Revisiting Global Subsumption

Giles Reger\textsuperscript{1} and Martin Suda\textsuperscript{2}

(1) School of Computer Science, University of Manchester, UK
(2) Institute for Information Systems, Vienna University of Technology, Austria

The 3rd Vampire Workshop
Introduction

In this talk we will

- Remind ourselves what Global Subsumption (GS) is
- Discuss five related questions

The following five questions will hopefully make more sense when we have explained GS

1. What groundings are good groundings?
2. What subclause are good subclauses?
3. Can GS play nicely with AVATAR?
4. Can GS play with theories?
5. Can we apply this lookahead idea to GS?
Overview
Global Subsumption: the Ground Case

- Assume a set of first order clauses $S$
- Let $S_{gr}$ be a set of ground clauses implied by $S$
  i.e. instances of clauses in $S$
- The ground clause $D \lor D'$ can be replaced by $D$ in $S$ if $S_{gr} \models D$
- This is sound as $D$ follows from $S$ and subsumes $D \lor D'$
- If $D$ is empty then $S_{gr}$ is unsatisfiable and so is $S$
Global Subsumption: the Ground Case, an Example

Consider

\[ S = \begin{cases} 
    p(x) \lor q(a) \\
    \neg p(x) \lor q(c) \\
    f(a) = a \\
    f(f(a)) \neq a 
\end{cases} \]

\[ S_{gr} = \begin{cases} 
    p(\bot) \lor q(a) \\
    \neg p(\bot) \lor q(c) \\
    f(a) = a \\
    f(f(a)) \neq a 
\end{cases} \]

- \((D \lor D') = q(a) \lor q(b) \lor q(c)\)
- \(S_{gr} \models q(a) \lor q(c)\)
- \(q(a) \lor q(b) \lor q(c)\) can be replaced by \(q(a) \lor q(c)\)
Global Subsumption: the Ground Case, an Example

Consider

\[ S = \begin{cases} 
    p(x) \lor q(a) \\
    \neg p(x) \lor q(c) \\
    f(a) = a \\
    f(f(a)) \neq a 
\end{cases} \]

\[ S_{gr} = \begin{cases} 
    1 \lor 2 \\
    \neg 1 \lor 3 \\
    4 \\
    \neg 5 
\end{cases} \]

- \((D \lor D') = 2 \lor 6 \lor 3\)
- \(S_{gr} \models 2 \lor 3\)
- \(q(a) \lor q(b) \lor q(c)\) can be replaced by \(q(a) \lor q(c)\)
Global Subsumption: the Ground Case, an Example

Consider

\[ S = \left\{ \begin{array}{l}
p(x) \lor q(a) \\
\neg p(x) \lor q(c)
\end{array} \right\} \quad S_{gr} = \left\{ \begin{array}{l}
1 \lor 2 \\
\neg 1 \lor 3
\end{array} \right\} \]

\[ (D \lor D') = 2 \lor 6 \lor 3 \]

\[ S_{gr} \models 2 \lor 3 \]

\[ q(a) \lor q(b) \lor q(c) \text{ can be replaced by } q(a) \lor q(c) \]
Global Subsumption: the Non-Ground Case

- We can lift this to give the non-ground global subsumption rule:

\[
\frac{C \lor C'}{C}
\]

where \( S_{gr} \models C \gamma \) for non-empty \( C' \) and injective substitution \( \gamma \) from variables in \( C \) to fresh constants.

- For every generated clause \( C \) we:
  1. Let \( \gamma = [x_1 \mapsto c_1, \ldots, x_n \mapsto c_n] \) for \( x_i \) in \( C \) and fresh \( c_i \)
  2. Add \( C \gamma \) to \( S_{gr} \)
  3. Search for a minimal \( C' \subset C \) such that \( S_{gr} \models C' \)

- Why an injective substitution?
  - \( S_{gr} \models C \) is the same as \( S_{gr}, \neg C \) being inconsistent
  - \( \neg C \) is \( \neg(\forall x C[x]) \) is \( \exists x \neg C[x] \) so \( \gamma \) looks like the result of Skolemization
Example

- Take the following case:
  - \( C = p(x, y) \lor r(x) \)
  - \( S = \{ p(x, y) \lor r(x), p(x, x) \} \)

