EXERCISE

BOOLEAN SATISFIABILITY REVIEW

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

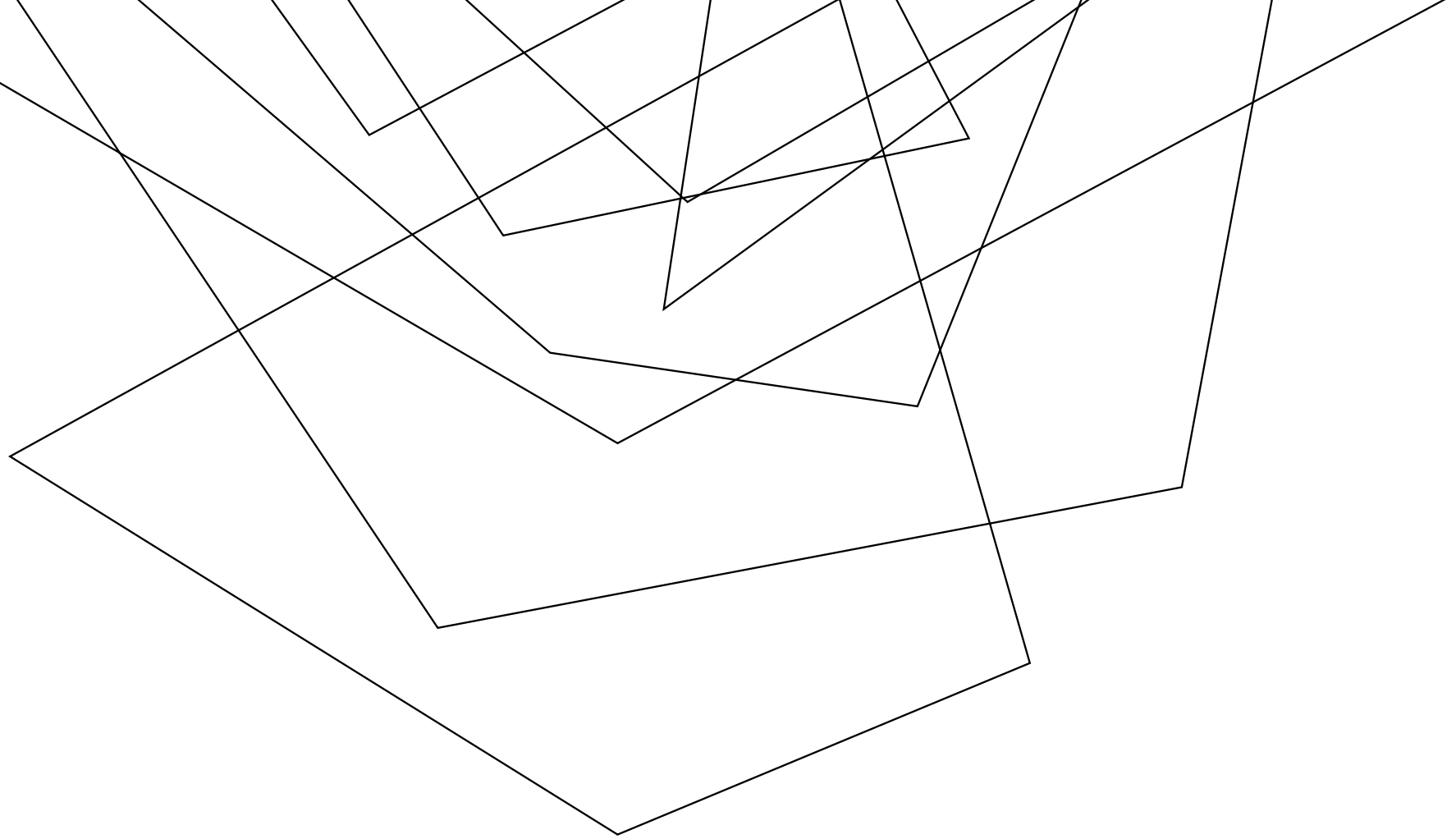
Apply DPLL to determine if there is a satisfying assignment to the following Boolean formula

$$(a \vee b) \wedge (a \vee c) \wedge (\neg b \vee \neg c) \wedge (\neg d \vee \neg c) \wedge (\neg d \vee \neg b) \wedge (c)$$

