Efficient Heuristic for SAT-Based (Variants of) Subsumption

Vampire Workshop 2024

Robin Coutelier [1]

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

01 July 2024

TU WIEN Informatics 20 YEARS

Acknowledgements

Joint work with Jakob Rath, Michael Rawson, Laura Kovács and Armin Biere.
We thank Pascal Fontaine (University of Liège, Belgium) for fruitful discussions. We acknowledge partial support from the ERC Consolidator Grant ARTIST 101002685, the FWF SFB project SpyCoDe F8504, the Austrian FWF project W1255-N23, the WWTF ICT22-007 Grant ForSmart, and the TU Wien Trustworthy Autonomous Cyber-Physical Systems Doctoral College. This research was funded in whole or in part by the Austrian Science Fund (FWF) [10.55776/F85, 10.55776/W1255]. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission.

European Research Council
Established by the European Commission

Der Wissenschaftsfonds.

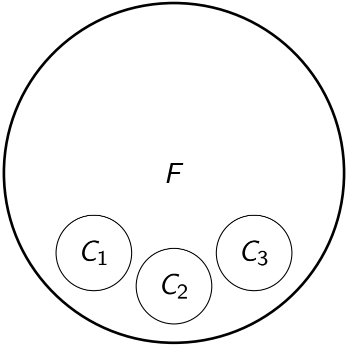
Vienna Science
and Technology Fund

Introduction

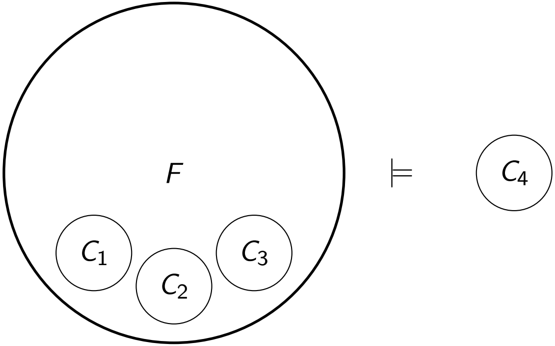
Related work

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

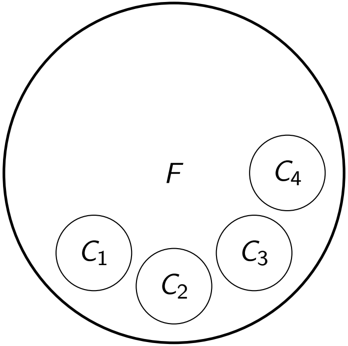
Saturation in FOL Theorem Proving



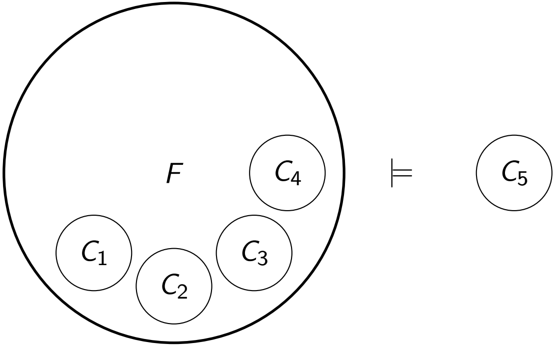
Saturation in FOL Theorem Proving



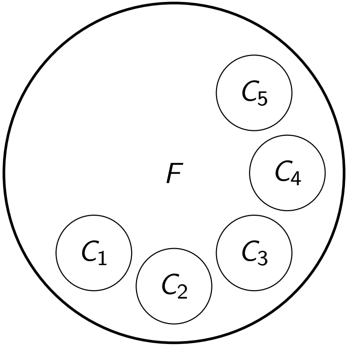
