SAT-based Simplification Inferences in First-Order Theorem Proving

SMITE Seminar, Belo Horizonte, Brazil

Robin Coutelier

robin.coutelier@tuwien.ac.at
TU Wien, Vienna, Austria

06 November 2026



Acknowledgements

This presentation is a modernized and merged version of the presentations given at the *Workshop on Alignment of Proof Systems and Machine Learning* (EuroProofNet 2024) in Vienna, Austria and the original CADE 2023 presentation [Coutelier et al., 2023]. It contains content from the following papers:

- [Rath et al., 2022] - First-Order Subsumption via SAT solving
- [Coutelier et al., 2023] - SAT-based Subsumption Resolution
- [Coutelier et al., 2024] - SAT Solving for Variants of First-Order Subsumption

The author acknowledge support from the ERC Consolidator Grant ARTIST 101002685; the TU Wien Doctoral College TrustACPS; the FWF SpyCoDe SFB projects F8504; the WWTF Grant ForSmart 10.47379/ICT22007



European Research Council
Established by the European Commission



Der Wissenschaftsfonds.









Why Should SMT People Care About Saturation?

SMT

- ✓ Theories are native
- × Quantifiers: guess ground instances (triggers, MBQI)
- × Matching loops, brittleness

Saturation

- ✓ Quantifiers are native (unification)
- × Theories are harder
- × Clause explosion

Why Should SMT People Care About Saturation?

SMT

- ✓ Theories are native
- × Quantifiers: guess ground instances (triggers, MBQI)
- × Matching loops, brittleness

Saturation

- ✓ Quantifiers are native (unification)
- × Theories are harder
- × Clause explosion

This talk

Keeping the clause set small with *SAT-based* subsumption.

A Small Example

Groups with left axioms only

$\forall x, y, z.\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$	associativity
$\forall x.\ e \cdot x = x$	left identity
$\forall x.\ x^{-1} \cdot x = e$	left inverse
$\vdash\ a \cdot e = a$	right identity?

A Small Example

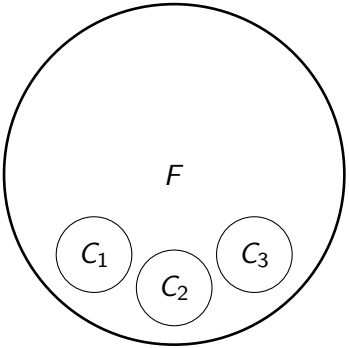
Groups with left axioms only

$\forall x, y, z.\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$	associativity
$\forall x.\ e \cdot x = x$	left identity
$\forall x.\ x^{-1} \cdot x = e$	left inverse
$\vdash\ a \cdot e = a$	right identity?

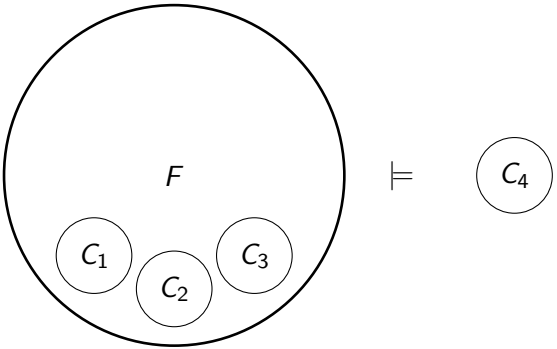
Results (`UF`, 30 s timeout)

Solver	**Result**	**Time**
Z3 4.15.3	timeout	—
cvc5 1.3 (default)	unknown	0.02 s
cvc5 (`--enum-inst`, `--mbqi`, ...)	timeout	—
Vampire 5.0	**unsat**	0.02 s