EXERCISE SOLUTION

BOOLEAN SATISFIABILITY REVIEW

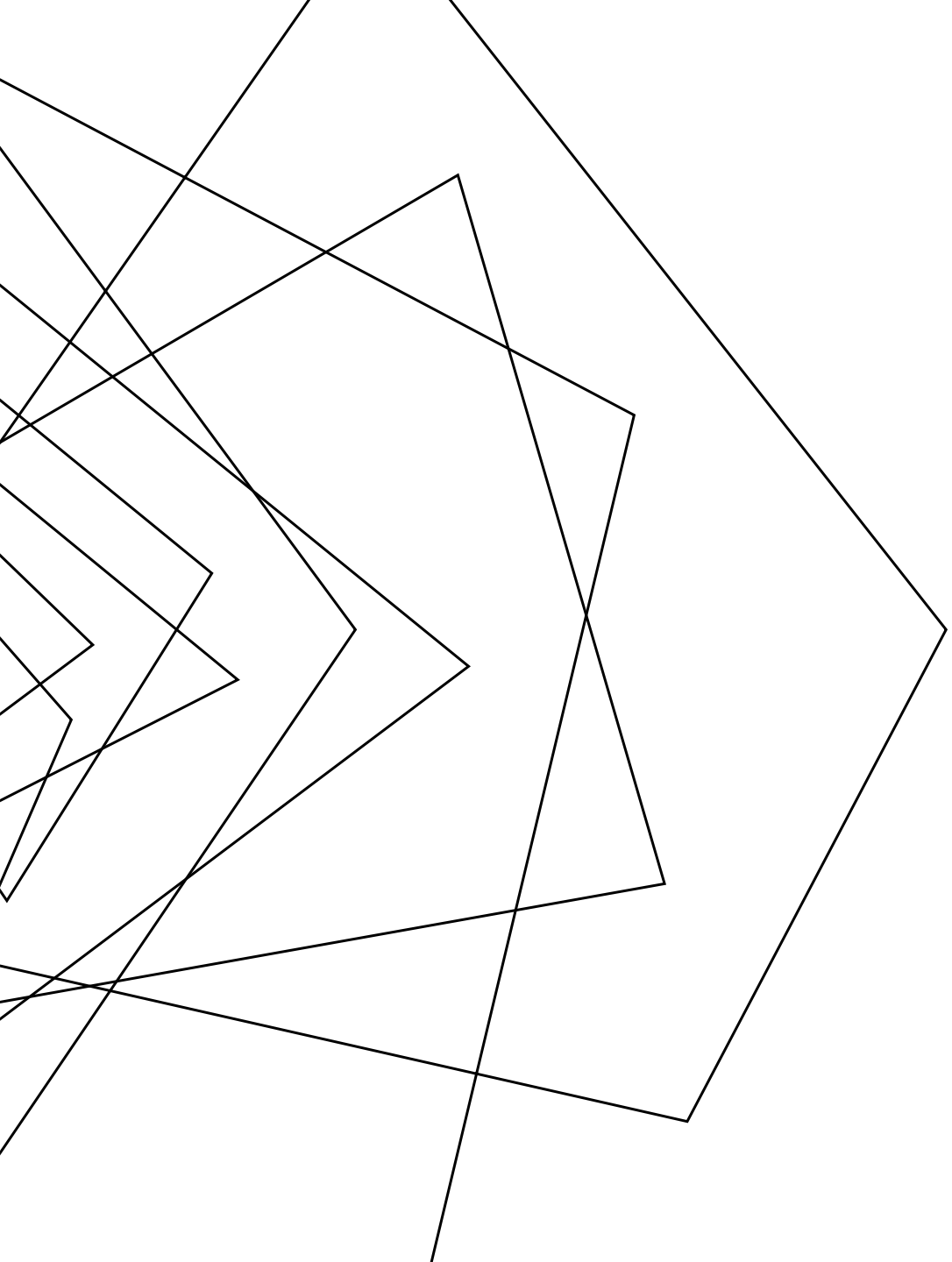
ADMINISTRIVIA AND ANNOUNCEMENTS



SMT SOLVING

EECS 677: Software Security Evaluation

Drew Davidson



WHERE WE'RE AT

TOOLS / TECHNIQUES UNDERLYING
SYMBOLIC EXECUTION

