SAT Preprocessing for MUS Extraction and MaxSAT

Anton Belov

Joint work with: Matti Järvisalo, António Morgado, Joao Marques-Silva

University College Dublin, Ireland
University of Helsinki, Finland
IST/INESC-ID, Lisbon, Portugal

EPCL Basic Training Camp
November 13-19, 2013
Dresden, Germany

Abstract

State-of-the-art algorithms for industrial instances of MUS extraction problem and MaxSAT rely on iterative calls to a SAT solver. Preprocessing is crucial for the acceleration of SAT solving, and the key preprocessing techniques rely on the application of resolution and subsumption elimination. Additionally, satisfiability-preserving clause elimination procedures are often used. Since the computation typically involves a large number of SAT calls, an interesting question is whether an input instance to a problem can be preprocessed *up-front*, i.e. prior to running an MUS extractor or a MaxSAT solver, rather than (or, in addition to) during each iterative SAT solver call. The key requirement in this setting is that the preprocessing has to be *sound*, i.e. so that the solution to the original problem can be reconstructed correctly and efficiently after the execution of an algorithm on the preprocessed instance. In this talk we will examine some of the obstacles to such up-front preprocessing, and will discuss a solution that involves re-casting the MUS and MaxSAT computation problems in a so-called labelled-CNF framework.

Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework

Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework

Introduction: MUSes, group-MUSes

$$\begin{array}{ccc} C_1 & C_2 & C_3 \\ \boxed{(p)} & \boxed{(q)} & \boxed{(\neg p \vee \neg q)} \end{array}$$

$M = \{C_1, C_2, C_3\}$ is UNSAT

Introduction: MUSes, group-MUSes

$$\begin{array}{ccc} C_1 & C_2 & C_3 \\ \boxed{(p)} & \boxed{(q)} & \boxed{(\neg p \vee \neg q)} \end{array}$$

$M = \{C_1, C_2, C_3\}$ is UNSAT, and $\forall C \in M$, $M \setminus \{C\}$ is SAT.

Introduction: MUSes, group-MUSes

C_1	C_2	C_3
(p)	(q)	$(\neg p \vee \neg q)$

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.
M is a *minimal unsatisfiable subformula (MUS)* of F.

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.
M is a *minimal unsatisfiable subformula (MUS)* of F. And there is another one.

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.
M is a *minimal unsatisfiable subformula (MUS)* of F. And there is another one.

G_0	G_1	G_2
(p) (q)	$(\neg p \vee \neg q)$ $(\neg p \vee r)$	$(p \vee q)$ $(\neg q \vee \neg r)$

A *group-CNF* formula — CNF partitioned into *groups* of clauses.

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.
M is a *minimal unsatisfiable subformula (MUS)* of F. And there is another one.

G_0	G_1	G_2
(p) (q)	$(\neg p \vee \neg q)$ $(\neg p \vee r)$	$(p \vee q)$ $(\neg q \vee \neg r)$

A *group-CNF* formula — CNF partitioned into *groups* of clauses.
$F = \{G_0, G_1, G_2\}$ is UNSAT, but not "group-MU" (can remove G_2).

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \ldots, C_6\}$ is UNSAT, but not MU.
M is a *minimal unsatisfiable subformula (MUS)* of F. And there is another one.

G_0	G_1	G_2
(p) (q)	$(\neg p \vee \neg q)$ $(\neg p \vee r)$	$(p \vee q)$ $(\neg q \vee \neg r)$

A *group-CNF* formula — CNF partitioned into *groups* of clauses.
$F = \{G_0, G_1, G_2\}$ is UNSAT, but not "group-MU" (can remove G_2).
$M = \{G_1\}$ is (the only) group-MUS of F.

Introduction: MUSes, group-MUSes

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

$M = \{C_1, C_2, C_3\}$ is *minimal unsatisfiable (MU)*.
$F = \{C_1, \dots, C_6\}$ is UNSAT, but not MU.
M is a *minimal unsatisfiable subformula (MUS)* of F. And there is another one.

G_0	G_1	G_2
$(\mathbf{p})$ $(\mathbf{q})$	$(\neg p \vee \neg q)$ $(\neg \mathbf{p} \vee \mathbf{r})$	$(p \vee q)$ $(\neg \mathbf{q} \vee \neg \mathbf{r})$

A *group-CNF* formula — CNF partitioned into *groups* of clauses.
$F = \{G_0, G_1, G_2\}$ is UNSAT, but not "group-MU" (can remove G_2).
$M = \{G_1\}$ is (the only) group-MUS of F.

Introduction: MaxSAT

A *MaxSAT solution* for F, is an assignment τ that *maximizes* $|Sat(F, \tau)|$.
Alternatively (and more customary): τ *minimizes* $|Unsat(F, \tau)|$.

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

Introduction: MaxSAT

A *MaxSAT solution* for F, is an assignment τ that *maximizes* $|Sat(F, \tau)|$.
Alternatively (and more customary): τ *minimizes* $|Unsat(F, \tau)|$.

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

$\tau_1 = \{\neg p, q, \neg r\}$

A *MaxSAT solution* for F, is an assignment τ that *maximizes* $|Sat(F, \tau)|$.
Alternatively (and more customary): τ *minimizes* $|Unsat(F, \tau)|$.

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

$\tau_1 = \{\neg p, q, \neg r\}$, $\tau_2 = \{p, \neg q, r\}$

Introduction: MaxSAT

A *MaxSAT solution* for F, is an assignment τ that *maximizes* $|Sat(F, \tau)|$.
Alternatively (and more customary): τ *minimizes* $|Unsat(F, \tau)|$.

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

$\tau_1 = \{\neg p, q, \neg r\}$, $\tau_2 = \{p, \neg q, r\}$

Note: a MaxSAT solution is a model of an MSS of *maximum* cardinality $\Rightarrow$ MaxSAT is about finding an MCS of a *minimum* cardinality.

Introduction: MaxSAT

A *MaxSAT solution* for F, is an assignment τ that *maximizes* $|Sat(F, \tau)|$.
Alternatively (and more customary): τ *minimizes* $|Unsat(F, \tau)|$.

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

$\tau_1 = \{\neg p, q, \neg r\}$, $\tau_2 = \{p, \neg q, r\}$

Note: a MaxSAT solution is a model of an MSS of *maximum* cardinality $\Rightarrow$ MaxSAT is about finding an MCS of a *minimum* cardinality.

Weighted Partial MaxSAT:

- each clause C has a non-negative weight ($\in \mathbb{N}^+$), $w(C)$;
- some clauses are *hard* — *must* be satisfied (akin to group-0);
- weighted CNF (WCNF) formula: $F = F^H \cup F^S$;
- τ is a weighted partial MaxSAT solution for WCNF F if $\tau(F^H) = 1$, and τ minimizes $cost(\tau) = \sum_{C \in Unsat(F^S, \tau)} w(C)$.

Introduction: applications

Applications

- Identification and repair of sources of inconsistency
  - circuit error diagnosis; error localization in product configuration.
- Identification of relevant features of systems:
  - automatic abstraction in model checking;
  - environmental assumptions in formal equivalence checking.
- MaxSAT: debugging, optimization, bioinformatics, etc.

Introduction: applications

Applications

- Identification and repair of sources of inconsistency
  - circuit error diagnosis; error localization in product configuration.
- Identification of relevant features of systems:
  - automatic abstraction in model checking;
  - environmental assumptions in formal equivalence checking.
- MaxSAT: debugging, optimization, bioinformatics, etc.

Computation of (group-)MUSes and MaxSAT solutions

- Algorithms aimed at industrial instances are based on iterative calls to a SAT solver.
- SAT solving is the main bottleneck.
- Number of SAT calls is a function of the size of the input formula.
- In the SAT world: *preprocessing* is essential for efficient SAT solving.

Introduction: preprocessing for SAT

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

Introduction: preprocessing for SAT

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

- *Subsumption elimination*

  C subsumes C' if $C \subset C'$.

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

- *Subsumption elimination*

  C subsumes C' if $C \subset C'$.

- *Blocked clause elimination (BCE)*

  C is blocked if every resolvent of C on some $l \in C$ is a tautology.

Introduction: preprocessing for SAT

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

- *Subsumption elimination*

  C subsumes C' if $C \subset C'$.

- *Blocked clause elimination (BCE)*

  C is blocked if every resolvent of C on some $l \in C$ is a tautology.

Resolution-based preprocessing

- *Boolean constraint propagation (BCP)*

Introduction: preprocessing for SAT

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

- *Subsumption elimination*

  C subsumes C' if $C \subset C'$.

- *Blocked clause elimination (BCE)*

  C is blocked if every resolvent of C on some $l \in C$ is a tautology.

Resolution-based preprocessing

- *Boolean constraint propagation (BCP)*
- *Variable elimination (VE)* (aka DP-reduction)

  $\mathrm{VE}(F, x) = F \cup (F_x \otimes_x F_{\neg x}) \setminus (F_x \cup F_{\neg x})$

Introduction: preprocessing for SAT

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

- *Subsumption elimination*

  C subsumes C' if $C \subset C'$.

- *Blocked clause elimination (BCE)*

  C is blocked if every resolvent of C on some $l \in C$ is a tautology.

Resolution-based preprocessing

- *Boolean constraint propagation (BCP)*
- *Variable elimination (VE)* (aka DP-reduction)

  $\mathrm{VE}(F, x) = F \cup (F_x \otimes_x F_{\neg x}) \setminus (F_x \cup F_{\neg x})$

- *Self-subsuming resolution (SSR)*

  replace $(x \vee C)$ $(\neg x \vee C \vee D)$ with $(x \vee C)$ $(\quad C \vee D)$.

Introduction: preprocessing for SAT

Clause elimination procedures

$E : CNF \mapsto CNF$, $E(F) \subseteq F$, and $E(F)$ is equisatisfiable with F.

- *Subsumption elimination*

  C subsumes C' if $C \subset C'$.

- *Blocked clause elimination (BCE)*

  C is blocked if every resolvent of C on some $l \in C$ is a tautology.

