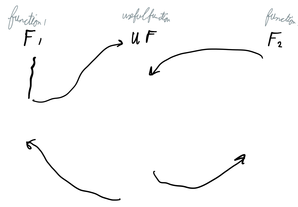
static code analysis

parsing → not helpful for finding bug's cause

expert review



hard to understand that F1 always returns to F1

(static code analysis in control control flow graph)

complete: will not classify bug-free as having bugs

sound: will not classify programs with bugs as bug-free

$$precision = \frac{true\ negatives}{rejects} \simeq completeness$$

$$= \frac{true\ negatives}{false\ positive + rejected}$$

	accept	reject
good	true negative	false positive
bad	false negative	true negative

$$recall = \frac{true\ positives}{accepted} \simeq soundness$$

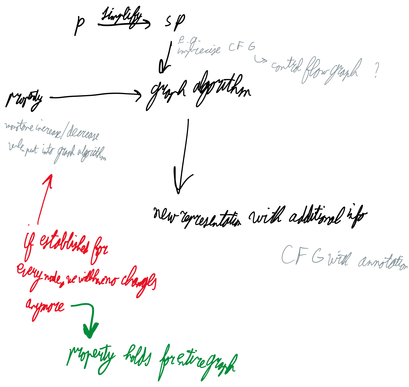
$$= \frac{true\ positive}{true\ positive + false\ negative}$$

$$F\text{-}measure = \frac{2 \times recall \times precision}{recall + precision}$$

deductive verification: program + specification given, verify whether program adheres to spec

(use e.g. theorem prover)

abstract interpretation



reaching definitions analysis

goal: determine, for each program point, which assignments have been made and not overwritten, when execution reaches that point

(defined variable name, defining node/label)

dataflow analysis: $IN[n]$ = set of facts at entry of node n

$OUT[n]$ = set of facts at exit of node n

- Compute $IN[n]$ and $OUT[n]$ for each node
- Repeat two operations until $IN[n]$ and $OUT[n]$ stop changing

"fixed point"

very busy analysis: determine very busy expressions at the exit from the node

no matter what path is taken, the expression is used before any of the variables occurring in it are redefined