Extensions and Generalizations of Minimal Unsatisfiability

Anton Belov

Complex and Adaptive Systems Laboratory
School of Computer Science and Informatics
University College Dublin, Ireland

EPCL Basic Training Camp
December 10-21, 2012
Dresden, Germany

Abstract

Motivated by practical applications, the concepts related to minimal unsatisfiability and minimal unsatisfiable subformulas (MUSes) have recently been extended beyond propositional formulas in CNF, and, also, generalized to the case of satisfiable formulas. In this talk I will discuss the algorithms and the optimization techniques for the computation of MUSes in some of these extended and generalized domains.

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$ $\quad$ $C_3 = (\neg p \vee \neg q)$ $\quad$ $C_5 = (\neg p \vee r)$

$C_2 = (q)$ $\quad$ $C_4 = (p \vee q)$ $\quad$ $C_6 = (\neg q \vee \neg r)$

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$ $\quad$ $C_3 = (\neg p \vee \neg q)$ $\quad$ $C_5 = (\neg p \vee r)$

$C_2 = (q)$ $\quad$ $C_4 = (p \vee q)$ $\quad$ $C_6 = (\neg q \vee \neg r)$

Which part of $\mathcal{F}$ is responsible for its inconsistency ?

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

Which part of $\mathcal{F}$ is responsible for its inconsistency ?

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

Which part of $\mathcal{F}$ is responsible for its inconsistency ?

The set $\{C_1, C_2, C_3\} \in \text{UNSAT}$, and is *minimal* w.r.t. to UNSAT.

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

Which part of $\mathcal{F}$ is responsible for its inconsistency ?

The set $\{C_1, C_2, C_3\} \in \text{UNSAT}$, and is *minimal* w.r.t. to UNSAT.

Def: $\mathcal{F}' \subseteq \mathcal{F}$ is *minimally unsatisfiable subformula (MUS)* of $\mathcal{F}$ if $\mathcal{F}' \in \text{UNSAT}$, and $\forall C \in \mathcal{F}', \mathcal{F}' \setminus \{C\} \in \text{SAT}$.

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

Which part of $\mathcal{F}$ is responsible for its inconsistency ?

The set $\{C_1, C_2, C_3\} \in \text{UNSAT}$, and is *minimal* w.r.t. to UNSAT.

Def: $\mathcal{F}' \subseteq \mathcal{F}$ is *minimally unsatisfiable subformula (MUS)* of $\mathcal{F}$ if $\mathcal{F}' \in \text{UNSAT}$, and $\forall C \in \mathcal{F}', \mathcal{F}' \setminus \{C\} \in \text{SAT}$.

$\{C_1, C_2, C_3\}$ is an MUS of $\mathcal{F}$.

Introduction: Minimal Unsatisfiable Subformulas (MUSes)

$\mathcal{F} = \{C_1, \ldots, C_6\} \in \text{UNSAT}$

$C_1 = (p)$

$C_2 = (q)$

$C_3 = (\neg p \vee \neg q)$

$C_4 = (p \vee q)$

$C_5 = (\neg p \vee r)$

$C_6 = (\neg q \vee \neg r)$

Which part of $\mathcal{F}$ is responsible for its inconsistency ?

The set $\{C_1, C_2, C_3\} \in \text{UNSAT}$, and is *minimal* w.r.t. to UNSAT.

Def: $\mathcal{F}' \subseteq \mathcal{F}$ is *minimally unsatisfiable subformula (MUS)* of $\mathcal{F}$ if $\mathcal{F}' \in \text{UNSAT}$, and $\forall C \in \mathcal{F}', \mathcal{F}' \setminus \{C\} \in \text{SAT}$.

$\{C_1, C_2, C_3\}$ is an MUS of $\mathcal{F}$. And so is $\{C_1, C_2, C_5, C_6\}$.

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.
- Variable MUSes [Chen&Ding, TAMC'06] — analysis in terms of variables of the formula.

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.
- Variable MUSes [Chen&Ding, TAMC'06] — analysis in terms of variables of the formula.
- Non-clausal and Circuit MUSes — extensions to non-clausal propositional formulas [Kleine Büning and Zhao, AIJ'07] and Boolean circuits [Belov and Marques-Silva, SAT'11].

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.
- Variable MUSes [Chen&Ding, TAMC'06] — analysis in terms of variables of the formula.
- Non-clausal and Circuit MUSes — extensions to non-clausal propositional formulas [Kleine Büning and Zhao, AIJ'07] and Boolean circuits [Belov and Marques-Silva, SAT'11].
- Minimal Equivalent Subformulas (MESes) [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05] — generalization to satisfiable formulas.

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.
- Variable MUSes [Chen&Ding, TAMC'06] — analysis in terms of variables of the formula.
- Non-clausal and Circuit MUSes — extensions to non-clausal propositional formulas [Kleine Büning and Zhao, AIJ'07] and Boolean circuits [Belov and Marques-Silva, SAT'11].
- Minimal Equivalent Subformulas (MESes) [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05] — generalization to satisfiable formulas.
- MUSes for LTL formulas [Schuppan, FSE'10].

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.
- Variable MUSes [Chen&Ding, TAMC'06] — analysis in terms of variables of the formula.
- Non-clausal and Circuit MUSes — extensions to non-clausal propositional formulas [Kleine Büning and Zhao, AIJ'07] and Boolean circuits [Belov and Marques-Silva, SAT'11].
- Minimal Equivalent Subformulas (MESes) [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05] — generalization to satisfiable formulas.
- MUSes for LTL formulas [Schuppan, FSE'10].
- Minimal unsatisfiability and falsity for QBF formulas [Kleine Büning and Zhao, SAT'06].

Introduction: Extensions and Generalizations

- Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10] — analysis in terms of subsets of clauses.
- Variable MUSes [Chen&Ding, TAMC'06] — analysis in terms of variables of the formula.
- Non-clausal and Circuit MUSes — extensions to non-clausal propositional formulas [Kleine Büning and Zhao, AIJ'07] and Boolean circuits [Belov and Marques-Silva, SAT'11].
- Minimal Equivalent Subformulas (MESes) [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05] — generalization to satisfiable formulas.
- MUSes for LTL formulas [Schuppan, FSE'10].
- Minimal unsatisfiability and falsity for QBF formulas [Kleine Büning and Zhao, SAT'06].

Related areas

- Minimal Unsatisfiable sets of Constraints (MUCs) — in CSP.
- Irreducible Infeasible Subsets (IISes) in linear programs Note: many MUS extraction algorithms can be traced back here.

Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10]

$\mathcal{F} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \in$ UNSAT — partitioned into *groups* (sets) of clauses

$$\mathcal{G}_1 = \{C_1, C_2\}, \quad \mathcal{G}_2 = \{C_3, C_4\}, \quad \mathcal{G}_3 = \{C_5, C_6\}.$$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10]

$\mathcal{F} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \in$ UNSAT — partitioned into *groups* (sets) of clauses

$$\mathcal{G}_1 = \{C_1, C_2\}, \quad \mathcal{G}_2 = \{C_3, C_4\}, \quad \mathcal{G}_3 = \{C_5, C_6\}.$$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

$\{\mathcal{G}_1, \mathcal{G}_2\}$ is a subset-minimal set of *groups* sufficient to refute $\mathcal{F}$.

Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10]

$\mathcal{F} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \in$ UNSAT — partitioned into *groups* (sets) of clauses

$$\mathcal{G}_1 = \{C_1, C_2\}, \quad \mathcal{G}_2 = \{C_3, C_4\}, \quad \mathcal{G}_3 = \{C_5, C_6\}.$$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

$\{\mathcal{G}_1, \mathcal{G}_2\}$ is a subset-minimal set of *groups* sufficient to refute $\mathcal{F}$.

$\{\mathcal{G}_1, \mathcal{G}_2\}$ is a *group-MUS* of (the partitioned) $\mathcal{F}$.

Group-oriented MUSes [Liffiton&Sakallah, JAR'08; Nadel, FMCAD'10]

$\mathcal{F} = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \in$ UNSAT — partitioned into *groups* (sets) of clauses

$$\mathcal{G}_1 = \{C_1, C_2\}, \quad \mathcal{G}_2 = \{C_3, C_4\}, \quad \mathcal{G}_3 = \{C_5, C_6\}.$$

$C_1 = (p)$ $C_2 = (q)$

$C_3 = (\neg p \vee \neg q)$ $C_4 = (p \vee q)$

$C_5 = (\neg p \vee r)$ $C_6 = (\neg q \vee \neg r)$

$\{\mathcal{G}_1, \mathcal{G}_2\}$ is a subset-minimal set of *groups* sufficient to refute $\mathcal{F}$.

