Factoring Out Assumptions to Speed Up MUS Extraction

Jean-Marie Lagniez[1] **Armin Biere**[2]

11 July 2013

[1] Univ. Lille-Nord de France – CRIL/CNRS UMR8188 – Lens, F-62307, France
[2] Institute for Formal Models and Verification Johannes Kepler University, Linz, Austria

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

UNSAT

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$
	SAT	

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - → constructive
  - → **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

UNSAT

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		
$\neg x \vee \neg y$	$\neg x \vee \neg z$	

MUS!

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

Minimal Unsatisfiable Set (MUS)

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		
$\neg x \vee \neg y$	$\neg x \vee \neg z$	

MUS!

- The formula is **unsatisfiable** : why?
- Subset of constraints **minimally unsatisfiable**
- Two approaches:
  - $\rightarrow$ constructive
  - $\rightarrow$ **destructive**

SAT Incremental

From SAT to Incremental SAT

Solving the SAT problem

- Modern SAT solvers are based on the CDCL paradigm
- Dynamic heuristics:
  - → VSIDS, polarity, cleaning learned clauses and restart

Solving incrementally SAT

- Successive calls of a SAT solver
- Keeping a lot of information between the different runs
  - → VSIDS, polarity, cleaning learned clauses and restart

From SAT to Incremental SAT

Solving the SAT problem

- Modern SAT solvers are based on the CDCL paradigm
- Dynamic heuristics:

  → VSIDS, polarity, cleaning learned clauses and restart

Solving incrementally SAT

- Successive calls of a SAT solver
- Keeping a lot of information between the different runs

  → VSIDS, polarity, cleaning learned clauses and restart
  → **learned clauses**

From SAT to Incremental SAT

Solving the SAT problem

- Modern SAT solvers are based on the CDCL paradigm
- Dynamic heuristics:

  $\rightarrow$ VSIDS, polarity, cleaning learned clauses and restart

Solving incrementally SAT

- Successive calls of a SAT solver
- Keeping a lot of information between the different runs

  $\rightarrow$ VSIDS, polarity, cleaning learned clauses and restart
  $\rightarrow$ **learned clauses**

Adding selectors

Selectors

$a_1 \lor x \lor y \lor z$ $a_2 \lor x \lor \neg y$ $a_3 \lor x \lor \neg z$

$a_4 \lor \neg x \lor y \lor z$ $a_5 \lor x \lor w$ $a_6 \lor w \lor z \lor \neg y$

$a_7 \lor \neg x \lor \neg y$ $a_8 \lor \neg x \lor \neg z$ $a_9 \lor w \lor \neg x \lor \neg z$

- To activate/deactivate the i^{th} clause :
  - $\rightarrow$ assign a_i to **false** to **activate** the clause
  - $\rightarrow$ assign a_i to **true** to **deactivate** the clause
- Used to know which initial clauses participating to the creation of each learned clause

$$
\begin{tikzcd}
a_1 \lor x \lor y \lor z \arrow[dr] & & a_2 \lor x \lor \neg y \arrow[dl] \\
& a_1 \lor a_2 \lor x \lor z &
\end{tikzcd}
$$

Selectors

$a_1 \vee x \vee y \vee z$	$a_2 \vee x \vee \neg y$	$a_3 \vee x \vee \neg z$
$a_4 \vee \neg x \vee y \vee z$	$a_5 \vee x \vee w$	$a_6 \vee w \vee z \vee \neg y$
$a_7 \vee \neg x \vee \neg y$	$a_8 \vee \neg x \vee \neg z$	$a_9 \vee w \vee \neg x \vee \neg z$

- To activate/deactivate the ith clause :
  - $\rightarrow$ assign a_i to **false** to **activate** the clause
  - $\rightarrow$ assign a_i to **true** to **deactivate** the clause
- Used to know which initial clauses participating to the creation of each learned clause

$$
\begin{tikzcd}
a_1 \vee x \vee y \vee z \arrow[dr] & & a_2 \vee x \vee \neg y \arrow[dl] \\
& a_1 \vee a_2 \vee x \vee z &
\end{tikzcd}
$$

Selectors impact on the size of the clauses

Factoring-out Assumptions

Introducing abbreviations to factor out assumptions

- The replaced part consists of all assumptions and previously added abbreviations
- Connections between the abbreviations and the replaced literals is stored in a **definition map**

$$(p_1 \vee \cdots \vee p_n \vee a_1 \vee \cdots \vee a_m)$$

is factored out into

$$(p_1 \vee \cdots \vee p_n \vee \ell) \qquad \text{and} \qquad \ell \mapsto \underbrace{a_1 \vee \cdots \vee a_m}_{\mathcal{G}[\ell]}$$

Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses | antecedents | factored clauses

factoring $\Longrightarrow$

$\mathcal{G}$

Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents	factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$	

$\xrightarrow{\textbf{factoring}}$

$\mathcal{G}$

Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$	**factoring** $\Longrightarrow$	$\alpha_1' : p_2 \vee p_7 \vee \ell_1$

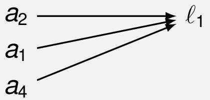
$\mathcal{G}$

Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$	$\xRightarrow{\textbf{factoring}}$	$\alpha_1' : p_2 \vee p_7 \vee \ell_1$

$\mathcal{G}$

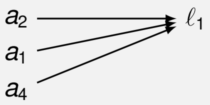


Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$	**factoring** $\Longrightarrow$	$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$		$\alpha'_2 : p_2 \vee \ell_2$

$\mathcal{G}$

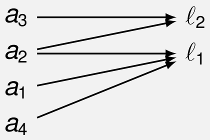


Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$		$\alpha_1' : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$	**factoring** $\Longrightarrow$	$\alpha_2' : p_2 \vee \ell_2$

$\mathcal{G}$

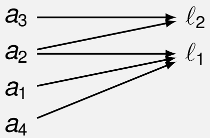


Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$		$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$		$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$	$\xRightarrow{\textbf{factoring}}$	$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$

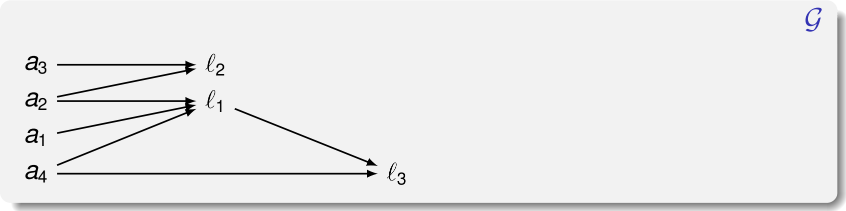
$\mathcal{G}$



Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$		$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$	**factoring** $\Longrightarrow$	$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$		$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$



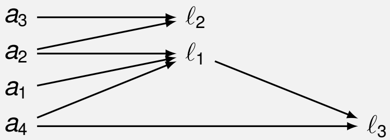
Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents	factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$	$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$	$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$	$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4 : p_6 \vee p_8 \vee \ell_2 \vee a_5$	$\{\alpha_2, \ldots\}$	$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$

factoring $\Longrightarrow$

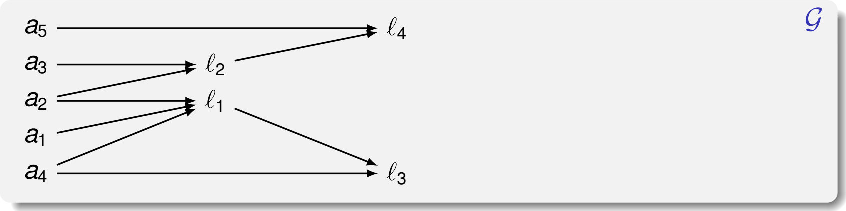
$\mathcal{G}$



Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

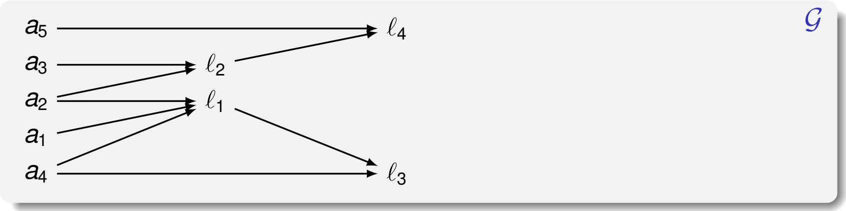
learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$		$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$		$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$	**factoring** $\Longrightarrow$	$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4 : p_6 \vee p_8 \vee \ell_2 \vee a_5$	$\{\alpha_2, \ldots\}$		$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$



Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$		$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$		$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$	**factoring** $\Longrightarrow$	$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4 : p_6 \vee p_8 \vee \ell_2 \vee a_5$	$\{\alpha_2, \ldots\}$		$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha_5 : p_2 \vee p_5 \vee a_2$	$\{\ldots\}$		$\alpha'_5 : p_2 \vee p_5 \vee a_2$