PREVIOUSLY : SATISFIABILITY

OUTLINE / OVERVIEW

THE MAGIC THAT MADE SYMBOLIC EXECUTION WORK WAS THE SOLVER

A COMPUTATIONALLY HARD PROBLEM

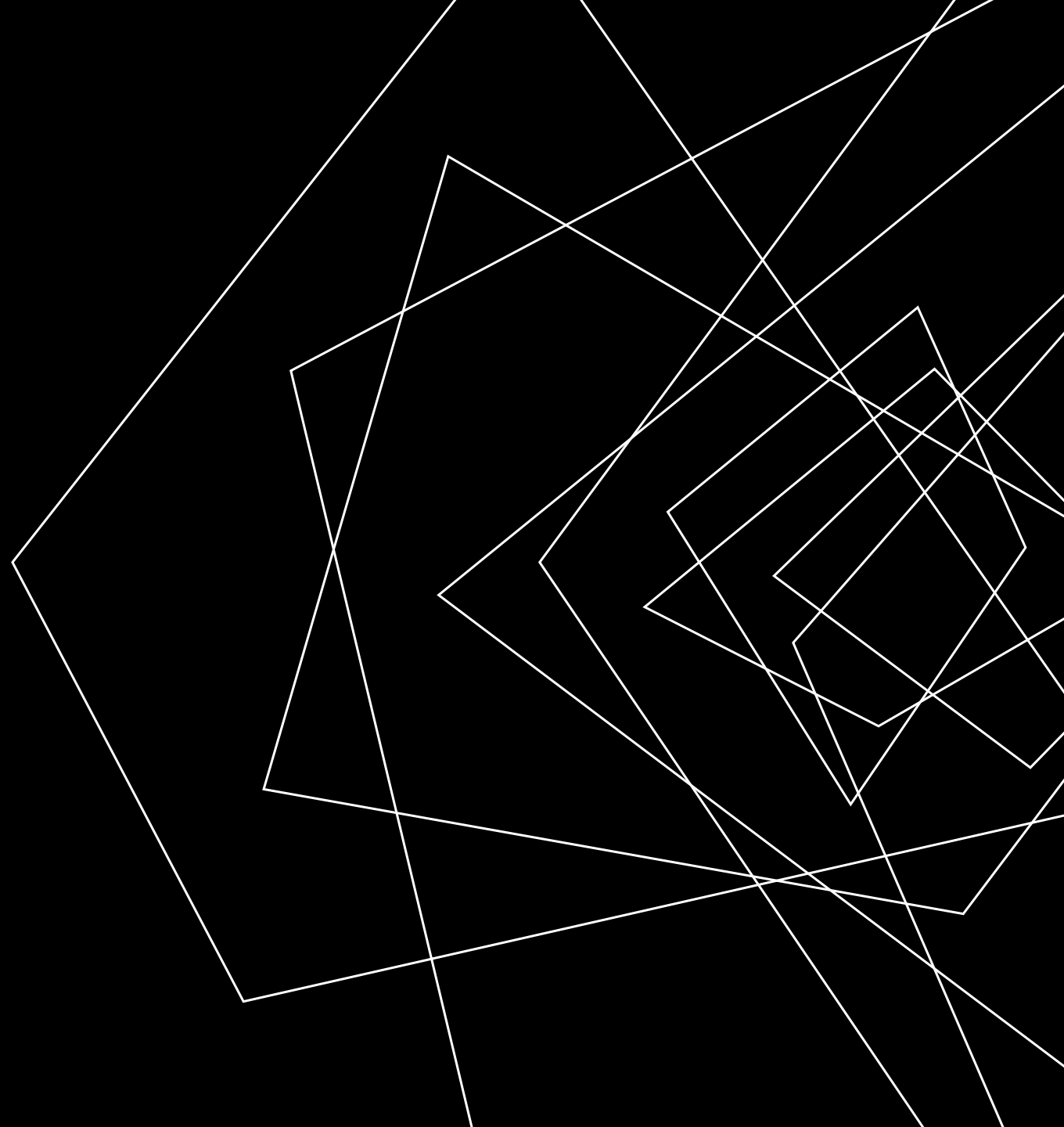
Famously NP-complete (the progenitor of that complexity class!)