$\{\mathcal{G}_1, \mathcal{G}_2\}$ is a *group-MUS* of (the partitioned) $\mathcal{F}$.

$\{\mathcal{G}_1, \mathcal{G}_3\}$ is also a *group-MUS* of (the partitioned) $\mathcal{F}$.

Group-oriented MUSes

Def: Given a *group-CNF* formula $\mathcal{F} = \mathcal{G}_0 \cup \mathcal{G}_1 \cup \cdots \cup \mathcal{G}_n \in$ UNSAT, a *group oriented MUS (GMUS)* of $\mathcal{F}$ is a set $\{\mathcal{G}_{i_1}, \ldots, \mathcal{G}_{i_k}\}$ such that $\mathcal{F}' = \mathcal{G}_0 \cup \bigcup_{1 \leq j \leq k} G_{i_j} \in$ UNSAT, and for every $1 \leq j \leq k$, $\mathcal{F}' \setminus \mathcal{G}_{i_j} \in$ SAT.

Group-oriented MUSes

Def: Given a *group-CNF* formula $\mathcal{F} = \mathcal{G}_0 \cup \mathcal{G}_1 \cup \cdots \cup \mathcal{G}_n \in$ UNSAT, a *group oriented MUS (GMUS)* of $\mathcal{F}$ is a set $\{\mathcal{G}_{i_1}, \ldots, \mathcal{G}_{i_k}\}$ such that $\mathcal{F}' = \mathcal{G}_0 \cup \bigcup_{1 \leq j \leq k} G_{i_j} \in$ UNSAT, and for every $1 \leq j \leq k$, $\mathcal{F}' \setminus \mathcal{G}_{i_j} \in$ SAT.

Example: a group = CNF representation of a gate in a circuit.

Group-oriented MUSes

Def: Given a *group-CNF* formula $\mathcal{F} = \mathcal{G}_0 \cup \mathcal{G}_1 \cup \cdots \cup \mathcal{G}_n \in$ UNSAT, a *group oriented MUS (GMUS)* of $\mathcal{F}$ is a set $\{\mathcal{G}_{i_1}, \ldots, \mathcal{G}_{i_k}\}$ such that $\mathcal{F}' = \mathcal{G}_0 \cup \bigcup_{1 \le j \le k} G_{i_j} \in$ UNSAT, and for every $1 \le j \le k$, $\mathcal{F}' \setminus \mathcal{G}_{i_j} \in$ SAT.

Example: a group = CNF representation of a gate in a circuit.

- Note the special role of $\mathcal{G}_0$ — the "background" clauses. Example: in PBA a group = CNF representation of a latch, $\mathcal{G}_0$ is the rest of the circuit.
- $\mathcal{G}_0 \in$ UNSAT $\iff$ the GMUS is $\emptyset$.
- MUS computation is a special case of group-MUS computation.

Group-oriented MUSes

Def: Given a *group-CNF* formula $\mathcal{F} = \mathcal{G}_0 \cup \mathcal{G}_1 \cup \cdots \cup \mathcal{G}_n \in$ UNSAT, a *group oriented MUS (GMUS)* of $\mathcal{F}$ is a set $\{\mathcal{G}_{i_1}, \ldots, \mathcal{G}_{i_k}\}$ such that $\mathcal{F}' = \mathcal{G}_0 \cup \bigcup_{1 \le j \le k} G_{i_j} \in$ UNSAT, and for every $1 \le j \le k$, $\mathcal{F}' \setminus \mathcal{G}_{i_j} \in$ SAT.

Example: a group = CNF representation of a gate in a circuit.

- Note the special role of $\mathcal{G}_0$ — the "background" clauses. Example: in PBA a group = CNF representation of a latch, $\mathcal{G}_0$ is the rest of the circuit.
- $\mathcal{G}_0 \in$ UNSAT $\iff$ the GMUS is $\emptyset$.
- MUS computation is a special case of group-MUS computation.

Perhaps the *most* useful (from practical standpoint) and versatile generalization of MUSes.

Computing Group-oriented MUSes

All current MUS extraction algorithms can be lifted to group-oriented setting:

- A group $\mathcal{G}$ is *necessary* for GCNF $\mathcal{F}$ if $\mathcal{F} \in$ UNSAT and $\mathcal{F} \setminus \mathcal{G} \in$ SAT.

Computing Group-oriented MUSes

All current MUS extraction algorithms can be lifted to group-oriented setting:

- A group $\mathcal{G}$ is *necessary* for GCNF $\mathcal{F}$ if $\mathcal{F} \in$ UNSAT and $\mathcal{F} \setminus \mathcal{G} \in$ SAT.
- Fact: A group $\mathcal{G}$ is necessary for GCNF $\mathcal{F}$ iff $\exists \tau$, such that $\tau(\mathcal{F} \setminus \mathcal{G}) = 1$ and $\tau(\mathcal{G}) = 0$. Note: τ need to falsify only *some* clauses of $\mathcal{G}$.

Computing Group-oriented MUSes

All current MUS extraction algorithms can be lifted to group-oriented setting:

- A group $\mathcal{G}$ is *necessary* for GCNF $\mathcal{F}$ if $\mathcal{F} \in$ UNSAT and $\mathcal{F} \setminus \mathcal{G} \in$ SAT.
- Fact: A group $\mathcal{G}$ is necessary for GCNF $\mathcal{F}$ iff $\exists \tau$, such that $\tau(\mathcal{F} \setminus \mathcal{G}) = 1$ and $\tau(\mathcal{G}) = 0$. Note: τ need to falsify only *some* clauses of $\mathcal{G}$.
- Now one can do deletion-based, insertion-based, dichotomic, etc. group-MUS extraction.

Computing Group-oriented MUSes

All current MUS extraction algorithms can be lifted to group-oriented setting:

- A group $\mathcal{G}$ is *necessary* for GCNF $\mathcal{F}$ if $\mathcal{F} \in$ UNSAT and $\mathcal{F} \setminus \mathcal{G} \in$ SAT.
- Fact: A group $\mathcal{G}$ is necessary for GCNF $\mathcal{F}$ iff $\exists \tau$, such that $\tau(\mathcal{F} \setminus \mathcal{G}) = 1$ and $\tau(\mathcal{G}) = 0$. Note: τ need to falsify only *some* clauses of $\mathcal{G}$.
- Now one can do deletion-based, insertion-based, dichotomic, etc. group-MUS extraction.
- Some low-level modification to a SAT solver are possible to optimize SAT solver use [Ryvchin and Strichman, SAT'11].

Computing Group-oriented MUSes: optimizations

Group-set refinement:

- When $\mathcal{F} \setminus \mathcal{G} \in$ UNSAT, get an unsatisfiable core $\mathcal{U}$ from the SAT solver.

Computing Group-oriented MUSes: optimizations

Group-set refinement:

- When $\mathcal{F} \setminus \mathcal{G} \in$ UNSAT, get an unsatisfiable core $\mathcal{U}$ from the SAT solver.
- Remove any group $\mathcal{G}_i$ for which $\mathcal{U} \cap \mathcal{G}_i = \emptyset$.

Computing Group-oriented MUSes: optimizations

Group-set refinement:

- When $\mathcal{F} \setminus \mathcal{G} \in$ UNSAT, get an unsatisfiable core $\mathcal{U}$ from the SAT solver.
- Remove any group $\mathcal{G}_i$ for which $\mathcal{U} \cap \mathcal{G}_i = \emptyset$.
- In assumption-based implementations, one can get a "group-core" right away.

