CS 243 Section 8(SMT)

Winter 2021

March 12, 2021

Path Sensitive Analysis with SMT

Consider the following C++ function:

```
void small_network(double a, double b, double &g, double &h) {
    double c, d, e, f;
    c = a - b;
    d = -2a + 3b;
    if (c <= 0){
        e = 0;
    } else {
        e = c;
    }
    if (d <= 0){
        f = 0;
    } else {
        f = d;
    }
    g = 2e - 3f;
    h = e - f;
}
```

1. Rewrite the program in SSA form.

```
c_0 = a_0 - b_0;
d_0 = -2a_0 + 3b_0;
phi_0 = (c_0 <= 0);
e_0 = 0;
e_1 = c_0;
e_2 = phi_0 ? e_0 : e_1;
phi_1 = (d0 <= 0);
f_0 = 0;
f_1 = d_0;
f_2 = phi_1 ? f_0 : f_1;
g_0 = 2e_2 - 3f_2;
h_0 = e_2 - f_2;
```

2. Translate the program to SMT.

3. Check whether g can be greater than h when a is non-positive and b is non-negative.

4. Check whether g can be greater than h when a and b are both non-negative.

```
(set-logic ALL)
(set-option :produce-models true)
(set-option :incremental true)
(declare-const a_0 Real)
(declare-const b_0 Real)
(declare-const c_0 Real)
(declare-const d_0 Real)
(declare-const e_0 Real)
(declare-const e_1 Real)
(declare-const e_2 Real)
(declare-const f_0 Real)
(declare-const f_1 Real)
(declare-const f_2 Real)
(declare-const g_0 Real)
(declare-const h_0 Real)
(declare-const phi_0 Bool)
(declare-const phi_1 Bool)


(assert (= c_0 (- a_0 b_0)))
(assert (= d_0 (+ (* (- 2) a_0) (* 3 b_0))))
(assert (= phi_0 (<= c_0 0)))
(assert (= e_0 0))
(assert (= e_1 c_0))
(assert (= e_2 (ite phi_0 e_0 e_1)))
(assert (= phi_1 (<= d_0 0)))

(assert (= f_0 0))
(assert (= f_1 d_0))
(assert (= f_2 (ite phi_1 f_0 f_1)))
(assert (= g_0 (- (* 2 e_2) (* 3 f_2))))
(assert (= h_0 (- e_2 f_2)))

(check-sat)
(get-model)

(push)
(assert (<= a_0 0))
```

```
(assert (>= b_0 0))
(assert (> g_0 h_0))
(check-sat)
(get-model)
(pop)


(push)
(assert (>= a_0 0))
(assert (>= b_0 0))
(assert (> g_0 h_0))

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