On the proof complexity of MCSAT
MCSAT vs. Res*(T) vs. CDCL(T)

Gereon Kremer, Erika Ábrahám, Vijay Ganesh
July 10th, 2019 – SC² Workshop 2019 – University of Bern
Satisfiability problem (for first-order logic)

Is an existentially quantified first-order formula $\varphi$ valid?

$$\exists x. \varphi(x) \equiv true$$

Applications:
- Software verification, test-case generation
- Termination proving
- Controller synthesis
- Scheduling and planning
- Product design automation
- And growing...
Satisfiability problem (for first-order logic)

Is an existentially quantified first-order formula $\varphi$ valid?

$$\exists x. \varphi(x) \equiv true$$

Applications:
- Software verification, test-case generation
- Termination proving
- Controller synthesis
- Scheduling and planning
- Product design automation
- And growing ...
Proof systems

Definition (Proof rule and proof systems)

Proof rule:
\[
\frac{A_1 \ldots A_n}{C_1 \ldots C_n} \quad \text{if } S_1, \ldots, S_m
\]

Proof system: set of proof rules

Example: resolution proof system

Resolution:
\[
\frac{(C \lor l) \quad (D \lor \neg l)}{(C \lor D)} \quad \text{if } \text{true}
\]

On the proof complexity of MCSAT: MCSAT vs. Res*(T) vs. CDCL(T)
Proofs

Let us prove \((a \lor b \lor \neg c) \land (a \lor \neg b) \land (a \lor c) \land (\neg a) \equiv \square\)
Let us prove \((a \lor b \lor \neg c) \land (a \lor \neg b) \land (a \lor c) \land (\neg a) \equiv \square\)

\[
\frac{(a \lor b \lor \neg c) \quad (a \lor \neg b)}{(a \lor \neg c) \quad (a \lor c)}\quad \frac{(a)}{(\neg a)}
\]

\[
\square
\]
Let us prove \((a \lor b \lor \neg c) \land (a \lor \neg b) \land (a \lor c) \land (\neg a) \equiv \square\)

\[
\frac{(a \lor b \lor \neg c) \quad (a \lor \neg b)}{(a \lor \neg c) \quad (a \lor c)}
\frac{(a) \quad (a \lor c)}{(\neg a)}
\square
\]

**Definition (Proof size and proof complexity)**

**Proof size:** number of proof rule applications.
Proofs

Let us prove \((a \lor b \lor \neg c) \land (a \lor \neg b) \land (a \lor c) \land (\neg a) \equiv \square\)

\[
\frac{(a \lor b \lor \neg c) \quad (a \lor \neg b)}{(a \lor \neg c) \quad (a \lor c)} \quad \frac{(a)}{(a \lor c) \quad (a)}
\]

\[
\frac{(a \lor \neg c) \quad (a \lor c)}{(a) \quad (\neg a)} \quad \square
\]

Definition (Proof size and proof complexity)

**Proof size:** number of proof rule applications.

**Proof complexity:** asymptotic proof size of the shortest proof.
Let us prove \((a \lor b \lor \neg c) \land (a \lor \neg b) \land (a \lor c) \land (\neg a) \equiv \square\)

\[
\begin{array}{c}
(a \lor b \lor \neg c) \quad (a \lor \neg b) \\
\hline
(a \lor \neg c) \\
\hline
(a) \\
\hline
(a \lor c) \\
\hline
(\neg a) \\
\hline
\square
\end{array}
\]

**Definition (Proof size and proof complexity)**

**Proof size:** number of proof rule applications.

**Proof complexity:** asymptotic proof size of the shortest proof.

**Assumption:** every rule costs the same.
MCSAT proof system – overview

Three groups of proof rules:

- **Search**: CDCL-style SAT solving.
- **Conflict**: CDCL-style conflict resolution.
- **Theory**: adds theory reasoning.

*de Moura and Jovanović (2013)*

- Decide
- Propagate
- Conflict
- Resolve
- Backjump
- Learn
- T-Propagate
- T-Decision
- T-Conflict

Important concepts:

- **Theory decisions**: like Boolean decisions, but for theory variables.
- **Trail**: $J \mid M, l_1, l_2, C \alpha x_1, \ldots, K$
- **States**: $x_M, \alpha C y$ and $x_M, \alpha C y, C$ (initially $x_J K, C y$)
MCSAT proof system – overview

Three groups of proof rules:

- **Search**: CDCL-style SAT solving.
- **Conflict**: CDCL-style conflict resolution.
- **Theory**: adds theory reasoning.

Important concepts:

- **Theory decisions**: like Boolean decisions, but for theory variables.
- **Trail**: $[M, l_1, \neg l_2, C \rightarrow l, x \leftarrow \alpha_x, \ldots]$.
- **States**: $\langle M, C \rangle$ and $\langle M, C \rangle \models C$ (initially $\langle [], C \rangle$).
- **value($l$)**: assigned by Boolean or theory model.