Computing Group-oriented MUSes: optimizations

Group-set refinement:

- When $\mathcal{F} \setminus \mathcal{G} \in$ UNSAT, get an unsatisfiable core $\mathcal{U}$ from the SAT solver.
- Remove any group $\mathcal{G}_i$ for which $\mathcal{U} \cap \mathcal{G}_i = \emptyset$.
- In assumption-based implementations, one can get a "group-core" right away.
- Very effective.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in \text{SAT}$, get a model τ — this is a witness for necessity $\mathcal{G}$.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in \text{SAT}$, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in$ SAT, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.
- Issue: τ may satisfy more than one clause. Solutions:

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in$ SAT, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.
- Issue: τ may satisfy more than one clause. Solutions:
  - ignore (i.e. do not rotate) - actually, "works" quite well.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in$ SAT, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.
- Issue: τ may satisfy more than one clause. Solutions:
  - ignore (i.e. do not rotate) - actually, "works" quite well.
  - use group structure - e.g. a group is a set of unrelated equivalences.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in$ SAT, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.
- Issue: τ may satisfy more than one clause. Solutions:
  - ignore (i.e. do not rotate) - actually, "works" quite well.
  - use group structure - e.g. a group is a set of unrelated equivalences.
  - use mini-SLS.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in$ SAT, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.
- Issue: τ may satisfy more than one clause. Solutions:
  - ignore (i.e. do not rotate) - actually, "works" quite well.
  - use group structure - e.g. a group is a set of unrelated equivalences.
  - use mini-SLS.
- Problem: if $\mathcal{G}_0$ is really large, the next step will get into $\mathcal{G}_0$, and rotation will get "lost" inside it. No satisfactory solution yet.

Computing Group-oriented MUSes: optimizations

Model rotation:

- When $\mathcal{F} \setminus \mathcal{G} \in$ SAT, get a model τ — this is a witness for necessity $\mathcal{G}$.
- As with MUSes, try to modify τ into a witness for another group $\mathcal{G}'$.
- Issue: τ may satisfy more than one clause. Solutions:
  - ignore (i.e. do not rotate) - actually, "works" quite well.
  - use group structure - e.g. a group is a set of unrelated equivalences.
  - use mini-SLS.
- Problem: if $\mathcal{G}_0$ is really large, the next step will get into $\mathcal{G}_0$, and rotation will get "lost" inside it. No satisfactory solution yet.
- Effectiveness varies, depending on the structure of GCNFs.

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \mathsf{CNF}(\neg\mathcal{G})$).

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \text{CNF}(\neg\mathcal{G})$).
- Can build CNF($\neg\mathcal{G}$) as follows (Plaisted-Greenbaum transform):

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \text{CNF}(\neg\mathcal{G})$).
- Can build CNF($\neg\mathcal{G}$) as follows (Plaisted-Greenbaum transform):
  - For each $C \in \mathcal{G}$ create a new variable u_C, and

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \mathrm{CNF}(\neg\mathcal{G})$.
- Can build $\mathrm{CNF}(\neg\mathcal{G})$ as follows (Plaisted-Greenbaum transform):
  - For each $C \in \mathcal{G}$ create a new variable u_C, and
    for each literal $l \in C$ create a clause $(\neg u_C \vee \neg l)$, i.e. $u_C \rightarrow \neg l$.

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \mathrm{CNF}(\neg\mathcal{G})$.
- Can build $\mathrm{CNF}(\neg\mathcal{G})$ as follows (Plaisted-Greenbaum transform):
  - For each $C \in \mathcal{G}$ create a new variable u_C, and
    for each literal $l \in C$ create a clause $(\neg u_C \vee \neg l)$, i.e. $u_C \rightarrow \neg l$.
  - Create a clause $\bigvee_{C \in \mathcal{G}} u_c$.

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \mathrm{CNF}(\neg\mathcal{G})$.
- Can build $\mathrm{CNF}(\neg\mathcal{G})$ as follows (Plaisted-Greenbaum transform):
  - For each $C \in \mathcal{G}$ create a new variable u_C, and
    for each literal $l \in C$ create a clause $(\neg u_C \vee \neg l)$, i.e. $u_C \rightarrow \neg l$.
  - Create a clause $\bigvee_{C \in \mathcal{G}} u_c$.
- Same problem with tainted cores as in MUS.

Computing Group-oriented MUSes: optimizations

Redundancy removal:

- Instead of SAT checking $\mathcal{F} \setminus \mathcal{G}$, check $(\mathcal{F} \setminus \mathcal{G}) \cup \mathrm{CNF}(\neg\mathcal{G})$.
- Can build $\mathrm{CNF}(\neg\mathcal{G})$ as follows (Plaisted-Greenbaum transform):
  - For each $C \in \mathcal{G}$ create a new variable u_C, and
    for each literal $l \in C$ create a clause $(\neg u_C \vee \neg l)$, i.e. $u_C \rightarrow \neg l$.
  - Create a clause $\bigvee_{C \in \mathcal{G}} u_c$.
- Same problem with tainted cores as in MUS.
- Effectiveness is not clear.

Variable MUSes [Zhen-Yu Chen and De-Cheng Ding, TAMC'06]

$\mathcal{F} = \{C_1, \ldots, C_6\}$, $Var(\mathcal{F}) = \{p, q, r\}$

$$C_1 = (p) \qquad C_3 = (\neg p \vee \neg q) \qquad C_5 = (\neg p \vee r)$$

$$C_2 = (q) \qquad C_4 = (p \vee q) \qquad C_6 = (\neg q \vee \neg r)$$

What is a subset-minimal set of variables sufficient to refute $\mathcal{F}$?

Variable MUSes [Zhen-Yu Chen and De-Cheng Ding, TAMC'06]

$\mathcal{F} = \{C_1, \ldots, C_6\}$, $Var(\mathcal{F}) = \{p, q, r\}$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

What is a subset-minimal set of variables sufficient to refute $\mathcal{F}$?

$\{p, q\}$ is such a subset of variables.

Variable MUSes [Zhen-Yu Chen and De-Cheng Ding, TAMC'06]

$\mathcal{F} = \{C_1, \ldots, C_6\}$, $Var(\mathcal{F}) = \{p, q, r\}$

$C_1 = (p)$	$C_3 = (\neg p \vee \neg q)$	$C_5 = (\neg p \vee r)$
$C_2 = (q)$	$C_4 = (p \vee q)$	$C_6 = (\neg q \vee \neg r)$

What is a subset-minimal set of variables sufficient to refute $\mathcal{F}$?

$\{p, q\}$ is such a subset of variables.

$\{p, q\}$ is a *variable-MUS* of $\mathcal{F}$.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

$$C_1 = (p) \qquad C_3 = (\neg p \vee \neg q) \qquad C_5 = (\neg p \vee r)$$

$$C_2 = (q) \qquad C_4 = (p \vee q) \qquad C_6 = (\neg q \vee \neg r)$$

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

$$C_1 = (p) \qquad C_3 = (\neg p \vee \neg q) \qquad C_5 = (\neg p \vee r)$$

$$C_2 = (q) \qquad C_4 = (p \vee q) \qquad C_6 = (\neg q \vee \neg r)$$

- The subformula induced by $\{p, q\}$ is $\mathcal{F}|_{\{p,q\}} = \{C_1, C_2, C_3, C_4\}$.
  - variable r is "removed" from $\mathcal{F}$.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

$C_1 = (p)$ $C_3 = (\neg p \vee \neg q)$ $C_5 = (\neg p \vee r)$

$C_2 = (q)$ $C_4 = (p \vee q)$ $C_6 = (\neg q \vee \neg r)$

- The subformula induced by $\{p, q\}$ is $\mathcal{F}|_{\{p,q\}} = \{C_1, C_2, C_3, C_4\}$.
  - variable r is "removed" from $\mathcal{F}$.
- The subformula induced by $\{p\}$, $\mathcal{F}|_{\{p\}} = \{C_1\}$.
  - variables q, r are "removed" from $\mathcal{F}$.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Def: A set $V \subseteq Var(\mathcal{F})$ is a *variable-MUS (VMUS)* of $\mathcal{F}$ if $\mathcal{F}|_V \in$ UNSAT, and for any $V' \subset V$, $\mathcal{F}|_{V'} \in$ SAT.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Def: A set $V \subseteq Var(\mathcal{F})$ is a *variable-MUS (VMUS)* of $\mathcal{F}$ if $\mathcal{F}|_V \in$ UNSAT, and for any $V' \subset V$, $\mathcal{F}|_{V'} \in$ SAT.

$$C_1 = (p) \qquad C_3 = (\neg p \vee \neg q) \qquad C_5 = (\neg p \vee r)$$

$$C_2 = (q) \qquad C_4 = (p \vee q) \qquad C_6 = (\neg q \vee \neg r)$$

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Def: A set $V \subseteq Var(\mathcal{F})$ is a *variable-MUS (VMUS)* of $\mathcal{F}$ if $\mathcal{F}|_V \in$ UNSAT, and for any $V' \subset V$, $\mathcal{F}|_{V'} \in$ SAT.

