Subair Kirimow 12321260

Analysis of my encoding

What Are the Constraints in the SMT-LIB2 Encoding Expressing?

The encoding has the following key components:

Input Constraints: These enforce the preconditions that both x and y must be greater than 0, assume(x > 0) and assume(y > 0) statements in the original program.

Program Semantics: The program changes the array A in two steps. First, it sets A[x] to y. Then, it checks if x is equal to y. If they are equal, it sets A[y] to x. If not, it sets A[x] to y again. In the SMT encoding, these changes are done step by step using the store operation, which updates the array. The final result is a new version of the array that shows what it looks like after the program finishes.

Formula F: The goal is to prove that after executing the program, the formula $2 * read(A, x) = 2 * y \wedge 2 * read(A, x) > 0$ holds. This is encoded as a logical conjunction, with read(A, x) interpreted as selecting the value at index x from the „final“ array.

How Do You Conclude Validity of the Formula F with Respect to p from Your Encoding, Using Z3?

To show that the formula F is always true after the program runs, we check if F holds for all values of x and y that meet the input conditions. Since Z3 is a tool that looks for examples where something *can* happen, we ask it the opposite of what we want: we tell Z3 to try and find a case where the input conditions and the program behavior are correct, but F is false.

If Z3 can't find such a case, it means there's no way for F to be false when the inputs and program logic are followed. In that situation, Z3 returns unsat, which tells us that F is always true under those conditions.

In our case, Z3 returns unsat, so we've confirmed that F is always valid after the program runs. This shows that our encoding is correct and the formula is logically guaranteed.