*de Moura and Jovanović (2013)*

- Decide
- Propagate
- Conflict
- Resolve
- Backjump
- Learn
- T-Propagate
- T-Decide
- T-Conflict
MCSAT proof system – search rules

Decide: \[ \frac{\langle M, C \rangle}{\langle [M, l], C \rangle} \]
if \( l \) is unassigned

Propagate: \[ \frac{\langle M, C \rangle}{\langle [M, C \rightarrow l], C \rangle} \]
if \( C \) is unit and implies \( l \)

Conflict: \[ \frac{\langle M, C \rangle}{\langle M, C \rangle \vdash C} \]
if \( C \in C \) is conflicting

Sat: \[ \frac{\langle M, C \rangle}{SAT} \]
if \( M \) is complete and satisfies \( C \)

Forget: \[ \frac{\langle M, C \rangle}{\langle M, C \setminus \{C\} \rangle} \]
if \( C \) is a learned clause
MCSAT proof system – conflict rules

Resolve: \[ \frac{\langle [M, D \rightarrow l], \mathcal{C} \rangle \models C}{\langle M, \mathcal{C} \rangle \models R} \]

if \( R = \text{resolve}(C, D, l) \)

Consume: \[ \frac{\langle [M, l \text{ or } D \rightarrow l], \mathcal{C} \rangle \models C}{\langle M, \mathcal{C} \rangle \models C} \]

if \( -l \notin C \)

Backjump: \[ \frac{\langle [M, N], \mathcal{C} \rangle \models C}{\langle [M, C \rightarrow l], \mathcal{C} \rangle} \]

if \( C \) is unit on \( M \) and \( N \) starts with a decision

Unsat: \[ \frac{\langle M, \mathcal{C} \rangle \models \text{false}}{\text{UNSAT}} \]

if \( \text{true} \)

Learn: \[ \frac{\langle M, \mathcal{C} \rangle \models C}{\langle M, \mathcal{C} \cup \{C\} \rangle \models C} \]

if \( C \) is a new clause

Restart: \[ \frac{\langle M, \mathcal{C} \rangle \models C}{\langle [], \mathcal{C} \rangle} \]

if \( \text{true} \)
infeasible($M$): checks whether $M$ can be extended to a full model.
explain($M$) $\leftrightarrow$ $C$: clause $C$ excludes a region around $M$. 

***: Terms and conditions may apply.
infeasible(M): checks whether $M$ can be extended to a full model.

explain(M) $\iff C$: clause $C$ excludes a region around $M$.

$$C = (x \leq -2 \lor x \geq 2 \lor y \leq l(x) \lor y \geq u(x))$$
infeasible($M$): checks whether $M$ can be extended to a full model.

explain($M$) $\hookrightarrow C$: clause $C$ excludes a region around $M$.

\[ \text{T-Propagate:} \frac{\langle M, C \rangle}{\langle [M, E \rightarrow l], C \rangle} \]

if infeasible([$M, \neg l$]) and $E = \text{explain}([M, \neg l])$

\[ \frac{\langle M, C \rangle}{\langle [M, x \mapsto \alpha_x], C \rangle} \]

if $x$ is unassigned and $[M, x \mapsto \alpha_x]$ is consistent

\[ \frac{\langle M, C \rangle}{\langle [M, x \mapsto \alpha_x], C \rangle} \]

if infeasible($M$) and $E = \text{explain}(M)$

\[ \frac{\langle [M, x \mapsto \alpha_x], C \rangle}{\langle M, C \rangle} \]

if $C$ is infeasible on $M$

\[ \frac{\langle [M, x \mapsto \alpha_x, N], C \rangle}{\langle [M, l], C \rangle} \]

if Backtracking $x \mapsto \alpha_x$ “unassigns” multiple literals from $C$ at once

***: Terms and conditions may apply.
Res*(T) proof system

Resolution: \[ \frac{(C \lor l) \land (D \lor \neg l)}{(C \lor D)} \] if true

(Regular) Theory Derivation: \[ \frac{\varphi}{\varphi \land C} \] if \( T \models C \), \( l \in \varphi \) for all \( l \in C \)

Strong Theory Derivation: \[ \frac{\varphi}{\varphi \land C} \] if \( T \models C \)

- SAT if no new clause can be generated by Resolution or Regular Theory Derivation.
- UNSAT if \( \square \) was generated.
Res*(T) proof system

Resolution: \[
\frac{(C \lor l) \lor (D \lor \neg l)}{(C \lor D)} \quad \text{if true}
\]

(Regular) Theory Derivation: \[
\frac{\varphi}{\varphi \land C} \quad \text{if } T \models C, l \in \varphi \text{ for all } l \in C
\]

Strong Theory Derivation: \[
\frac{\varphi}{\varphi \land C} \quad \text{if } T \models C
\]

- **SAT** if no new clause can be generated by Resolution or Regular Theory Derivation.
- **UNSAT** if \( \square \) was generated.