Resolution-based preprocessing

- *Boolean constraint propagation (BCP)*
- *Variable elimination (VE)* (aka DP-reduction)

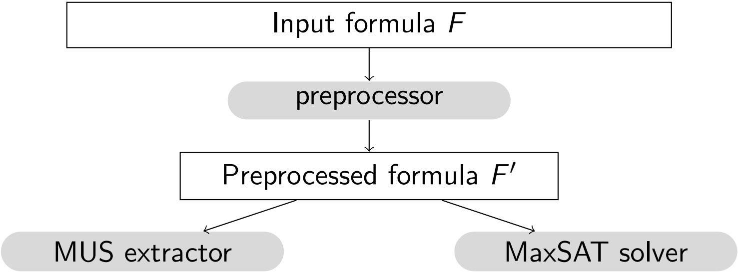
  $\mathrm{VE}(F, x) = F \cup (F_x \otimes_x F_{\neg x}) \setminus (F_x \cup F_{\neg x})$

- *Self-subsuming resolution (SSR)*

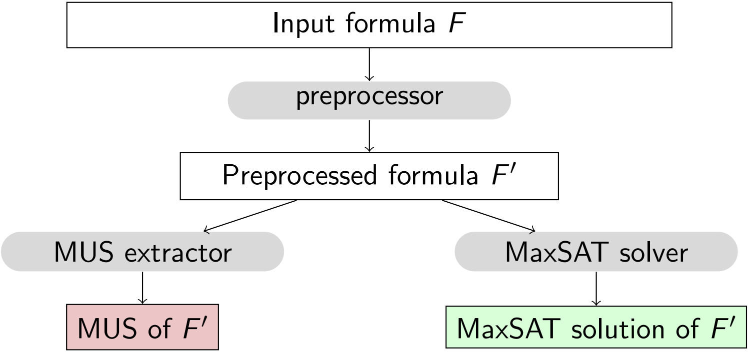
  replace $(x \vee C)$ $(\neg x \vee C \vee D)$ with $(x \vee C)$ $(\quad C \vee D)$.

Important: a model of original formula can be reconstructed *efficiently*.

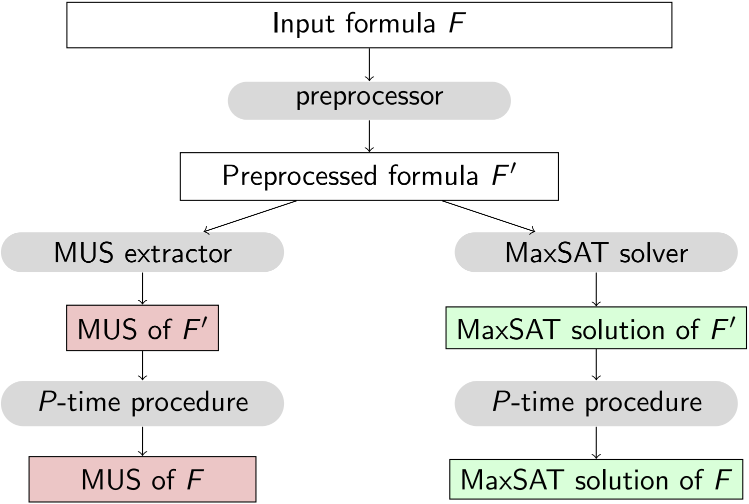
Introduction: preprocessing for MUSes and MaxSAT



Introduction: preprocessing for MUSes and MaxSAT



Introduction: preprocessing for MUSes and MaxSAT



Outline

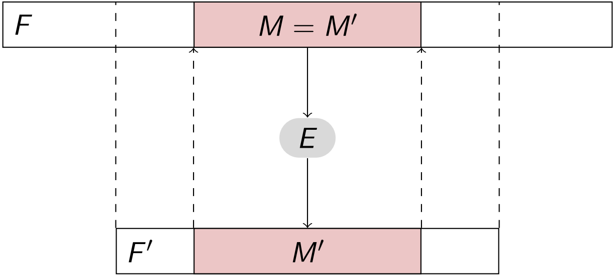
Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

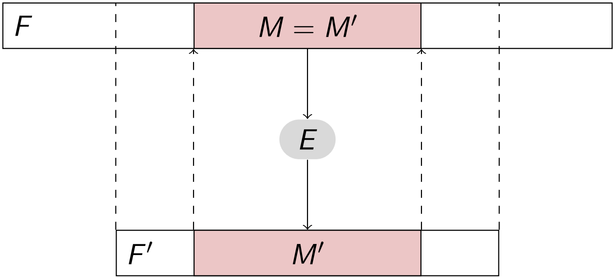
Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework

Plain MUS: clause elimination



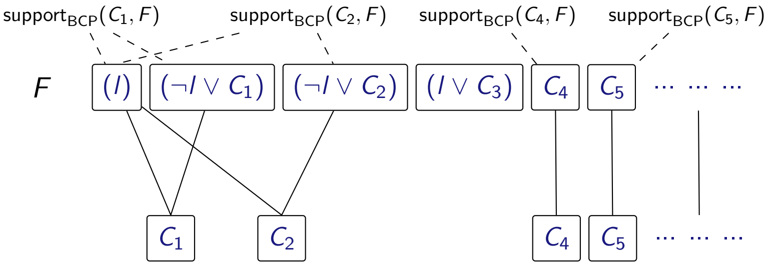
Plain MUS: clause elimination



Prop: Any MUS of $F' = E(F)$ is an MUS of F.

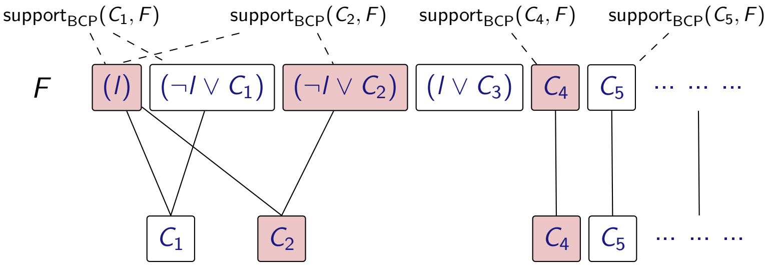
Reconstruction is trivial.

Plain MUS: BCP



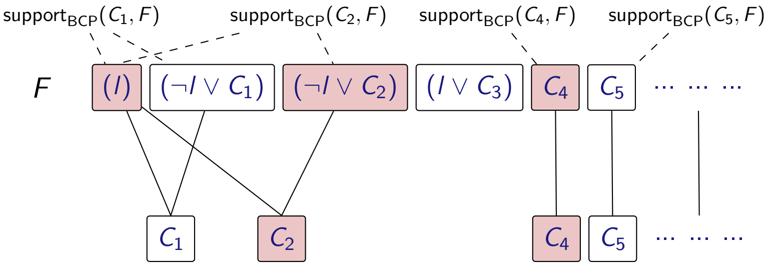
$$F' = \text{BCP}(F, l)$$

Plain MUS: BCP



$$F' = \text{BCP}(F, l)$$

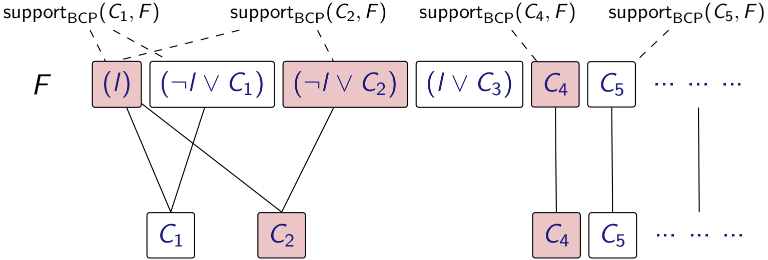
Plain MUS: BCP



$$F' = \mathrm{BCP}(F, l)$$

Prop: If M' is an MUS of $F' = \mathrm{BCP}(F, l)$, then
$M = \bigcup_{C \in M'} \text{support}_{\text{BCP}}(C, F)$ is an MUS of F.

Plain MUS: BCP



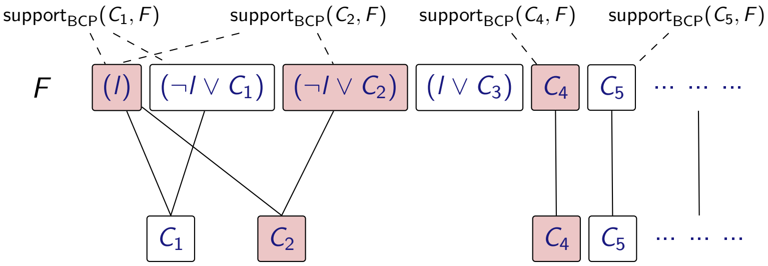
$F' = \mathrm{BCP}(F, l)$

Prop: If M' is an MUS of $F' = \mathrm{BCP}(F, l)$, then

$$M = \bigcup_{C \in M'} \mathrm{support}_{\mathrm{BCP}}(C, F) \text{ is an MUS of } F.$$

Pf: Take a witness τ for C_2 in F'. Then, $\tau \cup \{l\}$ is a witness for $(\neg l \vee C_2)$, while $\tau \cup \{\neg l\}$ is a witness for (l). If τ is a witness for C_4 in F', then $\tau \cup \{l\}$ is a witness for C_4 in F.

Plain MUS: BCP



$$F' = \text{BCP}(F, l)$$

Prop: If M' is an MUS of $F' = \text{BCP}(F, l)$, then

$$M = \bigcup_{C \in M'} \text{support}_{\text{BCP}}(C, F) \text{ is an MUS of } F.$$

Pf: Take a witness τ for C_2 in F'. Then, $\tau \cup \{l\}$ is a witness for $(\neg l \vee C_2)$, while $\tau \cup \{\neg l\}$ is a witness for (l). If τ is a witness for C_4 in F', then $\tau \cup \{l\}$ is a witness for C_4 in F.

Reconstruction can be done in P-time.

Group-MUS: *monotone* clause elimination

Def: A clause elimination procedure E is *monotone* iff for any $F' \subseteq F$, $E(F') \subseteq E(F)$.

Example: BCE — if C is blocked in F, its blocked in any $F' \subseteq F$.

Non-example: SUB — if $C_1 \subset C_2$ in F, but C_1 is not in F'.

Group-MUS: *monotone* clause elimination

Def: A clause elimination procedure E is *monotone* iff for any $F' \subseteq F$, $E(F') \subseteq E(F)$.

