Exercise 1

Encoding sketches

Reliable and Interpretable Artificial Intelligence 2017
ETH Zürich

September 27, 2017

```
1  t    := a[0]
2  a[0] := a[0] + a[1]
3  a[1] := a[1] + a[2]
4  a[2] := a[2] + t
5  c    := b[a[0]] + b[a[1]] + b[a[2]]
```

Figure 1: A program with integer arrays.

Problem 1. (A warm-up for Problem 2.) Consider the program in Figure 1. Encode the program's semantics in the Z3 SMT solver (`http://rise4fun.com/z3/tutorial`). Assume that the array bounds of b are from 0 to 63 included. Write down a formula asserting that `b` is never accessed outside its bounds. Use Z3 to find concrete values for `a[0]`, `a[1]`, and `a[2]` that guarantee that satisfy the formula.

```
1  Word Pop_Count(Word x)
2  {
3      Word n;
4      for (n = ??; x {| < | = | > |} ??; n += ??) {
5          x = ?? && (?? - ??);
6      }
7      return ??;
8  }
```

Figure 2: A sketch of *Pop_Count*.

Problem 2. Encode in Z3 the `Pop_Count` sketch in Figure 2. Each hole can be replaced by either a fresh constant or any program variable in scope. You can assume 8-bit words and a bounded number of loop iterations. Provide a set of input-output examples as a specification. Run Z3 to synthesize a correct implementation of `Pop_Count`.