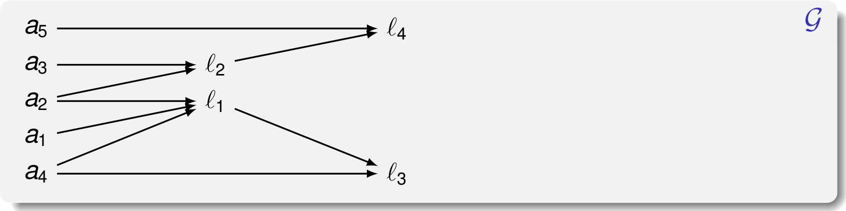


Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

learned clauses	antecedents	factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$	$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$	$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$	$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4 : p_6 \vee p_8 \vee \ell_2 \vee a_5$	$\{\alpha_2, \ldots\}$	$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha_5 : p_2 \vee p_5 \vee a_2$	$\{\ldots\}$	$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha_6 : p_7 \vee p_4 \vee \ell_3 \vee \ell_4$	$\{\alpha_3, \alpha_4, \ldots\}$	$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$

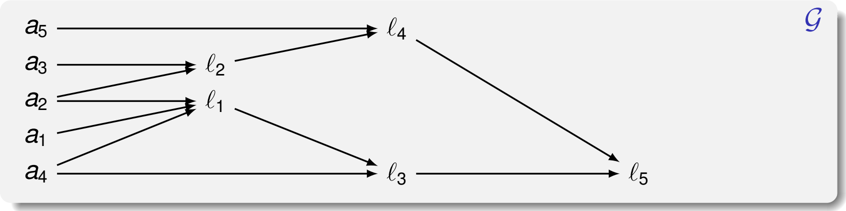
factoring $\Longrightarrow$



Definition Map

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

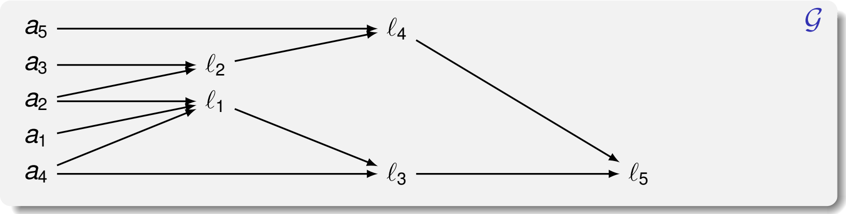
learned clauses	antecedents		factored clauses
$\alpha_1 : p_2 \vee p_7 \vee a_1 \vee a_2 \vee a_4$	$\{\ldots\}$		$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha_2 : p_2 \vee a_2 \vee a_3$	$\{\ldots\}$		$\alpha'_2 : p_2 \vee \ell_2$
$\alpha_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_1 \vee a_4$	$\{\alpha_1, \ldots\}$	**factoring** $\Longrightarrow$	$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4 : p_6 \vee p_8 \vee \ell_2 \vee a_5$	$\{\alpha_2, \ldots\}$		$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha_5 : p_2 \vee p_5 \vee a_2$	$\{\ldots\}$		$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha_6 : p_7 \vee p_4 \vee \ell_3 \vee \ell_4$	$\{\alpha_3, \alpha_4, \ldots\}$		$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$



Initialisation

- The definition map $\mathcal{G}$ can be interpreted as a non-cyclic circuit
- Abbreviations can be computed after all assumptions have been assigned
- In the MUS behaviour, the set of assumptions equals to the set of entries and it remains the same over all incremental calls

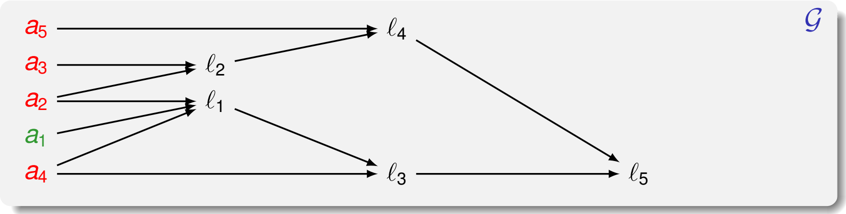
Example: under assumptions $\{a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



Initialisation

- The definition map $\mathcal{G}$ can be interpreted as a non-cyclic circuit
- Abbreviations can be computed after all assumptions have been assigned
- In the MUS behaviour, the set of assumptions equals to the set of entries and it remains the same over all incremental calls

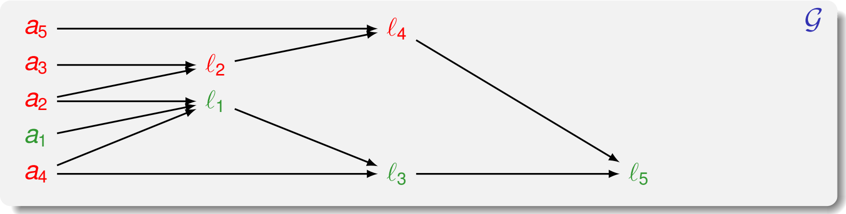
Example: under assumptions $\{a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