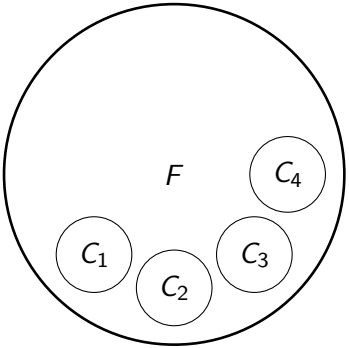
Saturation in FOL Theorem Proving



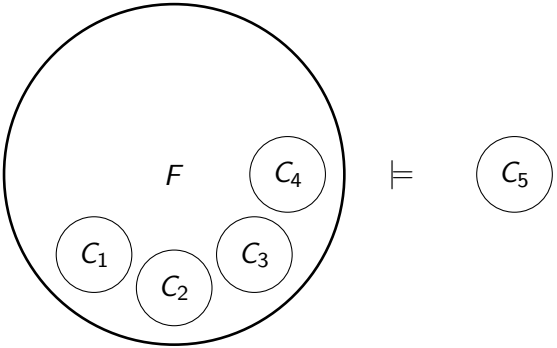
Saturation in FOL Theorem Proving



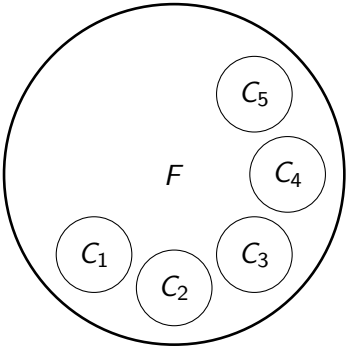
Saturation in FOL Theorem Proving



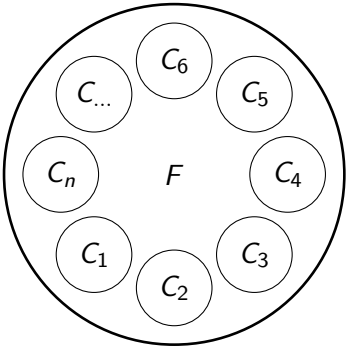
Saturation in FOL Theorem Proving



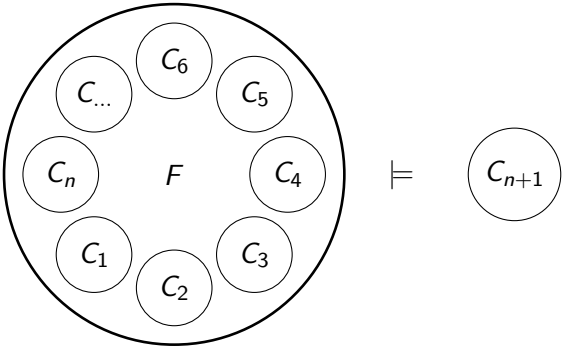
Saturation in FOL Theorem Proving



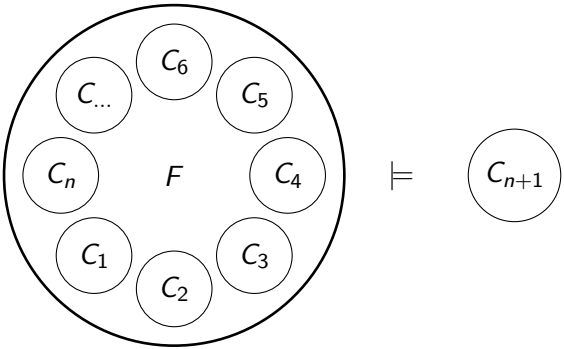
Saturation in FOL Theorem Proving



Saturation in FOL Theorem Proving



Saturation in FOL Theorem Proving



Out of memory!

Notations

Variables x, y, z are universally and implicitly quantified.

Example

$$\big[p(x) \vee q(x, f(a, y))\big] \wedge \big[p(g(x))\big]$$

is actually

$$\forall x, y.\ \big[p(x) \vee q(x, f(a, y))\big] \wedge \forall x.\ \big[p(g(x))\big]$$

Subsumption

Definition

A clause S *subsumes* a distinct clause M iff there is a substitution σ such that

$$\sigma(S) \sqsubseteq M$$

where $\sqsubseteq$ is the sub-multiset inclusion relation.

If S subsumes M, then M is redundant and can be removed from the formula.

Subsumption - Examples

Example (propositional logic)

$$S = a \vee b$$
$$M = a \vee b \vee c$$

S subsumes M. It is "stronger" than M.

Subsumption - Examples

Example (propositional logic)

$$S = a \vee b$$
$$M = a \vee b \vee c$$

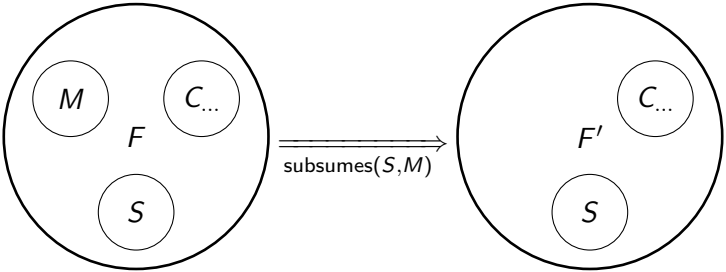
S subsumes M. It is "stronger" than M.

Example (FOL)

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(c), d) \vee p(f(y), c) \vee p(f(c), g(d))$$

S subsumes M with the substitution $\sigma = \{x_1 \mapsto f(y), x_2 \mapsto c, x_3 \mapsto g(d)\}$.

Subsumption - Intuition



Subsumption Resolution

Resolution (Simplified)

$$\frac{S^* \vee s' \qquad \neg\sigma(s') \vee M^*}{\sigma(S^*) \vee M^*}$$

Subsumption Resolution

Resolution (Simplified)

$$\frac{S^* \vee s' \qquad \neg\sigma(s') \vee M^*}{\sigma(S^*) \vee M^*}$$

Definition

Clauses M and S are said to be the main and side premise of subsumption resolution, respectively, iff there is a substitution σ, a set of literals $S' \subseteq S$ and a literal $m' \in M$ such that

$$\sigma(S') = \{\neg m'\} \quad \text{and} \quad \sigma(S \setminus S') \subseteq M \setminus \{m'\}.$$

Subsumption Resolution aims to remove a literal from the main premise.

Subsumption Resolution - Example 1

Example (propositional logic)

$$\frac{S := \boxed{a} \vee b \qquad M := \boxed{\neg a} \vee b \vee c}{M^* := b \vee c}$$

$\neg a$ is the resolution literal. M^* subsumes M and can replace M in the clause set.

Subsumption Resolution - Example 1

Example (propositional logic)

$$\frac{S := \boxed{a} \vee b \qquad M := \cancel{\boxed{\neg a} \vee b \vee c}}{M^* := b \vee c}$$

$\neg a$ is the resolution literal. M^* subsumes M and can replace M in the clause set.

