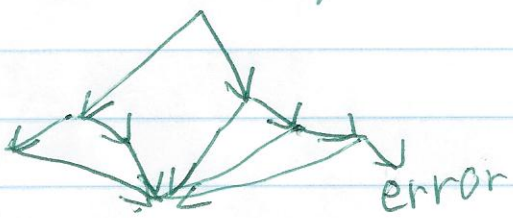
CREST/CUTE concolic unit testing engine

Motivation

Manual unit testing is expensive
⇒ Use automatic unit testing
- Random input not good
- Symbolic execution is to expensive for large units

⇒ Use concrete AND symbolic execution

Branch coverage



How it works

Run concretly while generating constraints for conditionals

Example

```
void test(Node*p, int x)      Node{
{                                 node* n
  if(p ≠ null)                    int x  }
    if(p.x == x)
      err
}
```

First run p = null, x = 17 ⇒ negate p ≠ null
Second run p = {n = null, x = 33} x = 17 ⇒ negate p.x ≠ x
Third run p = {n = null; x = 1} x = 1 } ⇒ Error found

Constraints

integers Linear Programs
$a_1x_1 + \dots + a_nx_n + c \bowtie 0$
$\bowtie$: $>, <, =, \dots$

Pointers
$x \cong y$ $x \cong null$
$=, \neq$

IF to hard, use concrete values. Might negate wrong branch.

Limitations

```
a[i]=0
a[j]=1
if(a[i]=1)
```

← Can't see that i=j ⇒ true since not in conditionals

Datastructures

Need correct input (reverse linked list on cyclic input ⇒ bad)

Use ← solve using symbolic execution

- RepOK
- Create, add, delete, contains

Classification

Sound but incomplete ← might not cover all code

↖ Finds errors using concrete execution

Expresiveness

Safety properties (assert)

Domain ~~in~~ independent (null pointer deref...)

Modularity

Interprocedural

Scalability

Seems OK for small programs

ILDENDC