Initialisation

- The definition map $\mathcal{G}$ can be interpreted as a non-cyclic circuit
- Abbreviations can be computed after all assumptions have been assigned
- In the MUS behaviour, the set of assumptions equals to the set of entries and it remains the same over all incremental calls

Example: under assumptions $\{a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

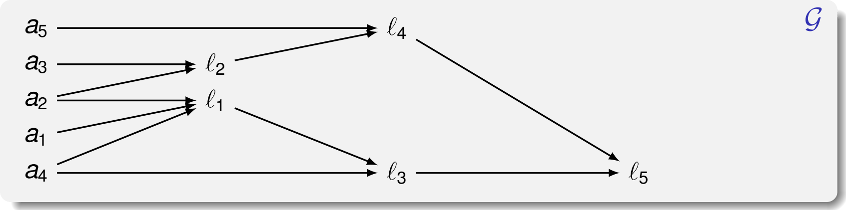


Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



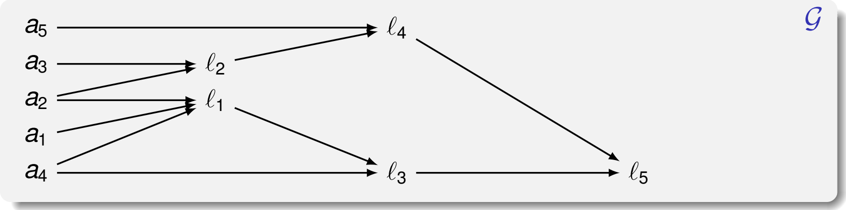
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$

$\perp$

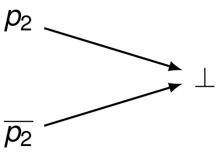
factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



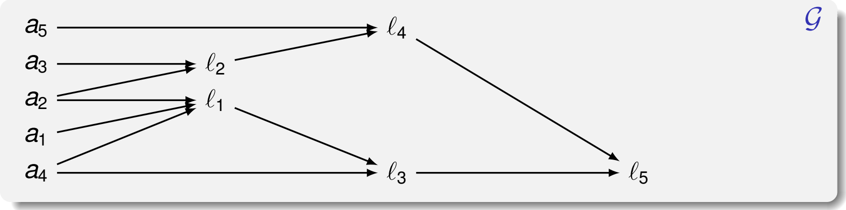
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



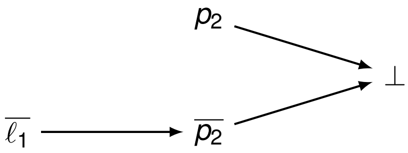
factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



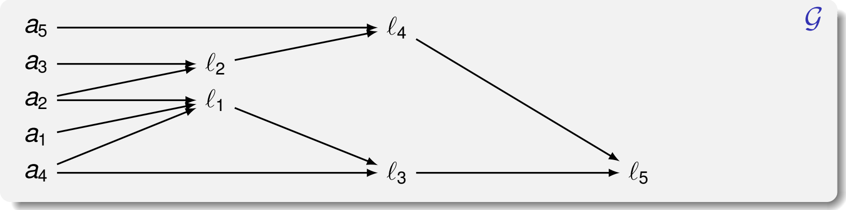
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



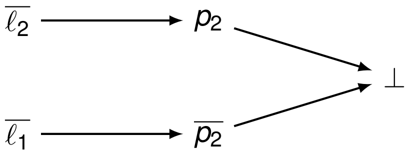
factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



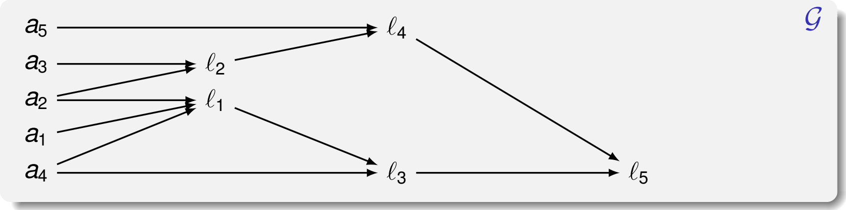
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



