Loops and Fixpoints
what about loops?

• Symbolically execute each statement in the program

• Treat loops as a **fixpoint** problem
  • If the inputs to a statement change, re-execute statement
  • Keep going until inputs stop changing

• Claim: this will handle loops
• Claim: inputs will eventually stop changing
loop example

First time through loop, $x = 1$
Subsequent times, $x = T$
loop example

First time through loop, x = 1
Subsequent times, x = T
loop example

First time through loop, $x = 1$
Subsequent times, $x = T$
loop example

First time through loop, \( x = 1 \)
Subsequent times, \( x = T \)
loop example

First time through loop, \( x = 1 \)
Subsequent times, \( x = T \)

Why does this work?
• Symbolic values during execution can be organized according to “amount of information” in a lattice

• has more information than any constant; any constant has more information than $\bot$
merge in lattices

• Rules for merging basically say merge the information coming from the two branches: “find the smallest symbol that has at least as much information as the two symbols”

• Special symbol for this join operation: $\sqcup$

1. $v_1 \sqcup v_1 \rightarrow v_1$
2. $T \sqcup * \rightarrow T$
3. $\bot \sqcup * \rightarrow *$
4. $v_1 \sqcup v_2 \rightarrow T$
how can symbols change?

- Fixpoint algorithm: keep re-executing when a symbol changes
- What happens when a statement executes?
  - If input symbol is “higher” in the lattice, output symbol is “higher” in the lattice
- How can symbols change?
  - $\bot \rightarrow$ some other symbol the first time the statement is executed
  - some symbol $\rightarrow \top$ due to merge operations
- Symbols only get larger as symbolic execution continues $\rightarrow$ symbols can only get as large as $\top$ then stop
next: can we generalize this?