We use Strong Theory Derivation!
Recall: Proof size and proof complexity

Proof size: number of proof rule applications.
Proof complexity: asymptotic proof size, depending on formula size.
Recall: Proof size and proof complexity

Proof size: number of proof rule applications.
Proof complexity: asymptotic proof size, depending on formula size.

Definition ($P_1$ simulates $P_2$)
Proof complexity of $P_1$ is at most polynomially larger for all inputs.

Definition ($P_2$ is $P_1$ derivable)
$P_1$ can simulate every rule of $P_2$ individually.
Recall: Proof size and proof complexity

**Proof size:** number of proof rule applications.
**Proof complexity:** asymptotic proof size, depending on formula size.

Definition ($P_1$ simulates $P_2$)

Proof complexity of $P_1$ is at most polynomially larger for all inputs.

Definition ($P_2$ is $P_1$ derivable)

$P_1$ can simulate every rule of $P_2$ individually.

Example: Res*(T) simulates CDCL(T). *Robere et al. (2018)*
Theorem

The Res*(T) proof system and the MCSAT proof system are bisimilar with respect to their proof complexity on first-order logic with any theory.

We show: MCSAT is Res*(T) derivable and Res*(T) is MCSAT derivable.
Theorem

The Res*(T) proof system and the MCSAT proof system are bisimilar with respect to their proof complexity on first-order logic with any theory.

We show: MCSAT is Res*(T) derivable and Res*(T) is MCSAT derivable.

Note that we actually show a slightly stronger statement:
MCSAT and Res*(T) are not only bisimilar but “algorithmically equivalent”.
Resolution of \((C \lor l) \land (D \lor \neg l)\):

- Decide literals of \(C\) and \(D\) to \textit{false}.
- Propagate \((C \lor l)\).
- Use \((D \lor \neg l)\) for Conflict.
- Apply Resolve.
- Learn the clause and Restart.
MCSAT simulates Res*(T)

Resolution of \((C \lor l) \land (D \lor \neg l)\):

- Decide literals of \(C\) and \(D\) to \(false\).
- Propagate \((C \lor l)\).
- Use \((D \lor \neg l)\) for Conflict.
- Apply Resolve.

Learn the clause and Restart.

Let \(\mathcal{C} = \{(a \lor l), (b \lor \neg l)\}\).

\[
\begin{align*}
\langle \text{[]} \rangle, \mathcal{C} & \\
\langle \neg a, \neg b, (a \lor l) \rightarrow l \rangle, \mathcal{C} & \\
\langle \neg a, \neg b, (a \lor l) \rightarrow l \rangle, \mathcal{C} \models (b \lor \neg l) & \\
\langle \neg a, \neg b \rangle, \mathcal{C} \models (a \lor b) & \\
\langle \neg a, \neg b \rangle, \mathcal{C} \cup \{a \lor b\} \models (a \lor b) & \\
\langle \text{[]} \rangle, \mathcal{C} \cup \{a \lor b\} &
\end{align*}
\]
Resolution of \((C \lor l) \land (D \lor \lnot l)\):

- Decide literals of \(C\) and \(D\) to \(false\).
- Propagate \(C \lor l\).
- Use \((D \lor \lnot l)\) for Conflict.
- Apply Resolve.
- Learn the clause and Restart.

**Strong Theory Derivation** of some clause \(C\):

- Decide all literals of \(C\) to \(false\).
- Apply T-Conflict to obtain \(C\).
- Learn the clause and Restart.
MCSAT simulates $\text{Res}^*(T)$

Resolution of $(C \lor l) \land (D \lor \neg l)$:

- Decide literals of $C$ and $D$ to false.
- Propagate $(C \lor l)$.
- Use $(D \lor \neg l)$ for Conflict.
- Apply Resolve.
- Learn the clause and Restart.

Strong Theory Derivation of some clause $C$:

- Decide all literals of $C$ to false.
- Apply T-Conflict to obtain $C$.
- Learn the clause and Restart.

\[
\begin{align*}
\langle [], C \rangle &\quad \langle [x < 0, x > 1], C \rangle \\
\langle [x < 0, x > 1], C \rangle &\vdash (x \geq 0 \lor x \leq 1) \\
\langle [], C \cup \{x \geq 0 \lor x \leq 1\} \rangle
\end{align*}
\]
MCSAT simulates Res*(T)

Resolution of \((C \lor l) \land (D \lor \neg l)\):

- Decide literals of \(C\) and \(D\) to \textit{false}.
- Propagate \((C \lor l)\).
- Use \((D \lor \neg l)\) for Conflict.
- Apply Resolve.
- Learn the clause and Restart.

