EXERCISE 31

CONCOLIC EXECUTION REVIEW

What is the benefit of concolic execution over symbolic execution? How does it compare in terms of soundness / completeness of vulnerability finding?

EXERCISE 31 SOLUTION

CONCOLIC EXECUTION REVIEW

EXERCISE 31

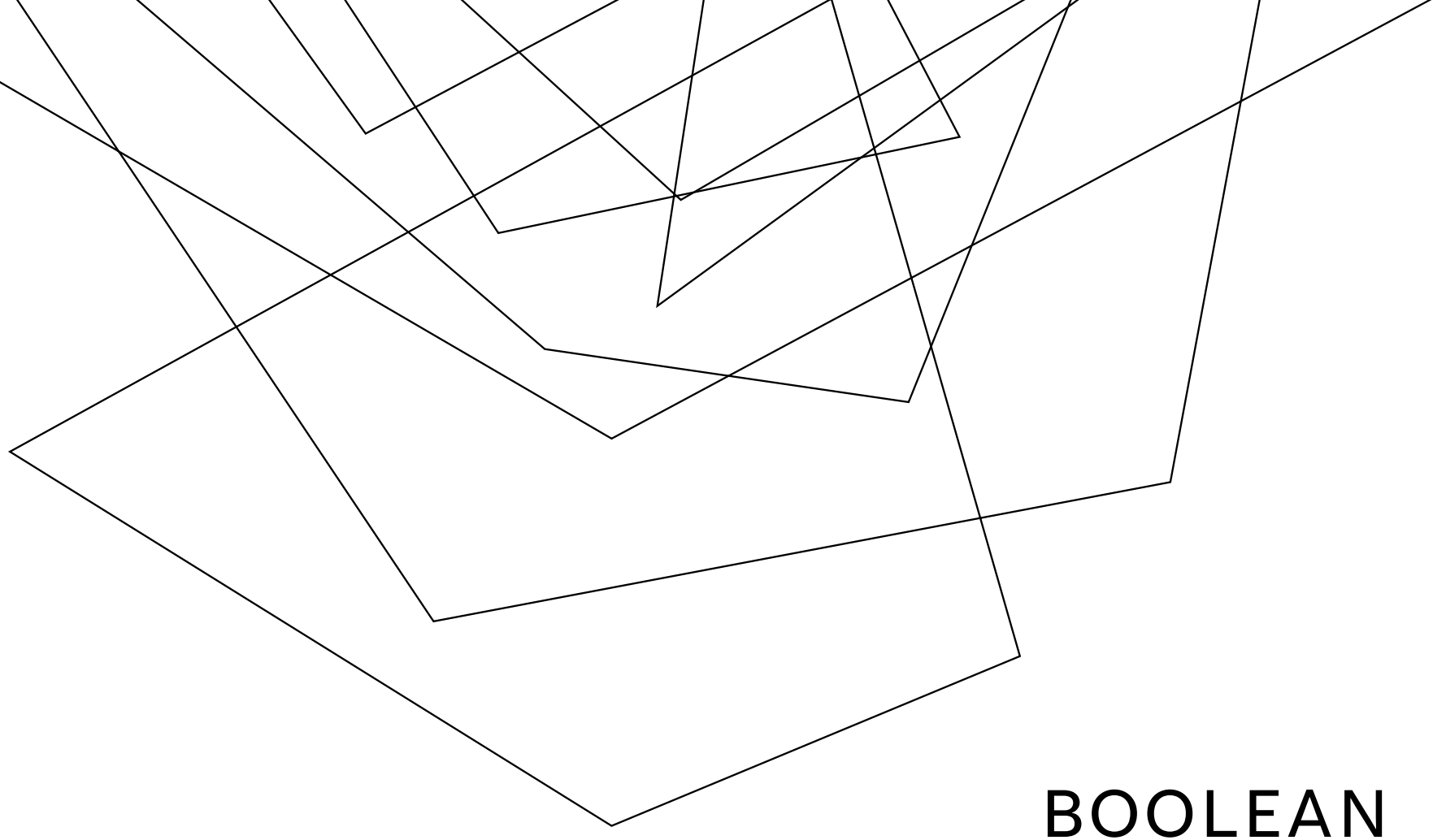
CONCOLIC EXECUTION REVIEW

Write your name and answer the following on a piece of paper

What is the benefit of concolic execution over symbolic execution? How does it compare in terms of soundness / completeness of vulnerability finding?