Subsumption Resolution - Example 2

Example (FOL)

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$
$$\sigma = \{x_1 \mapsto g(y), x_2 \mapsto c, x_3 \mapsto e\}$$

Subsumption Resolution - Example 2

Example (FOL)

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$
$$\sigma = \{x_1 \mapsto g(y), x_2 \mapsto c, x_3 \mapsto e\}$$

$$\frac{\dfrac{p(x_1, x_2) \vee p(f(x_2), x_3)}{p(g(y), c) \vee \boxed{p(f(c), e)}} \qquad \neg p(f(y), d) \vee p(g(y), c) \vee \boxed{\neg p(f(c), e)}}{M^* := \neg p(f(y), d) \vee p(g(y), c)}$$

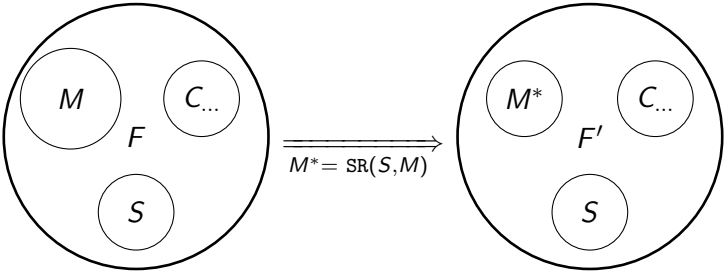
Subsumption Resolution - Example 2

Example (FOL)

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$
$$\sigma = \{x_1 \mapsto g(y), x_2 \mapsto c, x_3 \mapsto e\}$$

$$\frac{p(x_1, x_2) \vee p(f(x_2), x_3) \qquad \neg p(f(y), d) \vee p(g(y), c) \vee \boxed{\neg p(f(c), e)}}{p(g(y), c) \vee \boxed{p(f(c), e)} \qquad M^* := \neg p(f(y), d) \vee p(g(y), c)}$$

Subsumption Resolution - Intuition



Importance of Redundancy Elimination

```
$ vampire Problems/GRP/GRP140-1.p -fsr off -t 30
...
132544.  $ false
% Termination reason:  Refutation
% Memory used [KB]: 308054
% Time elapsed:  6.654 s
------------------------------------------------------------
```

Importance of Redundancy Elimination

```
$ vampire Problems/GRP/GRP140-1.p -fsr off -t 30
...
132544.  $ false
% Termination reason:  Refutation
% Memory used [KB]: 308054
% Time elapsed:  6.654 s
-------------------------------------------------------------
$ vampire Problems/GRP/GRP140-1.p -fsr on -t 30
...
4918.  $ false
% Termination reason:  Refutation
% Memory used [KB]: 12025
% Time elapsed:  0.150 s
```

Relevance of Speed

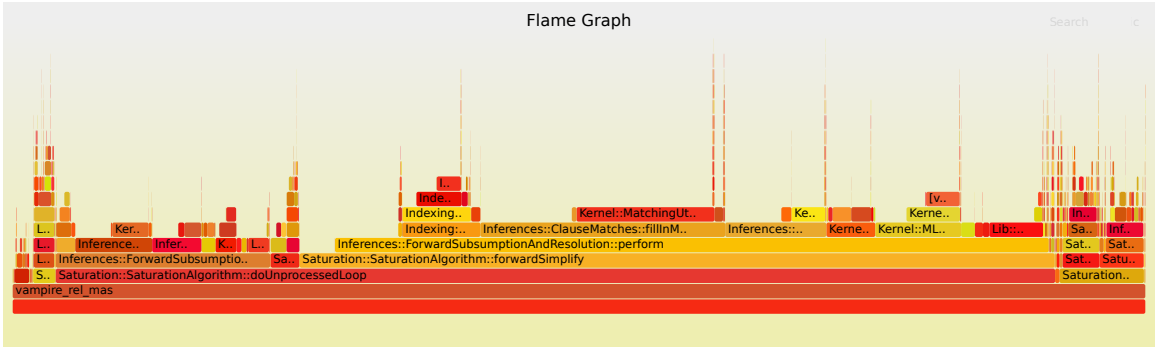


Figure: Typical profiling results for a TPTP problem (GRP001+6).

From Definition to Constraints

Definition (Subsumption Resolution (Reminder))

Clauses M and S are the main and side premise of SR, respectively, iff there is a substitution σ, a set of literals $S' \subseteq S$ and a literal $m' \in M$ such that

$$\sigma(S') = \{\neg m'\} \quad \text{and} \quad \sigma(S \setminus S') \subseteq M \setminus \{m'\}.$$

Theorem (Subsumption Resolution Constraints)

Clauses $S = s_1 \vee \cdots \vee s_{|S|}$ and $M = m_1 \vee \cdots \vee m_{|M|}$ apply SR iff there exists a substitution σ that satisfies the following four properties:

existence $\exists i\, j.\, \sigma(s_i) = \neg m_j$

From Definition to Constraints

Definition (Subsumption Resolution (Reminder))

Clauses M and S are the main and side premise of SR, respectively, iff there is a substitution σ, a set of literals $S' \subseteq S$ and a literal $m' \in M$ such that

$$\sigma(S') = \{\neg m'\} \quad \text{and} \quad \sigma(S \setminus S') \subseteq M \setminus \{m'\}.$$

Theorem (Subsumption Resolution Constraints)

Clauses $S = s_1 \vee \cdots \vee s_{|S|}$ and $M = m_1 \vee \cdots \vee m_{|M|}$ apply SR iff there exists a substitution σ that satisfies the following four properties:

existence	$\exists i\, j.\, \sigma(s_i) = \neg m_j$
uniqueness	$\exists j'.\, \forall i\, j.\, \big(\sigma(s_i) = \neg m_j \Rightarrow j = j'\big)$

From Definition to Constraints

Definition (Subsumption Resolution (Reminder))

Clauses M and S are the main and side premise of SR, respectively, iff there is a substitution σ, a set of literals $S' \subseteq S$ and a literal $m' \in M$ such that

$$\sigma(S') = \{\neg m'\} \quad \text{and} \quad \sigma(S \setminus S') \subseteq M \setminus \{m'\}.$$

Theorem (Subsumption Resolution Constraints)