- \( C \) cannot be reduced. Injectivity is important
  - If we do things wrong we can get \( S_{gr} = \{ p(a, b) \lor r(a), p(a, a) \} \)
  - We check \( \{ p(a, a) \lor r(a), p(a, a), \neg p(a, a) \} \)
  - We have \( S_{gr} \models p(a, a) \) but \( p(x, y) \) does not follow from \( S \)

- If we add \( p(x, y) \) to \( S \) then \( C \) can be reduced
  - The correct grounding of \( S \) is \( S_{gr} = \{ p(a, b) \lor r(a), p(a, a), p(a, b) \} \)
  - We check \( \{ p(a, b) \lor r(a), p(a, a), p(a, b), \neg p(a, b) \} \)
  - \( C \) can be replaced by \( p(x, y) \)
Note on Cheap SAT Solvers…. and Experiments!

- We make a note that GS is all about doing some very cheap stuff for big improvements
- This will influence our decisions generally
- And for this reason we only run SAT solver in unit propagation mode i.e. no guessing

- But maybe that assumption is wrong..

**Experiment**

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>Propagation Only</td>
<td>8935</td>
<td>61</td>
</tr>
<tr>
<td>Full</td>
<td>8920</td>
<td>46</td>
</tr>
<tr>
<td>Baseline</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Overview
What do we want?

Consider

$$C_1 = p(x) \lor \neg q(y) \lor r(y)$$

$$\neg p(x)$$
What do we want?

- Consider

\[
C_1 = p(a) \lor \neg q(b) \lor r(b) \\
\neg p(a)
\]

- \( \gamma = [x \mapsto a, y \mapsto b] \) for \( C_1 \)
What do we want?

- Consider

\[
\begin{align*}
C_1 &= p(a) \lor \neg q(b) \lor r(b) \\
\neg p(a) \\
\end{align*}
\]

\[
\{ \quad \} _{gr} \models \neg q(x) \lor r(x)
\]

- \( \gamma = [x \mapsto a, y \mapsto b] \) for \( C_1 \)
What do we want?

- Consider

\[
C_1 = p(a) \lor \neg q(b) \lor r(b) \\
\neg p(a)
\]

\[\text{gr} \models \neg q(a) \lor r(a)\]

- \(\gamma = [x \mapsto a, y \mapsto b]\) for \(C_1\)
What do we want?

- Consider

$$C_1 = p(b) \lor \neg q(a) \lor r(a)$$
$$\neg p(b)$$

$$\text{gr} \models \neg q(a) \lor r(a)$$

- $$\gamma = [x \mapsto b, y \mapsto a]$$ for $$C_1$$
What do we want?

- Consider

\[
\begin{align*}
C_1 &= p(a) \lor \neg q(b) \lor r(b) \\
\neg p(a) \\
\end{align*}
\]

\[
\text{gr} \models \neg q(b) \lor r(b)
\]

- \(\gamma = [x \mapsto a, y \mapsto b]\) for \(C_1\)
What do we want?

Consider

\[
C_1 = p(a) \lor \neg q(b) \lor r(b) \\
\neg p(a) \\
p(b) \lor \neg q(a) \lor r(a) \\
\neg p(b)
\]

\[
\gamma = \{x \mapsto a, y \mapsto b\} \text{ for } C_1
\]

\[
gr \models \neg q(a) \lor r(a)
\]
What do we do?

- A single substitution
- Order literals
  - Prefer fewer variables
  - Prefer lighter literals (complexity)
  - Order predicate symbols
  - Prefer negative
  - Break ties
Ideas

- Implemented
  - Reverse the ordering (backward) to see what happens
  - $n$ substitutions where there are $n$ clauses where we put each literal first

- Next ideas
  - Ground units in more than one way (i.e. $p(a), p(b), p(c)$)
  - Single constant substitution (i.e. $\{x_1, \ldots, x_n \mapsto a\}$)
  - Lookahead (see last question)
### Experiment

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Unique A</th>
<th>Unique T</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>AVATAR on</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard</td>
<td>8873</td>
<td>36</td>
<td>23</td>
</tr>
<tr>
<td>Backward</td>
<td>8882</td>
<td>54</td>
<td>38</td>
</tr>
<tr>
<td>First</td>
<td>8845</td>
<td>31</td>
<td>25</td>
</tr>
<tr>
<td><strong>AVATAR off</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Standard</td>
<td>8110</td>
<td>26</td>
<td>5</td>
</tr>
<tr>
<td>Backward</td>
<td>8099</td>
<td>24</td>
<td>7</td>
</tr>
<tr>
<td>First</td>
<td>8029</td>
<td>20</td>
<td>6</td>
</tr>
</tbody>
</table>

- Interesting relation with AVATAR
- Kind of demonstrates the point about difficulty with experiments
Finding the subclause

Given $D \lor D'$ we need to decide which bit is $D$ and which bit is $D'$

Clearly trying all combinations will get boring (expensive)

Initial idea is to go linearly i.e. first 1,2,3...