Saturation in FOL Theorem Proving



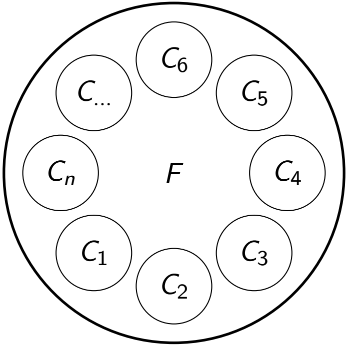
Saturation in FOL Theorem Proving



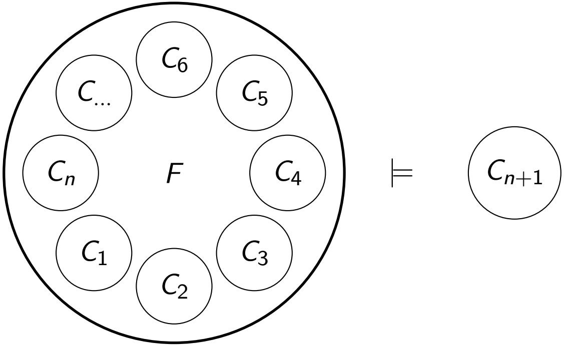
Saturation in FOL Theorem Proving



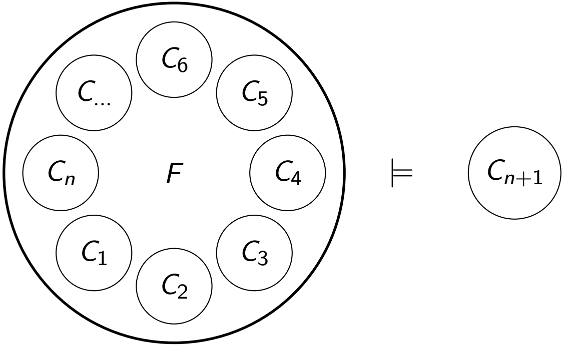
Saturation in FOL Theorem Proving



Saturation in FOL Theorem Proving



Saturation in FOL Theorem Proving



Out of memory!

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)

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

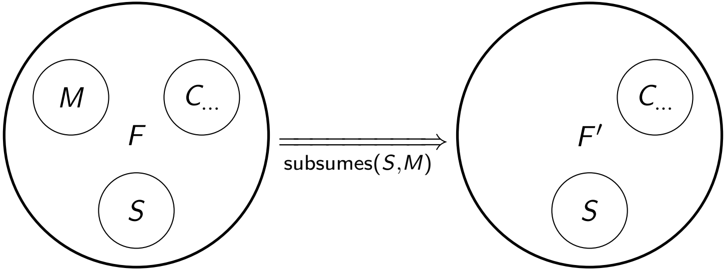
S subsumes M. It is "stronger" than M.

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)

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

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)

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

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

Subsumption Resolution - Example 2

Example (FOL)

$$
\begin{aligned}
S &= p(x_1, x_2) \lor p(f(x_2), x_3) \\
M &= \neg p(f(y), d) \lor p(g(y), c) \lor \neg p(f(c), e) \\
\sigma &= \{x_1 \mapsto g(y), x_2 \mapsto c, x_3 \mapsto e\}
\end{aligned}
$$

Subsumption Resolution - Example 2

Example (FOL)

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

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

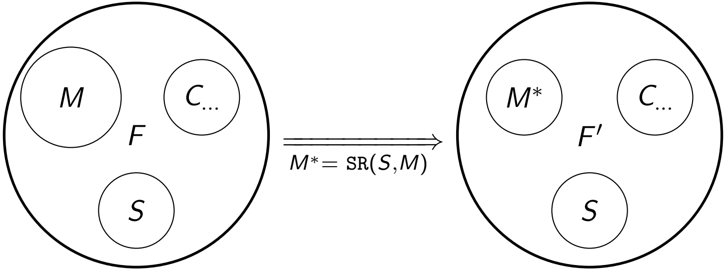
Subsumption Resolution - Example 2

Example (FOL)

