On Efficient Computation of Variable MUSes

Anton Belov\textsuperscript{1}, Alexander Ivrii\textsuperscript{2},
Arie Matsliah\textsuperscript{2}, Joao Marques-Silva\textsuperscript{1}

\textsuperscript{1}Complex and Adaptive Systems Laboratory
University College Dublin, Ireland

\textsuperscript{2}IBM Research – Haifa, Israel

SAT 2012
June 20, 2012
Trento, Italy
$\mathcal{F} = \{ C_1, \ldots, C_6 \} \in \text{UNSAT}$

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

$C_2 = (q)$  $C_4 = (p \lor q)$  $C_6 = (\neg q \lor \neg r)$
\( \mathcal{F} = \{ C_1, \ldots, C_6 \} \in \text{UNSAT} \)

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

Which part of \( \mathcal{F} \) is responsible for its inconsistency?
Introduction

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

\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}

Which part of \( \mathcal{F} \) is responsible for its inconsistency?
\[ \mathcal{F} = \{ C_1, \ldots, C_6 \} \in \text{UNSAT} \]

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

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

\( \{ C_1, C_2, C_3 \} \) is a subset-minimal set of clauses required to refute \( \mathcal{F} \).
$F = \{ C_1, \ldots, C_6 \} \in \text{UNSAT}$

\[
\begin{align*}
C_1 &= (p) \\
C_2 &= (q) \\
C_3 &= (\neg p \lor \neg q) \\
C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) \\
C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

Which part of $F$ is responsible for its inconsistency?

$\{ C_1, C_2, C_3 \}$ is a subset-minimal set of clauses required to refute $F$.

$F' \subseteq F$ is minimally unsatisfiable subformula (MUS) of $F$ if $F' \in \text{UNSAT}$, and $\forall C \in F', F' \setminus \{ C \} \in \text{SAT}$. 
\[ F = \{ C_1, \ldots, C_6 \} \in \text{UNSAT} \]

\[
\begin{align*}
C_1 &= (p) \\
C_2 &= (q) \\
C_3 &= (\neg p \lor \neg q) \\
C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) \\
C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

Which part of \( F \) is responsible for its inconsistency?

\( \{ C_1, C_2, C_3 \} \) is a subset-minimal set of clauses required to refute \( F \).

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

\( \{ C_1, C_2, C_5, C_6 \} \) is also an MUS of \( F \).
\( F = \mathcal{G}_1 \cup \mathcal{G}_2 \cup \mathcal{G}_3 \in \text{UNSAT} \) — partitioned into groups (sets) of clauses

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

\[ C_1 = (p) \]
\[ C_2 = (q) \]
\[ C_3 = (\neg p \lor \neg q) \]
\[ C_4 = (p \lor q) \]
\[ C_5 = (\neg p \lor r) \]
\[ C_6 = (\neg q \lor \neg r) \]
\[ F = G_1 \cup G_2 \cup G_3 \in \text{UNSAT} \quad \text{— partitioned into groups (sets) of clauses} \]

\[ G_1 = \{ C_1, C_2 \}, \quad G_2 = \{ C_3, C_4 \}, \quad G_3 = \{ C_5, C_6 \}. \]

\[
\begin{align*}
C_1 &= (p) \\
C_2 &= (q) \\
C_3 &= (\neg p \lor \neg q) \\
C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) \\
C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

What is a subset-minimal set of groups required to refute \( F \)?
\( F = G_1 \cup G_2 \cup G_3 \in \text{UNSAT} \) — partitioned into \textit{groups} (sets) of clauses

\[
G_1 = \{ C_1, C_2 \}, \quad G_2 = \{ C_3, C_4 \}, \quad G_3 = \{ C_5, C_6 \}.
\]

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) \\
C_2 &= (q) & C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

What is a subset-minimal set of \textit{groups} required to refute \( F \)?

\( \{ G_1, G_2 \} \) is a \textit{group-MUS} of (the partitioned) \( F \).
Introduction

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

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

\[ C_1 = (p) \quad C_3 = (\neg p \lor \neg q) \quad C_5 = (\neg p \lor r) \]
\[ C_2 = (q) \quad C_4 = (p \lor q) \quad C_6 = (\neg q \lor \neg r) \]

What is a subset-minimal set of groups required 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) \( \mathcal{F} \).
What about the variables of $\mathcal{F}$?

\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}

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

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

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

\{p, q\} is such a subset of variables.
What about the variables of $\mathcal{F}$?

\[
\begin{align*}
C_1 &= (p) \\
C_2 &= (q) \\
C_3 &= (\neg p \lor \neg q) \\
C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) \\
C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

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

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

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

What about the variables of $\mathcal{F}$?

\begin{align*}
C_1 &= (p) \\
C_2 &= (q) \\
C_3 &= (\neg p \lor \neg q) \\
C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) \\
C_6 &= (\neg q \lor \neg r)
\end{align*}

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

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

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

This paper is about algorithms for efficient computation of variable-MUSes.
What about the variables of $\mathcal{F}$?

$$
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) \\
C_2 &= (q) & C_4 &= (p \lor q) \\
C_5 &= (\neg p \lor r) & C_6 &= (\neg q \lor \neg r)
\end{align*}
$$

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

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

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

This paper is about algorithms for efficient computation of variable-MUSes.

