

Conclusions

- This work provides new insights on *model rotation*
- The key is to look at the algorithm as a graph search
- Typical benchmarks have properties that guarantee effectiveness of model rotation
- The technique was improved, parallelized and evaluated