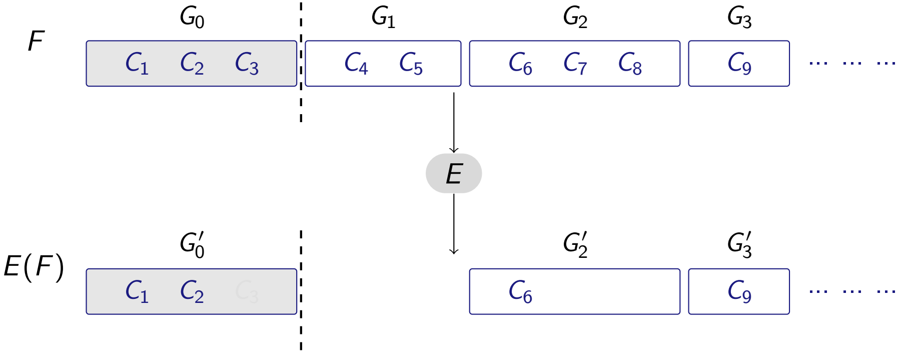
Example: BCE — if C is blocked in F, its blocked in any $F' \subseteq F$.

Non-example: SUB — if $C_1 \subset C_2$ in F, but C_1 is not in F'.

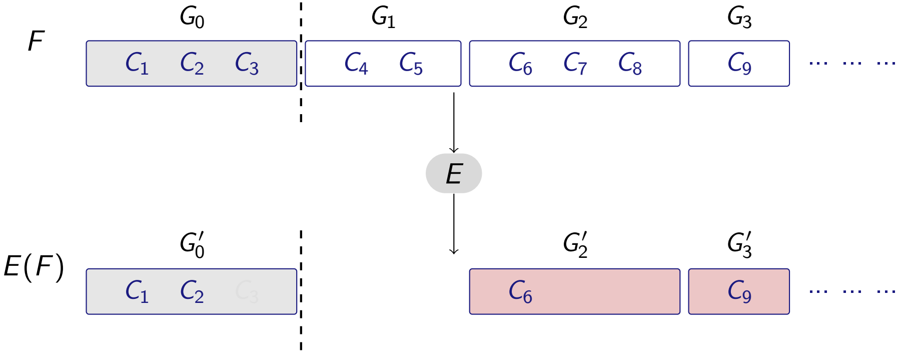
Prop: If E is monotone, then $\text{MUS}(E(F)) = \text{MUS}(F)$, i.e. any monotone clause elimination procedure is *MUS-preserving*.

Pf: if $M \in \text{MUS}(E(F))$, then $M \subseteq E(F) \subseteq F$, i.e. $M \in \text{MUS}(F)$ (doesn't matter that E is monotone); if $M \in \text{MUS}(F)$, then $E(M) = M$ because E is SAT-preserving, and since $M \subseteq F$, we have $E(M) \subseteq E(F)$ by monotonicity, i.e. $M \in \text{MUS}(E(F))$.

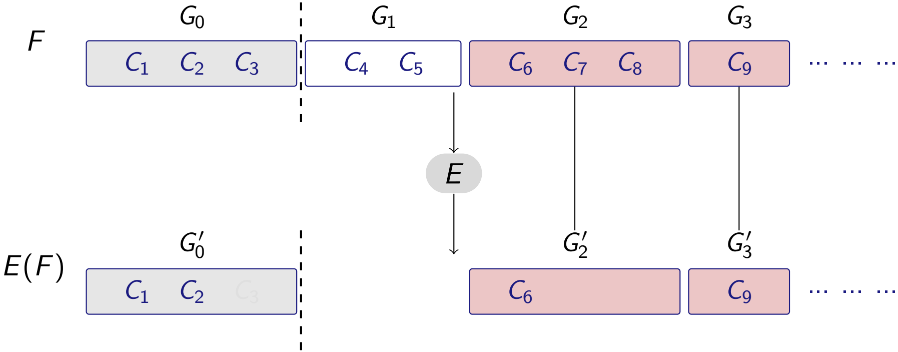
Group-MUS: *monotone* clause elimination



Group-MUS: *monotone* clause elimination

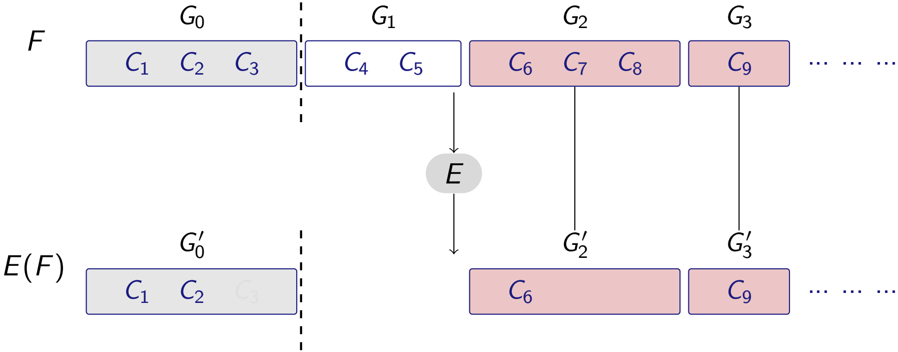


Group-MUS: *monotone* clause elimination



Given a group-MUS M' of $F' = E(F)$, let $M = \{G_i \in F \mid G'_i \in M'\}$.

Group-MUS: *monotone* clause elimination

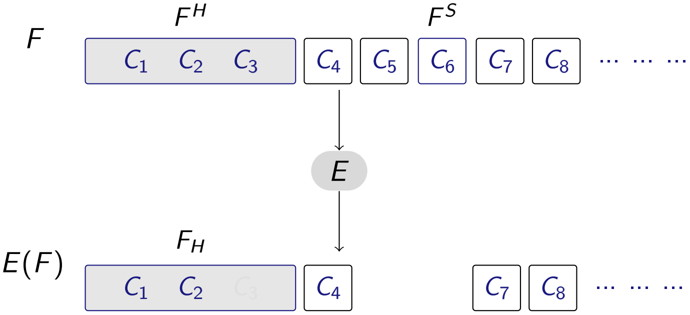


Given a group-MUS M' of $F' = E(F)$, let $M = \{G_i \in F \mid G_i' \in M'\}$.

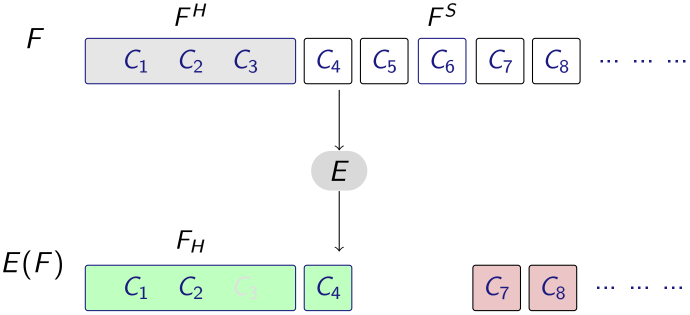
Prop: If E is MUS-preserving, then M is a group-MUS of F.

Note: in particular, this is true if E is monotone (e.g. BCE).

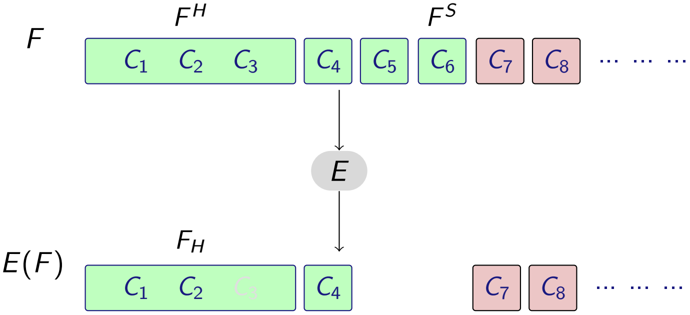
MaxSAT: *monotone* clause elimination



MaxSAT: *monotone* clause elimination



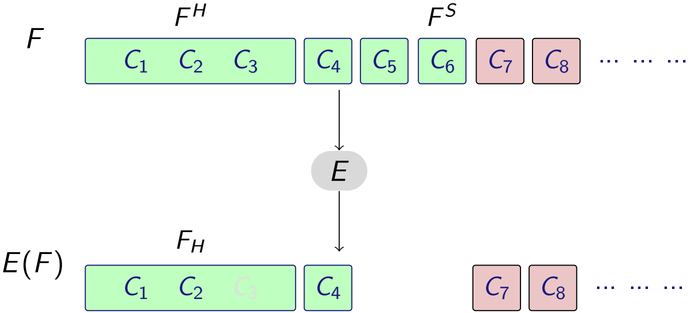
MaxSAT: *monotone* clause elimination



Prop: If E is MUS-preserving, and τ is a MaxSAT solution for $E(F)$, then $\alpha_E(\tau)$ is a MaxSAT solution for F.

Pf: E is MUS-preserving $\Rightarrow$ E is MCS-preserving (HS-duality), and so min-cost MCS is the same. Only MSS clauses will be eliminated. τ is a model of the MSS of $E(F)$, $\alpha_E(\tau)$ must be a model of corresponding MSS of F.

MaxSAT: *monotone* clause elimination



Prop: If E is MUS-preserving, and τ is a MaxSAT solution for $E(F)$, then $\alpha_E(\tau)$ is a MaxSAT solution for F.

Pf: E is MUS-preserving $\Rightarrow$ E is MCS-preserving (HS-duality), and so min-cost MCS is the same. Only MSS clauses will be eliminated. τ is a model of the MSS of $E(F)$, $\alpha_E(\tau)$ must be a model of corresponding MSS of F.

Note: in particular, this is true if E is monotone (e.g. BCE).

Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework

Plain MUS: SSR

$$F \quad \begin{array}{ccccc} C_1 & C_2 & & & \\ \boxed{(\neg x \vee p)} & \boxed{(x \vee p \vee q)} & \boxed{(\neg p)} & \boxed{(x \vee \neg q)} & \boxed{(\neg x)} \end{array}$$

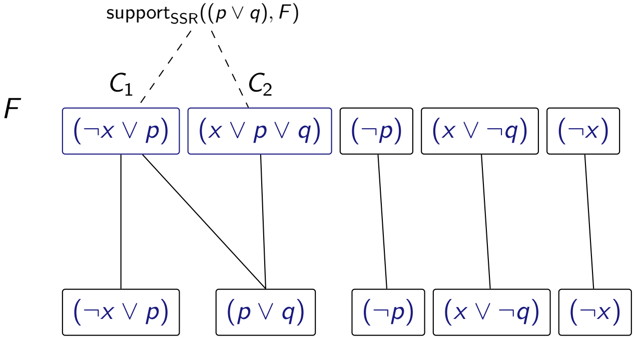
Plain MUS: SSR

$$F \quad \overset{C_1}{\boxed{(\neg x \vee p)}}\ \overset{C_2}{\boxed{(x \vee p \vee q)}}\ \boxed{(\neg p)}\ \boxed{(x \vee \neg q)}\ \boxed{(\neg x)}$$

$$\boxed{(\neg x \vee p)}\quad \boxed{(p \vee q)}\quad \boxed{(\neg p)}\ \boxed{(x \vee \neg q)}\ \boxed{(\neg x)}$$

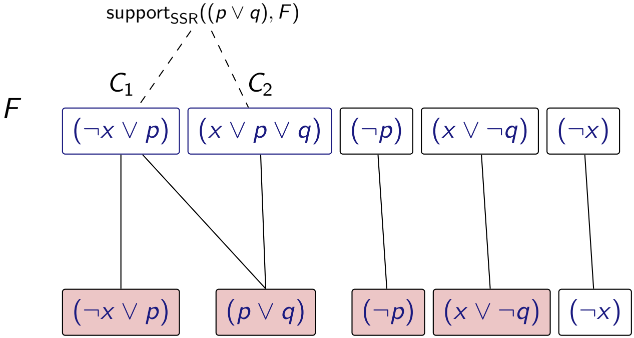
$$F' = \mathrm{SSR}(F, C_1, C_2, x)$$

Plain MUS: SSR



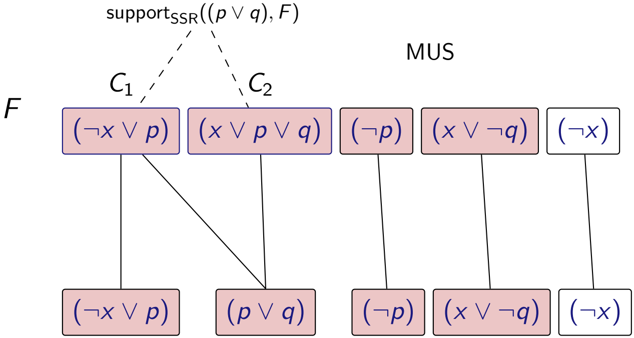
$F' = \text{SSR}(F, C_1, C_2, x)$

Plain MUS: SSR



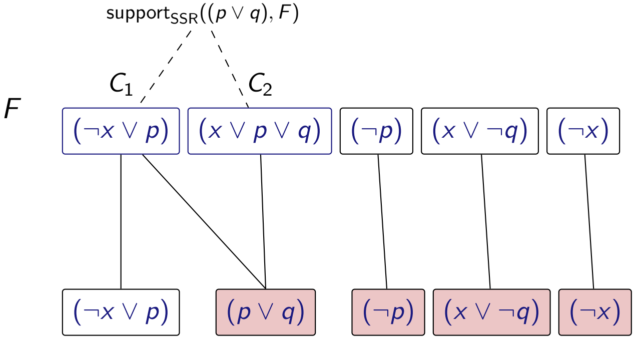
$F' = \text{SSR}(F, C_1, C_2, x)$

Plain MUS: SSR



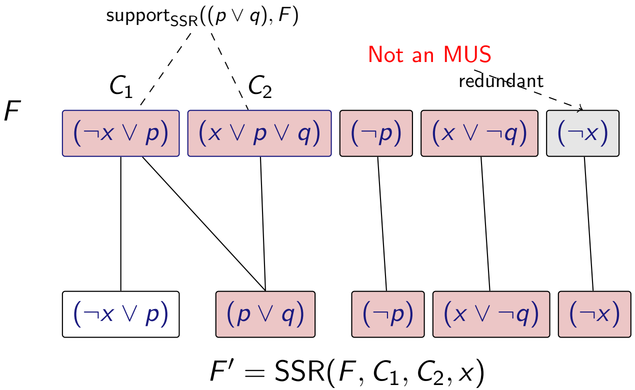
$F' = \text{SSR}(F, C_1, C_2, x)$

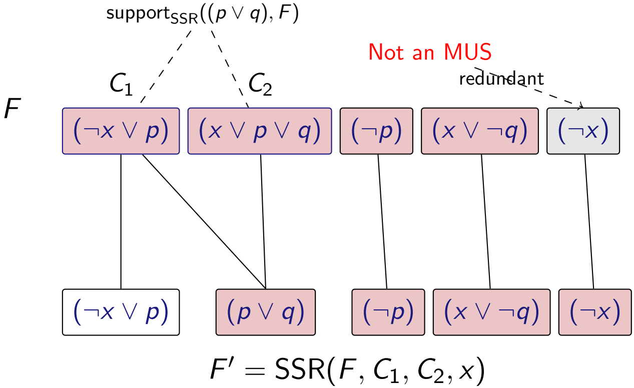
Plain MUS: SSR



$F' = \mathrm{SSR}(F, C_1, C_2, x)$

Plain MUS: SSR

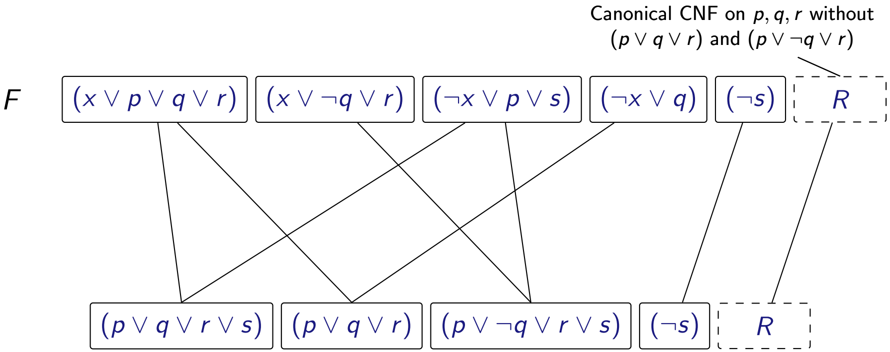




For an MUS M' of $F' = \text{SSR}(F, C, D, l)$,

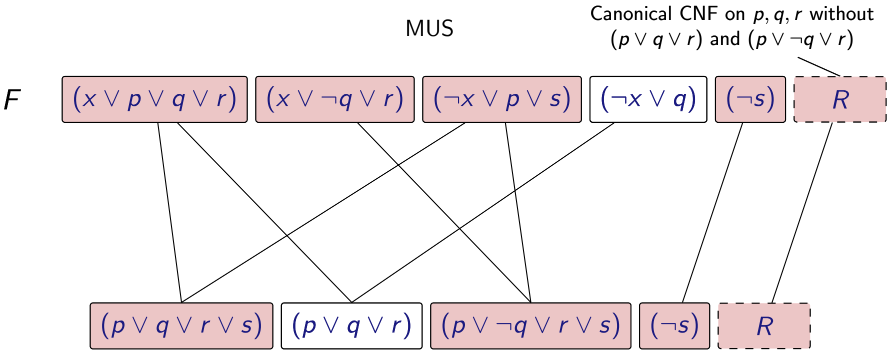
$M = \bigcup_{C \in M'} \text{support}_{\text{SSR}}(C, F)$ might be not an MUS of F.

Plain MUS: VE



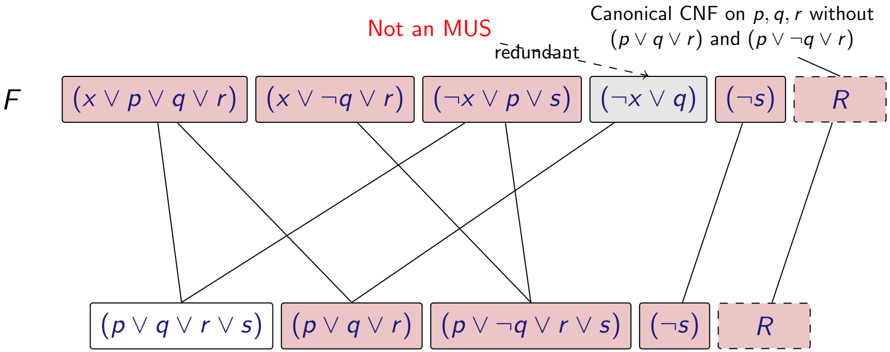
$F' = \mathrm{VE}(F, x)$

Plain MUS: VE



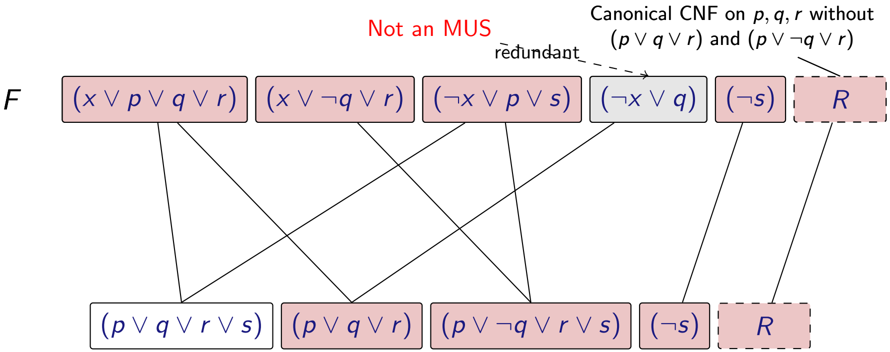
$F' = \mathrm{VE}(F, x)$

Plain MUS: VE



$F' = \mathrm{VE}(F, x)$

Plain MUS: VE



$F' = \mathrm{VE}(F, x)$

For an MUS M' of $F' = \mathrm{VE}(F, x)$,

$M = \bigcup_{C \in M'} \mathrm{support}_{\mathrm{VE}}(C, F)$ might be not an MUS of F.

Doesn't work even if M is greedily minimized to avoid duplicate resolvents.