ADMINISTRIVIA AND ANNOUNCEMENTS

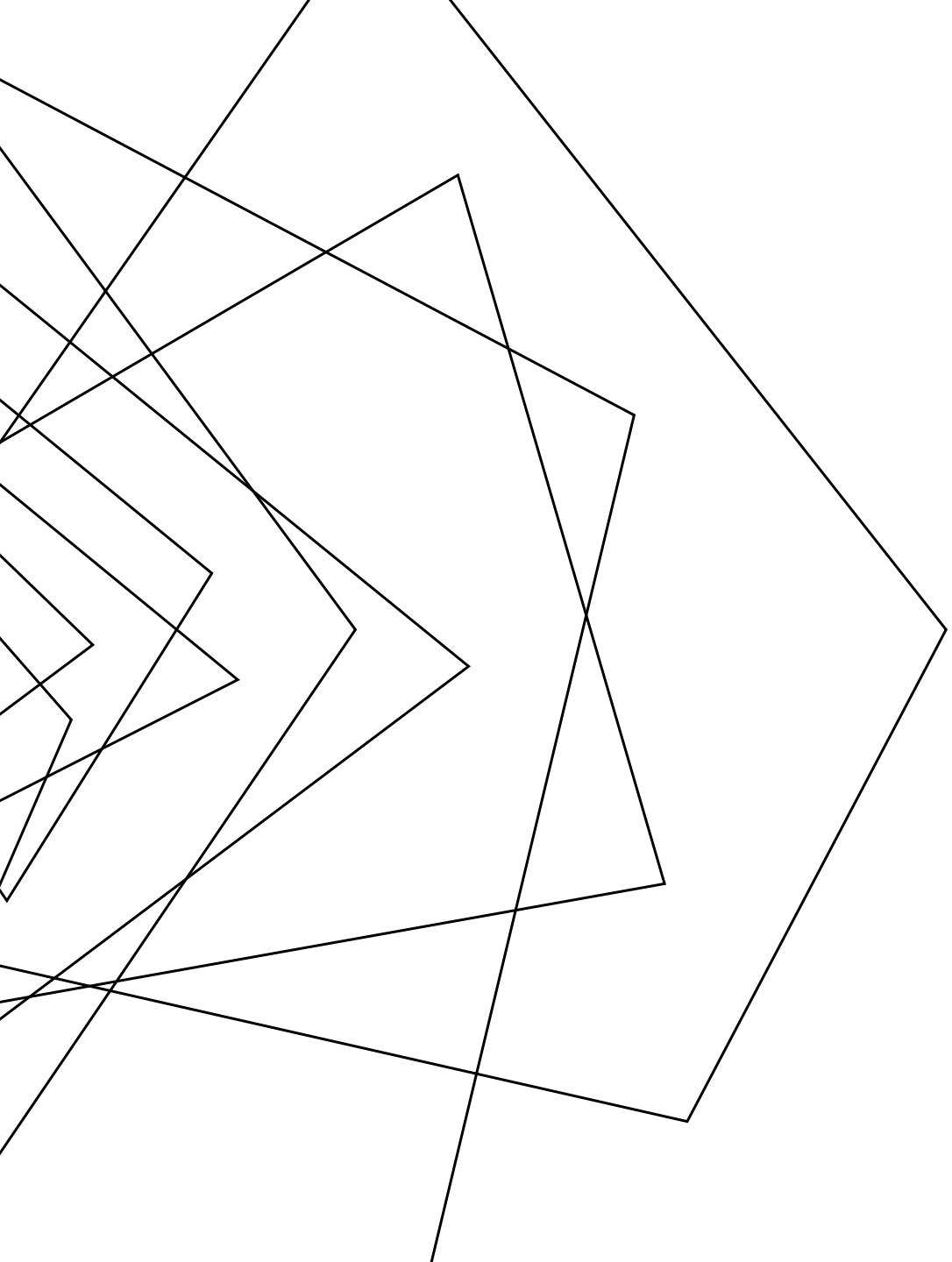
Quiz 3 is on Friday



BOOLEAN SATISFIABILITY

EECS 677: Software Security Evaluation

Drew Davidson



WHERE WE'RE AT

TOOLS / TECHNIQUES UNDERLYING
SYMBOLIC EXECUTION

PREVIOUSLY: ENHANCING SYMBOLIC EXECUTION

OUTLINE / OVERVIEW

GENERATING TEST CASES

PRIORITIZING STATES IN THE SYMBOLIC EXECUTION TREE

PRUNING DUPLICATE STATES

CONCRETIZING (SOME) INPUT TO MAKE PROGRESS

THIS TIME: SATISFIABILITY

OUTLINE / OVERVIEW

THE MAGIC THAT MADE SYMBOLIC EXECUTION WORK WAS THE SOLVER

Determines if a path constraint is feasible

Induces a test case that satisfies the path constraint

Allows for consistent concretization

BOOLEAN SATISFIABILITY

SAT AND SMT

AT THE ROOT OF THE SOLVER IS A MECHANISM FOR SOLVING A HARD PROBLEM:

Given a Boolean expression, provide a satisfying assignment to its variables or indicate no such assignment is possible

The search for a solution requires a lot of computation

$B \wedge A$

$B \vee A$

$\neg A \wedge A$

Constant time

Linear time

n log n time

polynomial time

Exponential time

Search scales rapidly with the size of the problem

NP
SAT AND SMT

The class of problems where...

A solution can be generated in polynomial time by a nondeterministic Turing machine

A solution can be verified in polynomial time by a deterministic Turing machine

$$P \stackrel{?}{=} NP$$

NP-COMPLETENESS

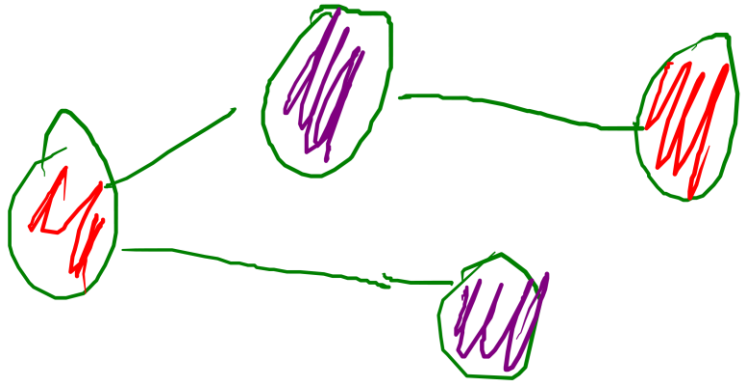
SAT AND SMT

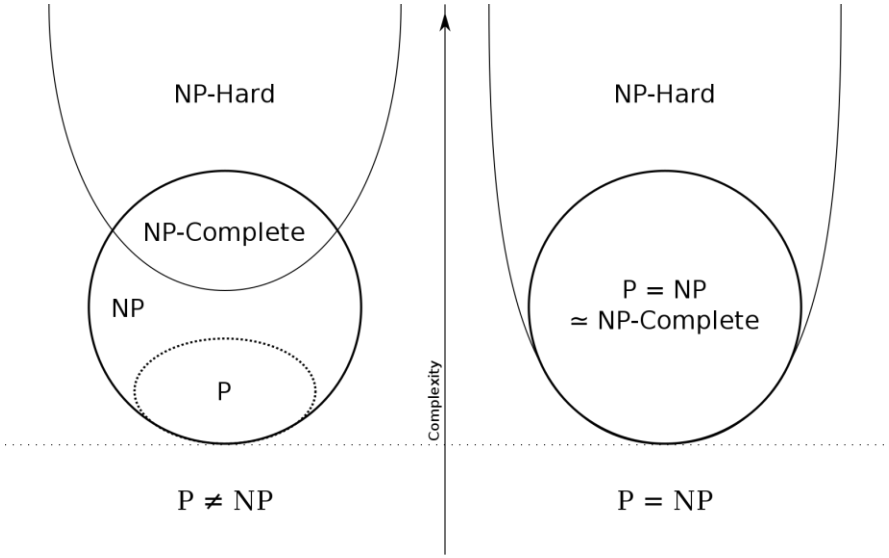
THE CLASS OF PROBLEMS WHERE...