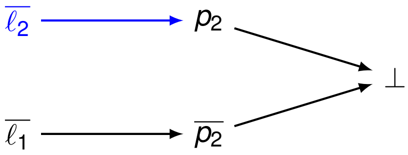
factored clauses

$\alpha_1' : p_2 \vee p_7 \vee \ell_1$
$\alpha_2' : p_2 \vee \ell_2$
$\alpha_3' : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4' : p_6 \vee p_8 \vee \ell_4$
$\alpha_5' : p_2 \vee p_5 \vee a_2$
$\alpha_6' : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



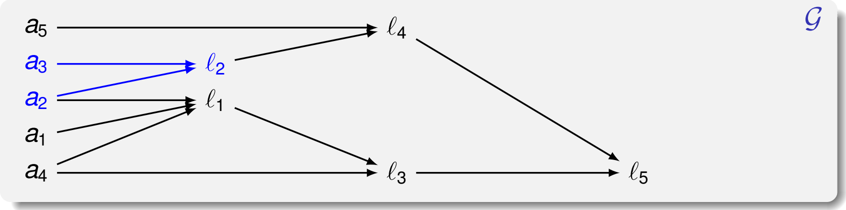
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



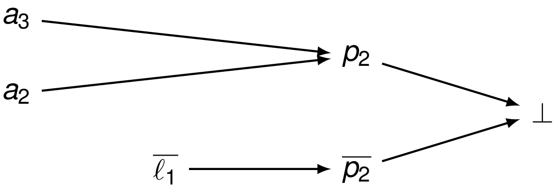
factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



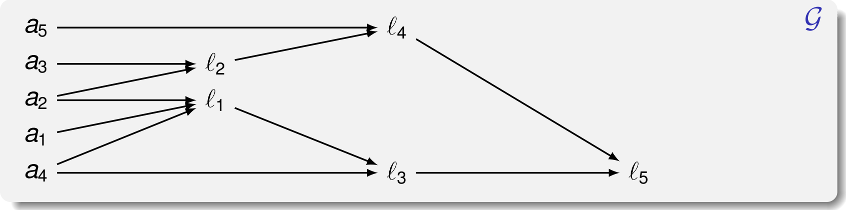
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



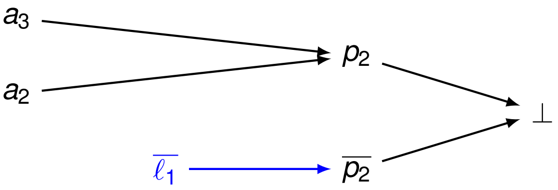
factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



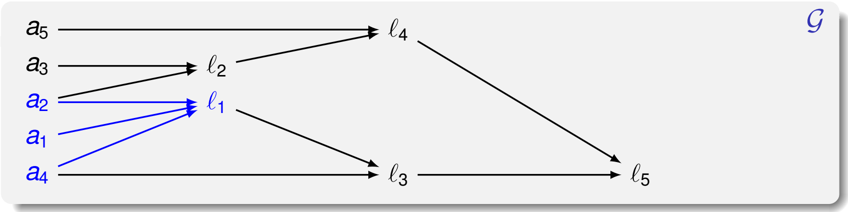
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



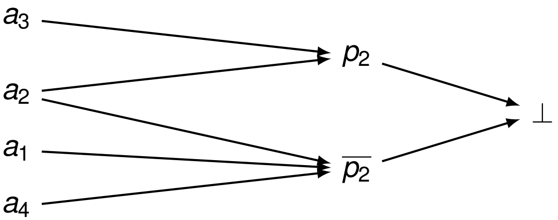
factored clauses

$\alpha_1' : p_2 \vee p_7 \vee \ell_1$
$\alpha_2' : p_2 \vee \ell_2$
$\alpha_3' : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha_4' : p_6 \vee p_8 \vee \ell_4$
$\alpha_5' : p_2 \vee p_5 \vee a_2$
$\alpha_6' : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



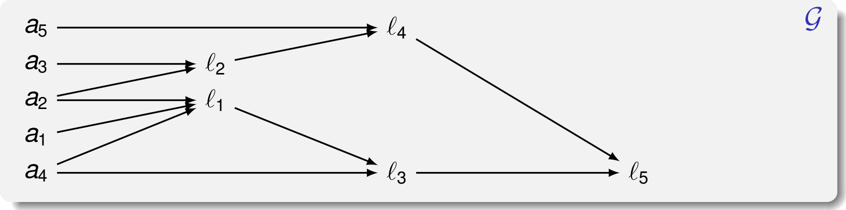
Assumptions Core Analysis

Under assumptions $\{\neg a_1, \neg a_2, \neg a_3, \neg a_4, \neg a_5, \neg a_6, \ldots\}$



factored clauses

$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$



Experiments: MUS Competition

- 300 instances from the MUS competition 2011
- Timeout limited to 1800 seconds
- Memory limited to 7800 Mo