Some applications: finding vertex-critical subgraphs; abstraction in abstraction refinement framework; satisfying assignments minimization.
Definition
Let $V \subseteq \text{Var}(\mathcal{F})$. The subformula of $\mathcal{F}$ induced by $V$ is the formula $\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}$.

I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in $V$. 
Definition
Let $V \subseteq Var(\mathcal{F})$. The subformula of $\mathcal{F}$ \textit{induced} by $V$ 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 $V$.

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]
Definition
Let \( V \subseteq \text{Var}(\mathcal{F}) \). The subformula of \( \mathcal{F} \) \textit{induced} by \( V \) is the formula
\[
\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}.
\]
I.e. \( \mathcal{F}|_V \) includes only those clauses of \( \mathcal{F} \) whose variables are in \( V \).

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

The subformula induced by \( \{p, q\} \) is \( \mathcal{F}|_{\{p,q\}} = \{C_1, C_2, C_3, C_4\} \).

Variable \( r \) is “removed” from \( \mathcal{F} \).
Definition
Let $V \subseteq \text{Var}(\mathcal{F})$. The subformula of $\mathcal{F}$ \textit{induced} by $V$ is the formula $\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}$. I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in $V$.

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

\begin{itemize}
  \item The subformula induced by \{p, q\} is $\mathcal{F}|_{\{p, q\}} = \{ C_1, C_2, C_3, C_4 \}$.
  \begin{itemize}
    \item variable $r$ is “removed” from $\mathcal{F}$.
  \end{itemize}
  \item The subformula induced by \{p\}, $\mathcal{F}|_p = \{ C_1 \}$.
  \begin{itemize}
    \item variables $q, r$ are “removed” from $\mathcal{F}$.
  \end{itemize}
\end{itemize}
Definition
Let $V \subseteq \text{Var}(\mathcal{F})$. The subformula of $\mathcal{F}$ induced by $V$ is the formula $\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}$.
I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in $V$.

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

**Definition**

Let $V \subseteq \text{Var}(\mathcal{F})$. The subformula of $\mathcal{F}$ *induced* by $V$ is the formula

$$\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}.$$

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

**Definition**

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

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]
Definition
Let $V \subseteq Var(F)$. The subformula of $F$ induced by $V$ is the formula
$F|_V = \{ C \mid C \in F \text{ and } Var(C) \subseteq V \}$.
I.e. $F|_V$ includes only those clauses of $F$ whose variables are in $V$.

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

$$
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
$$

$\Rightarrow F|_{\{p, q\}} = \{ C_1, C_2, C_3, C_4 \} \in UNSAT.$
Definition
Let $V \subseteq \text{Var}(\mathcal{F})$. The subformula of $\mathcal{F}$ induced by $V$ is the formula
$\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}$.
I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in $V$.

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

$C_1 = (p)$ \hspace{1cm} $C_3 = (\neg p \lor \neg q)$ \hspace{1cm} $C_5 = (\neg p \lor r)$

$C_2 = (q)$ \hspace{1cm} $C_4 = (p \lor q)$ \hspace{1cm} $C_6 = (\neg q \lor \neg r)$

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

[Zhen-Yu Chen and De-Cheng Ding, TAMC’06]

**Definition**

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

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

**Definition**

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

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

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

A. Belov, A. Ivrii, A. Matsliah and J. Marques-Silva  
Variable MUSes  
SAT 2012  
6 / 24
**Variable-MUSes** [Zhen-Yu Chen and De-Cheng Ding, TAMC’06]

**Definition**
Let $V \subseteq \text{Var}(\mathcal{F})$. The subformula of $\mathcal{F}$ *induced* by $V$ is the formula $\mathcal{F}|_V = \{ C \mid C \in \mathcal{F} \text{ and } \text{Var}(C) \subseteq V \}$. I.e. $\mathcal{F}|_V$ includes only those clauses of $\mathcal{F}$ whose variables are in $V$.

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

\[
\begin{align*}
C_1 &= (p) & C_3 &= (\neg p \lor \neg q) & C_5 &= (\neg p \lor r) \\
C_2 &= (q) & C_4 &= (p \lor q) & C_6 &= (\neg q \lor \neg r)
\end{align*}
\]

Hence, $\{p, q\}$ is a VMUS of $\mathcal{F}$. Notation: $\{p, q\} \in \text{VMUS}(\mathcal{F})$. 
Basic algorithms are similar to MUS extraction algorithms: based on detection of \textit{necessary} variables.

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

**Definition**

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

Basic algorithms are similar to MUS extraction algorithms: based on detection of \textit{necessary} variables.