A solution could be used as a solver for any problem in NP

The “most difficult problems in NP”

SAT is the canonical example of an NP-Complete problem





THE MARVEL OF ENGINEERING

SAT AND SMT

NP Reductions once were used to to show that a problem was difficult, now they are used to show that a problem is do-able

HOW DO SAT SOLVERS WORK?

SAT AND SMT

Naïve Solution: Exponential time 2^n

$$(a) \land (b \lor c) \land (\neg a \lor c \lor d) \land (\neg c \lor d) \land (\neg c \lor \neg d \lor \neg a) \land (b \lor d)$$

SOLVER STRATEGIES

SAT AND SMT

WE "OFTEN" CAN DO BETTER THAN THE WORST CASE

CONJUNCTIVE NORMAL FORM

SAT AND SMT

A Conjunction of clauses

Each clause may include 1 or more literals, 0 or more negations, 0 or more disjunctions

CONJUNCTIVE NORMAL FORM

SAT AND SMT

A CONJUNCTION OF CLAUSES

Each clause may include 1 or more literals, 0 or more negations, 0 or more disjunctions

P

(P ∧ Q)

P ∧ (Q ∨ R)

P ∧ (¬Q ∨ R)

P ∨ Q ∨ (R)

CONJUNCTIVE NORMAL FORM

SAT AND SMT

Any Boolean expression can be represented as a conjunction of disjunctions using the standard Boolean transformations

Boolean transformations:

$\neg(P \vee Q) \Longleftrightarrow \neg P \wedge \neg Q$

$\neg(P \wedge Q) \Longleftrightarrow \neg P \vee \neg Q$

$\neg\neg P \Longleftrightarrow P$

$(P \wedge (Q \vee R)) \Longleftrightarrow ((P \wedge Q) \vee (P \wedge R))$

$(P \vee (Q \wedge R)) \Longleftrightarrow ((P \vee Q) \wedge (P \vee R))$

CONJUNCTIVE NORMAL FORM: STRATEGY

SAT AND SMT

EVERY CLAUSE MUST BE SATISFIED

$(P \vee Q) \wedge (P \vee R)$

UNIT PROPAGATION

SAT AND SMT

a literal that exists all alone in a clause with only one literal is a unit

absolute unit

(a) ∧ (b ∨ c) ∧ (¬a ∨ c ∨ d) ∧ (¬c ∨ d) ∧ (¬c ∨ ¬d ∨ ¬a) ∧ (b ∨ d)