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

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

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

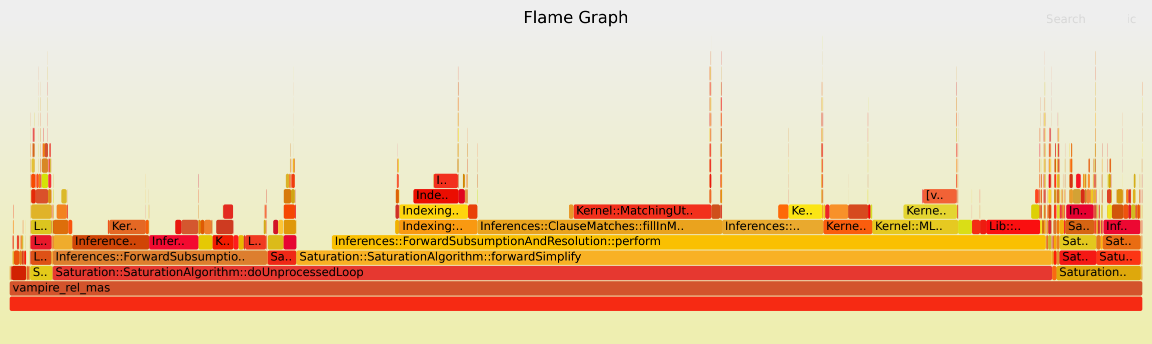
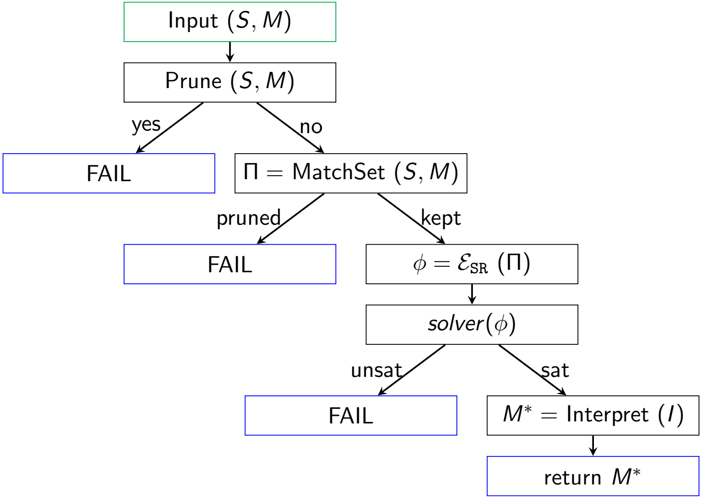
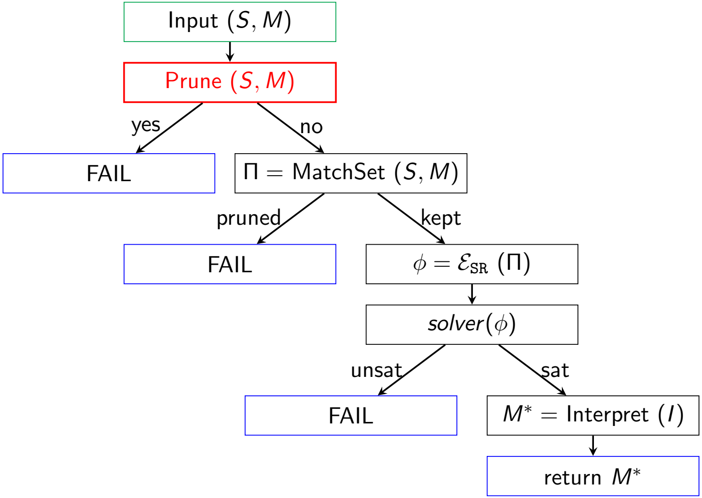


Figure: Typical profiling results for a TPTP problem (GRP001+6).

SAT-Based Subsumption Resolution



SAT-Based Subsumption Resolution



Multi-Step Pruning – (multi-)set check

$$\{(\mathcal{P}(s_i), \mathcal{Q}(s_i)) \mid s_i \in S\} \sqsubseteq \{(\mathcal{P}(m_j), \mathcal{Q}(m_j)) \mid m_j \in M\} \tag{1}$$

Theorem (Pruning Subsumption)

If the pruning criterion (1) *is unsat, then S does not subsume M.*

Multi-Step Pruning – (multi-)set check

$$\{(\mathcal{P}(s_i), \mathcal{Q}(s_i)) \mid s_i \in S\} \sqsubseteq \{(\mathcal{P}(m_j), \mathcal{Q}(m_j)) \mid m_j \in M\} \tag{1}$$

Theorem (Pruning Subsumption)

If the pruning criterion (1) is unsat, then S does not subsume M.

$$\{\mathcal{P}(s_i) \mid s_i \in S\} \subseteq \{\mathcal{P}(m_j) \mid m_j \in M\} \tag{2}$$

Theorem (Pruning Subsumption Resolution)

If the pruning criterion (2) is unsat, then S and M are not side and main premises of subsumption resolution.