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

Definition

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

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 $\mathcal{F}' \subseteq \mathcal{F}$.
Hybrid VMUS Computation w/o optimizations

**Input** $\mapsto$ **Output**: Unsatisfiable CNF Formula $\mathcal{F} \mapsto V \in \text{VMUS}(\mathcal{F})$

\[
egin{align*}
V & \leftarrow \emptyset \quad /\!/ \text{ VMUS under-approximation} \\
V_w & \leftarrow \text{Var}(\mathcal{F}) \quad /\!/ \text{ Working (‘‘untested’’) set of variables} \\
\mathcal{F}_w & \leftarrow \mathcal{F} \quad /\!/ \text{ Working formula} \\
\text{while } V_w \neq \emptyset & \quad /\!/ \text{ Inv: } \mathcal{F}_w = \mathcal{F}|_{V \cup V_w} \text{ and } \forall v \in V \text{ is nec. for } \mathcal{F}_w
\end{align*}
\]

\[
\begin{align*}
\forall v & \leftarrow \text{PickVariable}(V_w) \\
V_w & \leftarrow V_w \setminus \{v\} \quad /\!/ \text{ Redundancy removal} \\
\mathcal{R} & \leftarrow \text{CNF}(\neg \mathcal{F}_w^v) \\
\text{st} & = \text{SAT}(\mathcal{F}_w \setminus \mathcal{F}_w^v) \\
\text{if } \text{st} = \text{false} & \quad /\!/ v \text{ is not necessary for } \mathcal{F}_w
\end{align*}
\]

\[
\begin{align*}
V_w & \leftarrow V_w \cap \text{Var}(\mathcal{U}) \quad /\!/ \text{ Variable-set refinement} \\
\mathcal{F}_w & \leftarrow \mathcal{F}_w \setminus \mathcal{F}_w^v
\end{align*}
\]

\[
\text{else} \quad /\!/ v \text{ is necessary for } \mathcal{F}_w
\]

\[
\begin{align*}
V & \leftarrow V \cup \{v\} \\
V_w & \leftarrow V_w \setminus \{v\}
\end{align*}
\]

\[
\text{return } V \quad /\!/ V \in \text{VMUS}(\mathcal{F}) \text{ and } \mathcal{F}_w = \mathcal{F}|_{V \in \text{VMU}}
\]
Hybrid VMUS Computation w/o optimizations


Does not scale:

- Number of SAT solver calls: $|\text{Var}(\mathcal{F})|$.
- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.

Bottleneck: SAT solver calls.

Optimizations are aimed at:

- Making fewer SAT solver calls.
- Making SAT solver calls easier.
Hybrid VMUS Computation w/o optimizations


Does not scale:

- Number of SAT solver calls: $|\text{Var}(\mathcal{F})|$.
- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.
Hybrid VMUS Computation w/o optimizations


Does not scale:

- Number of SAT solver calls: $|\text{Var}(\mathcal{F})|$
- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.
- Bottleneck: SAT solver calls.
Hybrid VMUS Computation w/o optimizations


Does not scale:

▶ Number of SAT solver calls: $|\text{Var}(\mathcal{F})|$.
▶ 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.
▶ Bottleneck: SAT solver calls.

Optimizations are aimed at:

▶ Making fewer SAT solver calls.
▶ Making SAT solver calls easier.
Optimizations: variable-set refinement

- **Fact:** Let \( U \) be an unsatisfiable core of \( F \). Then, \( \text{Var}(U) \) contains at least one VMUS of \( F \).
Fact: Let $\mathcal{U}$ be an unsatisfiable core of $\mathcal{F}$. Then, $\text{Var}(\mathcal{U})$ contains at least one VMUS of $\mathcal{F}$.

Hence, if $\mathcal{U}$ is an unsatisfiable core of $\mathcal{F}$, all variables outside of $\text{Var}(\mathcal{U})$ can be removed from $\mathcal{F}$ — variable-set refinement.
Fact: Let $\mathcal{U}$ be an unsatisfiable core of $\mathcal{F}$. Then, $\text{Var}(\mathcal{U})$ contains at least one VMUS of $\mathcal{F}$.

Hence, if $\mathcal{U}$ is an unsatisfiable core of $\mathcal{F}$, all variables outside of $\text{Var}(\mathcal{U})$ can be removed from $\mathcal{F}$ — variable-set refinement.

Relies on the capability of SAT solvers to return unsatisfiable core.
Fact: Let $\mathcal{U}$ be an unsatisfiable core of $\mathcal{F}$. Then, $\text{Var}(\mathcal{U})$ contains at least one VMUS of $\mathcal{F}$.

Hence, if $\mathcal{U}$ is an unsatisfiable core of $\mathcal{F}$, all variables outside of $\text{Var}(\mathcal{U})$ can be removed from $\mathcal{F}$ — variable-set refinement.

Relies on the capability of SAT solvers to return unsatisfiable core.

Applied to the working set of variables $V_w$ inside the main loop, $\mathcal{F}_w$ is updated accordingly.
Optimizations: variable-set refinement

▷ **Fact:** Let $\mathcal{U}$ be an unsatisfiable core of $\mathcal{F}$. Then, $\text{Var}(\mathcal{U})$ contains at least one VMUS of $\mathcal{F}$.

▷ Hence, if $\mathcal{U}$ is an unsatisfiable core of $\mathcal{F}$, all variables outside of $\text{Var}(\mathcal{U})$ can be removed from $\mathcal{F}$ — *variable-set refinement*.

▷ Relies on the capability of SAT solvers to return unsatisfiable core.

▷ Applied to the working set of variables $V_w$ inside the main loop, $\mathcal{F}_w$ is updated accordingly.

