EECS 219C: Formal Methods — Assignment 2

Parker Ziegler

March 1, 2022

1. Bit Twiddling Hacks

a. Equivalence of `f1` and `f2`

My SMT-LIB encoding to check equivalence of `f1` and `f2` is located in `1a.smt2`. Based on my encoding, `f1` and `f2` **are not equivalent**. Z3 provides the counterexample $x = 1$, in which `f1(x)` evlauates to -1 while `f2(x)` evaluates to 1. The full output returned by Z3 is:

```
(
  (define-fun v_1 () (_ BitVec 32)
    #x00000000)
  (define-fun x () (_ BitVec 32)
    #x00000001)
  (define-fun v_0 () (_ BitVec 32)
    #xffffffff)
  (define-fun ret_2 () (_ BitVec 32)
    #x00000001)
  (define-fun v_2 () (_ BitVec 32)
    #x00000001)
  (define-fun ret_1 () (_ BitVec 32)
    #xffffffff)
)
```

A more readable output provided by Z3 using its Python API yields the following `list` showing assignments in the counterexample:

```
[x = 1, v_0 = -1, ret_1 = -1, v_1 = 0, v_2 = 1, ret_2 = 1]
```

b. Equivalence of `f3` and `f4`

My SMT-LIB encoding to check equivalence of `f3` and `f4` is located in `1b.smt2`. Based on my encoding, `f3` and `f4` **are equivalent**.

c. Synthesizing an equivalent function for `f2` without bit-wise operators

To synthesize an equivalent function for `f2`, we use syntax-guided synthesis (SyGuS) with Linear Integer Arithmetic (LIA) as the background theory. My encoding of the synthesis problem in SyGuS-IF format is located in `1c.sy`. I used the CVC4 solver to synthesize the following solution.

```
(define-fun f2Equiv ((x Int)) Int (ite (<= 0 x) x (- 0 x)))
```

I restricted the grammar to basic arithmetic operators, conditionals, and arithmetic comparison operators. Reading through the implementation of `f2` made it seem that `f2` returns the absolute value of its input `x`. Therefore, I used the constraint `(= (f2Equiv x) (abs x))` to describe the semantics of the function we aim to synthesize.

2. Sum-Sudoku

a. Formulate an SMT instance that finds a solution to Sum-Sudoku puzzles

My encoding of the Sum-Sudoku problem is located in `sumsudoku.py`. My encoding applies the following constraints:

1. Ensure that every integer in a row is distinct; that is, the same integer cannot appear twice in a given row. We encode this using `z3.Distinct`.
2. Ensure that every integer in a column is distinct; as above, we encode this with `z3.Distinct`.
3. Ensure that all values in a row sum up to their required row sum. We encode this using `z3.Sum` to sum variables and compare the resulting value using numerical equality to its row sum (which is given).

We apply the theories of equality and uninterpreted functions (EUF) and linear integer arithmetic (LIA) to specify these constraints. EUF can be used to represent the `Distinct` API in `Z3`; for example, we can think of `Distinct(x, y, z)` as a conjunction of pairwise disequalities $x \neq y \wedge y \neq z \wedge x \neq z$. We use LIA as the backing theory for `Sum`.

b. Assigning row and column values for a unique Sum-Sudoku problem

3. Bag of Chips

a. Prove that the choose-and-replace process on the bag of chips always halts

Our goal is to show that you can only execute the described choose-and-replace process on our bag of chips a finite number of times. To do this, we define a relation $<_{ybr}$ over our state tuples $(\#red_i, \#blue_i, \#yellow_i)$. Our relation has the following semantics:

$(\#red_{i+1}, \#blue_{i+1}, \#yellow_{i+1}) <_{ybr} (\#red_i, \#blue_i, \#yellow_i)$ iff

1. $\#yellow_{i+1} < \#yellow_i$, OR
2. $\#yellow_{i+1} = \#yellow_i \wedge \#blue_{i+1} < \#blue_i$, OR
3. $\#yellow_{i+1} = \#yellow_i \wedge \#blue_{i+1} = \#blue_i \wedge \#red_{i+1} < \#red_i$

To prove termination, we need to show that for an arbitrary state $(\#red_i, \#blue_i, \#yellow_i)$, a finite number of choose-and-replace operations remain; that is, that $<_{ybr}$ is a well-founded relation. The proof is by induction on the triple $(\#red_i, \#blue_i, \#yellow_i)$.

Base Case

There are four base cases to consider. If $(\#red_i, \#blue_i, \#yellow_i)$ is equal to $(0, 0, 0)$, the process has terminated. If $(\#red_i, \#blue_i, \#yellow_i)$ is in the set $\{(0, 0, 1), (0, 1, 0), (1, 0, 0)\}$, then only one step remains — removing the last chip. This proves termination of the base case.

Inductive Step

We take as the inductive hypothesis that for any state $(\#red_i, \#blue_i, \#yellow_i)$, the $i + 1$ state:

$$(\#red_{i+1}, \#blue_{i+1}, \#yellow_{i+1})$$

only has a finite number of choose-and-replace operations that remain. We now apply case analysis based on the semantics of the choose-and-replace operation.