Obvious exponential loose upper bound (brute force)



LECTURE OUTLINE

- Introducing Theories
- Nelson-Oppen



THIS LECTURE

SMT SOLVING

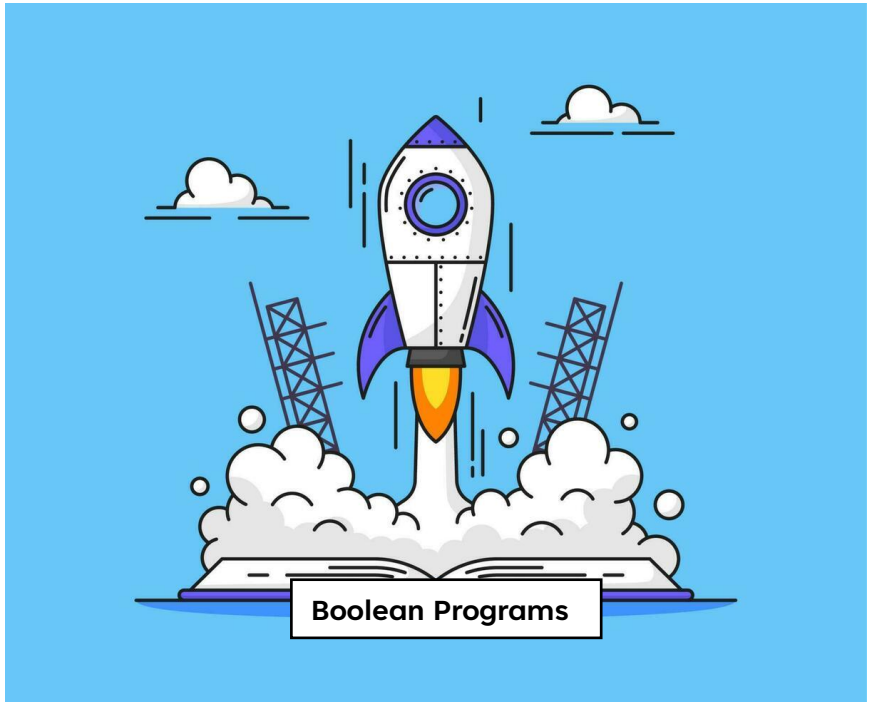
SATISFIABILITY BEYOND SIMPLE BOOLEAN EXPRESSIONS