Multi-Step Pruning – (multi-)set check

$$\{(\mathcal{P}(s_i), \mathcal{Q}(s_i)) \mid s_i \in S\} \sqsubseteq \{(\mathcal{P}(m_j), \mathcal{Q}(m_j)) \mid m_j \in M\} \tag{1}$$

Theorem (Pruning Subsumption)

If the pruning criterion (1) *is unsat, then S does not subsume M.*

$$\{\mathcal{P}(s_i) \mid s_i \in S\} \subseteq \{\mathcal{P}(m_j) \mid m_j \in M\} \tag{2}$$

Theorem

Validity of the subsumption pruning criterion (1) *implies validity of the subsumption resolution pruning criterion* (2).

Previous Implementation

$N \leftarrow$ number of predicate symbols
procedure *pruneSubsumption(S, M)*
```
    A ← zeros(2 · N)
    for m ∈ M do
        idx ← HEADERINDEX(m)
        A[idx] ← max(0, A[idx]) + 1
    for s ∈ S do
        idx ← HEADERINDEX(s)
        if A[idx] ≤ 0 then
            return ⊤
        A[idx] ← A[idx] − 1
    return ⊥
```

Previous Implementation

$N \leftarrow$ number of predicate symbols
procedure *pruneSubsumption(S, M)*
    $\mathcal{A} \leftarrow zeros(2 \cdot N)$
    for $m \in M$ **do**
        $idx \leftarrow \textsc{HeaderIndex}(m)$
        $\mathcal{A}[idx] \leftarrow \max(0, \mathcal{A}[idx]) + 1$
    for $s \in S$ **do**
        $idx \leftarrow \textsc{HeaderIndex}(s)$
        if $\mathcal{A}[idx] \leq 0$ **then**
            return $\top$
        $\mathcal{A}[idx] \leftarrow \mathcal{A}[idx] - 1$
    return $\perp$

Fast Implementation

$N \leftarrow$ number of predicate symbols

$t \leftarrow 0, \mathcal{A} \leftarrow \mathit{zeros}(2 \cdot N)$

procedure *pruneSubsumption(S, M)*

$\quad$ **if** $t + |M| > \mathtt{UINT\_MAX}$ **then**

$\quad\quad t \leftarrow 0,\ \mathcal{A} \leftarrow \mathit{zeros}(2 \cdot N)$

$\quad$ **for** $m \in M$ **do**

$\quad\quad \mathit{idx} \leftarrow \textsc{headerIndex}(m)$

$\quad\quad \mathcal{A}[\mathit{idx}] \leftarrow \max(t, \mathcal{A}[\mathit{idx}]) + 1$

$\quad$ **for** $s \in S$ **do**

$\quad\quad \mathit{idx} \leftarrow \textsc{headerIndex}(s)$

$\quad\quad$ **if** $\mathcal{A}[\mathit{idx}] \leq t$ **then**

$\quad\quad\quad t \leftarrow t + |M|$, **return** $\top$

$\quad\quad \mathcal{A}[\mathit{idx}] \leftarrow \mathcal{A}[\mathit{idx}] - 1$

$\quad t \leftarrow t + |M|$, **return** $\bot$

Variance Drop Explanation

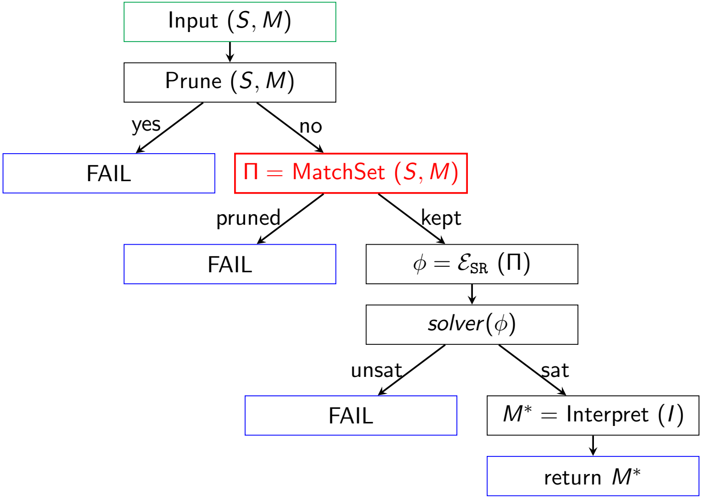
Prover	Average	Std. Dev.	Boost
$\textsc{Vampire}_M$	42.63 μs	1609.06 μs	1.00
$\textsc{Vampire}_I$	40.13 μs	1554.52 μs	1.06
$\textsc{Vampire}_I^*$	34.55 μs	250.25 μs	1.23