▷ Effect: remove multiple unnecessary variables in one SAT solver call.
Hybrid VMUS Computation: variable-set refinement

**Input** $\mapsto$ **Output**: Unsatisfiable CNF Formula $\mathcal{F}$ $\mapsto$ $V \in \text{VMUS}(\mathcal{F})$

- $V \leftarrow \emptyset$ // VMUS under-approximation
- $V_w \leftarrow \text{Var}(\mathcal{F})$ // Working (‘‘untested’’) set of variables
- $\mathcal{F}_w \leftarrow \mathcal{F}$ // Working formula

**while** $V_w \neq \emptyset$ **do** // Inv: $\mathcal{F}_w = \mathcal{F}|_{V \cup V_w}$ and $\forall v \in V$ is nec. for $\mathcal{F}_w$

- $v \leftarrow \text{PickVariable}(V_w)$
- $V_w \leftarrow V_w \setminus \{v\}$
- $\mathcal{R} \leftarrow \text{CNF}(\neg \mathcal{F}_w^v)$ // Redundancy removal
- $\text{st} = \text{SAT}(\mathcal{F}_w \setminus \mathcal{F}^v)$
- **if** $\text{st} = \text{false}$ **then** // $v$ is not necessary for $\mathcal{F}_w$
  - $V_w \leftarrow V_w \cap \text{Var}(U)$ // Variable-set refinement
  - $\mathcal{F}_w \leftarrow \mathcal{F}_w \setminus \mathcal{F}^v$
- **else** // $v$ is necessary for $\mathcal{F}_w$
  - $V \leftarrow V \cup \{v\}$
  - $V_w \leftarrow V_w \setminus \{v\}$

**return** $V$ // $V \in \text{VMUS}(\mathcal{F})$ and $\mathcal{F}_w = \mathcal{F}|_{V \in \text{VMU}}$
Hybrid VMUS Computation: variable-set refinement

**Input** $\mapsto$ **Output**: Unsatisfiable CNF Formula $\mathcal{F} \mapsto V \in \text{VMUS}(\mathcal{F})$

\[
\begin{align*}
V & \leftarrow \emptyset \quad \text{ // VMUS under-approximation} \\
V_w & \leftarrow \text{Var}(\mathcal{F}) \quad \text{ // Working (‘‘untested’’) set of variables} \\
\mathcal{F}_w & \leftarrow \mathcal{F} \quad \text{ // Working formula}
\end{align*}
\]

\textbf{while} $V_w \neq \emptyset$ \textbf{do} \quad // Inv: $\mathcal{F}_w = \mathcal{F}|_{V \cup V_w}$ and $\forall v \in V$ is nec. for $\mathcal{F}_w$

\[
\begin{align*}
& v \leftarrow \text{PickVariable}(V_w) \\
& V_w \leftarrow V_w \setminus \{v\} \\
& \mathcal{R} \leftarrow \text{CNF}(\neg \mathcal{F}_w^v) \\
& \langle \text{st}, U \rangle = \text{SAT}(\mathcal{F}_w \setminus \mathcal{F}_w^v) \\
& \textbf{if} \ \text{st} = \text{false} \ \textbf{then} \\
& \quad V_w \leftarrow V_w \cap \text{Var}(U) \\
& \quad \mathcal{F}_w \leftarrow \mathcal{F}_w|_{V \cup V_w} \\
& \textbf{else} \\
& \quad V \leftarrow V \cup \{v\} \\
& \quad V_w \leftarrow V_w \setminus \{v\} \\
\end{align*}
\]

\textbf{return} $V$ \quad // $V \in \text{VMUS}(\mathcal{F})$ and $\mathcal{F}_w = \mathcal{F}|_V \in \text{VMU}$
Impact of variable-set refinement

- 295 benchmarks from SAT Comp 2011, $T_O = 1800$ sec, $M_O = 4$ GB.

- CPU Time, w/o optimizations (#sol = 157) vs refinement (#sol = 239)
- Color: VMUS size (% of the number of variables in the input).
Fact: Take $\mathcal{F} \in$ UNSAT and an assignment $\tau$ to $\text{Var}(\mathcal{F})$. Then, any variable shared among the clauses falsified by $\tau$ is necessary for $\mathcal{F}$.
Fact: Take $\mathcal{F} \in \text{UNSAT}$ and an assignment $\tau$ to $\text{Var}(\mathcal{F})$. Then, any variable shared among the clauses falsified by $\tau$ is necessary for $\mathcal{F}$.

Why?

- Let $\mathcal{F}'$ be the clauses of $\mathcal{F}$ falsified by $\tau$.
- Let $v$ be any variable shared among the clauses of $\mathcal{F}'$
  - i.e. $\forall C \in \mathcal{F}', v \in \text{Var}(C)$.
- Remove $v$ from $\mathcal{F}$.
- All clauses of $\mathcal{F}'$ will be gone, because they all contain $v$.
- We get a subformula of $\mathcal{F} \setminus \mathcal{F}'$.
- But, $\mathcal{F} \setminus \mathcal{F}' \in \text{SAT}$, since $\tau$ is its model.
- Hence, $\mathcal{F} \setminus \mathcal{F}' \setminus v \in \text{SAT}$, and so $v$ is necessary for $\mathcal{F}$. 
Fact: Take $\mathcal{F} \in \text{UNSAT}$ and an assignment $\tau$ to $\text{Var}(\mathcal{F})$. Then, any variable shared among the clauses falsified by $\tau$ is necessary for $\mathcal{F}$.