Case 1: One of the removed chips is red

Starting from an arbitrary state $(\#red, \#blue, \#yellow)$ and removing at least one red chip from the bag yields one of three possible states:

1. $(\#red - 1, \#blue - 1, \#yellow)$
2. $(\#red - 1, \#blue, \#yellow - 1)$
3. $(\#red - 2, \#blue, \#yellow)$

The second restriction of the semantics of the $<_{ybr}$ relation shows that:

$$(\#red - 1, \#blue - 1, \#yellow) <_{ybr} (\#red, \#blue, \#yellow)$$

The first restriction shows that:

$$(\#red - 1, \#blue, \#yellow - 1) <_{ybr} (\#red, \#blue, \#yellow)$$

Finally, the third restriction shows that:

$$(\#red - 2, \#blue, \#yellow) <_{ybr} (\#red, \#blue, \#yellow)$$

Therefore, all new states abide by our well-founded relation for the first case.

Case 2: Both of the removed chips are yellow

Starting from an arbitrary state $(\#red, \#blue, \#yellow)$ and removing two yellow chips from the bag yields a new state $(\#red, \#blue + 5, \#yellow - 1)$. By the first resitrction of the semantics of the $<_{ybr}$ relation, we can indeed see that:

$$(\#red, \#blue + 5, \#yellow - 1) <_{ybr} (\#red, \#blue, \#yellow)$$

Case 3: One of the removed chips is blue

Starting from an arbitrary state $(\#red, \#blue, \#yellow)$ and removing one blue chip from the bag yields one of two potential states:

1. $(\#red + 10, \#blue - 2, \#yellow)$
2. $(\#red + 10, \#blue - 1, \#yellow - 1)$

For the first case, the second restriction of the semantics of the $<_{ybr}$ relation allow us to conclude that:

$$(\#red + 10, \#blue - 2, \#yellow) <_{ybr} (\#red, \#blue, \#yellow)$$

For the second case, the firt restriction of the semantics of the $<_{ybr}$ relation allow us to conclude that:

$$(\#red + 10, \#blue - 1, \#yellow - 1) <_{ybr} (\#red, \#blue, \#yellow)$$

In all cases, we've shown that:

$$(\#red_{i+1}, \#blue_{i+1}, \#yellow_{i+1}) <_{ybr} (\#red_i, \#blue_i, \#yellow_i)$$

This completes the inductive argument and, with it, the proof of termination.

b. Prove that the variant of the bag of chips problem always halts

Our goal is the same as in part (a), but we need to modify our relation slightly to prove termination in this variant of the bag of chips problem. We define this new relation, $<_{ryb}$, to have the following semantics:

$(\#red_{i+1}, \#blue_{i+1}, \#yellow_{i+1}) <_{ryb} (\#red_i, \#blue_i, \#yellow_i)$ iff

1. $\#red_{i+1} < \#red_i$, OR
2. $\#red_{i+1} = \#red_i \wedge \#yellow_{i+1} < \#yellow_i$, OR
3. $\#red_{i+1} = \#red_i \wedge \#yellow_{i+1} = \#yellow_i \wedge \#blue_{i+1} < \#blue_i$

This relation is similar to $<_{ybr}$ with the exception of comparison order. While $<_{ybr}$ compares values of two state triples in the order yellow $\rightarrow$ blue $\rightarrow$ red, our new relation $<_{ryb}$ compares these values in the order red $\rightarrow$ yellow $\rightarrow$ blue. As above, our proof is by induction on the triple $(\#red_i, \#blue_i, \#yellow_i)$.

Base Case

The base case is identical to the proof in (a); therefore, we elide it here.

Inductive Step

We take as the inductive hypothesis that for any state $(\#red_i, \#blue_i, \#yellow_i)$, the $i+1$ state:

$$(\#red_{i+1}, \#blue_{i+1}, \#yellow_{i+1})$$

only has a finite number of choose-and-replace operations that remain. We now apply case analysis based on the semantics of the choose-and-replace operation.

Case 1: One of the removed chips is red and the other is yellow

Starting from an arbitrary state $(\#red, \#blue, \#yellow)$ and removing one yellow chip and one red chip yields a new state $(\#red, \#blue, \#yellow - 1)$. By the second restriction of the semantics of the $<_{ryb}$ relation, we can see that:

$$(\#red, \#blue, \#yellow - 1) <_{ryb} (\#red, \#blue, \#yellow)$$

Case 2: Both removed chips are yellow

Starting from an arbitrary state $(\#red, \#blue, \#yellow)$ and removing two yellow chips yields a new state $(\#red, \#blue + 5, \#yellow - 2)$. By the second restriction of the semantics of the $<_{ryb}$ relation, we can see that:

$$(\#red, \#blue + 5, \#yellow - 2) <_{ryb} (\#red, \#blue, \#yellow)$$

Case 3: One of the removed chips is blue and the other is red

Starting from an arbitrary state $(\#red, \#blue, \#yellow)$ and removing one blue chip and one red chip yields a new state $(\#red - 1, \#blue + 9, \#yellow)$. By the first restriction of the semantics of the $<_{ryb}$ relation, we can see that:

$$(\#red - 1, \#blue + 9, \#yellow) <_{ryb} (\#red, \#blue, \#yellow)$$

This completes the inductive argument and, with it, the proof of termination.