- Use of the MUS extractor MUSer.2
  - → default options (destructive + model rotation)
  - → use of **MINISAT** solver
- Plug our approach MINISAT+abr to MUSer.2

- Intel® Core™2 Quad Processor Q9550 with 2.83 GHz CPU frequency with 8 GB memory and running Ubuntu 12.04

Experiments: Factoring Out Assumptions

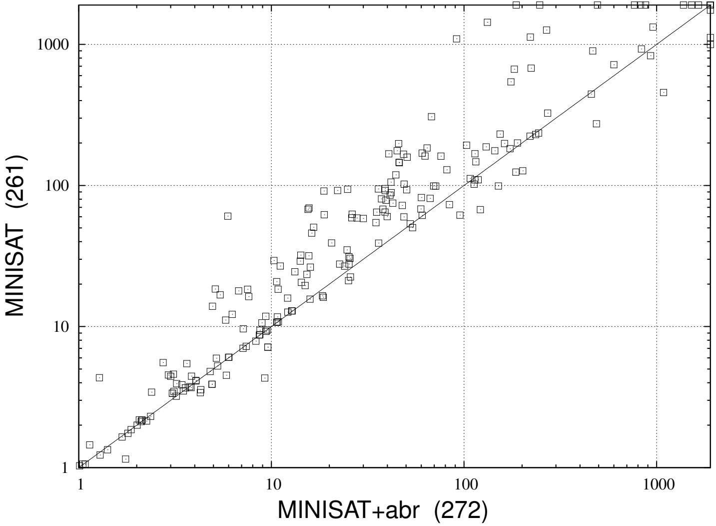
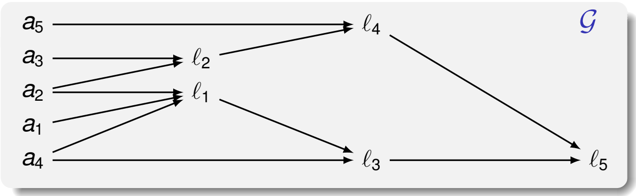


Figure: Running time

Reduce learned clause database

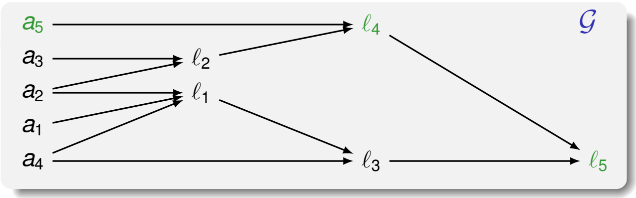
- Keeping all learned clauses **slows down** the solver
- Determining which learned clauses to keep is **essential**
- What are the necessary clauses **to prove the inconsistancy**?
  - $\rightarrow$ **use abbreviation** information to refine the approximation



$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$
$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$
$\alpha'_4 : p_6 \vee p_8 \vee \ell_4$
$\alpha'_5 : p_2 \vee p_5 \vee a_2$
$\alpha'_6 : p_7 \vee p_4 \vee \ell_5$
$\alpha_7 : \overline{p_2} \vee \ell_1$

Reduce learned clause database

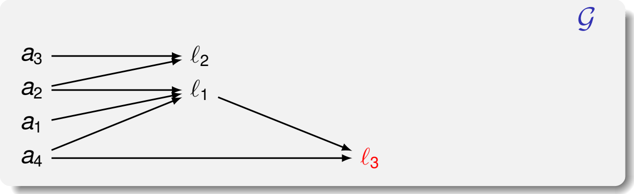
- Keeping all learned clauses **slows down** the solver
- Determining which learned clauses to keep is **essential**
- What are the necessary clauses **to prove the inconsistancy**?
  - $\rightarrow$ **use abbreviation** information to refine the approximation



$$
\begin{aligned}
\alpha'_1 &: p_2 \vee p_7 \vee \ell_1 \\
\alpha'_2 &: p_2 \vee \ell_2 \\
\alpha'_3 &: p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3 \\
\alpha'_4 &: p_6 \vee p_8 \vee \ell_4 \\
\alpha'_5 &: p_2 \vee p_5 \vee a_2 \\
\alpha'_6 &: p_7 \vee p_4 \vee \ell_5 \\
\alpha_7 &: \overline{p_2} \vee \ell_1
\end{aligned}
$$

Reduce learned clause database

- Keeping all learned clauses **slows down** the solver
- Determining which learned clauses to keep is **essential**
- What are the necessary clauses **to prove the inconsistancy**?
  - $\rightarrow$ **use abbreviation** information to refine the approximation