Gets us (closer) to the real programs that we want to analyze

KEY PRINCIPLES

Individual theory solvers

Formulating constraints modularize a concern to a theory



Blasting off beyond simple Boolean programs

THEORIES: MODULARIZED ANALYSIS

SMT SOLVING

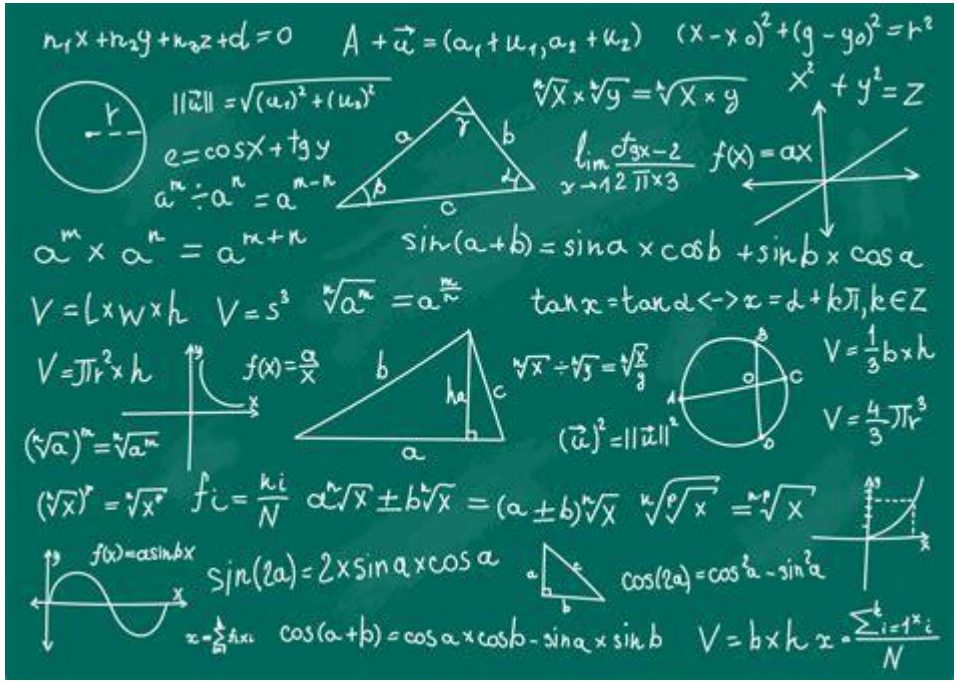
TOOLS EXIST TO REASON (COMPUTATIONALLY) ABOUT DIFFERENT "CONCEPTS"

We will refer to these concepts as *theories*

e.g. Theory of linear integer arithmetic:
$(0,1,+,-,\leq)$ interpreted over $\mathbb{Z}$

Good news: if your program fits into a theory, there may be a solver for it!

e.g. Phase I of the simplex method for optimizing linear constraints



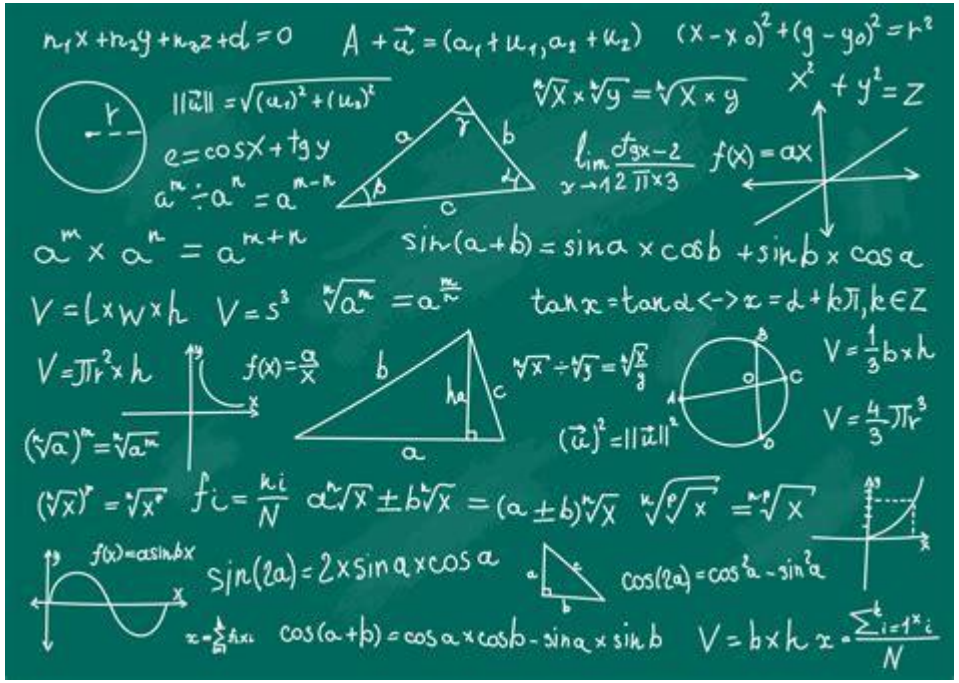
A theory (I guess of geometry in this case)