$C_1 = (p)$ $\quad$ $C_3 = (\neg p \vee \neg q)$ $\quad$ $C_5 = (\neg p \vee r)$

$C_2 = (q)$ $\quad$ $C_4 = (p \vee q)$ $\quad$ $C_6 = (\neg q \vee \neg r)$

- $\mathcal{F}|_{\{p,q\}} = \{C_1, C_2, C_3, C_4\} \in$ UNSAT.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Def: A set $V \subseteq Var(\mathcal{F})$ is a *variable-MUS (VMUS)* of $\mathcal{F}$ if $\mathcal{F}|_V \in$ UNSAT, and for any $V' \subset V$, $\mathcal{F}|_{V'} \in$ SAT.

$C_1 = (p)$ $C_3 = (\neg p \vee \neg q)$ $C_5 = (\neg p \vee r)$

$C_2 = (q)$ $C_4 = (p \vee q)$ $C_6 = (\neg q \vee \neg r)$

- $\mathcal{F}|_{\{p\}} = \{C_1\} \in$ SAT.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Def: A set $V \subseteq Var(\mathcal{F})$ is a *variable-MUS (VMUS)* of $\mathcal{F}$ if $\mathcal{F}|_V \in$ UNSAT, and for any $V' \subset V$, $\mathcal{F}|_{V'} \in$ SAT.

$C_1 = (p)$ $C_3 = (\neg p \vee \neg q)$ $C_5 = (\neg p \vee r)$

$C_2 = (q)$ $C_4 = (p \vee q)$ $C_6 = (\neg q \vee \neg r)$

- $\mathcal{F}|_{\{q\}} = \{C_2\} \in$ SAT.

Variable MUSes

Def: The subformula of CNF $\mathcal{F}$ *induced* by $V \subseteq Var(\mathcal{F})$ is the formula $\mathcal{F}|_V = \{C \mid C \in \mathcal{F} \text{ and } Var(C) \subseteq V\}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in V.

Def: A set $V \subseteq Var(\mathcal{F})$ is a *variable-MUS (VMUS)* of $\mathcal{F}$ if $\mathcal{F}|_V \in$ UNSAT, and for any $V' \subset V$, $\mathcal{F}|_{V'} \in$ SAT.

$$C_1 = (p) \qquad C_3 = (\neg p \vee \neg q) \qquad C_5 = (\neg p \vee r)$$

$$C_2 = (q) \qquad C_4 = (p \vee q) \qquad C_6 = (\neg q \vee \neg r)$$

- Hence, $\{p, q\}$ is a VMUS of $\mathcal{F}$. Notation: $\{p, q\} \in \text{VMUS}(\mathcal{F})$.

Computing Variable MUSes

Basic algorithm is similar to deletion-based MUS extraction algorithm: based on detection of *necessary* variables.

Notation: for $v \in Var(\mathcal{F})$, $\mathcal{F}^v = \{C \mid C \in \mathcal{F} \text{ and } v \in Var(C)\}$.

Definition

$v \in Var(\mathcal{F})$ is *necessary* for $\mathcal{F}$ if $\mathcal{F} \in$ UNSAT and $\mathcal{F} \setminus \mathcal{F}^v \in$ SAT.

Computing Variable MUSes

Basic algorithm is similar to deletion-based MUS extraction algorithm: based on detection of *necessary* variables.

Notation: for $v \in Var(\mathcal{F})$, $\mathcal{F}^v = \{C \mid C \in \mathcal{F} \text{ and } v \in Var(C)\}$.

Definition

$v \in Var(\mathcal{F})$ is *necessary* for $\mathcal{F}$ if $\mathcal{F} \in$ UNSAT and $\mathcal{F} \setminus \mathcal{F}^v \in$ SAT.

Properties:

1. $V \in \text{VMUS}(\mathcal{F})$ if and only if every $v \in V$ is necessary for $\mathcal{F}|_V$.
2. If v is necessary for $\mathcal{F}$, then v is necessary for any unsatisfiable $\mathcal{F}' \subseteq \mathcal{F}$.
3. Fact: v is necessary for $\mathcal{F}$ iff for some τ, $\tau(\mathcal{F} \setminus \mathcal{F}^v) = 1$, and $\tau(\mathcal{F}^v) = 0$.

Computing Variable MUSes: optimizations [Belov et al, SAT'12]

Model rotation — needs to be modified:

- When $\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$, the model falsifies some clauses with v.

Computing Variable MUSes: optimizations [Belov et al, SAT'12]

Model rotation — needs to be modified:

- When $\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$, the model falsifies some clauses with v.
- If we flip v (as in MUS extraction) the only clauses that will get falsified are those with variable v as well (in the opposite polarity). Does this give us anything ?

Computing Variable MUSes: optimizations [Belov et al, SAT'12]

Model rotation — needs to be modified:

- When $\mathcal{F} \setminus \mathcal{F}^v \in$ SAT, the model falsifies some clauses with v.
- If we flip v (as in MUS extraction) the only clauses that will get falsified are those with variable v as well (in the opposite polarity). Does this give us anything ?
- Fact: For any τ, any variable shared among clauses of $\mathit{Unsat}(\mathcal{F}, \tau)$ is necessary for $\mathcal{F}$. Note: In particular, when $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$ is a singleton – all variables of C are necessary.

Computing Variable MUSes: optimizations [Belov et al, SAT'12]

Model rotation — needs to be modified:

- When $\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$, the model falsifies some clauses with v.
- If we flip v (as in MUS extraction) the only clauses that will get falsified are those with variable v as well (in the opposite polarity). Does this give us anything ?
- Fact: For any τ, any variable shared among clauses of $\mathit{Unsat}(\mathcal{F}, \tau)$ is necessary for $\mathcal{F}$. Note: In particular, when $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$ is a singleton – all variables of C are necessary.
- For the assignment returned by SAT solver, v satisfies this condition, but maybe other variables as well.

Computing Variable MUSes: optimizations [Belov et al, SAT'12]

Model rotation — needs to be modified:

- When $\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$, the model falsifies some clauses with v.
- If we flip v (as in MUS extraction) the only clauses that will get falsified are those with variable v as well (in the opposite polarity). Does this give us anything ?
- Fact: For any τ, any variable shared among clauses of $\mathit{Unsat}(\mathcal{F}, \tau)$ is necessary for $\mathcal{F}$. Note: In particular, when $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$ is a singleton – all variables of C are necessary.
- For the assignment returned by SAT solver, v satisfies this condition, but maybe other variables as well.
- So, flip v, but collect variables shared falsified clauses. Continue recursively.

Computing Variable MUSes: optimizations

Variable-set refinement:

- If a variable is not in the unsatisfiable core, remove it.

Computing Variable MUSes: optimizations

Variable-set refinement:

- If a variable is not in the unsatisfiable core, remove it.

Redundancy removal — same trick as for groups, but with $CNF(\neg \mathcal{F}^v)$.

Computing Variable MUSes: optimizations

Variable-set refinement:

- If a variable is not in the unsatisfiable core, remove it.

Redundancy removal — same trick as for groups, but with $CNF(\neg\mathcal{F}^v)$.

All optimizations are very effective in this setting.

Computing Variable MUSes: optimizations

Variable-set refinement:

- If a variable is not in the unsatisfiable core, remove it.

Redundancy removal — same trick as for groups, but with $CNF(\neg\mathcal{F}^v)$.

All optimizations are very effective in this setting.

VMUSes can often be computed faster than MUSes (not always !).

Computing Variable MUSes: optimizations

Variable-set refinement:

- If a variable is not in the unsatisfiable core, remove it.

Redundancy removal — same trick as for groups, but with $CNF(\neg \mathcal{F}^v)$.

All optimizations are very effective in this setting.

VMUSes can often be computed faster than MUSes (not always !).
Another idea: translate to group-MUS.

- Doesn't work well, compared to the dedicated algorithm.

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

What happens to M if we remove some clause C from $\mathcal{F}$?