Clauses $S = s_1 \vee \cdots \vee s_{|S|}$ and $M = m_1 \vee \cdots \vee m_{|M|}$ apply SR iff there exists a substitution σ that satisfies the following four properties:

existence	$\exists i\, j.\, \sigma(s_i) = \neg m_j$
uniqueness	$\exists j'.\, \forall i\, j.\, \big(\sigma(s_i) = \neg m_j \Rightarrow j = j'\big)$
completeness	$\forall i.\, \exists j.\, \big(\sigma(s_i) = \neg m_j \vee \sigma(s_i) = m_j\big)$

From Definition to Constraints

Definition (Subsumption Resolution (Reminder))

Clauses M and S are the main and side premise of SR, respectively, iff there is a substitution σ, a set of literals $S' \subseteq S$ and a literal $m' \in M$ such that

$$\sigma(S') = \{\neg m'\} \quad \text{and} \quad \sigma(S \setminus S') \subseteq M \setminus \{m'\}.$$

Theorem (Subsumption Resolution Constraints)

Clauses $S = s_1 \vee \cdots \vee s_{|S|}$ and $M = m_1 \vee \cdots \vee m_{|M|}$ apply SR iff there exists a substitution σ that satisfies the following four properties:

existence	$\exists i\, j.\, \sigma(s_i) = \neg m_j$
uniqueness	$\exists j'.\, \forall i\, j.\, \big(\sigma(s_i) = \neg m_j \Rightarrow j = j'\big)$
completeness	$\forall i.\, \exists j.\, \big(\sigma(s_i) = \neg m_j \vee \sigma(s_i) = m_j\big)$
coherence	$\forall j.\, \big(\exists i.\, \sigma(s_i) = m_j \Rightarrow \forall i.\, \sigma(s_i) \neq \neg m_j\big)$

SAT variables

Let

$$S = s_1 \vee \cdots \vee s_{|S|} \quad M = m_1 \vee \cdots \vee m_{|M|}$$

We define the following SAT variables:

- $b^{+}_{i,j} \Leftrightarrow \sigma(s_i) = m_j$
- $b^{-}_{i,j} \Leftrightarrow \sigma(s_i) = \neg m_j$

$\sigma(m_i) = s_j$ means that the substitution $\sigma_{i,j}$ used to bind m_i to s_j is compatible with the other substitutions.

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^{-}_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^{-}_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$
- $b^{+}_{1,2} \Leftrightarrow \{x_1 \mapsto g(y), x_2 \mapsto c\} \subseteq \sigma$

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^-_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$
- $b^+_{1,2} \Leftrightarrow \{x_1 \mapsto g(y), x_2 \mapsto c\} \subseteq \sigma$
- $b^-_{1,3} \Leftrightarrow \{x_1 \mapsto f(c), x_2 \mapsto e\} \subseteq \sigma$

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^-_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$
- $b^+_{1,2} \Leftrightarrow \{x_1 \mapsto g(y), x_2 \mapsto c\} \subseteq \sigma$
- $b^-_{1,3} \Leftrightarrow \{x_1 \mapsto f(c), x_2 \mapsto e\} \subseteq \sigma$
- $b^-_{2,1} \Leftrightarrow \{x_2 \mapsto y, x_3 \mapsto d\} \subseteq \sigma$

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^-_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$
- $b^+_{1,2} \Leftrightarrow \{x_1 \mapsto g(y), x_2 \mapsto c\} \subseteq \sigma$
- $b^-_{1,3} \Leftrightarrow \{x_1 \mapsto f(c), x_2 \mapsto e\} \subseteq \sigma$
- $b^-_{2,1} \Leftrightarrow \{x_2 \mapsto y, x_3 \mapsto d\} \subseteq \sigma$
- $b^+_{2,2} \Leftrightarrow \{\bot\} \subseteq \sigma$

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^-_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$
- $b^+_{1,2} \Leftrightarrow \{x_1 \mapsto g(y), x_2 \mapsto c\} \subseteq \sigma$
- $b^-_{1,3} \Leftrightarrow \{x_1 \mapsto f(c), x_2 \mapsto e\} \subseteq \sigma$
- $b^-_{2,1} \Leftrightarrow \{x_2 \mapsto y, x_3 \mapsto d\} \subseteq \sigma$
- $b^+_{2,2} \Leftrightarrow \{\bot\} \subseteq \sigma$
- $b^-_{2,3} \Leftrightarrow \{x_2 \mapsto c, x_3 \mapsto e\} \subseteq \sigma$

SAT variables - Example

$$S = p(x_1, x_2) \vee p(f(x_2), x_3)$$
$$M = \neg p(f(y), d) \vee p(g(y), c) \vee \neg p(f(c), e)$$

- $b^-_{1,1} \Leftrightarrow \{x_1 \mapsto f(y), x_2 \mapsto d\} \subseteq \sigma$
- $b^+_{1,2} \Leftrightarrow \{x_1 \mapsto g(y), x_2 \mapsto c\} \subseteq \sigma$
- $b^-_{1,3} \Leftrightarrow \{x_1 \mapsto f(c), x_2 \mapsto e\} \subseteq \sigma$
- $b^-_{2,1} \Leftrightarrow \{x_2 \mapsto y, x_3 \mapsto d\} \subseteq \sigma$
- $b^+_{2,2} \Leftrightarrow \{\bot\} \subseteq \sigma$
- $b^-_{2,3} \Leftrightarrow \{x_2 \mapsto c, x_3 \mapsto e\} \subseteq \sigma$