THEORIES: MODULARIZED ANALYSIS

SMT SOLVING

Good news: if your program fits into a theory, there may be a solver for it!

e.g. Phase I of the simplex method for optimizing linear constraints



A theory (I guess of geometry in this case)

DETOUR: THE SIMPLEX METHOD

SMT SOLVING

Good news: if your program fits into a theory, there may be a solver for it!

e.g. Phase I of the simplex method for optimizing linear constraints

an iterative algorithm used in linear programming to find the optimal solution (maximum or minimum) of a linear objective function subject to a set of linear equality and inequality constraints

Overview [edit]

Further information: Linear programming

The simplex algorithm operates on linear programs in the canonical form

maximize $\mathbf{c}^{\mathbf{T}}\mathbf{x}$

subject to $A\mathbf{x} \leq \mathbf{b}$ and $\mathbf{x} \geq 0$

with $\mathbf{c} = (c_1, \ldots, c_n)$ the coefficients of the objective function, $(\cdot)^{\mathrm{T}}$ is the matrix transpose, and $\mathbf{x} = (x_1, \ldots, x_n)$ are the variables of the problem, A is a $p \times n$ matrix, and $\mathbf{b} = (b_1, \ldots, b_p)$. There is a straightforward process to convert any linear program into one in standard form, so using this form of linear programs results in no loss of generality.

In geometric terms, the feasible region defined by all values of $\mathbf{x}$ such that $A\mathbf{x} \leq \mathbf{b}$ and $\forall i, x_i \geq 0$ is a (possibly unbounded) convex polytope. An extreme point or vertex of this polytope is known as *basic feasible solution* (BFS).

THEORY SOLVERS AS PURPOSE-BUILT TOOLS

SMT SOLVING

Good news: if your program fits into a theory, there may be a solver for it!

More good news: there are lots of theories with solvers

Theory of linear integer arithmetic

Theory of bitvectors

Theory of arrays

Theory of strings

Theory of equality on uninterpreted (mathematical) functions

Often possible (+ convenient / necessary) to abstract away the actual behavior of a function



A shelf of tools for different jobs

THEORY SOLVERS AS PURPOSE-BUILT TOOLS

SMT SOLVING

Good news: if your program fits into a theory, there may be a solver for it!

More good news: there are lots of theories with solvers

Bad news: the solvers don't really "mix"

