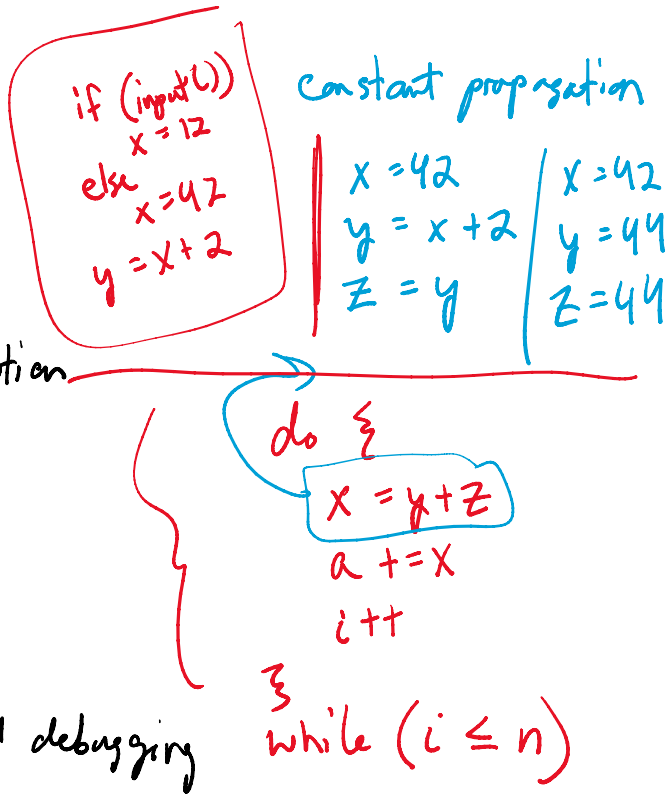
Week 1.1

Tuesday, January 9, 2024 12:02 AM

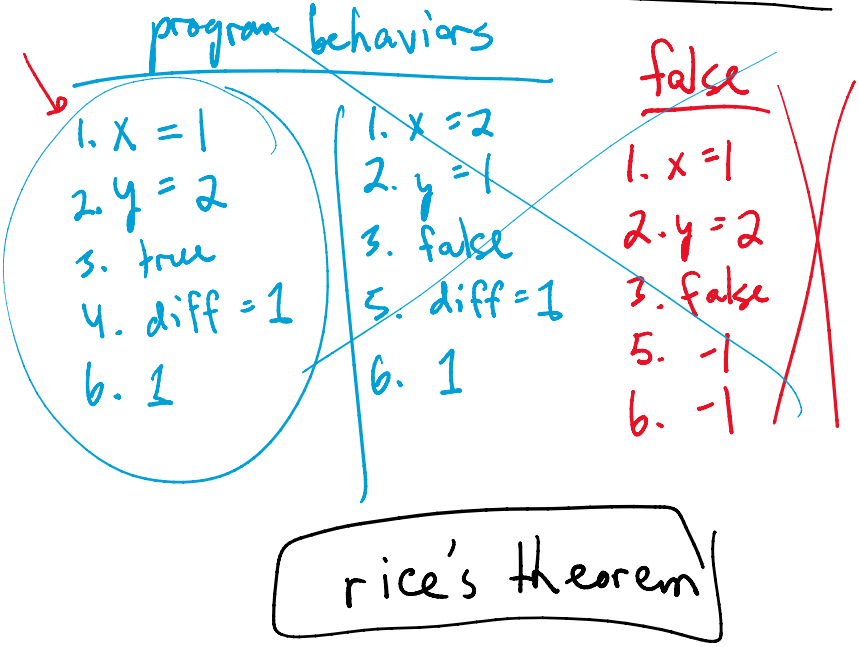
program analysis: compute invariants about program behavior

examples

- compiler optimization
  - constant propagation
  - loop invariant code motion
- security
  - sql injection
- bug checking
- program understanding and debugging



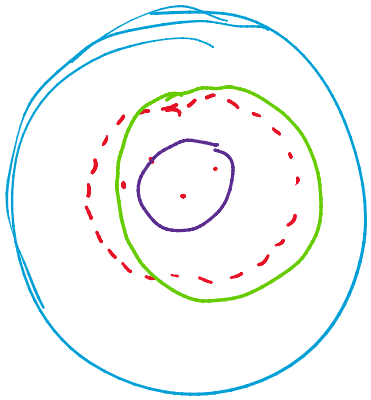
```
1. x = input()
2. y = input()
3. if (x < y) {
4.     diff = y - x
   }
   else {
5.     diff = x - y
   }
6. output(diff)
```



---

three strategies

three strategies



- over-approximate
  - guarantee: if an invariant is true of the over-approximation, it's true of the program
- under-approximate
  - guarantee: if an invariant is false for the under-approximation, it's false of the program
  - testing
- neither
  - no guarantee

which one we use depends on context of the analysis
- ↳ guarantees
- ↳ performance
- ↳ precision

a number of different approaches
- ↳ dataflow analysis (DFA)
- ↳ set constraint-based analysis

Language:

language

C_0 :

well-typed

integers
pointers
structs

conditionals
loops
functions

compile C_0 → LIR

(low-level intermediate representation)

L_0 designed around LLVM

Dataflow Analysis (DFA)

- intraprocedural analysis
- treat pointers conservatively

```
x = 42
*y = 42
x = *y
```

kildall 70's

abstract interpretation (Cousot's)