Why?

- Let $\mathcal{F}'$ be the clauses of $\mathcal{F}$ falsified by $\tau$.
- Let $v$ be any variable shared among the clauses of $\mathcal{F}'$
  - i.e. $\forall C \in \mathcal{F}', \ v \in \text{Var}(C)$.
- Remove $v$ from $\mathcal{F}$.
- All clauses of $\mathcal{F}'$ will be gone, because they all contain $v$.
- We get a subformula of $\mathcal{F} \setminus \mathcal{F}'$.
- But, $\mathcal{F} \setminus \mathcal{F}' \in \text{SAT}$, since $\tau$ is its model.
- Hence, $\mathcal{F} \setminus \mathcal{F}' \in \text{SAT}$, and so $v$ is necessary for $\mathcal{F}$.

So, any assignment $\tau$, such that $v \in \bigcap_{C \in \text{Unsat}(\mathcal{F},\tau)} \text{Var}(C)$, is a witness of necessity of $v$ in $\mathcal{F}$. 

A. Belov, A. Ivrii, A. Matsliakh and J. Marques-Silva
Note: in VHYB, when $\mathcal{F}_w \setminus \mathcal{F}^{\nu}_w \in SAT$, the assignment returned by the SAT solver (call it $\tau$) is a witness of $\nu$. 

Variable-based model rotation (VMR):

1. Flip $\nu$ to get new assignment $\tau'$: all clauses in $\text{Unsat}(\mathcal{F}_w, \tau)$ are now satisfied.
2. But, some other clauses must be falsified by $\tau'$.
3. Any variable shared among the clauses falsified by $\tau'$ is necessary.
4. Continue recursively (but watch for loops).

Effect: detect multiple necessary variables in a single SAT solver call.

Details and an "extended" (improved) version of VMR are in the paper.

A. Belov, A. Ivrii, A. Matsliah and J. Marques-Silva
Variable MUSes
SAT 2012
Note: in VHYB, when $F_w \setminus F^v_w \in \text{SAT}$, the assignment returned by the SAT solver (call it $\tau$) is a witness of $v$.

$\tau$ might be a witness for another variable too. E.g. when $\text{Unsat}(F_w, \tau) = \{C\}$, every variable in $C$ is necessary for $F_w$. 

Variable-based model rotation (VMR): take $\tau$ and try to modify it into a witness $\tau'$ for another variable. How?

1. Flip $v$ to get new assignment $\tau'$: all clauses in $\text{Unsat}(F_w, \tau)$ are now satisfied.
2. But, some other clauses must be falsified by $\tau'$.
3. Any variable shared among the clauses falsified by $\tau'$ is necessary.
4. Continue recursively (but watch for loops).

Effect: detect multiple necessary variables in a single SAT solver call. Details and an "extended" (improved) version of VMR are in the paper.
Note: in VHYB, when $F_w \setminus F^v_w \in \text{SAT}$, the assignment returned by the SAT solver (call it $\tau$) is a witness of $v$.

$\tau$ might be a witness for another variable too. E.g. when $\text{Unsat}(F_w, \tau) = \{C\}$, every variable in $C$ is necessary for $F_w$.

Variable-based model rotation: take $\tau$ and try to modify it into a witness $\tau'$ for another variable. How?

1. Flip $v$ to get new assignment $\tau'$: all clauses in $\text{Unsat}(F_w, \tau)$ are now satisfied.
2. But, some other clauses must be falsified by $\tau'$.
3. Any variable shared among the clauses falsified by $\tau'$ is necessary.
4. Continue recursively (but watch for loops).
Optimizations: variable-based model rotation (VMR)

- Note: in VHYB, when $\mathcal{F}_w \setminus \mathcal{F}_w^v \in \text{SAT}$, the assignment returned by the SAT solver (call it $\tau$) is a witness of $v$.

- $\tau$ might be a witness for another variable too. E.g. when $\text{Unsat}(\mathcal{F}_w, \tau) = \{ C \}$, every variable in $C$ is necessary for $\mathcal{F}_w$.

- **Variable-based model rotation**: take $\tau$ and try to modify it into a witness $\tau'$ for another variable. How?
  1. Flip $v$ to get new assignment $\tau'$: all clauses in $\text{Unsat}(\mathcal{F}_w, \tau)$ are now satisfied.
  2. But, some other clauses must be falsified by $\tau'$.
  3. Any variable shared among the clauses falsified by $\tau'$ is necessary.
  4. Continue recursively (but watch for loops).

- Effect: detect multiple necessary variables in a single SAT solver call.
Note: in VHYB, when $F_w \setminus F^v_w \in \text{SAT}$, the assignment returned by the SAT solver (call it $\tau$) is a witness of $v$.

$\tau$ might be a witness for another variable too. E.g. when $\text{Unsat}(F_w, \tau) = \{C\}$, every variable in $C$ is necessary for $F_w$.

