ETH zürich

Alexander J. Summers

Program Verification

Exercise Sheet 4: Quantifiers

NOTE: with the material covered on 14th of March, you should be able to do the first two assignments.

Assignment 1 (Rewriting and Skolemization)

Transform the following formula into (an equisatisfiable formula in) extended CNF (all variables/constants are of some uninterpreted sort T; c, g and s are uninterpreted):

$$\exists z.\neg((\forall n.g(n,z) \wedge \exists m.(\neg n = z \Rightarrow s(m) = n)) \Rightarrow c = z) \wedge \forall w.\neg s(s(s(w))) = s(s(c))$$

When transforming the formula, try to simplify it as much as you can (this will help in the assignment 3).

Assignment 2 (Applying MBQI)

Consider the following formula:

$$\exists x.(\forall y.\neg\ f(y){=}x) \wedge (\forall z.\neg\ f(z){=}z \wedge \neg\ f(f(z)){=}z)$$

Does this formula have a model? If so, is there a lower bound on the size of possible models?
Apply Model-Based Quantifier Elimination to try to find a model for this formula – what happens? You should try not to "guess" the right model straight away – keep the model as simple as possible with respect to the ground constraints it needs to satisfy so far, as you work.

Assignment 3 (E-graphs and E-matching)

Take your answer from Assignment 1, and construct an E-graph to represent the ground facts that will be added to the E-graph during initial DPLL search (without quantifier instantiations). What would be appropriate triggers to add to the (two) $\forall$-quantifiers? Show how, once equipped with these triggers, E-matching can show that the original formula is unsatisfiable.

Assignment 4 (Axiomatising Duplicate-Freeness)

Suppose we model infinite integer arrays (as a uninterpreted sort), using a function *lookup*(a, i) to represent the value of looking-up (integer) index i of array a. Suppose further that we want to express that an array a contains no duplicate values.

One way to do this, would be via a quantifier:

$$\forall i : \mathrm{Int}, j : \mathrm{Int}. \neg\ i{=}j \Rightarrow \neg\ \textit{lookup}(a, i){=}\textit{lookup}(a, j)$$

Suppose now that we want to use e-matching with this quantifier, for example to deduce that conjoining *lookup*$(a, 0)$ = *lookup*$(a, 1)$ gives us `unsat`. What triggers would we choose? How many quantifier instantiations will potentially be made, in terms of the number of ground *lookup* function applications in the input problem?

Can you think of an alternative way to express having no duplicate values, which would reduce the potential number of quantifier instantiations?