$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$

$\alpha'_2 : p_2 \vee \ell_2$

$\alpha'_3 : p_7 \vee p_4 \vee \overline{p_6} \vee \ell_3$

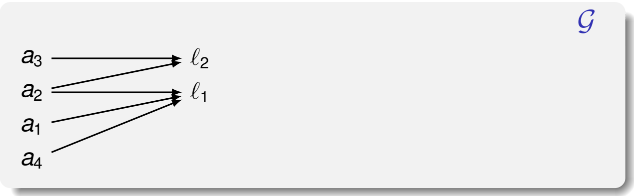
$\alpha'_5 : p_2 \vee p_5 \vee a_2$

$\alpha_7 : \overline{p_2} \vee \ell_1$

Reduce learned clause database

- Keeping all learned clauses **slows down** the solver
- Determining which learned clauses to keep is **essential**
- What are the necessary clauses **to prove the inconsistancy**?

  $\rightarrow$ **use abbreviation** information to refine the approximation



$\alpha'_1 : p_2 \vee p_7 \vee \ell_1$
$\alpha'_2 : p_2 \vee \ell_2$

$\alpha'_5 : p_2 \vee p_5 \vee a_2$

$\alpha_7 : \overline{p_2} \vee \ell_1$

Experiments

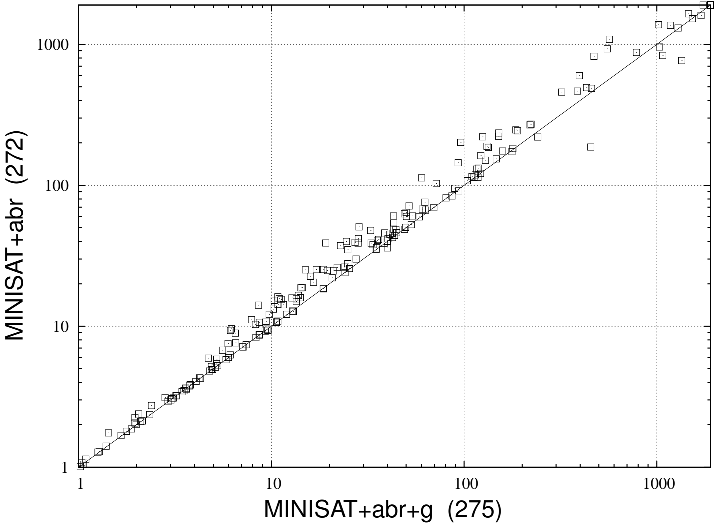


Figure: Running time

Minimization of the learned clauses

- Learned clauses can be minimized: **recursive minimization**
- Clause minimization usually improves SAT solver performance

- With **many assumptions**, clause minimization is not effective
  - $\rightarrow$ assumptions are not obtained by unit propagation
  - $\rightarrow$ non-assumption literals are often blocked by assumptions
  - $\rightarrow$ the number of deleted literals is rather small

- **Ignoring** assumptions during the minimization step
  - $\rightarrow$ the resulting “minimized” clause might even increase in size
  - $\rightarrow$ no more non-assumption literals than the original clause

	MINISAT	MINISAT+abr	MINISAT+abr+g
	#solved(MO)	#solved(MO)	#solved(MO)
without	259(15)	272(3)	273(3)
classic	261(13)	272(3)	275(1)
full	238(25)	276(0)	**281(0)**

Conclusion and perspectives

- Introduction of the factoring out assumptions in the context of incremental SAT solving under assumptions: **MINISAT+abr**
  - → techniques that work well for a large number of assumptions
  - → improve the speed of the BCP procedure
- Additionnal information collected from the definition map to reduce the learned clause database
- Application of new form of clause minimization
- Good results when our approach is combined with MUSer.2

- Combine our techniques with more recent results on MUS preprocessing (inprocessing)
- Apply our approach to high-level MUS extraction
- Improve the data structure used to save the definition map


QUESTIONS ?