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

What happens to M if we remove some clause C from $\mathcal{F}$?

Case 1: M grows — the clause C is *irredundant* in $\mathcal{F}$.

C "blocks" some assignments that no other clause does.

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

What happens to M if we remove some clause C from $\mathcal{F}$?

Case 1: M grows — the clause C is *irredundant* in $\mathcal{F}$.

C "blocks" some assignments that no other clause does.

Case 2: M does not change — the clause C is *redundant* in $\mathcal{F}$.

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

What happens to M if we remove some clause C from $\mathcal{F}$?

Case 1: M grows — the clause C is *irredundant* in $\mathcal{F}$.

C "blocks" some assignments that no other clause does.

Case 2: M does not change — the clause C is *redundant* in $\mathcal{F}$.

$$C_1 = (x) \qquad C_3 = (y \vee z) \qquad C_5 = (\neg z \vee t)$$

$$C_2 = (x \vee t) \qquad C_4 = (\neg y \vee \neg z) \qquad C_6 = (y \vee t)$$

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

What happens to M if we remove some clause C from $\mathcal{F}$?

Case 1: M grows — the clause C is *irredundant* in $\mathcal{F}$.

C "blocks" some assignments that no other clause does.

Case 2: M does not change — the clause C is *redundant* in $\mathcal{F}$.

$C_1 = (x)$ $C_3 = (y \vee z)$ $C_5 = (\neg z \vee t)$

$C_2 = (x \vee t)$ $C_4 = (\neg y \vee \neg z)$ $C_6 = (y \vee t)$

C_1 is irredundant in $\mathcal{F}$.

Minimal Equivalent Subformulas

Let M be the set of all models of CNF formula $\mathcal{F}$.

What happens to M if we remove some clause C from $\mathcal{F}$?

Case 1: M grows — the clause C is *irredundant* in $\mathcal{F}$.

C "blocks" some assignments that no other clause does.

Case 2: M does not change — the clause C is *redundant* in $\mathcal{F}$.

$$C_1 = (x) \qquad C_3 = (y \vee z) \qquad C_5 = (\neg z \vee t)$$

$$C_2 = (x \vee t) \qquad C_4 = (\neg y \vee \neg z) \qquad C_6 = (y \vee t)$$

C_1 is irredundant in $\mathcal{F}$.
C_2 is redundant in $\mathcal{F}$.

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

$C_1 = (x)$ $C_3 = (y \vee z)$ $C_5 = (\neg z \vee t)$

$C_2 = (x \vee t)$ $C_4 = (\neg y \vee \neg z)$ $C_6 = (y \vee t)$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

$C_1 = (x)$ $C_3 = (y \vee z)$ $C_5 = (\neg z \vee t)$

$C_2 = (x \vee t)$ $C_4 = (\neg y \vee \neg z)$ $C_6 = (y \vee t)$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

$C_1 = (x)$ $C_3 = (y \vee z)$ $C_5 = (\neg z \vee t)$

$C_2 = (x \vee t)$ $C_4 = (\neg y \vee \neg z)$ $C_6 = (y \vee t)$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

$C_1 = (x)$ $C_3 = (y \vee z)$ $C_5 = (\neg z \vee t)$

$C_2 = (x \vee t)$ $C_4 = (\neg y \vee \neg z)$ $C_6 = (y \vee t)$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1)

$$C_1 = (x) \qquad C_3 = (y \vee z) \qquad C_5 = (\neg z \vee t)$$

$$C_2 = (x \vee t) \qquad C_4 = (\neg y \vee \neg z) \qquad C_6 = (y \vee t)$$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1) : $\mathcal{F}' = \mathcal{F} \setminus \{C_2\}$

$$C_1 = (x) \qquad C_3 = (y \vee z) \qquad C_5 = (\neg z \vee t)$$

$$C_2 = (x \vee t) \qquad C_4 = (\neg y \vee \neg z) \qquad C_6 = (y \vee t)$$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1) : $\mathcal{F}' = \mathcal{F} \setminus \{C_2\}$
- C_6 is redundant in $\mathcal{F}'$ (resolve C_3 and C_5)

$$C_1 = (x) \qquad C_3 = (y \vee z) \qquad C_5 = (\neg z \vee t)$$

$$C_2 = (x \vee t) \qquad C_4 = (\neg y \vee \neg z) \qquad C_6 = (y \vee t)$$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1) : $\mathcal{F}' = \mathcal{F} \setminus \{C_2\}$
- C_6 is redundant in $\mathcal{F}'$ (resolve C_3 and C_5) : $\mathcal{F}'' = \mathcal{F}' \setminus \{C_6\}$

$$C_1 = (x) \qquad C_3 = (y \vee z) \qquad C_5 = (\neg z \vee t)$$

$$C_2 = (x \vee t) \qquad C_4 = (\neg y \vee \neg z) \qquad C_6 = (y \vee t)$$

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1) : $\mathcal{F}' = \mathcal{F} \setminus \{C_2\}$
- C_6 is redundant in $\mathcal{F}'$ (resolve C_3 and C_5) : $\mathcal{F}'' = \mathcal{F}' \setminus \{C_6\}$
- the rest of clauses are irredundant in $\mathcal{F}''$

$C_1 = (x)$	$C_3 = (y \vee z)$	$C_5 = (\neg z \vee t)$
$C_2 = (x \vee t)$	$C_4 = (\neg y \vee \neg z)$	$C_6 = (y \vee t)$

$\mathcal{F}'' \subset \mathcal{F}$ is irredundant and logically equivalent to $\mathcal{F}$.

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1) : $\mathcal{F}' = \mathcal{F} \setminus \{C_2\}$
- C_6 is redundant in $\mathcal{F}'$ (resolve C_3 and C_5) : $\mathcal{F}'' = \mathcal{F}' \setminus \{C_6\}$
- the rest of clauses are irredundant in $\mathcal{F}''$

$C_1 = (x)$	$C_3 = (y \vee z)$	$C_5 = (\neg z \vee t)$
$C_2 = (x \vee t)$	$C_4 = (\neg y \vee \neg z)$	$C_6 = (y \vee t)$

$\mathcal{F}'' \subset \mathcal{F}$ is irredundant and logically equivalent to $\mathcal{F}$.
$\mathcal{F}''$ is a *Minimal Equivalent Subformula (MES)* of $\mathcal{F}$.

Minimal Equivalent Subformulas [Kleine Büning and Zhao, AMAI'05; Liberatore, AIJ'05]

A clause $C \in \mathcal{F}$ is *redundant* in $\mathcal{F}$ if $\mathcal{F} \setminus \{C\} \equiv \mathcal{F}$.

Alternatively: $C \in \mathcal{F}$ is *redundant* if $\mathcal{F} \setminus \{C\} \models C$.

A CNF formula $\mathcal{F}$ is *irredundant* if it does not have redundant clauses.

- I.e. every clause "serves a purpose"

Keep on removing redundant clauses, until the remainder is irredundant:

- C_2 is redundant in $\mathcal{F}$ (subsumed by C_1) : $\mathcal{F}' = \mathcal{F} \setminus \{C_2\}$
- C_6 is redundant in $\mathcal{F}'$ (resolve C_3 and C_5) : $\mathcal{F}'' = \mathcal{F}' \setminus \{C_6\}$
- the rest of clauses are irredundant in $\mathcal{F}''$

$C_1 = (x)$	$C_3 = (y \vee z)$	$C_5 = (\neg z \vee t)$
$C_2 = (x \vee t)$	$C_4 = (\neg y \vee \neg z)$	$C_6 = (y \vee t)$

$\mathcal{F}'' \subset \mathcal{F}$ is irredundant and logically equivalent to $\mathcal{F}$.
$\mathcal{F}''$ is a *Minimal Equivalent Subformula (MES)* of $\mathcal{F}$. There is another.

Minimal Equivalent Subformulas

Def: A *Minimal Equivalent Subformula (MES)* of a CNF $\mathcal{F}$ is an irredundant subformula of $\mathcal{F}$ logically equivalent to $\mathcal{F}$.

Minimal Equivalent Subformulas

Def: A *Minimal Equivalent Subformula (MES)* of a CNF $\mathcal{F}$ is an irredundant subformula of $\mathcal{F}$ logically equivalent to $\mathcal{F}$.