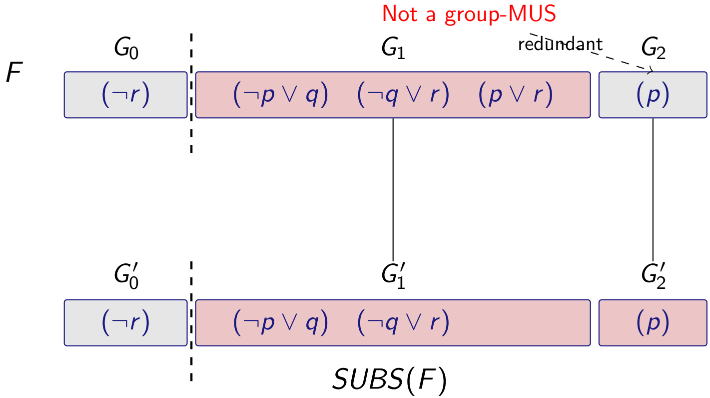
Group-MUS: subsumption

	G_0	G_1	G_2
F	$(\neg r)$	$(\neg p \vee q)$ $(\neg q \vee r)$ $(p \vee r)$	(p)

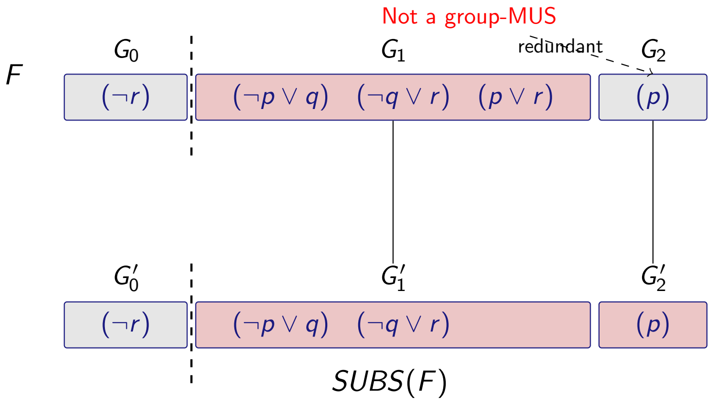
G_0'	G_1'	G_2'
$(\neg r)$	$(\neg p \vee q)$ $(\neg q \vee r)$	(p)

$SUBS(F)$

Group-MUS: subsumption



Group-MUS: subsumption



For a group-MUS M' of $F' = E(F)$,

$M = \{G_i \in F \mid G'_i \in F'\}$ might be not a group-MUS of F.

Note: can do *certain* things with subsumption: eliminate *within* a group, or from group-0 to other groups.

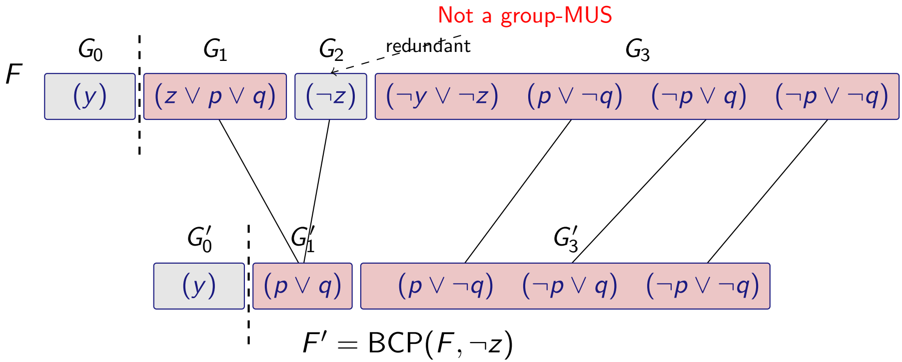
Group MUS: BCP

	G_0	G_1	G_2	G_3
F	(y)	$(z \vee p \vee q)$	$(\neg z)$	$(\neg y \vee \neg z)$ $(p \vee \neg q)$ $(\neg p \vee q)$ $(\neg p \vee \neg q)$

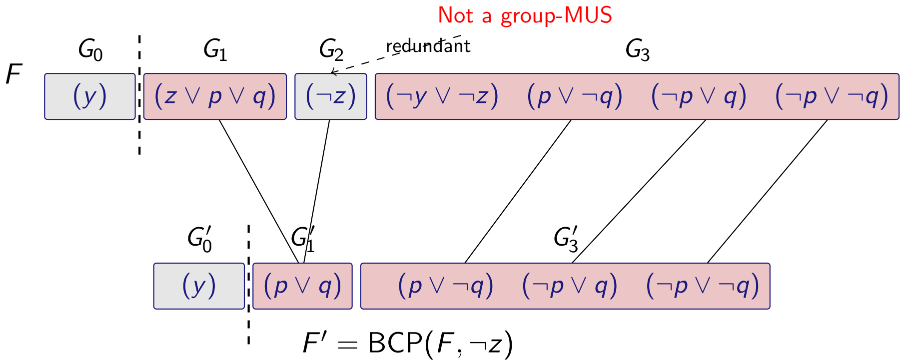
G'_0	G'_1	G'_3
(y)	$(p \vee q)$	$(p \vee \neg q)$ $(\neg p \vee q)$ $(\neg p \vee \neg q)$

$$F' = \mathrm{BCP}(F, \neg z)$$

Group MUS: BCP



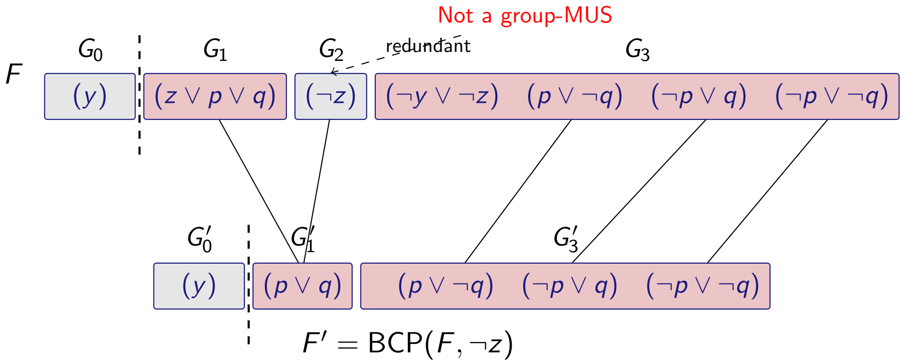
Group MUS: BCP



BCP in the group-MUS setting is problematic.

Note: again, can do *certain* things with BCP: propagate *within* a group, or from group-0 to other groups.

Group MUS: BCP



BCP in the group-MUS setting is problematic.

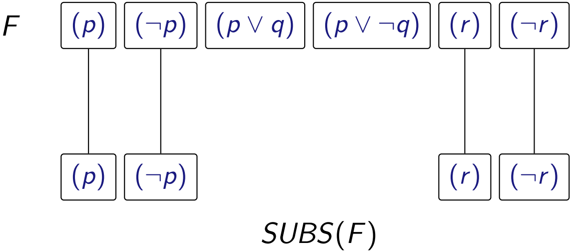
Note: again, can do *certain* things with BCP: propagate *within* a group, or from group-0 to other groups.

Note: VE and SSR didn't work for plain MUSes, and so won't work for group-MUSes either.

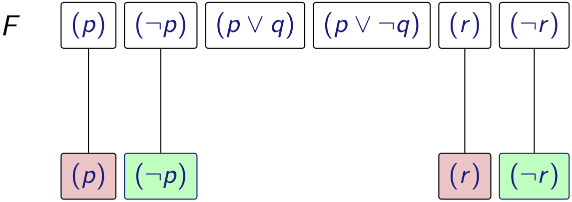
MaxSAT: subsumption

$F \quad \boxed{(p)}\ \boxed{(\neg p)}\ \boxed{(p \vee q)}\ \boxed{(p \vee \neg q)}\ \boxed{(r)}\ \boxed{(\neg r)}$

MaxSAT: subsumption



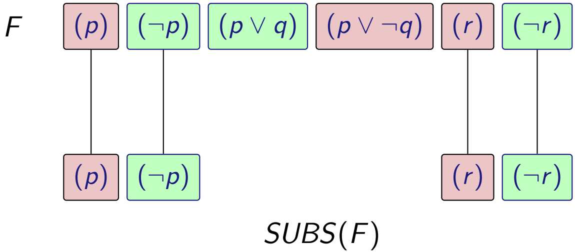
MaxSAT: subsumption



$SUBS(F)$

$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $SUBS(F)$, $cost(\tau) = 2$.

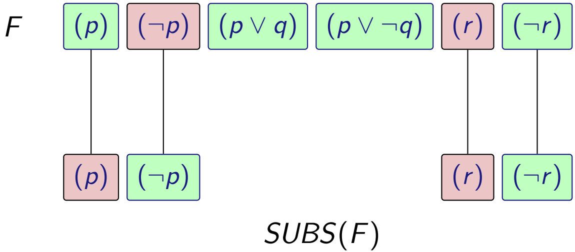
MaxSAT: subsumption



$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $SUBS(F)$, $cost(\tau) = 2$.

$\tau' = \{\neg p, \neg r, q\}$ is not a MaxSAT solution for F, $cost(\tau') = 3$, but $cost(\{p, \neg r, q\}) = 2$.

MaxSAT: subsumption



$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $SUBS(F)$, $cost(\tau) = 2$.

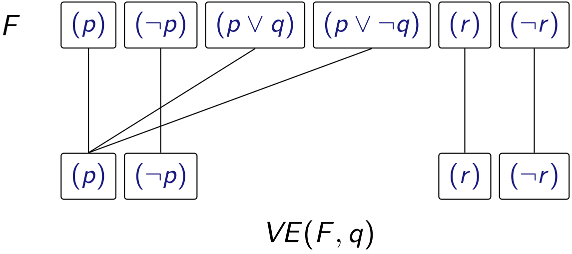
$\tau' = \{\neg p, \neg r, q\}$ is not a MaxSAT solution for F, $cost(\tau') = 3$, but $cost(\{p, \neg r, q\}) = 2$.

Reason: subsumption may remove some of the MUSes of F.

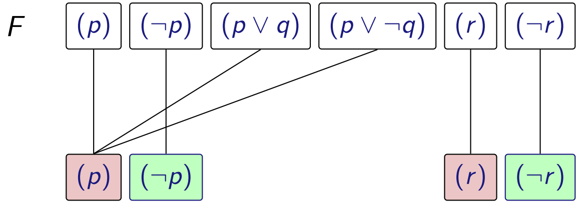
MaxSAT: VE

$F \quad \boxed{(p)}\ \boxed{(\neg p)}\ \boxed{(p \vee q)}\ \boxed{(p \vee \neg q)}\ \boxed{(r)}\ \boxed{(\neg r)}$

MaxSAT: VE



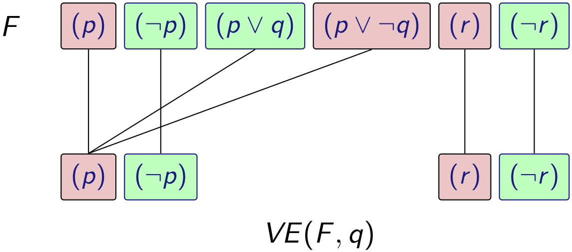
MaxSAT: VE



$VE(F, q)$

$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $VE(F, q)$, $cost(\tau) = 2$.

