

# Worked SMT Example

Alan J. Hu  
for CpSc 513  
Univ. of British Columbia

2011 October 6

These are supplementary notes, showing a worked example of how an SMT solver might solve a problem. It's nice to see an example worked out in its entirety sometimes, just to convince oneself that an idea might actually work.

## 1 Software Verification Example

The code we are going to verify is basically the running example we've used in lectures already:

```
if (a[i] > a[i+1]) {
  t = a[i];
  a[i] = a[i+1];
  a[i+1] = t;
}
assert a[i] <= a[i+1];
```

## 2 Verification Condition via Symbolic Execution

This is fairly straightforward. Let the initial values of  $a$ ,  $i$ , and  $t$  be  $a_0$ ,  $i_0$ , and  $t_0$ .

The test in the `if` statement is the path condition:

$$\text{select}(a_0, i_0) > \text{select}(a_0, i_0 + 1) \tag{1}$$

When this is false, we will follow the `else` branch, and the variables will have their original values.

Along the `then` branch, symbolic execution proceeds to update:

$$t : \text{select}(a_0, i_0) \tag{2}$$

$$a : \text{update}(a_0, i_0, \text{select}(a_0, i_0 + 1)) \tag{3}$$

and then updating `a` again, so we plug expression 3 into another update expression:

$$a : \text{update}(\text{expr 3}, i_0 + 1, \text{expr 2}) \quad (4)$$

or fully plugging everything in:

$$a : \text{update}(\text{update}(a_0, i_0, \text{select}(a_0, i_0 + 1)), i_0 + 1, \text{select}(a_0, i_0)) \quad (5)$$

For convenience, call this expression FOO.

So, after the code finishes, the value of `a` will be:

$$a : (\text{select}(a_0, i_0) > \text{select}(a_1, i_0 + 1)) ? \text{FOO} : a_0 \quad (6)$$

Call this expression BAR.

The assertion we are checking then becomes:

$$\text{select}(\text{BAR}, i_0) \leq \text{select}(\text{BAR}, i_0 + 1) \quad (7)$$

For the program to be correct, that expression should be valid, so we can check whether it's complement is unsatisfiable.

$$\text{select}(\text{BAR}, i_0) > \text{select}(\text{BAR}, i_0 + 1) \quad (8)$$

### 3 Purification

Expression 8 is a single term, but it mixes the array theory with the arithmetic theory, so we have to purify it. I am going to generate the terms as go along. At the top level, it's just a comparison in the arithmetic theory:

$$(x_0 > x_1) \quad (9)$$

and then we have to constrain  $x_0$  and  $x_1$  to have the correct values:

$$(x_0 = \text{select}(x_2, i_0)) \wedge (x_1 = \text{select}(x_2, x_3)) \quad (10)$$

where now  $x_2$  and  $x_3$  are placeholders for terms that we haven't purified yet.  $x_3$  is easy:

$$(x_3 = i_0 + 1) \quad (11)$$

and  $x_2$  is just the expression BAR (expression 6), which we still have to purify. I'm going to simplify the if-then-else expression a bit, just to save some time, so we have:

$$(x_4 > x_5) ? (x_2 = \text{FOO}) : (x_2 = a_0) \quad (12)$$

where we need to fill in the FOO and purify it, giving:

$$(x_4 > x_5) ? (x_2 = \text{update}(\text{update}(a_0, i_0, \text{select}(a_0, x_3)), x_3, \text{select}(a_0, i_0))) : (x_2 = a_0) \quad (13)$$

Note that I spotted that we had already defined  $x_3 = i_0 + 1$ , so I substituted that in, instead of introducing new variables also equal to  $i_0 + 1$  (which would have worked as well). Now, we still have to define  $x_4$  and  $x_5$ :

$$(x_4 = \text{select}(a_0, i_0)) \wedge (x_5 = \text{select}(a_0, x_3)) \quad (14)$$

Cool! That completes the purification. All terms are either purely linear arithmetic, or purely array theory.

