EECS 219C: Computer-Aided Verification

Satisfiability Modulo Theories

Examples Used in Lecture

Sanjit A. Seshia
EECS, UC Berkeley

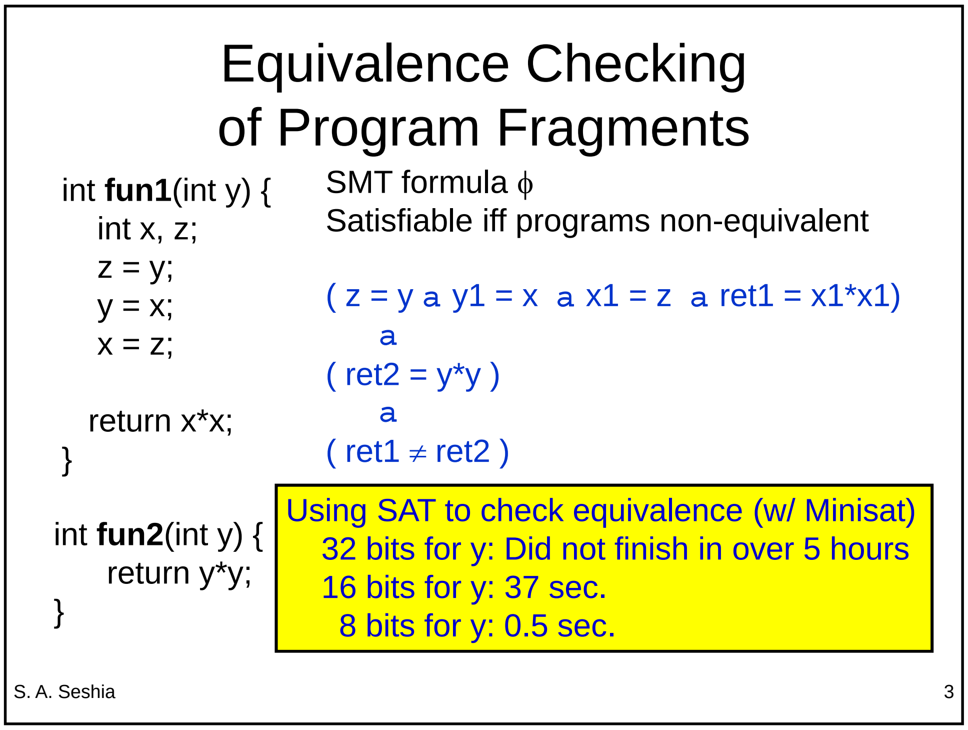
Equivalence Checking of Program Fragments

```
int fun1(int y) {
    int x, z;
    z = y;
    y = x;
    x = z;

   return x*x;
}

int fun2(int y) {
     return y*y;
}
```

SMT formula ϕ
Satisfiable iff programs non-equivalent

(z = y a y1 = x a x1 = z a ret1 = x1*x1)
a
(ret2 = y*y)
a
(ret1 $\neq$ ret2)

What if we use SAT to check equivalence?



Equivalence Checking of Program Fragments

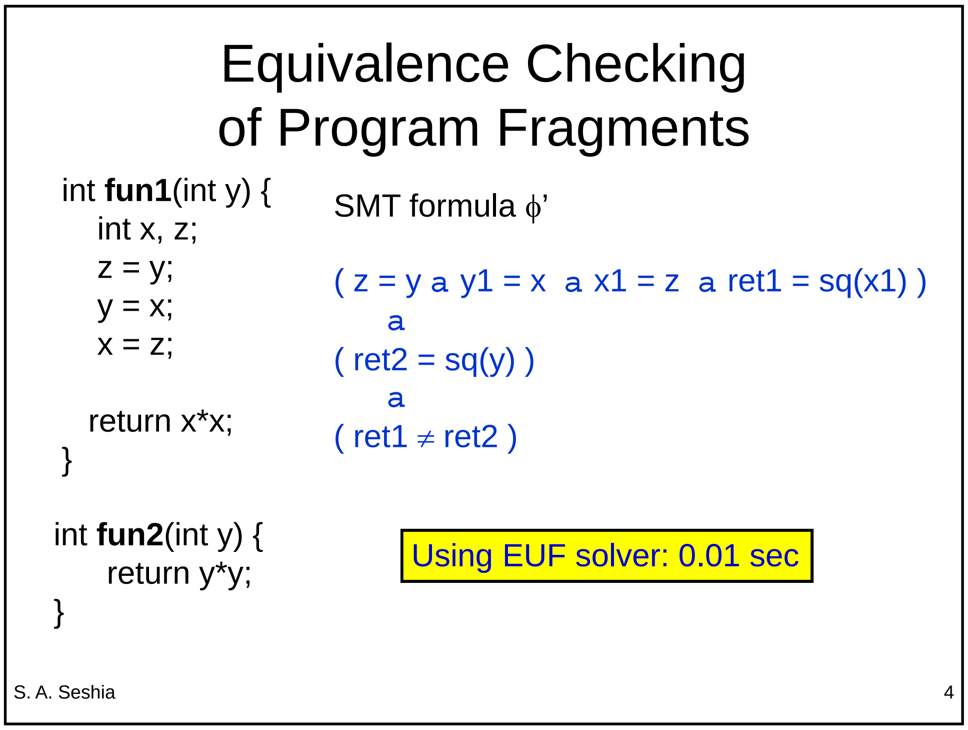
```
int fun1(int y) {
    int x, z;
    z = y;
    y = x;
    x = z;

   return x*x;
}

int fun2(int y) {
      return y*y;
}
```

