Revisiting Global Subsumption

Giles Reger[1] and Martin Suda[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 $D \vee D'$ can be replaced by D in S if $S_{gr} \models D$
- This is sound as D follows from S and subsumes $D \vee D'$
- If D is empty then S_{gr} is unsatisfiable and so is S

Global Subsumption: the Ground Case, an Example

- Consider

$$S = \left\{ \begin{array}{c} p(x) \vee q(a) \\ \neg p(x) \vee q(c) \\ f(a) = a \\ f(f(a)) \neq a \end{array} \right\} \quad S_{gr} = \left\{ \begin{array}{c} p(\bot) \vee q(a) \\ \neg p(\bot) \vee q(c) \\ f(a) = a \\ f(f(a)) \neq a \\ \end{array} \right\}$$

- $(D \vee D') = q(a) \vee q(b) \vee q(c)$
- $S_{gr} \models q(a) \vee q(c)$
- $q(a) \vee q(b) \vee q(c)$ can be replaced by $q(a) \vee q(c)$

Global Subsumption: the Ground Case, an Example

- Consider

$$S = \left\{ \begin{array}{c} p(x) \vee q(a) \\ \neg p(x) \vee q(c) \\ f(a) = a \\ f(f(a)) \neq a \end{array} \right\} \quad S_{gr} = \left\{ \begin{array}{c} 1 \vee 2 \\ \neg 1 \vee 3 \\ 4 \\ \neg 5 \\ \\ \end{array} \right\}$$

- $(D \vee D') = 2 \vee 6 \vee 3$
- $S_{gr} \models 2 \vee 3$
- $q(a) \vee q(b) \vee q(c)$ can be replaced by $q(a) \vee q(c)$

Global Subsumption: the Ground Case, an Example

- Consider

$$S = \left\{ \begin{array}{c} p(x) \vee q(a) \\ \neg p(x) \vee q(c) \\ f(a) = a \\ f(f(a)) \neq a \end{array} \right\} \quad S_{gr} = \left\{ \begin{array}{c} 1 \vee 2 \\ \neg 1 \vee 3 \\ 4 \\ \neg 5 \\ \neg 2, \neg 3 \end{array} \right\}$$

- $(D \vee D') = 2 \vee 6 \vee 3$
- $S_{gr} \models 2 \vee 3$
- $q(a) \vee q(b) \vee q(c)$ can be replaced by $q(a) \vee q(c)$

Global Subsumption: the Non-Ground Case

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

$$\frac{C \vee C'}{C}$$

  where $S_{gr} \models C\gamma$ for non-empty C' and injective substitution γ 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 $\neg(\forall \mathbf{x} C[\mathbf{x}])$ is $\exists \mathbf{x} \neg C[\mathbf{x}]$ so γ looks like the result of Skolemization

Example

- Take the following case:
  - $C = p(x, y) \vee r(x)$
  - $S = \{p(x, y) \vee r(x), p(x, x)\}$
- C cannot be reduced. Injectivity is important
  - If we do things wrong we can get $S_{gr} = \{p(a, b) \vee r(a), p(a, a)\}$
  - We check $\{p(a, a) \vee r(a), p(a, a), \neg p(a, a)\}$
  - 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 $S_{gr} = \{p(a, b) \vee r(a), p(a, a), p(a, b)\}$
  - We check $\{p(a, b) \vee r(a), p(a, a), p(a, b), \neg p(a, b)\}$
  - 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

	Total	Unique
Propagation Only	8935	61
Full	8920	46
Baseline	?	?

Overview

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(x) \vee \neg q(y) \vee r(y) \\ \neg p(x) \\ \\ \end{array} \right\}$$

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(a) \vee \neg q(b) \vee r(b) \\ \neg p(a) \\ \end{array} \right\}_{gr}$$

- $\gamma = [x \mapsto a, y \mapsto b]$ for C_1

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(a) \vee \neg q(b) \vee r(b) \\ \neg p(a) \\ \end{array} \right\}_{gr} \models \neg q(x) \vee r(x)$$

- $\gamma = [x \mapsto a, y \mapsto b]$ for C_1

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(a) \vee \neg q(b) \vee r(b) \\ \neg p(a) \\ \end{array} \right\}_{gr} \models \neg q(a) \vee r(a)$$

- $\gamma = [x \mapsto a, y \mapsto b]$ for C_1

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(b) \vee \neg q(a) \vee r(a) \\ \neg p(b) \\ \end{array} \right\}_{gr} \models \neg q(a) \vee r(a)$$

- $\gamma = [x \mapsto b, y \mapsto a]$ for C_1

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(a) \vee \neg q(b) \vee r(b) \\ \neg p(a) \\ \end{array} \right\}_{gr} \models \neg q(b) \vee r(b)$$

- $\gamma = [x \mapsto a, y \mapsto b]$ for C_1

What do we want?

- Consider

$$\left\{ \begin{array}{c} C_1 = p(a) \vee \neg q(b) \vee r(b) \\ \neg p(a) \\ p(b) \vee \neg q(a) \vee r(a) \\ \neg p(b) \end{array} \right\}_{gr} \models \neg q(a) \vee r(a)$$

- $\gamma = [x \mapsto a, y \mapsto b]$ for C_1

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

	Total	Unique A	Unique T
AVATAR on			
Standard	8873	36	23
Backward	**8882**	**54**	38
First	8845	31	25
AVATAR off			
Standard	8110	26	5
Backward	8099	24	7
First	8029	20	6

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

Overview

Finding the subclause

- Given $D \vee D'$ 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 $D \vee D'$ be $l_1 \vee \ldots \vee l_n$ such that the grounding is $v_1 \vee \ldots \vee v_n$
- Add $v_1 \vee \ldots \vee v_n$ 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

	Total	Unique
off	8959	16
on	8965	21
randomized	8981	38

- 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

	Total	Unique
ssnc=known		
off	9030	131
current	6149	6
full	3250	
ssnc=all		
off	8615	47
current	933	
full	699	
ssnc=all_dependent		
off	8678	16
current	5915	
full	3416	
ssnc=none		
off	8832	43
current	6853	
full	3586	

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 **E**quality bit)

Conclusions

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