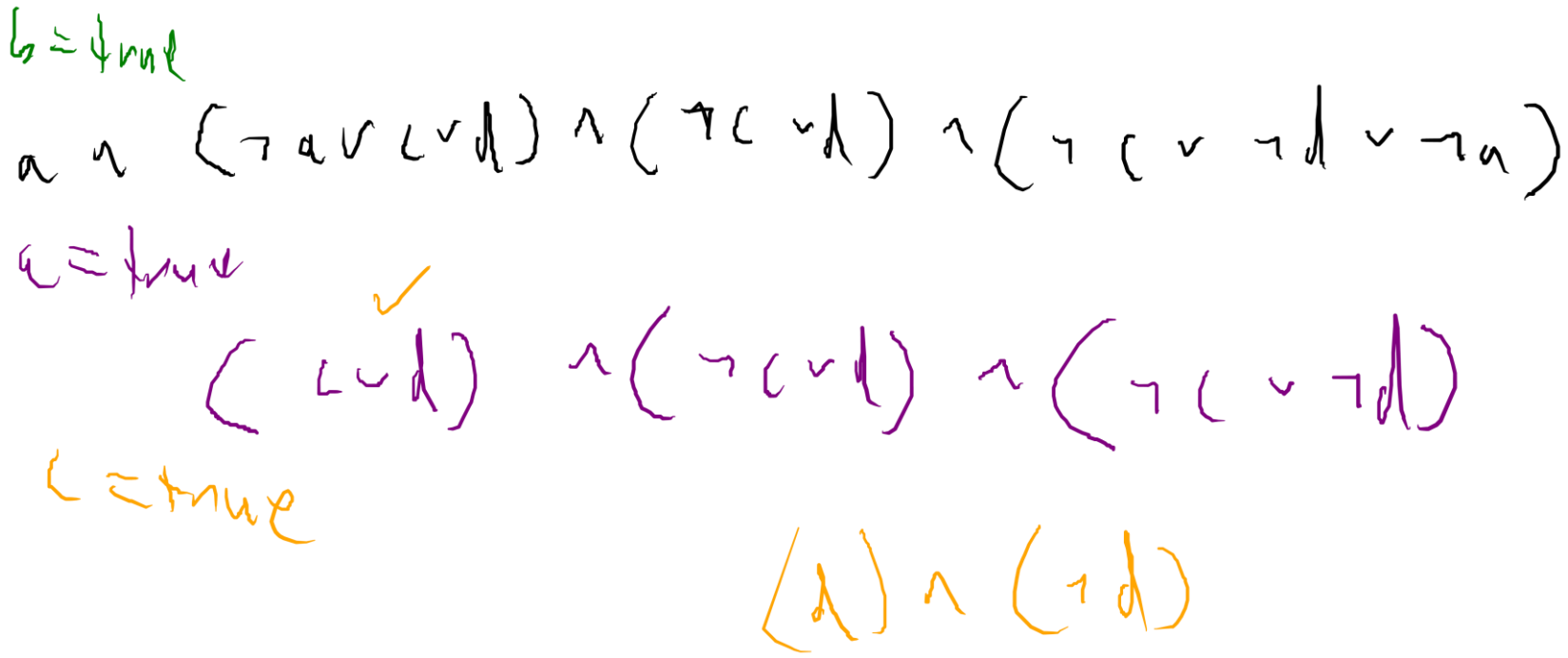
a = true a = 1

PURE LITERAL ELIMINATION

SAT AND SMT

a literal that occurs only positively, or only negatively, in a formula is pure

(a) ∧ (b ∨ c) ∧ (¬a ∨ c ∨ d) ∧ (¬c ∨ d) ∧ (¬c ∨ ¬d ∨ ¬a) ∧ (b ∨ d)



DPLL

SAT AND SMT

$$(a) \land (b \lor c) \land (\neg a \lor c \lor d) \land (\neg c \lor d) \land (\neg c \lor \neg d \lor \neg a) \land (b \lor d)$$

```
function DPLL(φ)
        if φ = true then
                return true
        end if
        if φ contains a false clause then
                return false
        end if
        for all unit clauses l in φ do
                φ ← UNIT-PROPAGATE(l, φ)
        end for
        for all literals l occurring pure in φ do
                φ ← PURE-LITERAL-ASSIGN(l, φ)
        end for
        l ← CHOOSE-LITERAL(φ)
        return DPLL(φ ∧ l) ∨ DPLL(φ ∧ ¬l)
end function
```

NO MAGIC BULLET

OUTLINE / OVERVIEW

WE KNOW SOME CONSTRAINTS ARE COMPUTATIONALLY HARD TO UNPACK

```
int main(){
  char s[80];
  scanf("%s", s);
  if (sha256sum(s) == c01b39c7a35ccc3b081a3e83d2c7
}
```

NO MAGIC BULLET

OUTLINE / OVERVIEW

WE KNOW SOME CONSTRAINTS ARE COMPUTATIONALLY HARD TO UNPACK

```
int main(){
  char s[80];
  scanf("%s", s);
  if (sha256sum(s) == c01b39c7a35ccc3b081a3e83d2c71fa9a767ebfeb45c69f08e17dfe3ef375a7b){
    return 1 / 0;
  }
}
```

FROM SAT TO SMT

OUTLINE / OVERVIEW

Next Time...

Symbolic execution requires path constraints far more complex than Boolean expressions.

Although a naïve reduction is somewhat straightforward, naivety does not gel well with NP-completeness