- $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{F}' \equiv \mathcal{F}$ and $\forall C \in \mathcal{F}'$, $\mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$

Minimal Equivalent Subformulas

Def: A *Minimal Equivalent Subformula (MES)* of a CNF $\mathcal{F}$ is an irredundant subformula of $\mathcal{F}$ logically equivalent to $\mathcal{F}$.

- $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{F}' \equiv \mathcal{F}$ and $\forall C \in \mathcal{F}'$, $\mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$

Note: If $\mathcal{F} \in$ UNSAT, then an MES is a Minimal Unsatisfiable Subformula (MUS) of $\mathcal{F}$.

Minimal Equivalent Subformulas

Def: A *Minimal Equivalent Subformula (MES)* of a CNF $\mathcal{F}$ is an irredundant subformula of $\mathcal{F}$ logically equivalent to $\mathcal{F}$.

- $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{F}' \equiv \mathcal{F}$ and $\forall C \in \mathcal{F}', \mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$

Note: If $\mathcal{F} \in$ UNSAT, then an MES is a Minimal Unsatisfiable Subformula (MUS) of $\mathcal{F}$.

Computation of MESes is about removing redundant clauses. Why would we want to do this ?

Minimal Equivalent Subformulas

Def: A *Minimal Equivalent Subformula (MES)* of a CNF $\mathcal{F}$ is an irredundant subformula of $\mathcal{F}$ logically equivalent to $\mathcal{F}$.

- $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{F}' \equiv \mathcal{F}$ and $\forall C \in \mathcal{F}', \mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$

Note: If $\mathcal{F} \in$ UNSAT, then an MES is a Minimal Unsatisfiable Subformula (MUS) of $\mathcal{F}$.

Computation of MESes is about removing redundant clauses. Why would we want to do this ?

- In some applications redundancy is undesirable:
  - knowledge bases [P. Liberatore, AIJ 2005]
  - conformant and contingent planning [S.T. To, et al, AAAI 2010-11]
  - probabilistic reasoning systems [M. Niepert, et al, J. of Approx. Reasoning, 2010]

Minimal Equivalent Subformulas

Def: A *Minimal Equivalent Subformula (MES)* of a CNF $\mathcal{F}$ is an irredundant subformula of $\mathcal{F}$ logically equivalent to $\mathcal{F}$.

- $\mathcal{F}' \subseteq \mathcal{F}$ and $\mathcal{F}' \equiv \mathcal{F}$ and $\forall C \in \mathcal{F}'$, $\mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$

Note: If $\mathcal{F} \in$ UNSAT, then an MES is a Minimal Unsatisfiable Subformula (MUS) of $\mathcal{F}$.

Computation of MESes is about removing redundant clauses. Why would we want to do this ?

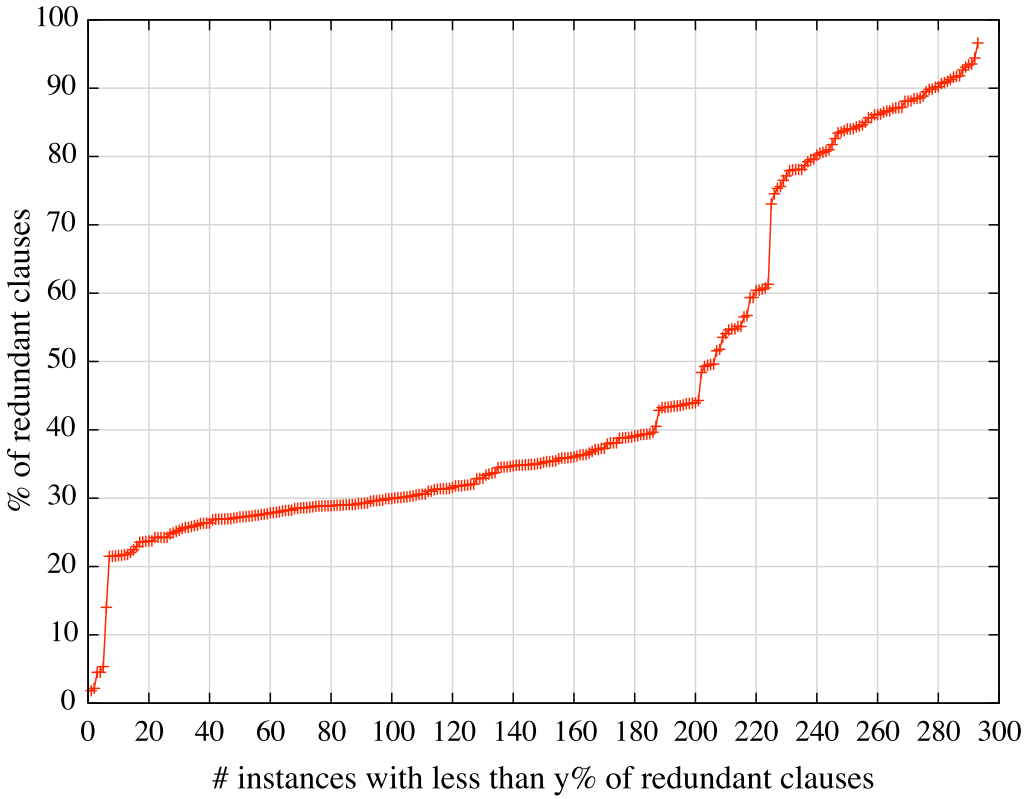
- In some applications redundancy is undesirable:
  - knowledge bases [P. Liberatore, AIJ 2005]
  - conformant and contingent planning [S.T. To, et al, AAAI 2010-11]
  - probabilistic reasoning systems [M. Niepert, et al, J. of Approx. Reasoning, 2010]
- Construction of concise CNF encodings

Is Redundancy Common ?

A sample of 300 benchmarks from various applications of SAT

Is Redundancy Common ?

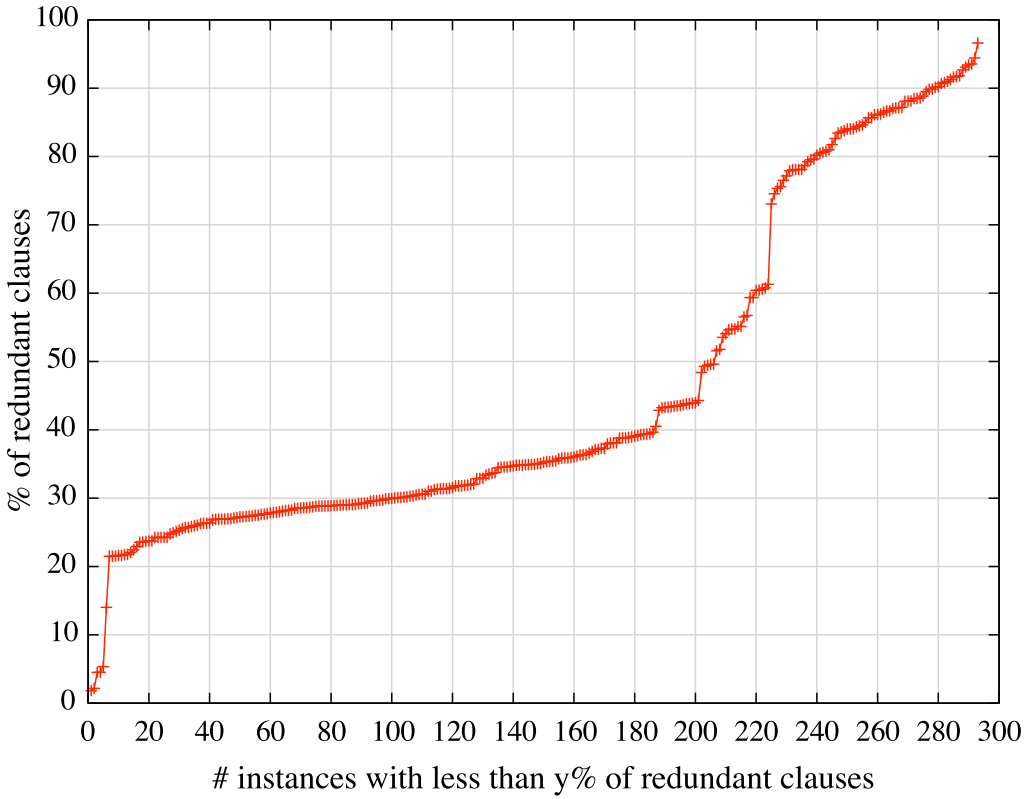
A sample of 300 benchmarks from various applications of SAT



- Number of instances (x) with less than y% redundant clauses

Is Redundancy Common ?