## 4 Boolean Skeleton

Fudge... it's hard for me in LaTeX to go back and label all the terms, but the basic Boolean skeleton is just the AND of a bunch of variables except for the if-then-else in term 13. So, the Boolean skeleton will look like (using capital letters for the Boolean variables):  $A \wedge B \wedge \dots \wedge (X?Y : Z)$ . So, there are only two satisfying assignments to consider: when  $X$  and  $Y$  are both true and  $Z$  is a don't-care, and when  $\neg X$  and  $Z$  are true and  $Y$  is a don't-care (and all the other variables are true).

## 5 Communicating Theory Solvers 1

OK, let's do the first case, where  $X$  is true. In this case, we have the following terms for the linear solver:

$$(x_0 > x_1) \tag{15}$$

$$(x_3 = i_0 + 1) \tag{16}$$

$$(x_4 > x_5) \tag{17}$$

and the following terms for the array solver:

$$(x_0 = \text{select}(x_2, i_0)) \tag{18}$$

$$(x_1 = \text{select}(x_2, x_3)) \tag{19}$$

$$(x_2 = \text{update}(\text{update}(a_0, i_0, \text{select}(a_0, x_3)), x_3, \text{select}(a_0, i_0))) \tag{20}$$

$$(x_4 = \text{select}(a_0, i_0)) \tag{21}$$

$$(x_5 = \text{select}(a_0, x_3)) \tag{22}$$

All the variables except  $a_0$  are interface variables.

The linear solver can immediately tell the array solver that  $(x_0 \neq x_1)$ ,  $(x_3 \neq i_0)$ , and  $(x_4 \neq x_5)$ , but it can't deduce anything more than that for now.

The array solver can simplify the expression for  $x_2$  a bit by using the definitions of  $x_4$  and  $x_5$ :

$$(x_2 = \text{update}(\text{update}(a_0, i_0, x_5), x_3, x_4)) \tag{23}$$

Next, the array solver can use the array axiom to see that

$$x_1 = \text{select}(x_2, x_3) \tag{24}$$

$$= x_4 \tag{25}$$

Slightly more complex, since the arithmetic solver told us that  $(x_3 \neq i_0)$ , the array solver also can apply the array axiom twice to get:

$$x_0 = \text{select}(x_2, i_0) \tag{26}$$

$$= \text{select}(\text{update}(a_0, i_0, x_5), i_0) \tag{27}$$

$$= x_5 \tag{28}$$

So, the array solver can now tell the linear solver that  $(x_1 = x_4)$  and  $(x_0 = x_5)$ .

Now, the linear solver can figure out that  $x_0 > x_1 = x_4 > x_5 = x_0$ . That's a contradiction! So, this case is unsatisfiable.

## 6 Communicating Theory Solvers 2

The other case we have to consider is when  $X$  is false and  $Z$  is true. In that case, we'd have the following terms for the linear solver:

$$(x_0 > x_1) \tag{29}$$

$$(x_3 = i_0 + 1) \tag{30}$$

$$\neg(x_4 > x_5) \tag{31}$$

and the following terms for the array solver:

$$(x_0 = \text{select}(x_2, i_0)) \tag{32}$$

$$(x_1 = \text{select}(x_2, x_3)) \tag{33}$$

$$(x_2 = a_0) \tag{34}$$

$$(x_4 = \text{select}(a_0, i_0)) \tag{35}$$

$$(x_5 = \text{select}(a_0, x_3)) \tag{36}$$

In this case, the array solver immediately deduces that  $(x_0 = x_4)$  and  $(x_1 = x_5)$ , which it communicates to the linear solver.

The linear solver then says  $x_4 = x_0 > x_1 = x_5$ , but it also has the term  $\neg(x_4 > x_5)$  which is a contradiction, so this case is unsatisfiable, too.

Since the only two satisfying assignments to the Boolean skeleton produce solutions that are unsatisfiable in the theory solvers, the whole expression is unsatisfiable, so the assertion about the program must be valid.