Table: Without Pruning Optimization

Prover	Average	Std. Dev.	Boost
$\textsc{Vampire}_M$	33.63 μs	1839.25 μs	1.00
$\textsc{Vampire}_I$	28.36 μs	243.38 μs	1.19
$\textsc{Vampire}_I^*$	24.93 μs	196.38 μs	1.35

Table: With Pruning Optimization

SAT-Based Subsumption Resolution



Match Set

Incompatible Substitution

The Incompatible Substitution $\tilde{\Sigma}$ is a substitution that is incompatible with all substitutions. If two literals s_i and m_j are not unifiable, then $\tilde{\Sigma}$ is the only substitution that can be applied to s_i to make it equal to m_j.

$$\tilde{\Sigma}(s_i) = m_j \vee \tilde{\Sigma}(s_i) = \neg m_j$$

.

Definition

The *match set* of S and M is the set of pairs $\left(b^{\pm}_{i,j}, \Sigma^{\pm}_{i,j}\right)$ such that $b^{\pm}_{i,j}$ is a propositional variables and $\Sigma^{\pm}_{i,j}$ is a substitution such that $\Sigma^{+}_{i,j}(s_i) = m_j$ and $\Sigma^{-}_{i,j}(s_i) = \neg m_j$.

Multi-Step Pruning – After Building the Match Set

$$\forall i \exists j.\ \Sigma^{+}_{i,j} \neq \tilde{\Sigma} \tag{3}$$

Theorem (Substitution Sets for Pruning Subsumption)

Let $\Pi(S, M) = \left\{ \left(b^{\pm}_{i,j}, \Sigma^{\pm}_{i,j} \right) \right\}$ *be the match set of* S *and* M*. If* (3) *is unsat, then* S *does not subsume* M*.*

Multi-Step Pruning – After Building the Match Set

$$\forall i \exists j.\ \Sigma^{+}_{i,j} \neq \tilde{\Sigma} \tag{3}$$

Theorem (Substitution Sets for Pruning Subsumption)

Let $\Pi(S, M) = \left\{ \left(b^{\pm}_{i,j}, \Sigma^{\pm}_{i,j} \right) \right\}$ be the match set of S and M. If (3) is unsat, then S does not subsume M.

$$\forall i \exists j.\ \Sigma^{+}_{i,j} \neq \tilde{\Sigma} \vee \Sigma^{-}_{i,j} \neq \tilde{\Sigma} \tag{4}$$

Theorem (Substitution Sets for Pruning Subsumption Resolution)

Let $\Pi(S, M) = \left\{ \left(b^{\pm}_{i,j}, \Sigma^{\pm}_{i,j} \right) \right\}$ be the match set of S, M. If (4) is unsat, then S and M are not side and main premises of subsumption resolution.

Multi-Step Pruning – After Building the Match Set

$$\forall i, i'.\ (i \neq i') \Rightarrow (\mathcal{P}(s_i) = \mathcal{P}(s_{i'}) \vee \exists j\ \Sigma^{+}_{i,j} \neq \tilde{\Sigma} \vee \exists j\ \Sigma^{+}_{i',j} \neq \tilde{\Sigma}) \tag{5}$$

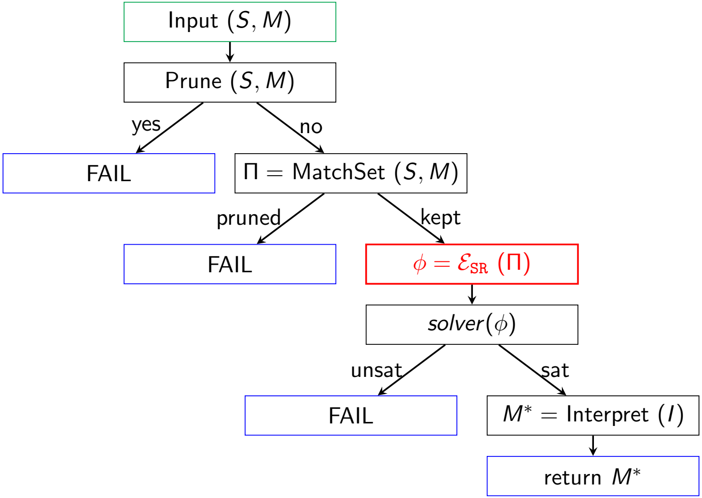
Theorem (Predicate Matches for Pruning Subsumption Resolution)