Constraint to SAT Encoding - Completeness

The constraints can be encoded using a simple procedure.
For example the completeness constraint:

$$\forall i.\, \exists j.\, \left(\sigma(s_i) = \neg m_j \vee \sigma(s_i) = m_j\right)$$

Constraint to SAT Encoding - Completeness

The constraints can be encoded using a simple procedure.
For example the completeness constraint:

$$\forall i.\, \exists j.\, \left(\sigma(s_i) = \neg m_j \vee \sigma(s_i) = m_j\right)$$
$$\forall i.\, \exists j.\, \left(b^{-}_{i,j} \vee b^{+}_{i,j}\right)$$

Constraint to SAT Encoding - Completeness

The constraints can be encoded using a simple procedure.
For example the completeness constraint:

$$\forall i.\, \exists j.\, \left(\sigma(s_i) = \neg m_j \vee \sigma(s_i) = m_j\right)$$
$$\forall i.\, \exists j.\, \left(b^-_{i,j} \vee b^+_{i,j}\right)$$
$$\bigwedge_i \bigvee_j b^-_{i,j} \vee b^+_{i,j}$$

Constraint to SAT Encoding - Completeness

The constraints can be encoded using a simple procedure.
For example the completeness constraint:

$$\forall i.\, \exists j.\, \left(\sigma(s_i) = \neg m_j \vee \sigma(s_i) = m_j\right)$$
$$\forall i.\, \exists j.\, \left(b^-_{i,j} \vee b^+_{i,j}\right)$$
$$\bigwedge_i \bigvee_j b^-_{i,j} \vee b^+_{i,j}$$
$$\bigwedge_i \bigvee_j b_{i,j}$$

SR Direct Encoding

SAT-based compatibility

$$\bigwedge_i \bigwedge_j [b_{i,j} \Rightarrow \sigma_{i,j} \subseteq \sigma]$$

SAT-based existence

$$\bigvee_i \bigvee_j b^-_{i,j}$$

SAT-based uniqueness

$$\bigwedge_j \bigwedge_i \bigwedge_{i' \geq i} \bigwedge_{j' > j} \neg b^-_{i,j} \vee \neg b^-_{i',j'}$$

SAT-based completeness

$$\bigwedge_i \bigvee_j b_{i,j}$$

SAT-based coherence

$$\bigwedge_j \bigwedge_i \bigwedge_{i'} \neg b^+_{i,j} \vee \neg b^-_{i',j}$$

Structuring Variables

We define the following SAT variables:

- c_j is true iff m_j is the resolution literal.

$$c_j \Leftrightarrow \exists i.\, \sigma(s_i) = \neg m_j$$

.

Illustration

$$\begin{aligned}
c_1 &\Leftrightarrow b^-_{1,1} \vee \ldots \vee b^-_{n,1} \\
&\Leftrightarrow \sigma(s_1) = \neg m_1 \vee \ldots \vee \sigma(s_n) = \neg m_1 \\
c_2 &\Leftrightarrow b^-_{1,2} \vee \ldots \vee b^-_{n,2} \\
&\Leftrightarrow \sigma(s_1) = \neg m_2 \vee \ldots \vee \sigma(s_n) = \neg m_2 \\
&\vdots
\end{aligned}$$

SR Indirect Encoding

SAT-based compatibility $\bigwedge_i \bigwedge_j [b_{i,j} \Rightarrow \sigma_{i,j} \subseteq \sigma]$

Structurality $\bigwedge_j \left[\neg c_j \vee \bigvee_i b^-_{i,j}\right] \wedge \bigwedge_j \bigwedge_i \left(c_j \vee \neg b^-_{i,j}\right)$

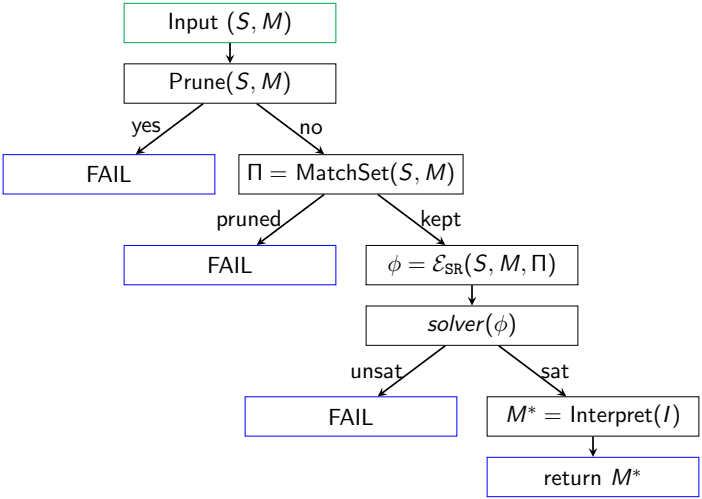
Revised existence $\bigvee_j c_j$

Revised uniqueness $AMO(\{c_j, j = 1, ..., |M|\})$

Revised completeness $\bigwedge_i \bigvee_j b_{i,j}$

Revised coherence $\bigwedge_j \bigwedge_i \left(\neg c_j \vee \neg b^+_{i,j}\right)$

SAT-Based Subsumption Resolution



Two Encodings

Direct Encoding $\mathcal{E}^{d}_{\mathtt{SR}}(S, M)$

positive compatibility	$\bigwedge_i \bigwedge_j \left(b^+_{i,j} \Rightarrow \Sigma^+_{i,j} \subseteq \sigma \right)$
negative compatibility	$\bigwedge_i \bigwedge_j \left(b^-_{i,j} \Rightarrow \Sigma^-_{i,j} \subseteq \sigma \right)$
existence	$\bigvee_i \bigvee_j b^-_{i,j}$
uniqueness	$\bigwedge_j \bigwedge_i \bigwedge_{i' \geq i\, j' > j} \neg b^-_{i,j} \vee \neg b^-_{i',j'}$
completeness	$\bigwedge_i \bigvee_j b^+_{i,j} \vee b^-_{i,j}$
coherence	$\bigwedge_j \bigwedge_i \bigwedge_{i'} \neg b^+_{i,j} \vee \neg b^-_{i',j}$