SMT formula ϕ
Satisfiable iff programs non-equivalent

(z = y a y1 = x a x1 = z a ret1 = x1*x1)
a
(ret2 = y*y)
a
(ret1 ≠ ret2)

Using SAT to check equivalence (w/ Minisat)
32 bits for y: Did not finish in over 5 hours
16 bits for y: 37 sec.
8 bits for y: 0.5 sec.

S. A. Seshia 3



Equivalence Checking of Program Fragments

```
int fun1(int y) {
    int x, z;
    z = y;
    y = x;
    x = z;

   return x*x;
}

int fun2(int y) {
      return y*y;
}
```

SMT formula ϕ’

(z = y a y1 = x a x1 = z a ret1 = sq(x1))
a
(ret2 = sq(y))
a
(ret1 ≠ ret2)

Using EUF solver: 0.01 sec

S. A. Seshia 4

Equivalence Checking of Program Fragments

```
int fun1(int y) {
    int x;
    x = x ^ y;
    y = x ^ y;
    x = x ^ y;

   return x*x;
}

int fun2(int y) {
      return y*y;
}
```

Does EUF still work?

No!
Must reason about bit-wise XOR.

Need a solver for bit-vector arithmetic.

Solvable in less than a sec. with a current bit-vector solver.

Equivalence Checking of Program Fragments

```
int fun1(int y) {
    int x[2];
    x[0] = y;
    y = x[1];
    x[1] = x[0];

   return x[1]*x[1];
}
```

```
int fun2(int y) {
      return y*y;
}
```

How can we express the equivalence checking problem as an SMT formula with arrays?

Equivalence Checking of Program Fragments

```
int fun1(int y) {
    int x[2];
    x[0] = y;
    y = x[1];
    x[1] = x[0];

   return x[1]*x[1];
}


 int fun2(int y) {
       return y*y;
 }
```

SMT formula ϕ''

[x1 = store(x,0,y) a y1 = select(x1,1)
a x2 = store(x1,1,select(x1,0))
a ret1 = sq(select(x2,1))]
a
(ret2 = sq(y))
a
(ret1 $\neq$ ret2)

EUF

- Example:

g(g(g(x))) = x
a g(g(g(g(g(x))))) = x
a g(x) $\neq$ x

Difference Logic

x_1 ¿ x_2

x_3 ‰ 0

$x_2 + 3$ ¿ x_1

$x_1 + 1$ ‰ x_3

$x_2 + 1$ ¿ 0

$x_4 + 2$ ¿ 0

x_4 ‰ $x_2 - 2$

Theory of Arrays

- Two main axioms: For all A, i, j, d
  - select(store(A,i,d), i) = d
  - select(store(A,i,d), j) = select(A,j), if i $\neq$ j
- Decision procedure operates by performing case-splits
- Example:

```
int a[10];
int fun3(int i) {
   int j;
   for(j=0; j<10; j++) a[j] = j;
   assert(a[i] <= 5);
}
```

Theory of Arrays

- Two main axioms: For all A, i, j, d
  - select(store(A,i,d), i) = d
  - select(store(A,i,d), j) = select(A,j), if i $\neq$ j
- Decision procedure operates by performing case-splits
- Example:

a[0] = 0 a a[1] = 1 a a[2] = 2 a … a[9] = 9 a a[i] > 5