Let $\Pi(S, M) = \left\{ \left(b^{\pm}_{i,j}, \Sigma^{\pm}_{i,j} \right) \right\}$ be the match set of S, M. If (5) is unsat, then S and M are not side and main premises of subsumption resolution.

Example

Let $S = \neg p(x) \vee q(x)$ and $M = p(a) \vee \neg q(a)$. There are two literals in S that only match negatively to literals in M, with a different predicate. Therefore, subsumption resolution is impossible.

SAT-Based Subsumption Resolution



Two Encodings

Direct Encoding $\mathcal{E}^{d}_{\mathrm{SR}}(\Pi)$

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 $\mathcal{E}^{i}_{\mathrm{SR}}(\Pi)$

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 $\mathcal{E}^{d}_{\mathrm{SR}}(\Pi)$

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

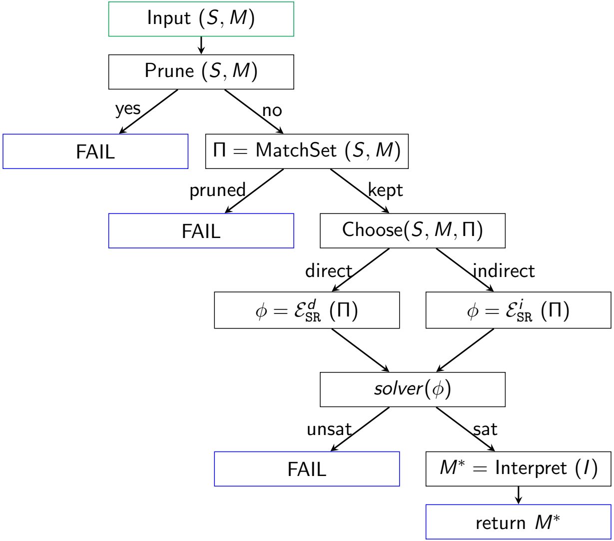
Indirect Encoding $\mathcal{E}^{i}_{\mathrm{SR}}(\Pi)$

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	$AMO(\{c_j, j = 1, \ldots, \lvert M \rvert\})$
completeness	$\bigwedge_i \bigvee_j b^{+}_{i,j} \vee b^{-}_{i,j}$
revised coherence	$\bigwedge_j \bigwedge_i \left(\neg c_j \vee \neg b^{+}_{i,j}\right)$

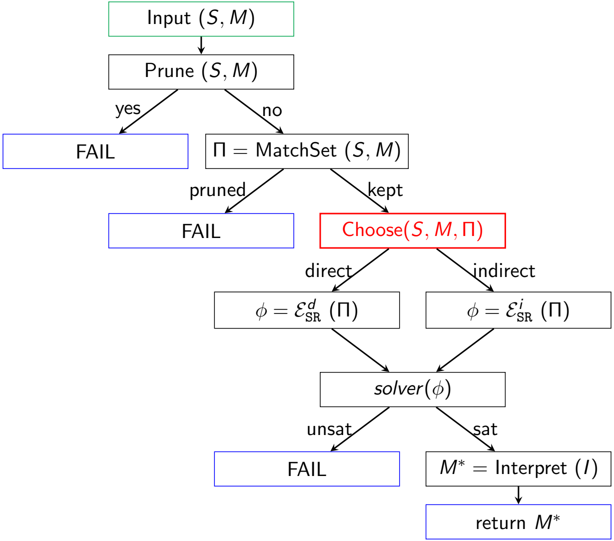
Complexities

- $\mathcal{E}^{d}_{\mathrm{SR}}(\Pi)$ has $O(|\Pi|)$ variables and $O(|\Pi|^2)$ clauses.
- $\mathcal{E}^{i}_{\mathrm{SR}}(\Pi)$ has $O(|\Pi| + |M|)$ variables and $O(|\Pi|)$ clauses.