Indirect Encoding $\mathcal{E}^{i}_{\mathtt{SR}}(S, M)$

positive compatibility	$\bigwedge_i \bigwedge_j \left(b^+_{i,j} \Rightarrow \Sigma^+_{i,j} \subseteq \sigma \right)$
negative compatibility	$\bigwedge_i \bigwedge_j \left(b^-_{i,j} \Rightarrow \Sigma^-_{i,j} \subseteq \sigma \right)$
structurality	$\bigwedge_j \left[\neg c_j \vee \bigvee_i b^-_{i,j} \right] \wedge \bigwedge_j \bigwedge_i \left(c_j \vee \neg b^-_{i,j} \right)$
revised existence	$\bigvee_j c_j$
revised uniqueness	$AMO(\{c_j, j = 1, ..., \lvert M \rvert\})$
completeness	$\bigwedge_i \bigvee_j b^+_{i,j} \vee b^-_{i,j}$
revised coherence	$\bigwedge_j \bigwedge_i \left(\neg c_j \vee \neg b^+_{i,j} \right)$

Two Encodings

Direct Encoding $\mathcal{E}^d_{\mathrm{SR}}(S, M)$

positive compatibility $\bigwedge_i \bigwedge_j \left(b^+_{i,j} \Rightarrow \Sigma^+_{i,j} \subseteq \sigma\right)$

negative compatibility $\bigwedge_i \bigwedge_j \left(b^-_{i,j} \Rightarrow \Sigma^-_{i,j} \subseteq \sigma\right)$

existence $\bigvee_i \bigvee_j b^-_{i,j}$

uniqueness $\bigwedge_j \bigwedge_i \bigwedge_{i' \geq i} \bigwedge_{j' > j} \neg b^-_{i,j} \vee \neg b^-_{i',j'}$

completeness $\bigwedge_i \bigvee_j b^+_{i,j} \vee b^-_{i,j}$

coherence $\bigwedge_j \bigwedge_i \bigwedge_{i'} \neg b^+_{i,j} \vee \neg b^-_{i',j}$

Indirect Encoding $\mathcal{E}^i_{\mathrm{SR}}(S, M)$

positive compatibility $\bigwedge_i \bigwedge_j \left(b^+_{i,j} \Rightarrow \Sigma^+_{i,j} \subseteq \sigma\right)$

negative compatibility $\bigwedge_i \bigwedge_j \left(b^-_{i,j} \Rightarrow \Sigma^-_{i,j} \subseteq \sigma\right)$

structurality $\bigwedge_j \left[\neg c_j \vee \bigvee_i b^-_{i,j}\right] \wedge \bigwedge_j \bigwedge_i \left(c_j \vee \neg b^-_{i,j}\right)$

revised existence $\bigvee_j c_j$

revised uniqueness $AMO(\{c_j, j = 1, ..., |M|\})$

completeness $\bigwedge_i \bigvee_j b^+_{i,j} \vee b^-_{i,j}$

revised coherence $\bigwedge_j \bigwedge_i \left(\neg c_j \vee \neg b^+_{i,j}\right)$

Complexities

- $\mathcal{E}^d_{\mathrm{SR}}(S, M)$ has $O(|\Pi|)$ variables and $O(|\Pi|^2)$ clauses.
- $\mathcal{E}^i_{\mathrm{SR}}(S, M)$ has $O(|\Pi| + |M|)$ variables and $O(|\Pi|)$ clauses.

Results

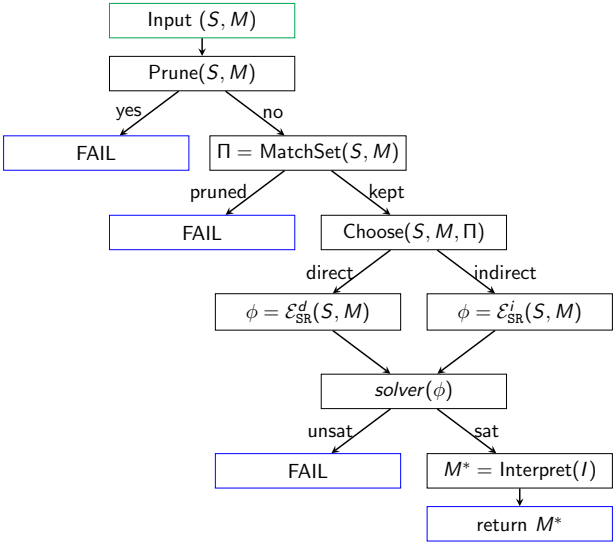
Prover	Average	Std. Dev.	Boost
VAMPIRE$_M$	33.63 μs	1839.25 μs	1.00
VAMPIRE$_D$	25.38 μs	1241.86 μs	1.32
VAMPIRE$_I$	24.93 μs	196.38 μs	1.35

Table: Average time spent in the forward simplify loop.