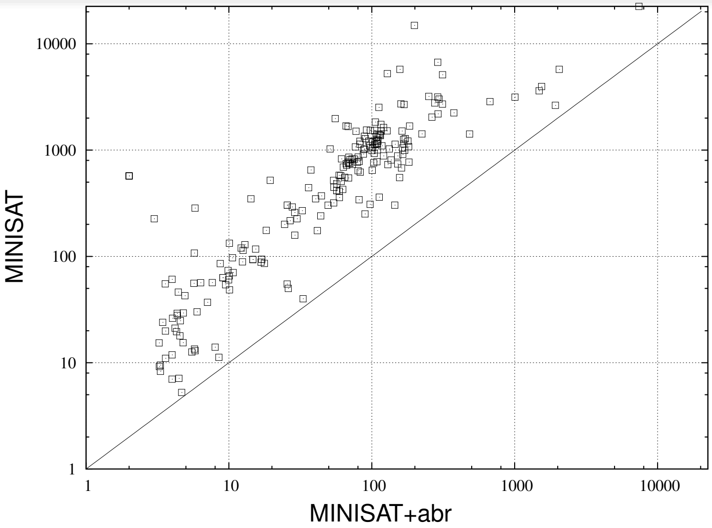


Figure: Average size of learned clauses

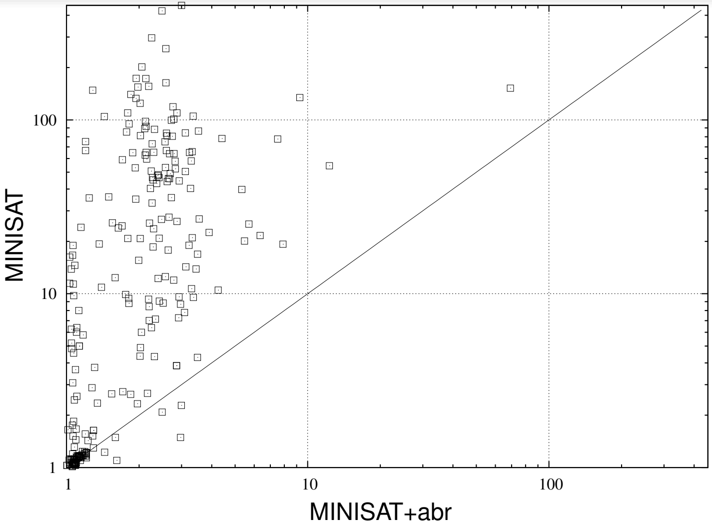


Figure: Average number of traversed literals

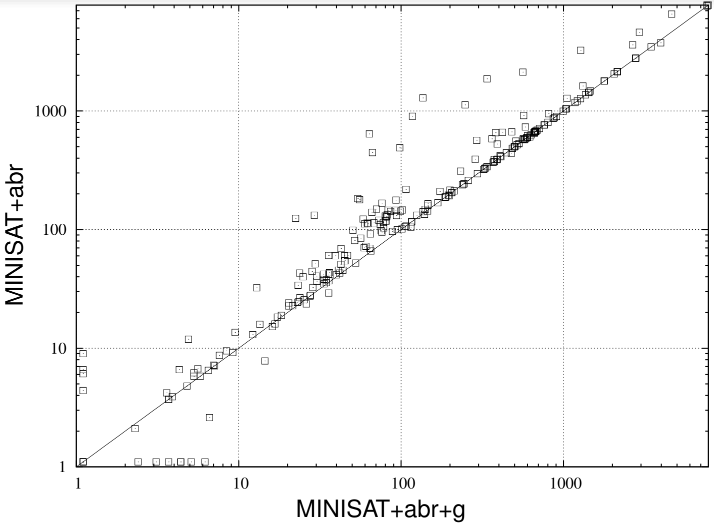


Figure: Memory used (in Mega Bytes)

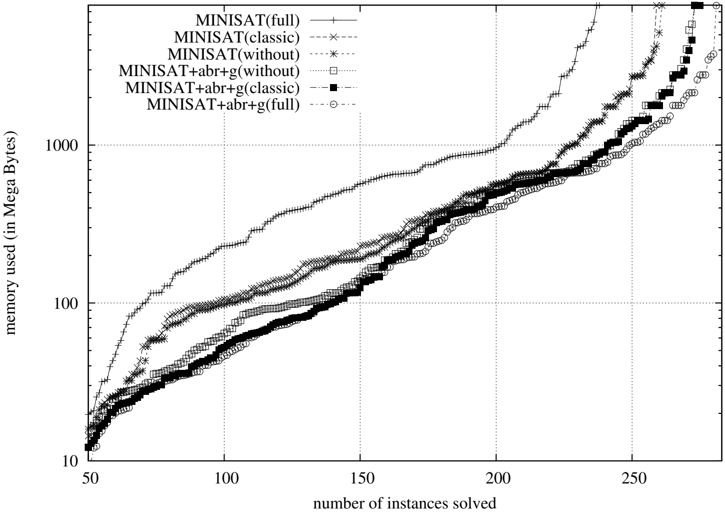


Figure: Memory usage of MUSer