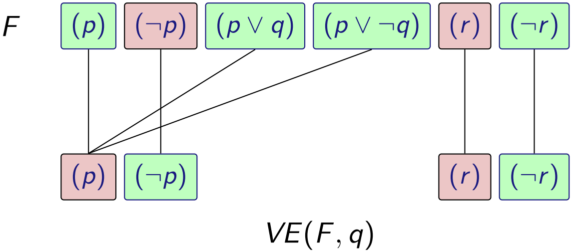
MaxSAT: VE



$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $VE(F, q)$, $cost(\tau) = 2$.

$\tau' = \{\neg p, \neg r, q\}$ is not a MaxSAT solution for F, $cost(\tau') = 3$, but $cost(\{p, \neg r, q\}) = 2$.

MaxSAT: VE



$\tau = \{\neg p, \neg r\}$ is a MaxSAT solution for $VE(F, q)$, $cost(\tau) = 2$.

$\tau' = \{\neg p, \neg r, q\}$ is not a MaxSAT solution for F, $cost(\tau') = 3$, but $cost(\{p, \neg r, q\}) = 2$.

Reason: VE *changes* the MUSes of F. Additional problem: what to do with weights ?

MaxSAT: resolution-based preprocessing

None of the resolution-based techniques are *sound* for MaxSAT, because the resolution rule itself is not sound.

$$(x \vee C_1) \otimes_x (\neg x \vee C_2) = (C_1 \vee C_2)$$

Consider an assignment τ s.t. $\tau(\neg x \vee C_2) = 0$ and $\tau(C_1) = 1$.

MaxSAT: resolution-based preprocessing

None of the resolution-based techniques are *sound* for MaxSAT, because the resolution rule itself is not sound.

$$(x \vee C_1) \otimes_x (\neg x \vee C_2) = (C_1 \vee C_2)$$

Consider an assignment τ s.t. $\tau(\neg x \vee C_2) = 0$ and $\tau(C_1) = 1$.

MaxSAT Resolution rule [Bonet et al, 07]

$$\begin{array}{l}
x \vee a_1 \vee \cdots \vee a_s \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \\
\hline\hline
a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_t \\
x \vee a_1 \vee \cdots \vee a_s \vee \bar{b}_1 \\
x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \bar{b}_2 \\
\cdots \\
x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_{t-1} \vee \bar{b}_t \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \vee \bar{a}_1 \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \bar{a}_2 \\
\cdots \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \cdots \vee a_{s-1} \vee \bar{a}_s
\end{array}$$

MaxSAT: resolution-based preprocessing

None of the resolution-based techniques are *sound* for MaxSAT, because the resolution rule itself is not sound.

$$(x \vee C_1) \otimes_x (\neg x \vee C_2) = (C_1 \vee C_2)$$

Consider an assignment τ s.t. $\tau(\neg x \vee C_2) = 0$ and $\tau(C_1) = 1$.

MaxSAT Resolution rule [Bonet et al, 07]

$$\begin{array}{l}
x \vee a_1 \vee \cdots \vee a_s \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \\
\hline\hline
a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_t \\
x \vee a_1 \vee \cdots \vee a_s \vee \bar{b}_1 \\
x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \bar{b}_2 \\
\cdots \\
x \vee a_1 \vee \cdots \vee a_s \vee b_1 \vee \cdots \vee b_{t-1} \vee \bar{b}_t \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \vee \bar{a}_1 \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \bar{a}_2 \\
\cdots \\
\bar{x} \vee b_1 \vee \cdots \vee b_t \vee a_1 \vee \cdots \vee a_{s-1} \vee \bar{a}_s
\end{array}$$

Although sound and complete for MaxSAT solving, is not suitable for preprocessing.

Summary of Direct Reconstruction

Techniques that work

- plain-MUS: any clause elimination, BCP.
- group-MUS: MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: MUS-preserving (sp. monotone) clause elimination.

Techniques that break

- plain-MUS: VE, SSR
- group-MUS: SUB, BCP, VE, SSR
- MaxSAT: SUB, BCP, VE, SSR

Summary of Direct Reconstruction

Techniques that work

- plain-MUS: any clause elimination, BCP.
- group-MUS: MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: MUS-preserving (sp. monotone) clause elimination.

Techniques that break

- plain-MUS: VE, SSR
- group-MUS: SUB, BCP, VE, SSR
- MaxSAT: SUB, BCP, VE, SSR

Important: not known that sound direct methods are *impossible*.

Summary of Direct Reconstruction

Techniques that work

- plain-MUS: any clause elimination, BCP.
- group-MUS: MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: MUS-preserving (sp. monotone) clause elimination.

Techniques that break

- plain-MUS: VE, SSR
- group-MUS: SUB, BCP, VE, SSR
- MaxSAT: SUB, BCP, VE, SSR

Important: not known that sound direct methods are *impossible*.

Note: Some special cases do work:

- group-MUS: within groups, or from group-0 to other groups.
- MaxSAT: within F^H, or from F^H to soft clauses.

Need a generic framework for guaranteed correctness-preserving application of preprocessing techniques.

Outline

Introduction: MUSes, MaxSAT, SAT preprocessing

Direct reconstruction: techniques that work

Direct reconstruction: techniques that break

Sound preprocessing in Labelled CNF framework

Labelled CNF (LCNF) formulas

Original motivation: generalize group-MUS to *intersecting* groups

Note: those arise naturally in circuit-MUS and variable-MUS settings.

Note: generalizes other related problems, such as MCS computation and MaxSAT.

Labelled CNF (LCNF) formulas

Original motivation: generalize group-MUS to *intersecting* groups

Note: those arise naturally in circuit-MUS and variable-MUS settings.

Note: generalizes other related problems, such as MCS computation and MaxSAT.

The components of the framework

- *Lbls*: a countable set of *labels*.
- *Labelled clause* C^L is a tuple $\langle C, L \rangle$, where C is a clause, and $L \subset Lbls$ is a finite set of labels.
- *LCNF formula* Φ is a set of labelled clauses.
  - $Cls(\Phi)$ - the "normal" clauses of Φ, i.e. $\bigcup_{C^L \in \Phi} C$.
  - $Lbls(\Phi)$ - the labels of Φ, i.e. $\bigcup_{C^L \in \Phi} L$.

Labelled CNF (LCNF) formulas

Original motivation: generalize group-MUS to *intersecting* groups

Note: those arise naturally in circuit-MUS and variable-MUS settings.

Note: generalizes other related problems, such as MCS computation and MaxSAT.

The components of the framework

- *Lbls*: a countable set of *labels*.
- *Labelled clause* C^L is a tuple $\langle C, L \rangle$, where C is a clause, and $L \subset Lbls$ is a finite set of labels.
- *LCNF formula* Φ is a set of labelled clauses.
  - $Cls(\Phi)$ - the "normal" clauses of Φ, i.e. $\bigcup_{C^L \in \Phi} C$.
  - $Lbls(\Phi)$ - the labels of Φ, i.e. $\bigcup_{C^L \in \Phi} L$.

Example ($Lbls = \mathbb{N}$):

Φ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(z \vee p \vee q)^{\{1\}}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$ $\boxed{(\neg p \vee q)^{\{2,3\}}}$ $\boxed{(\neg p \vee \neg q)^{\{3\}}}$

LCNF formulas: induced subformulas

Def: For an LCNF Φ, let $M \subseteq Lbls(\Phi)$. Then, $\Phi|_M = \{C^L \in \Phi \mid L \subseteq M\}$ is a subformula of Φ *induced* by M.

Alternatively: any clause that has a label *outside* of M is removed from Φ.

LCNF formulas: induced subformulas

Def: For an LCNF Φ, let $M \subseteq Lbls(\Phi)$. Then, $\Phi|_M = \{C^L \in \Phi \mid L \subseteq M\}$ is a subformula of Φ *induced* by M.

Alternatively: any clause that has a label *outside* of M is removed from Φ.

Φ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(z \vee p \vee q)^{\{1\}}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$ $\boxed{(\neg p \vee q)^{\{2,3\}}}$ $\boxed{(\neg p \vee \neg q)^{\{3\}}}$

$\Phi|_{\{1,2\}}$ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(z \vee p \vee q)^{\{1\}}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$

$\Phi|_{\{2\}}$ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$

LCNF formulas: induced subformulas

Def: For an LCNF Φ, let $M \subseteq Lbls(\Phi)$. Then, $\Phi|_M = \{C^L \in \Phi \mid L \subseteq M\}$ is a subformula of Φ *induced* by M.

Alternatively: any clause that has a label *outside* of M is removed from Φ.

Φ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(z \vee p \vee q)^{\{1\}}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$ $\boxed{(\neg p \vee q)^{\{2,3\}}}$ $\boxed{(\neg p \vee \neg q)^{\{3\}}}$

$\Phi|_{\{1,2\}}$ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(z \vee p \vee q)^{\{1\}}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$

$\Phi|_{\{2\}}$ $\boxed{(x)^{\emptyset}}$ $\boxed{(y)^{\emptyset}}$ $\boxed{(p \vee \neg q)^{\{2\}}}$

Note: Clauses with the label-set $\emptyset$ cannot be removed $\Rightarrow$ a convenient way to represents G_0 clauses in group-MUS, or F^H in MaxSAT.

LCNF formulas: minimal unsatisfiability and MUSes

Satisfiability: LCNF Φ is SAT iff $Cls(\Phi)$ is SAT.

Def: $M \subseteq Lbls(\Phi)$ is an *minimal unsatisfiable subformula (MUS)* of Φ if $\Phi|_M \in$ UNSAT, and $\forall l \in M$, $\Phi|_{M \setminus \{l\}} \in$ SAT

I.e. the removal of *any* label from $\Phi|_M$ makes it SAT.

LCNF formulas: minimal unsatisfiability and MUSes

Satisfiability: LCNF Φ is SAT iff $Cls(\Phi)$ is SAT.

Def: $M \subseteq Lbls(\Phi)$ is an *minimal unsatisfiable subformula (MUS)* of Φ if $\Phi|_M \in \text{UNSAT}$, and $\forall l \in M,\ \Phi|_{M \setminus \{l\}} \in \text{SAT}$

I.e. the removal of *any* label from $\Phi|_M$ makes it SAT.