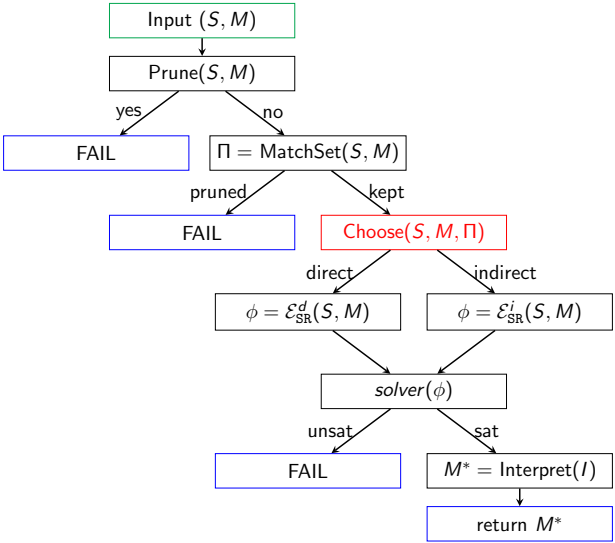
Prover	Total Solved	Gain/Loss
VAMPIRE$_M$	10 728	baseline
VAMPIRE$_D$	10 791	(+94, −31)
VAMPIRE$_I$	10 785	(+92, −35)

Table: Number of TPTP problems solved by the considered versions of VAMPIRE. The run was made using the options `-sa otter -av off` with a timeout of 60 s. The **Gain/Loss** column reports the difference of solved instances compared to VAMPIRE$_M$.

Choosing the Encoding



Choosing the Encoding



Choosing Features

The features should be

- fast to compute;
- informative;
- independent.

Choosing Features

The features should be

- fast to compute;
- informative;
- independent.

1. Number of literals of S;
2. Number of literals of M;
3. Sparsity of the match set $\frac{|\Pi|}{|S|\cdot|M|}$.

Choosing the Architecture

What do we want?

We want a model that is

- fast to compute;
- generalisable;
- interpretable;
- easy to train.

Choosing the Architecture

What do we want?

We want a model that is

- fast to compute;
- generalisable;
- interpretable;
- easy to train.

Decision Trees are (almost) perfect

- can be hard coded in a few lines;
- not prone to overfitting;
- can be visualised;
- ... but cannot easily be trained online ...

Big Dataset without Online Learning

Objective function

$$\arg\min_{f \in \mathcal{F}} \mathbb{E}_{\substack{x \sim \mathcal{X} \\ (y_0, y_1) \sim \mathcal{D}(\cdot|x)}} \left[y_{f(x)} \right]$$

Big Dataset without Online Learning

Objective function

$$\arg\min_{f \in \mathcal{F}} \mathbb{E}_{\substack{x \sim \mathcal{X} \\ (y_0, y_1) \sim \mathcal{D}(\cdot|x)}} \left[y_{f(x)} \right]$$

What if the dataset cannot be loaded in memory?

Big Dataset without Online Learning

Objective function

$$\arg\min_{f \in \mathcal{F}} \mathbb{E}_{\substack{x \sim \mathcal{X} \\ (y_0, y_1) \sim \mathcal{D}(\cdot|x)}} \left[y_{f(x)} \right]$$

What if the dataset cannot be loaded in memory?

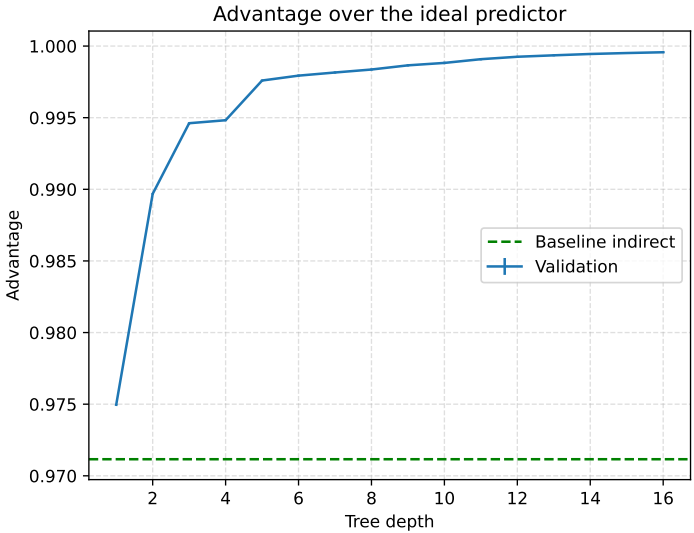
Revised objective function

We condense de dataset $\{(x, y_0, y_1)\}$ into $\mathcal{S} = \{(x, \hat{y}_0, \hat{y}_1)\}$ where $\hat{y}_0$ is the sum of the y_0 with the same x and $\hat{y}_1$ is the sum of the y_1 with the same x. Then we have

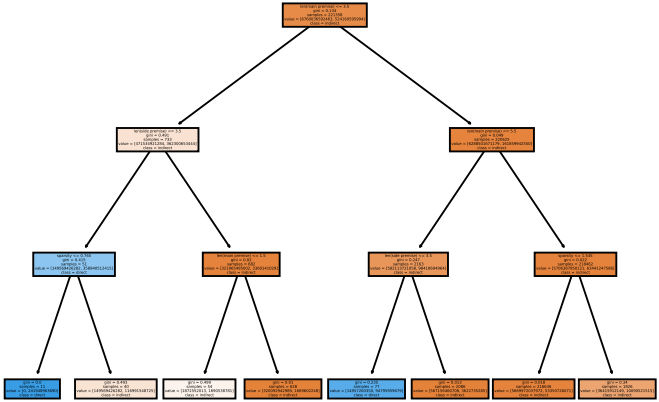
$$\arg\min_{f \in \mathcal{F}} \sum_{(x, \hat{y}_0, \hat{y}_1) \in \mathcal{S}} \left[|\hat{y}_0 - \hat{y}_1| * (f(x) - H(\hat{y}_0 - \hat{y}_1))^2 \right]$$

with H the step function

Setting Model Complexity



Final Tree



Results (1/2)

Prover	Average	Std. Dev.	Boost
VAMPIRE$_M$	33.63 μs	1839.25 μs	1.00
VAMPIRE$_D$	25.38 μs	1241.86 μs	1.32
VAMPIRE$_I$	24.93 μs	196.38 μs	1.35
VAMPIRE$_H$	24.73 μs	190.69 μs	1.36

Table: Average time spent in the forward simplify loop.