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

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

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?

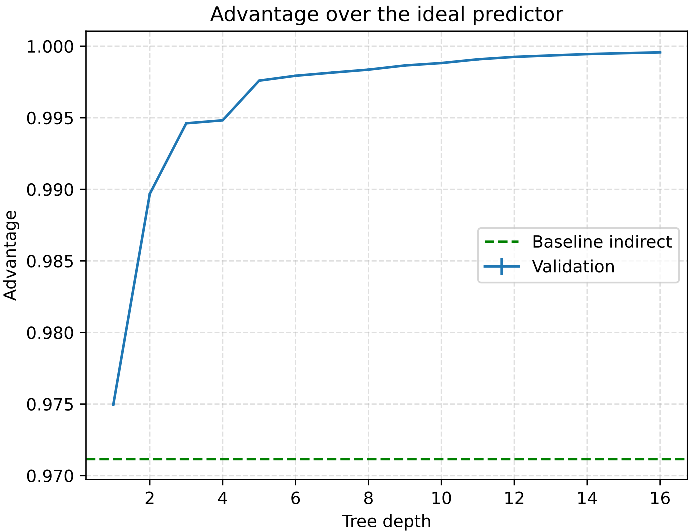
Revised objective function

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

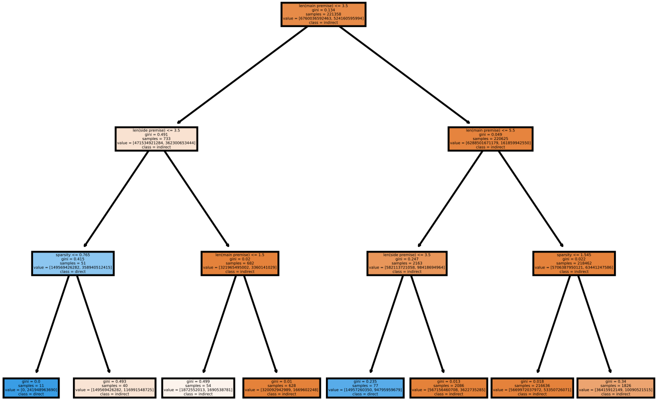
$$\arg\min_{f \in \mathcal{F}} \sum_{(x, \hat{y}_0, \hat{y}_1) \in \mathcal{S}} \left[|\hat{y}_0 - \hat{y}_1| * (f(x) - H(\hat{y}_0 - \hat{y}_1))^2 \right]$$

with H the step function

Setting Model Complexity



Final Tree


len(main premise) <= 3.5
gini = 0.134
samples = 221358
value = [6760036592463, 524160595994]
class = indirect
len(side premise) <= 3.5
gini = 0.491
samples = 733
value = [471534921284, 362300653444]
class = indirect
len(main premise) <= 5.5
gini = 0.049
samples = 220625
value = [6288501671179, 161859942550]
class = indirect
sparsity <= 0.765
gini = 0.415
samples = 51
value = [149569426282, 358940512415]
class = direct
len(main premise) <= 1.5
gini = 0.02
samples = 682
value = [321965495002, 3360141029]
class = indirect
len(side premise) <= 3.5
gini = 0.247
samples = 2163
value = [582113721058, 98418694964]
class = indirect
sparsity <= 1.545
gini = 0.022
samples = 218462
value = [5706387950121, 63441247586]
class = indirect
gini = 0.0
samples = 11
value = [0, 241948963690]
class = direct
gini = 0.493
samples = 40
value = [149569426282, 116991548725]
class = indirect
gini = 0.499
samples = 54
value = [1872552013, 1690538781]
class = indirect
gini = 0.01
samples = 628
value = [320092942989, 1669602248]
class = indirect
gini = 0.235
samples = 77
value = [14957260350, 94795959679]
class = direct
gini = 0.013
samples = 2086
value = [567156460708, 3622735285]
class = indirect
gini = 0.018
samples = 216636
value = [5669972037972, 53350726071]
class = indirect
gini = 0.34
samples = 1826
value = [36415912149, 10090521515]
class = indirect

Performance of the Simplification Loop

Prover	Average	Std. Dev.	Boost
$\textsc{Vampire}_M$	33.63 μs	1839.25 μs	1.00
$\textsc{Vampire}_D$	28.74 μs	1245.88 μs	1.17
$\textsc{Vampire}_I$	28.36 μs	243.38 μs	1.19
$\textsc{Vampire}_H$	28.16 μs	233.87 μs	1.19
$\textsc{Vampire}^*_D$	25.38 μs	1241.86 μs	1.32
$\textsc{Vampire}^*_I$	24.93 μs	196.38 μs	1.35
$\textsc{Vampire}^*_H$	24.73 μs	190.69 μs	1.36

Table: Average time spent in the forward simplify loop.