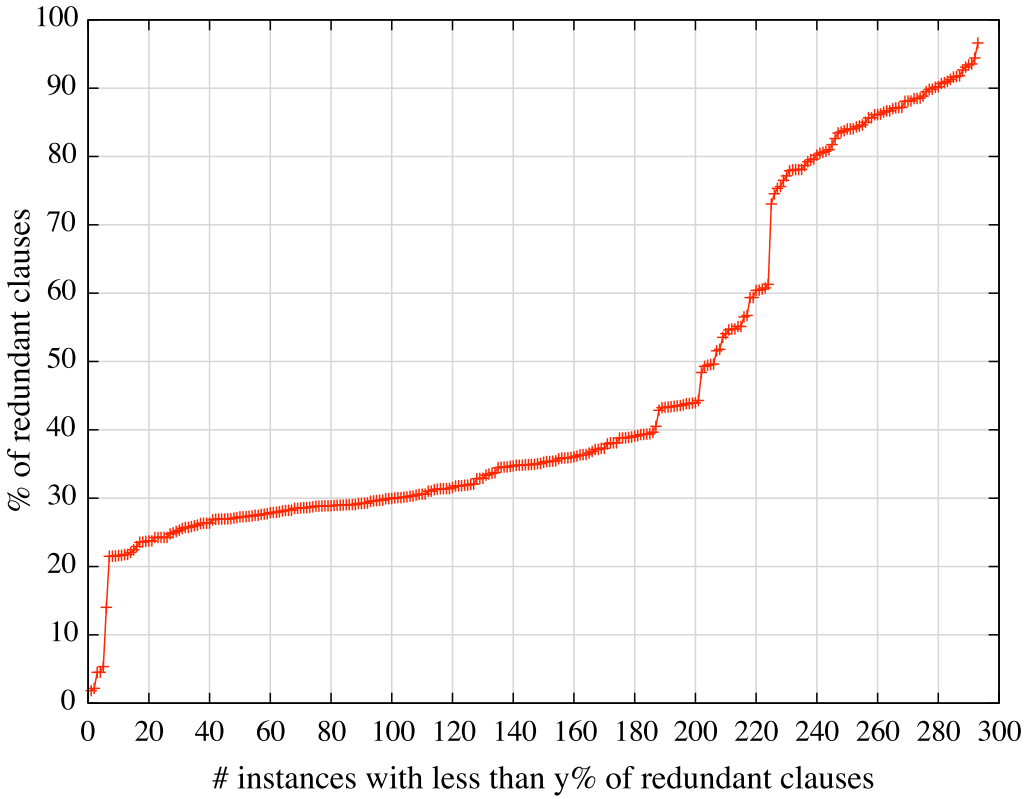
A sample of 300 benchmarks from various applications of SAT



- Number of instances (x) with less than y% redundant clauses
- 2/3 of instances have 20%-50% redundant clauses

Is Redundancy Common ?

A sample of 300 benchmarks from various applications of SAT



- Number of instances (x) with less than y% redundant clauses
- 2/3 of instances have 20%-50% redundant clauses
- 1/3 has $>$ 50%, in some cases $>$ 90%, of redundant clauses !

Computing MESes

How to check if C is redundant in $\mathcal{F}$, i.e. $\mathcal{F} \setminus \{C\} \models C$?

Computing MESes

How to check if C is redundant in $\mathcal{F}$, i.e. $\mathcal{F} \setminus \{C\} \models C$?

- In general, $\alpha \models \beta$ **iff** $\alpha \wedge \neg\beta \in$ UNSAT

Computing MESes

How to check if C is redundant in $\mathcal{F}$, i.e. $\mathcal{F} \setminus \{C\} \models C$?

- In general, $\alpha \models \beta$ **iff** $\alpha \wedge \neg\beta \in$ UNSAT
- Use SAT solver to check satisfiability of $\mathcal{F} \setminus \{C\} \cup \mathsf{CNF}(\neg C)$
  - UNSAT — C is redundant in $\mathcal{F}$. SAT — C is irredundant in $\mathcal{F}$.

Computing MESes

How to check if C is redundant in $\mathcal{F}$, i.e. $\mathcal{F} \setminus \{C\} \models C$?

- In general, $\alpha \models \beta$ **iff** $\alpha \wedge \neg\beta \in$ UNSAT
- Use SAT solver to check satisfiability of $\mathcal{F} \setminus \{C\} \cup \mathsf{CNF}(\neg C)$
  - UNSAT — C is redundant in $\mathcal{F}$. SAT — C is irredundant in $\mathcal{F}$.

Deletion-based MES Computation

Input $\mapsto$ **Output**: CNF Formula $\mathcal{F}$ $\mapsto$ an MES $\mathcal{M}$ of $\mathcal{F}$

$\mathcal{M} \leftarrow \mathcal{F}$ // Inv: $\mathcal{M}$ is a superset of some MES of $\mathcal{F}$
foreach $C \in \mathcal{M}$ **do**
  if not SAT($\mathcal{M} \setminus \{C\} \cup \mathsf{CNF}(\neg C)$) **then** // is C redundant ?
    // yes - delete it
    $\mathcal{M} \leftarrow \mathcal{M} \setminus \{C\}$
  // no - keep it
return $\mathcal{M}$ // Every $C \in \mathcal{M}$ is irredundant in $\mathcal{M}$

Computing MESes [Belov et al, CP'12]

But, insertion-based (and dichotomic) MUS extraction algorithms needs to be ported carefully.

Computing MESes [Belov et al, CP'12]

But, insertion-based (and dichotomic) MUS extraction algorithms needs to be ported carefully.

As per insertion-based MUS extraction algorithm:

- $\mathcal{F}$ – the input formula,
- $\mathcal{M} \subset \mathcal{F}$ – the set of irredundant clauses computed so far (under-approximation of an MES of $\mathcal{F}$)
- $\mathcal{S}$ – the set of untested clauses

Computing MESes [Belov et al, CP'12]

But, insertion-based (and dichotomic) MUS extraction algorithms needs to be ported carefully.

As per insertion-based MUS extraction algorithm:

- $\mathcal{F}$ – the input formula,
- $\mathcal{M} \subset \mathcal{F}$ – the set of irredundant clauses computed so far (under-approximation of an MES of $\mathcal{F}$)
- $\mathcal{S}$ – the set of untested clauses
- When computing MUS of $\mathcal{F} \in$ UNSAT, a clause $C \in \mathcal{F}$ is added to $\mathcal{M}$ when $\mathcal{M} \cup \mathcal{S} \in$ SAT but $\mathcal{M} \cup \mathcal{S} \cup \{C\} \in$ UNSAT.
  - transition from SAT to UNSAT.

Computing MESes [Belov et al, CP'12]

But, insertion-based (and dichotomic) MUS extraction algorithms needs to be ported carefully.

As per insertion-based MUS extraction algorithm:

- $\mathcal{F}$ – the input formula,
- $\mathcal{M} \subset \mathcal{F}$ – the set of irredundant clauses computed so far (under-approximation of an MES of $\mathcal{F}$)
- $\mathcal{S}$ – the set of untested clauses
- When computing MUS of $\mathcal{F} \in$ UNSAT, a clause $C \in \mathcal{F}$ is added to $\mathcal{M}$ when $\mathcal{M} \cup \mathcal{S} \in$ SAT but $\mathcal{M} \cup \mathcal{S} \cup \{C\} \in$ UNSAT.
  - transition from SAT to UNSAT.
- When computing MESes we want to add a clause C to $\mathcal{M}$ when $\mathcal{M} \cup \mathcal{S} \not\equiv \mathcal{F}$ but $\mathcal{M} \cup \mathcal{S} \cup \{C\} \equiv \mathcal{F}$.
  - transition from $\not\equiv \mathcal{F}$ to $\equiv \mathcal{F}$.

Computing MESes [Belov et al, CP'12]

But, insertion-based (and dichotomic) MUS extraction algorithms needs to be ported carefully.

