

Program Verification

Exercise Sheet 4: Quantifiers

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 2).

Hint: push all negations as far inside the formula as possible, including quantifiers, then apply skolemization.

Assignment 2 (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 3 (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:

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

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?