**Variable-based model rotation**: take $\tau$ and try to modify it into a witness $\tau'$ for another variable. How?

1. Flip $v$ to get new assignment $\tau'$: all clauses in $\text{Unsat}(F_w, \tau)$ are now satisfied.
2. But, some other clauses must be falsified by $\tau'$.
3. Any variable shared among the clauses falsified by $\tau'$ is necessary.
4. Continue recursively (but watch for loops).

Effect: detect multiple necessary variables in a single SAT solver call.

Details and an “extended” (improved) version of VMR are in the paper.
Hybrid VMUS Computation: VMR

**Input** $\mapsto$ **Output**: Unsatisfiable CNF Formula $\mathcal{F} \mapsto V \in \text{VMUS}(\mathcal{F})$

- $V \leftarrow \emptyset$  // VMUS under-approximation
- $V_w \leftarrow \text{Var}(\mathcal{F})$  // Working (‘‘untested’’) set of variables
- $\mathcal{F}_w \leftarrow \mathcal{F}$  // Working formula

while $V_w \neq \emptyset$ do  // Inv: $\mathcal{F}_w = \mathcal{F}|_{V \cup V_w}$ and $\forall v \in V$ is nec. for $\mathcal{F}_w$

- $v \leftarrow \text{PickVariable}(V_w)$
- $V_w \leftarrow V_w \setminus \{v\}$
- $\mathcal{R} \leftarrow \text{CNF}(\neg \mathcal{F}_w^v)$  // Redundancy removal
- $\langle \text{st}, U \rangle = \text{SAT}(\mathcal{F}_w \setminus \mathcal{F}_w^v)$
- if $\text{st} = \text{false}$ then
  - $V_w \leftarrow V_w \cap \text{Var}(U)$  // Variable-set refinement
  - $\mathcal{F}_w \leftarrow \mathcal{F}_w|_{V \cup V_w}$
- else
  - $V \leftarrow V \cup \{v\}$  // $v$ is necessary for $\mathcal{F}_w$
  - $V_w \leftarrow V_w \setminus v$

return $V$  // $V \in \text{VMUS}(\mathcal{F})$ and $\mathcal{F}_w = \mathcal{F}|_{V \in \text{VMU}}$
Hybrid VMUS Computation: VMR

**Input** $\mapsto$ **Output**: Unsatisfiable CNF Formula $\mathcal{F} \mapsto V \in VMUS(\mathcal{F})$

$V \leftarrow \emptyset$ // VMUS under-approximation

$V_w \leftarrow \text{Var}(\mathcal{F})$ // Working (‘‘untested’’) set of variables

$\mathcal{F}_w \leftarrow \mathcal{F}$ // Working formula

while $V_w \neq \emptyset$ do // Inv: $\mathcal{F}_w = \mathcal{F}|_{V \cup V_w}$ and $\forall v \in V$ is nec. for $\mathcal{F}_w$

\[ \begin{align*}
& v \leftarrow \text{PickVariable}(V_w) \\
& V_w \leftarrow V_w \setminus \{v\} \\
& \mathcal{R} \leftarrow \text{CNF}(\neg \mathcal{F}_w^v) \\
& \langle \text{st}, U, \tau \rangle = \text{SAT}(\mathcal{F}_w \setminus \mathcal{F}_w^v) \\
& \text{if } \text{st} = \text{false} \text{ then} \\
& \quad V_w \leftarrow V_w \cap \text{Var}(U) \\
& \quad \mathcal{F}_w \leftarrow \mathcal{F}_w|_{V \cup V_w} \\
& \text{else} \\
& \quad \text{VModelRotation}(\mathcal{F}_w, V, \tau) \quad \text{ // } v \in V \text{ after this call} \\
& \quad V_w \leftarrow V_w \setminus V \quad \text{ // } v \text{ is necessary for } \mathcal{F}_w
\end{align*} \]

return $V$ // $V \in VMUS(\mathcal{F})$ and $\mathcal{F}_w = \mathcal{F}|_{V \in VMU}$
Impact of variable-based model rotation

- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.

- CPU Time, refinement (#sol=239) vs ref.+EVMR (#sol=264)
- Color: VMUS size (% of the number of variables in the input).
Fact: If $\mathcal{F} \in \text{UNSAT}$, then for any $v \in \text{Var}(\mathcal{F})$, 
$\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$ if and only if $(\mathcal{F} \setminus \mathcal{F}^v) \cup \text{CNF}(\neg \mathcal{F}^v) \in \text{SAT}$.

$\neg \text{CNF}(\mathcal{F}^v)$ stands for a satisfiability-preserving transformation of the formula $\neg \mathcal{F}^v$ (we used Plaisted-Greenbaum).

During hybrid VMUS extraction: add the clauses of $\text{CNF}(\neg \mathcal{F}^v)$ to the formula before SAT solver call.
**Fact:** If $\mathcal{F} \in \text{UNSAT}$, then for any $v \in \text{Var}(\mathcal{F})$, $\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$ if and only if $(\mathcal{F} \setminus \mathcal{F}^v) \cup \text{CNF}(\neg \mathcal{F}^v) \in \text{SAT}$.