Strong Theory Derivation of some clause \(C\):

- Decide all literals of \(C\) to \textit{false}.
- Apply T–Conflict to obtain \(C\).
- Learn the clause and Restart.

Theory reasoning: T–Conflict checks infeasibility with infeasible. ***
Observation

All clauses in MCSAT “live” at: $C$, $M$, conflict clause $C$.
We only need to simulate rules that create completely new clauses: Resolve, T-Propagate and T-Conflict.

All other rules do not manipulate clauses or only move them around.
Observation

All clauses in MCSAT “live” at: $C, M$, conflict clause $C$.
We only need to simulate rules that create completely new clauses: Resolve, T-Propagate and T-Conflict.

All other rules do not manipulate clauses or only move them around.

- **Resolve**: essentially identical to Resolution.
- **T-Propagate and T-Conflict**: use explain to generate “a valid theory lemma”, we can use Strong Theory Derivation.
Some observations

- All the reductions are polynomial.
Some observations

- All the reductions are **polynomial**.
- Theory decisions completely **irrelevant** (for the proof).
Some observations

- All the reductions are *polynomial*.

- Theory decisions completely *irrelevant* (for the proof).

- Terms and Conditions:
  We assumed *infeasible* to be *complete*, though it is not in practice. *Incomplete infeasible* needs theory exploration (may be *exponential*).
Some observations

- All the reductions are polynomial.
- Theory decisions completely irrelevant (for the proof).
- Terms and Conditions:
  We assumed infeasible to be complete, though it is not in practice. Incomplete infeasible needs theory exploration (may be exponential).
  
  What did you expect?
  MCSAT moves theory reasoning into the proof system.
  The cost is not new, but only made explicit.
  Is this a problem?
What about CDCL(T)?

- Literature: $\text{Res}^*(T)$ and CDCL(T) are bisimilar. $\text{Robere et al. (2018)}$
- Now: $\text{Res}^*(T)$ and MCSAT are bisimilar.
- $\Rightarrow$ MCSAT and CDCL(T) are bisimilar ...

On the proof complexity of MCSAT: MCSAT vs. $\text{Res}^*(T)$ vs. CDCL(T)
What about CDCL(T)?

- Literature: Res*(T) and CDCL(T) are bisimilar. \textit{Robere et al. (2018)}
- Now: Res*(T) and MCSAT are bisimilar.
- \[ \implies \text{MCSAT and CDCL(T) are bisimilar … with respect to proof complexity.} \]
What about CDCL(T)?

- Literature: Res*(T) and CDCL(T) are bisimilar. \textit{Robere et al. (2018)}
- Now: Res*(T) and MCSAT are bisimilar.
- $\Rightarrow$ MCSAT and CDCL(T) are bisimilar ... with respect to proof complexity.
- Also: Res*(T) and MCSAT are “algorithmically equivalent”. ***
- What about MCSAT and CDCL(T)?
What about CDCL(T)?

- Literature: Res*(T) and CDCL(T) are bisimilar. \textit{Robere et al. (2018)}
- Now: Res*(T) and MCSAT are bisimilar.
- \Rightarrow MCSAT and CDCL(T) are bisimilar … with respect to \textit{proof complexity}.

- Also: Res*(T) and MCSAT are “algorithmically equivalent”. ***
- What about MCSAT and CDCL(T)?

- We claim: MCSAT and CDCL(T) are “algorithmically equivalent”. ***
With respect to proof complexity,

- Res*(T) and CDCL(T) are bisimilar, \( \text{Robere et al. (2018)} \)

- Res*(T) and MCSAT are bisimilar,

- thus CDCL(T) and MCSAT are bisimilar.
With respect to proof complexity,

- Res*(T) and CDCL(T) are bisimilar, \cite{Robere et al. (2018)}
- Res*(T) and MCSAT are bisimilar,
- thus CDCL(T) and MCSAT are bisimilar.

We have seen that

- theory decisions are “only” a heuristic,
- the MCSAT proof system is more powerful than any implementation,
- Res*(T) and MCSAT perform roughly the same theory reasoning.
With respect to proof complexity,

- Res*(T) and CDCL(T) are bisimilar, \cite{Robere2018}
- Res*(T) and MCSAT are bisimilar,
- thus CDCL(T) and MCSAT are bisimilar.

We have seen that

- theory decisions are “only” a heuristic,
- the MCSAT proof system is more powerful than any implementation,
- Res*(T) and MCSAT perform roughly the same theory reasoning.

We conjecture “algorithmic equivalency” of CDCL(T) and MCSAT.
References