Natural mapping between CNF/group-CNF and LCNF:

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

MUS is $\{C_1, C_2, C_4, C_6\}$

Φ $\quad (p)^{\{1\}} \quad (q)^{\{2\}} \quad (\neg p \vee \neg q)^{\{3\}} \quad (\neg p \vee r)^{\{4\}} \quad (p \vee q)^{\{5\}} \quad (\neg q \vee \neg r)^{\{6\}}$

LMUS is $\{1, 2, 4, 6\}$

LCNF formulas: minimal unsatisfiability and MUSes

Satisfiability: LCNF Φ is SAT iff $Cls(\Phi)$ is SAT.

Def: $M \subseteq Lbls(\Phi)$ is an *minimal unsatisfiable subformula (MUS)* of Φ if $\Phi|_M \in$ UNSAT, and $\forall l \in M$, $\Phi|_{M\setminus\{l\}} \in$ SAT

I.e. the removal of *any* label from $\Phi|_M$ makes it SAT.

Natural mapping between CNF/group-CNF and LCNF:

G_0	G_1	G_2
$(p) \quad (q)$	$(\neg p \vee \neg q) \quad (\neg p \vee r)$	$(p \vee q) \quad (\neg q \vee \neg r)$

group-MUS is $\{G_1\}$

$$\Phi \quad \boxed{(p)^{\emptyset}} \ \boxed{(q)^{\emptyset}} \ \boxed{(\neg p \vee \neg q)^{\{1\}}} \ \boxed{(\neg p \vee r)^{\{1\}}} \ \boxed{(p \vee q)^{\{2\}}} \ \boxed{(\neg q \vee \neg r)^{\{2\}}}$$

LMUS is $\{1\}$

LCNF formulas: MCSes and MaxSAT

Def: $S \subseteq Lbls(\Phi)$ is a *maximal satisfiable subformula (MSS)* of Φ if $\Phi|_S \in$ SAT, and $\forall l \in Lbls(\Phi) \setminus S, \quad \Phi|_{S \cup \{l\}} \in$ UNSAT.

Def: $R \subseteq Lbls(\Phi)$ is a *minimal correction subset (MCS)* of Φ if $\Phi|_{Lbls(\Phi) \setminus R} \in$ SAT, and $\forall l \in R, \Phi|_{(Lbls(\Phi) \setminus R) \cup \{l\}} \in$ UNSAT.

Note: hitting sets duality between MUSes and MCSes holds for LCNFs.

LCNF formulas: MCSes and MaxSAT

Def: $S \subseteq Lbls(\Phi)$ is a *maximal satisfiable subformula (MSS)* of Φ if $\Phi|_S \in \text{SAT}$, and $\forall l \in Lbls(\Phi) \setminus S, \quad \Phi|_{S \cup \{l\}} \in \text{UNSAT}$.

Def: $R \subseteq Lbls(\Phi)$ is a *minimal correction subset (MCS)* of Φ if $\Phi|_{Lbls(\Phi) \setminus R} \in \text{SAT}$, and $\forall l \in R, \Phi|_{(Lbls(\Phi) \setminus R) \cup \{l\}} \in \text{UNSAT}$.

Note: hitting sets duality between MUSes and MCSes holds for LCNFs.

MaxSAT for LCNFs:

- each *label* l has a *weight* $w(l) \in \mathbb{N}^+$;
- the *cost* of a set of labels L is $\sum_{l \in L} w(l)$.
- weighted LCNF: just an LCNF + the weights for labels.
- a *MaxSAT solution* for Φ is a model of an MSS of the *maximum cost*. Alternatively: falsifies only the clauses of an MCS of the *minimum cost*.

LCNF formulas: MCSes and MaxSAT

Def: $S \subseteq Lbls(\Phi)$ is a *maximal satisfiable subformula (MSS)* of Φ if $\Phi|_S \in \text{SAT}$, and $\forall l \in Lbls(\Phi) \setminus S, \quad \Phi|_{S \cup \{l\}} \in \text{UNSAT}$.

Def: $R \subseteq Lbls(\Phi)$ is a *minimal correction subset (MCS)* of Φ if $\Phi|_{Lbls(\Phi) \setminus R} \in \text{SAT}$, and $\forall l \in R, \Phi|_{(Lbls(\Phi) \setminus R) \cup \{l\}} \in \text{UNSAT}$.

Note: hitting sets duality between MUSes and MCSes holds for LCNFs.

MaxSAT for LCNFs:

- each *label* l has a *weight* $w(l) \in \mathbb{N}^+$;
- the *cost* of a set of labels L is $\sum_{l \in L} w(l)$.
- weighted LCNF: just an LCNF + the weights for labels.
- a *MaxSAT solution* for Φ is a model of an MSS of the *maximum cost*. Alternatively: falsifies only the clauses of an MCS of the *minimum cost*.

Note: a natural mapping between MaxSAT instances and weighted LCNFs.

LCNF formulas: preprocessing

Main point: For LCNF formulas MUSes and MCSes are *preserved* under a (suitably defined) preprocessing techniques.

LCNF formulas: preprocessing

Main point: For LCNF formulas MUSes and MCSes are *preserved* under a (suitably defined) preprocessing techniques.

Subsumption elimination

Theorem: Let $C_1^{L_1}$ and $C_2^{L_2}$ in Φ be such that $C_1 \subseteq C_2$ *and* $L_1 \subseteq L_2$. Then, any MUS of $\Phi \setminus \{C_2^{L_2}\}$ is an MUS of Φ, and vice versa.

Note: by hitting sets duality, $\text{MCS}(\Phi \setminus \{C_2^{L_2}\}) = \text{MCS}(\Phi)$. Any MaxSAT solution τ for $\Phi \setminus \{C_2^{L_2}\}$ is a MaxSAT solution for Φ.

LCNF formulas: preprocessing

Main point: For LCNF formulas MUSes and MCSes are *preserved* under a (suitably defined) preprocessing techniques.

Subsumption elimination

Theorem: Let $C_1^{L_1}$ and $C_2^{L_2}$ in Φ be such that $C_1 \subseteq C_2$ *and* $L_1 \subseteq L_2$. Then, any MUS of $\Phi \setminus \{C_2^{L_2}\}$ is an MUS of Φ, and vice versa.

Note: by hitting sets duality, $\mathrm{MCS}(\Phi \setminus \{C_2^{L_2}\}) = \mathrm{MCS}(\Phi)$. Any MaxSAT solution τ for $\Phi \setminus \{C_2^{L_2}\}$ is a MaxSAT solution for Φ.

- I.e. subsumption is OK, as long as the labels also have a subset relationship.
- Effectively, blocks subsumption across groups in group-MUS, and across soft clauses in MaxSAT (good).
- Blocks subsumption in plain MUS instances: a weaker condition that allows for it seems to be possible.

LCNF formulas: preprocessing

Def: Resolution: $(x \vee C_1)^{L_1} \otimes_x (\neg x \vee C_2)^{L_2} = (C_1 \vee C_2)^{L_1 \cup L_2}$.

Def: Variable elimination: $\mathrm{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})$.

LCNF formulas: preprocessing

Def: Resolution: $(x \vee C_1)^{L_1} \otimes_x (\neg x \vee C_2)^{L_2} = (C_1 \vee C_2)^{L_1 \cup L_2}$.

Def: Variable elimination: $\mathrm{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})$.

Theorem: Any LMUS of $\mathrm{LVE}(\Phi, x)$ is an LMUS of Φ, and vice versa.

LCNF formulas: preprocessing

Def: Resolution: $(x \vee C_1)^{L_1} \otimes_x (\neg x \vee C_2)^{L_2} = (C_1 \vee C_2)^{L_1 \cup L_2}$.

Def: Variable elimination: $\mathrm{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})$.

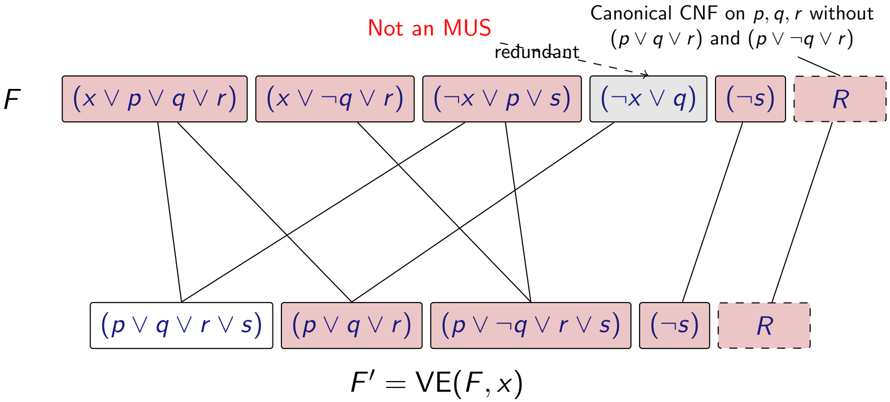
Theorem: Any LMUS of $\mathrm{LVE}(\Phi, x)$ is an LMUS of Φ, and vice versa.

Note: by hitting sets duality, $\mathrm{MCS}(\mathrm{VE}(\Phi, x)) = \mathrm{MCS}(\Phi)$.

For any MaxSAT solution τ for $\mathrm{VE}(\Phi, x)$ there is a MaxSAT solution for Φ with exactly the same cost (exactly the same MCS).

To get a solution for Φ (the model of the MSS) do the standard reconstruction after VE.

Remember the bad example for VE ?



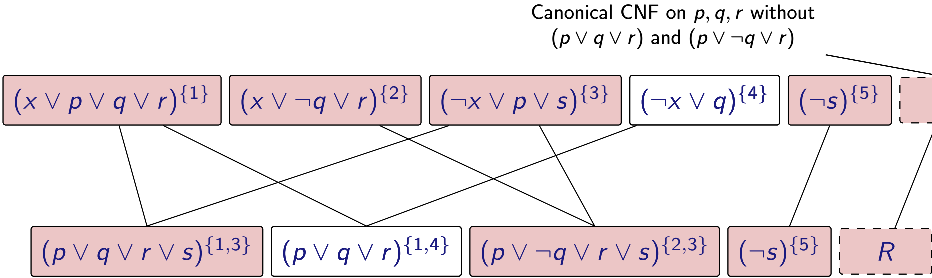
For an MUS M' of $F' = \mathrm{VE}(F, x)$,
$M = \bigcup_{C \in M'} \mathrm{support}_{\mathrm{VE}}(C, F)$ might be not an MUS of F.

LCNF formulas: preprocessing

Def: Resolution: $(x \vee C_1)^{L_1} \otimes_x (\neg x \vee C_2)^{L_2} = (C_1 \vee C_2)^{L_1 \cup L_2}$.
Def: Variable elimination: $\mathrm{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})$.
Theorem: Any LMUS of $\mathrm{LVE}(\Phi, x)$ is an LMUS of Φ, and vice versa.