As per insertion-based MUS extraction algorithm:

- $\mathcal{F}$ – the input formula,
- $\mathcal{M} \subset \mathcal{F}$ – the set of irredundant clauses computed so far (under-approximation of an MES of $\mathcal{F}$)
- $\mathcal{S}$ – the set of untested clauses
- When computing MUS of $\mathcal{F} \in$ UNSAT, a clause $C \in \mathcal{F}$ is added to $\mathcal{M}$ when $\mathcal{M} \cup \mathcal{S} \in$ SAT but $\mathcal{M} \cup \mathcal{S} \cup \{C\} \in$ UNSAT.
  - transition from SAT to UNSAT.
- When computing MESes we want to add a clause C to $\mathcal{M}$ when $\mathcal{M} \cup \mathcal{S} \not\equiv \mathcal{F}$ but $\mathcal{M} \cup \mathcal{S} \cup \{C\} \equiv \mathcal{F}$.
  - transition from $\not\equiv \mathcal{F}$ to $\equiv \mathcal{F}$.
- SAT check needs to be modified: e.g. the algorithm loops while SAT($\mathcal{M} \cup \mathcal{S} \cup$ CNF($\neg \mathcal{F}$) Note: in fact, do not need to negate the whole $\mathcal{F}$.

Computing MESes: optimizations

Model rotation:

- Fact: C is irredundant in $\mathcal{F}$ **iff** for τ, $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$. Note: the same property as in MUSes.

Computing MESes: optimizations

Model rotation:

- Fact: C is irredundant in $\mathcal{F}$ **iff** for τ, $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$. Note: the same property as in MUSes.
- Almost the same as with MUSes: may need to rotate through satisfied clauses.

Computing MESes: optimizations

Model rotation:

- Fact: C is irredundant in $\mathcal{F}$ **iff** for τ, $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$. Note: the same property as in MUSes.
- Almost the same as with MUSes: may need to rotate through satisfied clauses.
- Works well on medium-to-low redundancy instances

Computing MESes: optimizations

Model rotation:

- Fact: C is irredundant in $\mathcal{F}$ **iff** for τ, $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$. Note: the same property as in MUSes.
- Almost the same as with MUSes: may need to rotate through satisfied clauses.
- Works well on medium-to-low redundancy instances
- But: does not help for highly-redundant instances

Computing MESes: optimizations

Model rotation:

- Fact: C is irredundant in $\mathcal{F}$ **iff** for τ, $\mathit{Unsat}(\mathcal{F}, \tau) = \{C\}$. Note: the same property as in MUSes.
- Almost the same as with MUSes: may need to rotate through satisfied clauses.
- Works well on medium-to-low redundancy instances
- But: does not help for highly-redundant instances

Clause-set refinement cannot be applied soundly to MES extraction !

Example: check C_2 in $\mathcal{F} = \{C_1 = (p), C_2 = (p \vee q), C_3 = (r)\}$.

Computing MESes: reduction to group-MUS

Observation

- $\mathcal{F}' \subseteq \mathcal{F}$ is an MES of $\mathcal{F}$ **iff**:
  - $\mathcal{F}' \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{UNSAT}$, i.e. $\mathcal{F}' \models \mathcal{F}$, and so $\mathcal{F}' \equiv \mathcal{F}$, and
  - $\forall C \in \mathcal{F}'$, $(\mathcal{F}' \setminus \{C\}) \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{SAT}$, i.e. $\mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$

Computing MESes: reduction to group-MUS

Observation

- $\mathcal{F}' \subseteq \mathcal{F}$ is an MES of $\mathcal{F}$ **iff**:
  - $\mathcal{F}' \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{UNSAT}$, i.e. $\mathcal{F}' \models \mathcal{F}$, and so $\mathcal{F}' \equiv \mathcal{F}$, and
  - $\forall C \in \mathcal{F}'$, $(\mathcal{F}' \setminus \{C\}) \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{SAT}$, i.e. $\mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$
- So, to compute an MES, we can:
  - start with $\mathcal{F} \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{UNSAT}$, and
  - find $\mathcal{F}' \subseteq \mathcal{F}$ such that $\mathcal{F}' \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{UNSAT}$, and $\forall C \in \mathcal{F}'$, $(\mathcal{F}' \setminus \{C\}) \cup \mathsf{CNF}(\neg\mathcal{F}) \in \mathsf{SAT}$

Computing MESes: reduction to group-MUS

Observation

- $\mathcal{F}' \subseteq \mathcal{F}$ is an MES of $\mathcal{F}$ **iff**:
  - $\mathcal{F}' \cup \text{CNF}(\neg\mathcal{F}) \in \text{UNSAT}$, i.e. $\mathcal{F}' \models \mathcal{F}$, and so $\mathcal{F}' \equiv \mathcal{F}$, and
  - $\forall C \in \mathcal{F}'$, $(\mathcal{F}' \setminus \{C\}) \cup \text{CNF}(\neg\mathcal{F}) \in \text{SAT}$, i.e. $\mathcal{F}' \setminus \{C\} \not\equiv \mathcal{F}$
- So, to compute an MES, we can:
  - start with $\mathcal{F} \cup \text{CNF}(\neg\mathcal{F}) \in \text{UNSAT}$, and
  - find $\mathcal{F}' \subseteq \mathcal{F}$ such that $\mathcal{F}' \cup \text{CNF}(\neg\mathcal{F}) \in \text{UNSAT}$, and $\forall C \in \mathcal{F}'$, $(\mathcal{F}' \setminus \{C\}) \cup \text{CNF}(\neg\mathcal{F}) \in \text{SAT}$
- This is an instance of *group-MUS* computation problem !
  - take "group 0" to be $\text{CNF}(\neg\mathcal{F})$, and
  - make a singleton group for each $C \in \mathcal{F}$

Computing MESes: reduction to group-MUS

So what ?

- Efficient group-MUS algorithms and tools are available.

Computing MESes: reduction to group-MUS

So what ?

- Efficient group-MUS algorithms and tools are available.
- Clause-set refinement is sound and effective again.

Computing MESes: reduction to group-MUS

So what ?

- Efficient group-MUS algorithms and tools are available.
- Clause-set refinement is sound and effective again.
- Turns out that reduction can be done incrementally: family of algorithms (that includes the deletion-based algorithm).

MUSer2 — MUS, group-MUS, VMUS, MES extractor

Algorithms:

- Hybrid algorithm
- Insertion-based
- Dichotomic
- A number of additional specialized algorithms for MES extraction.

MUSer2 — MUS, group-MUS, VMUS, MES extractor

Algorithms:

- Hybrid algorithm
- Insertion-based
- Dichotomic
- A number of additional specialized algorithms for MES extraction.

Optimizations:

- Clause-set refinement and trimming
- Recursive model rotation
- (Adaptive) redundancy removal

MUSer2 — MUS, group-MUS, VMUS, MES extractor

Algorithms:

- Hybrid algorithm
- Insertion-based
- Dichotomic
- A number of additional specialized algorithms for MES extraction.

Optimizations:

- Clause-set refinement and trimming
- Recursive model rotation
- (Adaptive) redundancy removal

Control/heuristics for clause/group/variable ordering

MUSer2 — MUS, group-MUS, VMUS, MES extractor

Algorithms:

- Hybrid algorithm
- Insertion-based
- Dichotomic
- A number of additional specialized algorithms for MES extraction.

Optimizations:

- Clause-set refinement and trimming
- Recursive model rotation
- (Adaptive) redundancy removal

Control/heuristics for clause/group/variable ordering

Testing of computed MUSes (etc)

MUSer2 — MUS, group-MUS, VMUS, MES extractor

Algorithms:

- Hybrid algorithm
- Insertion-based
- Dichotomic
- A number of additional specialized algorithms for MES extraction.

Optimizations:

- Clause-set refinement and trimming
- Recursive model rotation
- (Adaptive) redundancy removal

Control/heuristics for clause/group/variable ordering

Testing of computed MUSes (etc)

SAT solvers are used in a black-box manner; can use various SAT solvers.

Current and Future Work

Heuristics

Current and Future Work

Heuristics

Scalability

- New algorithms, new optimization techniques.

Current and Future Work

Heuristics

Scalability

- New algorithms, new optimization techniques.

Applications

- New applications typically pose new types of problems.

Current and Future Work

Heuristics

Scalability

- New algorithms, new optimization techniques.

Applications

- New applications typically pose new types of problems.

Thank you for your attention !