Overfitting?

Not likely!

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

Overfitting?

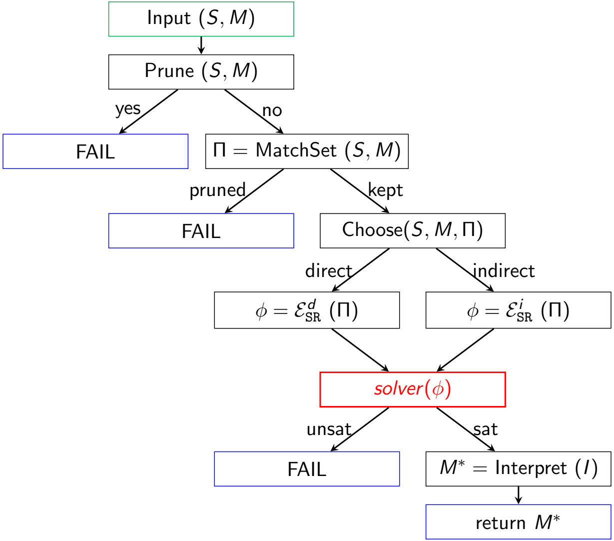
Not likely!

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

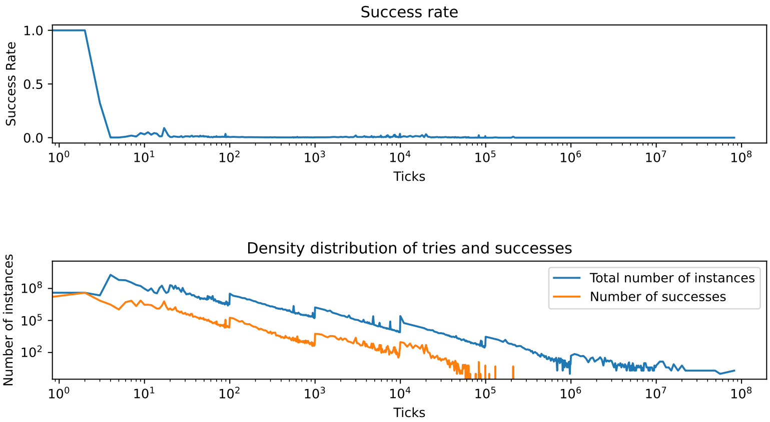
In any case...

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

SAT-Based Subsumption Resolution

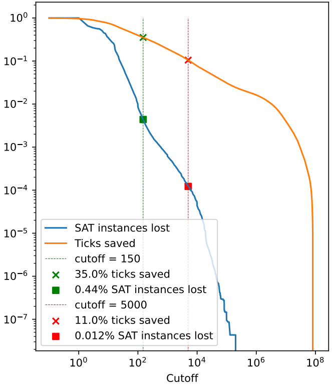


Cutting off Difficult Instances?



Success rate of direct SR with respect to difficulty

Computation Saved



Computation saved with direct SR with respect to difficulty threshold

Performance on TPTP

Prover	Total Solved	Gain/Loss
VAMPIRE$_M$	10 728	baseline
VAMPIRE$_D$	10 762	$(+62, -28)$
VAMPIRE$_I$	10 760	$(+63, -31)$
VAMPIRE$_H$	10 764	$(+64, -28)$
VAMPIRE$^*_D$	10 791	$(+94, -31)$
VAMPIRE$^*_I$	10 785	$(+92, -35)$
VAMPIRE$^*_H$	10 794	$(+97, -31)$
VAMPIRE–cutoff-5000$^*_H$	10 790	$(+97, -35)$
VAMPIRE–cutoff-150$^*_H$	10 768	$(+93, -53)$

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$_M$.

References I

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.