VU Programm- und Systemverifikation
Homework: SMT

(15 points)

May 13, 2015

Given the following code example:

```
double S = input();
assert(0 < S && S < 3);
double a = S;
double c = S - 1.0;

while (c >= 0.01 || c <= -0.01) {
    a = a - a * c / 2.0;
    c = c * c * (c - 3.0) / 4.0;
    assert(S * (1 + c) = a * a && S > 0 && S < 3);
}

printf("sqrt of %f is approx. %f\n", S, a);

assert(a * a > S * 0.99);
assert(a * a < S * 1.01);
```

and its loop invariant

$$S \cdot (1 + c) = a^2 \wedge S > 0 \wedge S < 3$$

Tasks: Following the example of the file `loop.smt` discussed during the lecture and also given on the next page:

1. encode the transition relation of the loop in Z3
2. write a Z3 assertion and check that the precondition (initialization) implies the invariant
3. write a Z3 assertion and check that if the invariant holds before one iteration, it also holds after the iteration
4. write a Z3 assertion and check that the assertion holds upon leaving the loop
5. write a Z3 assertion and check that $|c|$ is decreasing if the loop body is executed under the assumption $-1 < c < 2$.

Upload a text file called `assignment5.smt` with your solutions to TUWEL by May 27, 2015. Make sure that the file contains your name and ID (as a comment). Z3 **must not** report syntax errors when called with the `-smt2` option.

```
;;
;; By calling "z3 -smt2 loop.smt" one can check the assertion
;; in the following code using the loop invariant: m * x = n * y + z
;;
;; Igor Konnov, Josef Widder, 2013. v1.0
;;

;; int n = input();
;; int x = input();

;; int m = n;
;; int y = x;
;; int z = 0;

;; assume(n >= 0);

;; while (n > 0) {
;;   if (n % 2) {
;;      z += y;
;;   }
;;   y *= 2;
;;   n /= 2;
;; }

;; assert(z == m * x);


(declare-const n Int)
(declare-const x Int)
(declare-const m Int)
(declare-const y Int)
(declare-const z Int)


(declare-const n2 Int)
(declare-const y2 Int)
(declare-const z2 Int)

(define-fun loopcond () Bool (> n 0))

(define-fun loopbody () Bool
  (if loopcond
      (and (if (= 1 (mod n 2))
         (= z2 (+ z y))
       (= z2 z))
     (= y2 (* y 2))
     (= n2 (/ n 2)))
    (and (= z2 z)
   (= y2 y)
   (= n2 n))))

(define-fun invariant () Bool (and
              (>= n 0)
              (>= m 0)
              (= (* m x) (+ z (* n y)))))

(define-fun invariantpost () Bool (and
           (>= n2 0)
           (>= m 0)
           (= (* m x) (+ z2 (* n2 y2)))))
```

```
;;
;; check that the precondition implies invariant
;;
(push)
  (assert (not (=>
     (and (= m n) (= x y) (= z 0) (>= n 0) )
     invariant
     )))
 (check-sat)
(pop)


;;
;; check invariant:
;;
(push)
(assert (not (=>
        (and invariant loopbody)
        invariantpost)))
  (check-sat)
  ;;(get-model)
(pop)


;;
;; check assertion after leaving loop
;;
(push)
  (assert (not (=>
    (and invariant (not loopcond))
    (= z (* m x)))))

  (check-sat)
 ;; (get-model)
(pop)


;;
;; check that loop terminates (i.e., if n>0 then n decreases)
;;
(push)
  (assert (not (=>
    (and loopcond loopbody)
    (< n2 n))))

  (check-sat)
  ;;(get-model)
(pop)
```