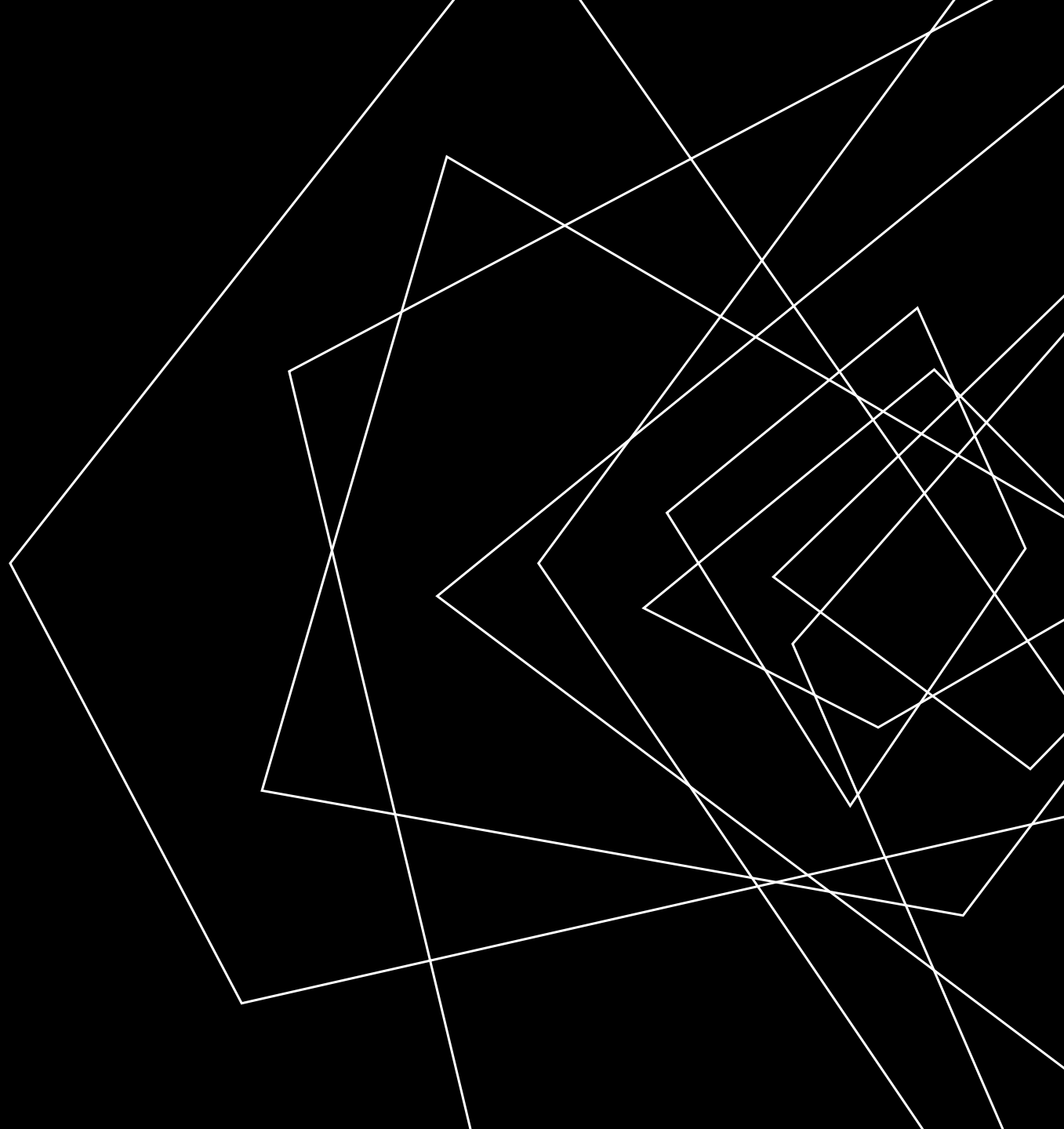
The solver won't work beyond the limit of its theory

Practical programs often mix theories together



LECTURE OUTLINE

- Introducing Theories
- Nelson-Oppen Theory Combination



THE NELSON-OPPEN METHOD

SMT SOLVING

THE HIGH-LEVEL IDEA

Restate the formula (i.e. path constraint for us) into a form where sub-formulae match individual theories