LCNF formulas: preprocessing

Def: Resolution: $(x \vee C_1)^{L_1} \otimes_x (\neg x \vee C_2)^{L_2} = (C_1 \vee C_2)^{L_1 \cup L_2}$.
Def: Variable elimination: $\mathrm{VE}(\Phi, x) = \Phi \cup (\Phi_x \otimes_x \Phi_{\neg x}) \setminus (\Phi_x \cup \Phi_{\neg x})$.
Theorem: Any LMUS of $\mathrm{LVE}(\Phi, x)$ is an LMUS of Φ, and vice versa.



$\Phi' = \mathrm{LVE}(\Phi, x)$

Label 4 *is* redundant ! The LMUS of Φ' is $\{1, 2, 3, 5, \dots\}$ – same as for Φ.

LCNF formulas: preprocessing

BCP and SSR can also be defined for LCNFs so that the correctness is preserved.

Bottom line: labels enable correctness-preserving preprocessing for MUS and group-MUS extraction, and for computing MaxSAT solutions.

LCNF formulas: preprocessing

BCP and SSR can also be defined for LCNFs so that the correctness is preserved.

Bottom line: labels enable correctness-preserving preprocessing for MUS and group-MUS extraction, and for computing MaxSAT solutions.

Preprocessing flow

Given F, a (group-)MUS instance or an instance of MaxSAT:

1. Run any MUS-preserving clause elimination procedure (like BCE).
2. For plain-MUSes may also run SUB and BCP (while keeping trace).
3. Convert the resulting formula F_1 to LCNF Φ_1.
4. Run VE, SSR on Φ_1 to get Φ_2.
5. Compute an MUS M of Φ_2 or a MaxSAT solution τ for Φ_2.
6. The labels in M represent a (group-)MUS of F_1. The cost of a MaxSAT solution of F_1 is the same as the cost of τ; to get a solution do the standard reconstruction after VE.

LCNF formulas: preprocessing

BCP and SSR can also be defined for LCNFs so that the correctness is preserved.

Bottom line: labels enable correctness-preserving preprocessing for MUS and group-MUS extraction, and for computing MaxSAT solutions.

Preprocessing flow

Given F, a (group-)MUS instance or an instance of MaxSAT:

1. Run any MUS-preserving clause elimination procedure (like BCE).
2. For plain-MUSes may also run SUB and BCP (while keeping trace).
3. Convert the resulting formula F_1 to LCNF Φ_1.
4. Run VE, SSR on Φ_1 to get Φ_2.
5. Compute an MUS M of Φ_2 or a MaxSAT solution τ for Φ_2.
6. The labels in M represent a (group-)MUS of F_1. The cost of a MaxSAT solution of F_1 is the same as the cost of τ; to get a solution do the standard reconstruction after VE.

LCNF formulas: computing MUSes

Given an LCNF Φ, for each label $l_i \in Lbls(\Phi)$, $1 \le i \le n$, introduce a fresh variable p_i.

Then, convert each labelled clause $C^{\{l_1,\dots,l_k\}}$ in Φ into a clause $(p_1 \vee \cdots \vee p_k \vee C)$. Let F_Φ denote the resulting CNF.

LCNF formulas: computing MUSes

Given an LCNF Φ, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable p_i.

Then, convert each labelled clause $C^{\{l_1,\ldots,l_k\}}$ in Φ into a clause $(p_1 \vee \cdots \vee p_k \vee C)$. Let F_Φ denote the resulting CNF.

Reduction to group-MUS problem

Let $G_0 = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, let $G_i = \{(\neg p_i)\}$.

Compute a group-MUS of $\{G_0, G_1, \ldots, G_n\}$: if G_i is in, then l_i is in.

LCNF formulas: computing MUSes

Given an LCNF Φ, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable p_i.

Then, convert each labelled clause $C^{\{l_1,\ldots,l_k\}}$ in Φ into a clause $(p_1 \vee \cdots \vee p_k \vee C)$. Let F_Φ denote the resulting CNF.

Reduction to group-MUS problem

Let $G_0 = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, let $G_i = \{(\neg p_i)\}$.

Compute a group-MUS of $\{G_0, G_1, \ldots, G_n\}$: if G_i is in, then l_i is in.

Direct computation

Load F_Φ into an incremental SAT solver, and use p_i's as assumptions.

Note: this is how most of the MUS extractors compute MUSes and group-MUSes anyway.

LCNF formulas: computing MaxSAT solutions

Given a weighted LCNF Φ, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable p_i.

Then, convert each labelled clause $C^{\{l_1,\ldots,l_k\}}$ in Φ into a clause $(p_1 \vee \cdots \vee p_k \vee C)$. Let F_Φ denote the resulting CNF.

LCNF formulas: computing MaxSAT solutions

Given a weighted LCNF Φ, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable p_i.

Then, convert each labelled clause $C^{\{l_1,\dots,l_k\}}$ in Φ into a clause $(p_1 \vee \dots \vee p_k \vee C)$. Let F_Φ denote the resulting CNF.

Reduction to weighted partial MaxSAT problem

Let $F^H = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, create a soft clause $C_i = (\neg p_i)$ with weight $w(l_i)$.

Compute a MaxSAT solution τ of $F^H \cup C_1 \cup \dots C_n$. τ is a MaxSAT solution of Φ.

LCNF formulas: computing MaxSAT solutions

Given a weighted LCNF Φ, for each label $l_i \in Lbls(\Phi)$, $1 \leq i \leq n$, introduce a fresh variable p_i.

Then, convert each labelled clause $C^{\{l_1,\dots,l_k\}}$ in Φ into a clause $(p_1 \vee \cdots \vee p_k \vee C)$. Let F_Φ denote the resulting CNF.

Reduction to weighted partial MaxSAT problem

Let $F^H = F_\Phi$, and for each $l_i \in Lbls(\Phi)$, create a soft clause $C_i = (\neg p_i)$ with weight $w(l_i)$.

Compute a MaxSAT solution τ of $F^H \cup C_1 \cup \dots C_n$. τ is a MaxSAT solution of Φ.

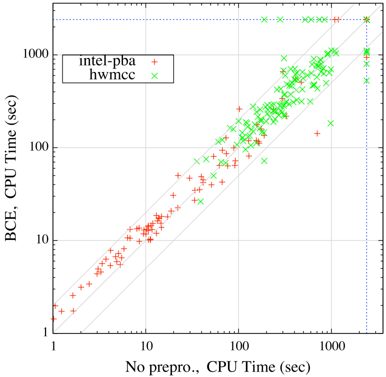
Direct computation

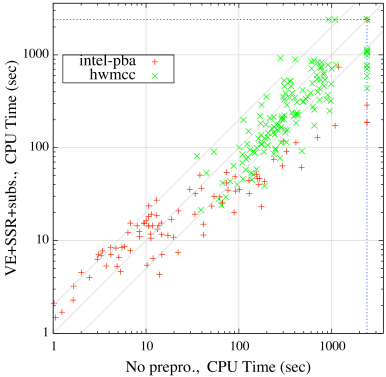
Load F_Φ into an incremental SAT solver, and use p_i's as assumptions, and run a modified MaxSAT algorithm (need to "relax" p_i's, not clauses).

Note: this also gives a nice incremental-SAT based MaxSAT algorithm.

Experimental evaluation: group-MUS

- 99 group-CNF instances from Intel (PBA), 148 instances derived from HWMCC.
- Time limit 1800 sec, memory limit 4 GB.





Experimental evaluation: MaxSAT

- 1000 instances from MaxSAT Competition 2013.
- Time limit 1800 sec, memory limit 4 GB.

	MaxSAT		Partial MaxSAT		Weighted Partial MaxSAT	
	#Sol.	A.CPU	#Sol.	A.CPU	#Sol.	A.CPU
Instances	55		627		397	
P_NI	37	172.76	254	152.04	233	131.32
P_NI+BCE	**41**	237.58	241	105.02	234	105.65
P_NI+BCE+RS	35	177.37	240	120.70	247	75.42
P_NI+RS	37	246.68	265	154.84	254	75.00
P	37	236.26	237	132.83	249	31.31
P+BCE	38	180.22	227	70.08	248	27.60
P+BCE+RS	37	209.48	221	85.36	259	32.19
P+RS	34	151.77	238	146.93	273	40.08
L	36	101.92	270	75.45	274	30.64
L+BCE	37	67.88	271	95.89	276	25.49
L+BCE+RS	38	96.02	271	73.90	275	15.90
L+RS	38	161.71	**276**	91.15	**289**	27.85
WMSU1	39	165.64	241	149.01	232	165.35

To sum up ...

Direct preprocessing

- plain-MUS: any clause elimination and BCP.
- group-MUS: only MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: only MUS-preserving (sp. monotone) clause elimination.

To sum up ...

Direct preprocessing

- plain-MUS: any clause elimination and BCP.
- group-MUS: only MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: only MUS-preserving (sp. monotone) clause elimination.

Labelled CNF (LCNF) framework

Allows for sound reconstruction of (group-)MUSes and MaxSAT solutions after preprocessing.

To sum up ...

Direct preprocessing

- plain-MUS: any clause elimination and BCP.
- group-MUS: only MUS-preserving (sp. monotone) clause elimination.
- MaxSAT: only MUS-preserving (sp. monotone) clause elimination.

Labelled CNF (LCNF) framework

Allows for sound reconstruction of (group-)MUSes and MaxSAT solutions after preprocessing.

Experimental evaluation

Good results on on industrially-relevant group-MUS computation problems

Improvements on weighted partial MaxSAT problems

Current and Future Work

- Analyze additional preprocessing techniques.
- Relax the conditions for labelled subsumption elimination.
- Proper implementation and additional MaxSAT algorithms.

Current and Future Work

- Analyze additional preprocessing techniques.
- Relax the conditions for labelled subsumption elimination.
- Proper implementation and additional MaxSAT algorithms.

Thank you for your attention !

Related Papers

1. A. Belov, M. Järvisalo and J. Marques-Silva. Formula Preprocessing in MUS Extraction. In Proc. TACAS 2013, 2013

2. A. Belov, A. Morgado, J. Marques-Silva. SAT-based Preprocessing for MaxSAT. In Proc. LPAR'19 (to appear), 2013.

3. A. Belov and J. Marques-Silva. Generalizing Redundancy in Propositional Logic: Foundations and Hitting Sets Duality. arXiv, 2012.