- $\neg \text{CNF}(\mathcal{F}^v)$ stands for a satisfiability-preserving transformation of the formula $\neg \mathcal{F}^v$ (we used Plaisted-Greenbaum).
- During hybrid VMUS extraction: add the clauses of $\text{CNF}(\neg \mathcal{F}^v)$ to the formula before SAT solver call.

**Effect:** make SAT calls easier.
Optimizations: redundancy removal

- **Fact:** If $\mathcal{F} \in \text{UNSAT}$, then for any $v \in \text{Var}(\mathcal{F})$, $\mathcal{F} \setminus \mathcal{F}^v \in \text{SAT}$ if and only if $(\mathcal{F} \setminus \mathcal{F}^v) \cup \text{CNF}(\neg\mathcal{F}^v) \in \text{SAT}$.

  - $\neg\text{CNF}(\mathcal{F}^v)$ stands for a satisfiability-preserving transformation of the formula $\neg\mathcal{F}^v$ (we used Plaisted-Greenbaum).
  - During hybrid VMUS extraction: add the clauses of $\text{CNF}(\neg\mathcal{F}^v)$ to the formula before SAT solver call.

  - **Effect:** make SAT calls easier.
  - **But:** if the clauses of $\text{CNF}(\neg\mathcal{F}^v)$ are included in the unsatisfiable core (in case of UNSAT outcome), the core cannot be used safely for the variable-set refinement.
Hybrid VMUS Extraction: redundancy removal

**Input** $\mapsto$ **Output**: Unsatisfiable CNF Formula $\mathcal{F} \mapsto V \in \text{VMUS}(\mathcal{F})$

$V \leftarrow \emptyset$  \hspace{1cm} // VMUS under-approximation

$V_w \leftarrow \text{Var}(\mathcal{F})$  \hspace{1cm} // Working (‘‘untested’’) set of variables

$\mathcal{F}_w \leftarrow \mathcal{F}$  \hspace{1cm} // Working formula

**while** $V_w \neq \emptyset$ **do**  \hspace{1cm} // Inv: $\mathcal{F}_w = \mathcal{F}|_{V \cup V_w}$ and $\forall v \in V$ is nec. for $\mathcal{F}_w$

1. $v \leftarrow \text{PickVariable}(V_w)$
2. $V_w \leftarrow V_w \setminus \{v\}$
3. $R \leftarrow \text{CNF}(\neg \mathcal{F}_w^v)$  \hspace{1cm} // Redundancy removal

4. $\langle \text{st}, \mathcal{U}, \tau \rangle = \text{SAT}(\mathcal{F}_w \setminus \mathcal{F}_w^v)$

5. **if** $\text{st} = \text{false}$ **then**

6. 1. $V_w \leftarrow V_w \cap \text{Var}(\mathcal{U})$
7. 2. $\mathcal{F}_w \leftarrow \mathcal{F}_w|_{V \cup V_w}$  \hspace{1cm} // $v$ is not necessary for $\mathcal{F}_w$

8. **else**

9. 1. $\text{VModelRotation}(\mathcal{F}_w, V, \tau)$  \hspace{1cm} // $v \in V$ after this call
10. 2. $V_w \leftarrow V_w \setminus V$  \hspace{1cm} // $v$ is necessary for $\mathcal{F}_w$

**return** $V$  \hspace{1cm} // $V \in \text{VMUS}(\mathcal{F})$ and $\mathcal{F}_w = \mathcal{F}|_{v \in \text{VMU}}$

A. Belov, A. Ivrii, A. Matsliah and J. Marques-Silva
Variable MUSes
Hybrid VMUS Extraction: redundancy removal

**Input $\mapsto$ Output**: Unsatisfiable CNF Formula $\mathcal{F} \mapsto V \in \text{VMUS}(\mathcal{F})$

\[
V \leftarrow \emptyset \quad \text{// VMUS under-approximation}
\]

\[
V_w \leftarrow \text{Var}(\mathcal{F}) \quad \text{// Working (‘‘untested’’) set of variables}
\]

\[
\mathcal{F}_w \leftarrow \mathcal{F} \quad \text{// Working formula}
\]

\[
\text{while } V_w \neq \emptyset \text{ do} \quad \text{// Inv: } \mathcal{F}_w = \mathcal{F}|_{V \cup V_w} \text{ and } \forall v \in V \text{ is nec. for } \mathcal{F}_w
\]

\[
\begin{align*}
v & \leftarrow \text{PickVariable}(V_w) \\
V_w & \leftarrow V_w \setminus \{v\} \\
\mathcal{R} & \leftarrow \text{CNF}(\neg \mathcal{F}_w^v) \quad \text{// Redundancy removal}
\end{align*}
\]

\[
\langle \text{st}, U, \tau \rangle = \text{SAT}((\mathcal{F}_w \setminus \mathcal{F}_w^v) \cup \mathcal{R})
\]

\[
\begin{align*}
\text{if } \text{st} = \text{false} \text{ then} & \quad \text{// } v \text{ is not necessary for } \mathcal{F}_w \\
& \begin{align*}
& \text{if } U \cap \mathcal{R} = \emptyset \text{ then } V_w \leftarrow V_w \cap \text{Var}(U) \quad \text{// ‘‘Safe’’ refinement} \\
& \quad \mathcal{F}_w \leftarrow \mathcal{F}_w|_{V \cup V_w}
\end{align*}
\end{align*}
\]

\[\text{else} \quad \text{// } v \text{ is necessary for } \mathcal{F}_w
\]

\[
\begin{align*}
& \quad \text{VModelRotation}(\mathcal{F}_w, V, \tau) \quad \text{// } v \in V \text{ after this call} \\
& \quad V_w \leftarrow V_w \setminus V
\end{align*}
\]

\[\text{return } V \quad \text{// } V \in \text{VMUS}(\mathcal{F}) \text{ and } \mathcal{F}_w = \mathcal{F}|_{V \in \text{VMU}}
\]
Impact of redundancy removal

- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.