It worked very well like this until we did something better...
Using Solving Under Assumptions

- Concept:
  - Assume some SAT variables $v_1, \ldots, v_n$ have a certain value
  - Run SAT solver and it finds unsat
  - Ask it for a minimal set of $x_i$ that were used in unsat

- In this context...
  - Let $D \lor D'$ be $l_1 \lor \ldots \lor l_n$ such that the grounding is $v_1 \lor \ldots \lor v_n$
  - Add $v_1 \lor \ldots \lor v_n$ as usual
  - Assume $\neg v_1, \ldots, \neg v_n$
  - $v_i$ is a first guess at $D'$
Going Further

- We can then minimise the set of assumptions
- Basically, step through the literals and see if they can be removed
- Three options
  - Don’t do it
  - In order
  - Randomized order (default)
## Experiment

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>off</td>
<td>8959</td>
<td>16</td>
</tr>
<tr>
<td>on</td>
<td>8965</td>
<td>21</td>
</tr>
<tr>
<td>randomized</td>
<td>8981</td>
<td>38</td>
</tr>
</tbody>
</table>

- So the default is best... that’s good
Overview
AVATAR Clauses

- In AVATAR with have A-Clauses i.e. clauses have assertions A that capture splitting context

- Reductions (like GS) need to be careful of assertions
Two Approaches

- Add assertions as additional SAT variables to every grounded clause

- Current Branch
  - Assume the current branch

- Full Model
  - Assume the full encoding of the model

- What we haven’t tried
  - Letting GS and AVATAR share a SAT solver
  - Using GS to reduce the assertions only
## Experiment

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Unique</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssnc=known</td>
<td></td>
<td></td>
</tr>
<tr>
<td>off</td>
<td>9030</td>
<td>131</td>
</tr>
<tr>
<td>current</td>
<td>6149</td>
<td>6</td>
</tr>
<tr>
<td>full</td>
<td>3250</td>
<td></td>
</tr>
<tr>
<td>ssnc=all</td>
<td></td>
<td></td>
</tr>
<tr>
<td>off</td>
<td>8615</td>
<td>47</td>
</tr>
<tr>
<td>current</td>
<td>933</td>
<td></td>
</tr>
<tr>
<td>full</td>
<td>699</td>
<td></td>
</tr>
<tr>
<td>ssnc=all_dependent</td>
<td></td>
<td></td>
</tr>
<tr>
<td>off</td>
<td>8678</td>
<td>16</td>
</tr>
<tr>
<td>current</td>
<td>5915</td>
<td></td>
</tr>
<tr>
<td>full</td>
<td>3416</td>
<td></td>
</tr>
<tr>
<td>ssnc=none</td>
<td></td>
<td></td>
</tr>
<tr>
<td>off</td>
<td>8832</td>
<td>43</td>
</tr>
<tr>
<td>current</td>
<td>6853</td>
<td></td>
</tr>
<tr>
<td>full</td>
<td>3586</td>
<td></td>
</tr>
</tbody>
</table>
Overview
Idea: replace SAT solver with SMT solver

- It’s a simple idea... we did it with AVATAR
- But, the idea of GS is to be cheap
- Let’s try it and find out!
Technical Issues

- Ground terms get translated into SMT language
- Non-ground terms get named propositionally again

- Make sure that assertions (which represent theory constraints) are also included!

- I had hoped to present some experimental results, but I forgot the last point so it was unsound
Overview
The Lookahead Idea

- Earlier when we were talking about good groundings to add we were trying to guess what groundings were already in the SAT solver.

- The next idea is to look and base our decision on what is actually there.

- Is this E-matching? (without the Equality bit)
Conclusions

- Global Subsumption is useful
- We can play with lots of bits
- There’s more playing to do