Results (2/2)

Prover	Total Solved	Gain/Loss
VAMPIRE$_M$	10 728	baseline
VAMPIRE$_D$	10 791	(+94, −31)
VAMPIRE$_I$	10 785	(+92, −35)
VAMPIRE$_H$	10 794	(+97, −31)

Table: Number of TPTP problems solved by the considered versions of VAMPIRE. The run was made using the options `-sa otter -av off` with a timeout of 60 s. The **Gain/Loss** column reports the difference of solved instances compared to VAMPIRE$_M$.

Overfitting?

Not likely!

- The model is simple;
- The dataset is large;
- The feature space is small.

Overfitting?

Not likely!

- The model is simple;
- The dataset is large;
- The feature space is small.

In any case...

We want to solve problems from TPTP, generalisation is not our main goal.

Success Rate of Queries

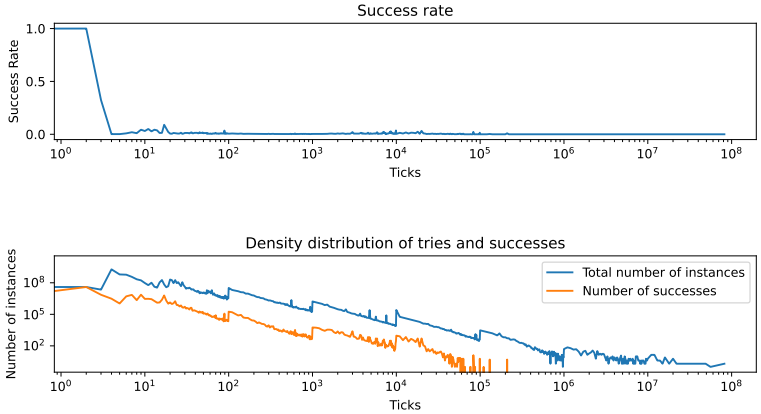


Figure: Success rate of the SAT query compared to how long the solver runs.

Saving SAT Runtime

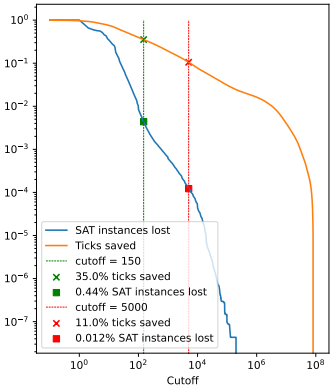


Figure: Cutoff time vs. number of problems lost.

Results

Prover	Total Solved	Gain/Loss
VAMPIRE$_M$	10 728	baseline
VAMPIRE	10 794	(+97, −31)
VAMPIRE-cutoff-150	10 768	(+93, −53)
VAMPIRE-cutoff-5000	10 790	(+97, −35)

Table: Number of TPTP problems solved by the considered versions of VAMPIRE. The run was made using the options `-sa otter -av off` with a timeout of 60 s. The **Gain/Loss** column reports the difference of solved instances compared to VAMPIRE$_M$.

Conclusion

SAT-based subsumption (resolution) in VAMPIRE

- Subsumption becomes a small SAT query, with two encodings (direct, indirect)
- 1.36× faster forward simplification, ∼10× lower variance
- +66 TPTP problems solved (+97, −31)

Conclusion

SAT-based subsumption (resolution) in VAMPIRE

- Subsumption becomes a small SAT query, with two encodings (direct, indirect)
- 1.36× faster forward simplification, ∼10× lower variance
- +66 TPTP problems solved (+97, −31)

Take-away

Saturation lives or dies by redundancy elimination,
and SAT technology is a great fit for it.
Giving up on hard problems is a bad idea.

Conclusion

SAT-based subsumption (resolution) in VAMPIRE

- Subsumption becomes a small SAT query, with two encodings (direct, indirect)
- 1.36× faster forward simplification, ∼10× lower variance
- +66 TPTP problems solved (+97, −31)

Take-away

Saturation lives or dies by redundancy elimination,
and SAT technology is a great fit for it.
Giving up on hard problems is a bad idea.

Open questions

- Subsumption/SR modulo AC?
- Can this be useful in SMT?

References


Coutelier, R., Kovács, L., Rawson, M., and Rath, J. (2023).
Sat-based subsumption resolution.
In Pientka, B. and Tinelli, C., editors, *Automated Deduction - CADE 29 - 29th International Conference on Automated Deduction, Rome, Italy, July 1-4, 2023, Proceedings*, volume 14132 of *Lecture Notes in Computer Science*, pages 190–206. Springer.


Coutelier, R., Rath, J., Rawson, M., and Kovács, A. B. L. (2024).
Sat solving for variants of first-order subsumption.


Rath, J., Biere, A., and Kovács, L. (2022).
First-order subsumption via SAT solving.
In Griggio, A. and Rungta, N., editors, *22nd Formal Methods in Computer-Aided Design, FMCAD 2022, Trento, Italy, October 17-21, 2022*, pages 160–169. IEEE.