- CPU Time, ref.+EVMR vs ref.+EVMR+RR. Average speedup: 15%.
- Color: VMUS size (% of the number of variables in the input).
Impact of all optimizations

- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.

Additional approaches (in the paper)

Relaxation-variable based approach.
- Lets the SAT solver to *find* a necessary variable.
- Probably won’t scale (have not tried it).

Translation to group-MUS computation problem.
- For dense formulas: for each $v \in \text{Var}(F)$ introduce two variables $v_p$ and $v_n$, and
  1. replace $v$ with $v_p$ and $\neg v$ with $v_n$;
  2. put the result into group 0;
  3. make a group that says $v_p \leftrightarrow v_n$.
- For sparse formulas: for each $v \in \text{Var}(F)$ introduce an activation variable $a_v$, and
  1. add an activation variable $a_v$ to clause $C$ if $v \in C$;
  2. put all translated clauses to group 0;
  3. for each variable make a group $\{a_v\}$.

Outperformed by Hybrid VMUS algorithm by a wide margin (see paper)
Additional approaches (in the paper)

Relaxation-variable based approach.

- Lets the SAT solver to find a necessary variable.
- Probably won’t scale (have not tried it).

Translation to group-MUS computation problem.

- For dense formulas: for each $v \in Var(F)$ introduce two variables $v_p$ and $v_n$, and
  1. replace $v$ with $v_p$ and $\neg v$ with $v_n$;
  2. put the result into group 0;
  3. make a group that says $v_p \leftrightarrow v_n$.

Outperformed by Hybrid VMUS algorithm by a wide margin (see paper)

A. Belov, A. Ivrii, A. Matsliah and J. Marques-Silva
Additional approaches (in the paper)

Relaxation-variable based approach.
- Lets the SAT solver to find a necessary variable.
- Probably won’t scale (have not tried it).

Translation to group-MUS computation problem.
- For dense formulas: for each $v \in \text{Var}(\mathcal{F})$ introduce two variables $v_p$ and $v_n$, and
  1. replace $v$ with $v_p$ and $\neg v$ with $v_n$;
  2. put the result into group 0;
  3. make a group that says $v_p \leftrightarrow v_n$.
- For sparse formulas: for each $v \in \text{Var}(\mathcal{F})$ introduce an activation variable $a_v$, and
  1. add an activation variable $a_v$ to clause $C$ if $v \in C$;
  2. put all translated clauses to group 0;
  3. for each variable make a group $\{a_v\}$. 
Additional approaches (in the paper)

Relaxation-variable based approach.
- Lets the SAT solver to *find* a necessary variable.
- Probably won’t scale (have not tried it).

Translation to group-MUS computation problem.
- For dense formulas: for each \( v \in \text{Var}(\mathcal{F}) \) introduce two variables \( v_p \) and \( v_n \), and
  1. replace \( v \) with \( v_p \) and \( \neg v \) with \( v_n \);
  2. put the result into group 0;
  3. make a group that says \( v_p \leftrightarrow v_n \).
- For sparse formulas: for each \( v \in \text{Var}(\mathcal{F}) \) introduce an activation variable \( a_v \), and
  1. add an activation variable \( a_v \) to clause \( C \) if \( v \in C \);
  2. put all translated clauses to group 0;
  3. for each variable make a group \( \{a_v\} \).

*Outperformed by Hybrid VMUS algorithm by a wide margin* (see paper)
Performance comparison: run-time

- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.
VMUS extraction vs MUS extraction

- 295 benchmarks from SAT Comp 2011, TO = 1800 sec, MO = 4GB.

- CPU Time, MUS (#sol=245) vs VMUS extraction (#sol=265).
Summary

- Hybrid VMUS Extraction — significant performance gains vs state-of-the-art due to optimizations.
Summary

- Hybrid VMUS Extraction — significant performance gains vs state-of-the-art due to optimizations.

Thank you for your attention!
Summary

- Hybrid VMUS Extraction — significant performance gains vs state-of-the-art due to optimizations.
- A number of alternative approaches (not as efficient).
Summary

- Hybrid VMUS Extraction — significant performance gains vs state-of-the-art due to optimizations.
- A number of alternative approaches (not as efficient).
- VMUS computation is in many cases faster than MUS computation ⇒ perhaps some applications of MUSes could be re-considered.
Summary

- Hybrid VMUS Extraction — significant performance gains vs state-of-the-art due to optimizations.
- A number of alternative approaches (not as efficient).
- VMUS computation is in many cases faster than MUS computation ⇒ perhaps some applications of MUSes could be re-considered.

Thank you for your attention!