CBMC

30/11/16

Kalev Alpernas

C Program

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

- Is there a potential bug in the program?

C Program

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

- Is there a potential bug in the program?
  - Array access

C Program

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

- Is there a potential bug in the program?
  - Array access
- Need to find inputs that cause an error

C Program

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

- Is there a potential bug in the program?
  - Array access
- Need to find inputs that cause an error:
  - Reach array access

C Program

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

- Is there a potential bug in the program?
  - Array access
- Need to find inputs that cause an error:
  - Reach array access
  - Index out of bounds

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

$(y > 0)$

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

$(y > 0)$

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

$(y > 0) \wedge$

$(y < 100)$

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

$(y > 0) \wedge$

$(y < 100)$

Lets Translate

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

$$(y > 0) \wedge$$

$$(y < 100) \wedge$$

$$x \notin [0 \dots 15]$$

Now we let the solver find an assignment

- $(y > 0) \wedge (y < 100) \wedge (x \notin [0 \dots 15])$

- A satisfying assignment:
  - x=20
  - y=10

C Program

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

- The values x=20 and y=10 will cause an error

CBMC Verification Condition

- Run CBMC with --show-vcc
- Shows the VC that CBMC produces for each checked property
- The VC is sent to a SAT solver (miniSAT)
- A satisfying assignment is a failing testcase
- If no assignment exists, the program is safe – **up to the bound**.

CBMC Verification Condition

```
int foo(int x, int y)
{
  int a[16];
  if (y > 0)
  {
    if (y < 100)
    {
      return a[x];
    }
  }
  return -1;
}
```

$$(y > 0) \wedge$$

$$(y < 100) \wedge$$

$$x \notin [0 \ldots 15]$$

CBMC Verification Condition

$$(y > 0) \wedge$$

$$(y < 100) \wedge$$

$$x < 15$$

CBMC Verification Condition

$(y > 0) \wedge$

$(y < 100) \wedge$

$x < 15$

```
...
{-14} \guard#1 == y!0@1#1 >= 1
{-15} \guard#2 == !(y!0@1#1 >= 100)
|-------------------------
{1} \guard#1 && \guard#2 ==>
 !((signed long int)x!0@1#1 >= 16l)
```

CBMC Verification Condition

$$(y > 0) \wedge$$

$$(y < 100) \wedge$$

$$x < 15$$

```
...
{-14} \guard#1 == y!0@1#1 >= 1
{-15} \guard#2 == !(y!0@1#1 >= 100)
|-------------------------
{1} \guard#1 && \guard#2 ==>
 !((signed long int)x!0@1#1 >= 16l)
```

CBMC Verification Condition

$(y > 0) \wedge$

$(y < 100) \wedge$

$x < 15$

```
...
{-14} \guard#1 == y!0@1#1 >= 1
{-15} \guard#2 == !(y!0@1#1 >= 100)
|-------------------------
{1} \guard#1 && \guard#2 ==>
!((signed long int)x!0@1#1 >= 16l)
```

CBMC Verification Condition

$$(y > 0) \wedge$$

$$(y < 100) \wedge$$

$$x < 15$$

```
...
{-14} \guard#1 == y!0@1#1 >= 1
{-15} \guard#2 == !(y!0@1#1 >= 100)
|-------------------------
{1} \guard#1 && \guard#2 ==>
 !((signed long int)x!0@1#1 >= 16l)
```