Check each sub-formula against its matching solver

Check the sub-solutions against each other

Determine if the formula is globally satisfiable

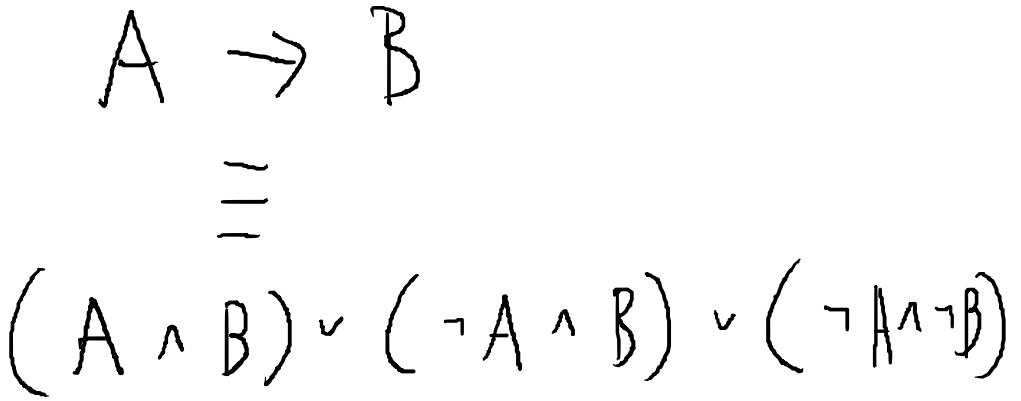
SEPARATING CONCERNS

SMT SOLVING

Note: we will only deal with constraints in Quantifier-Free First-Order Logic

Goal: break down the constraint system to match our core (logical) theory at the top level, with individual clauses potentially in our theory signatures

Logical symbols
– Parentheses: (,)
– Propositional connectives: ∨, ∧, ¬, →, ↔
– Variables: v1, v2, . . .
– ~~Quantifiers: ∀, ∃~~
Non-logical symbols
– Equality: =
– Functions: +, -, %, bit-wise &, f(), concat, ...
– Predicates: is_substring, ...
– Constant symbols: 0, 1.0, null`



EXAMPLE
SMT SOLVING

$$f(f(x) - f(y)) = a$$
$$\wedge$$
$$f(0) = a + 2$$
$$\wedge$$
$$x = y$$

Credit: this example due to Oliveras and Rodriguez-Carbonell, additional work by Aldrich

EXAMPLE
SMT SOLVING

Step 1: Nelson-Oppen procedure to separate theories

$f(f(x) - f(y)) = a$
$\wedge$
$f(0) = a + 2$
$\wedge$
$x = y$

$f(e_1) = a$
$\wedge$
$e_1 = f(x) - f(y)$
$\wedge$
$f(0) = a + 2$
$\wedge$
$x = y$

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(0) = a + 2$
$\wedge$
$x = y$

EXAMPLE
SMT SOLVING

Step 1: Nelson-Oppen procedure to separate theories

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(0) = a + 2$
$\wedge$
$x = y$

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(e_4) = a + 2$
$\wedge$
$e_4 = 0$
$\wedge$
$x = y$

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(e_4) = e_5$
$\wedge$
$e_4 = 0$
$\wedge$
$e_5 = a + 2$
$\wedge$
$x = y$

EXAMPLE

SMT SOLVING

$f(e_1) = a$ — Theory of EUF

$\wedge$

$e_1 = e_2 - e_3$ — Theory of integer arithmetic

$\wedge$

$e_2 = f(x)$ — Theory of EUF

$\wedge$

$e_3 = f(y)$ — Theory of EUF

$\wedge$

$f(e_4) = e_5$ — Theory of EUF

$\wedge$

$e_4 = 0$ — Theory of integer arithmetic

$\wedge$

$e_5 = a + 2$ — Theory of integer arithmetic

$\wedge$

$x = y$ — Theory of EUF AND Theory of integer arithmetic

EXAMPLE

SMT SOLVING

$f(e_1) = a$

$\wedge$

$e_1 = e_2 - e_3$

$\wedge$

$e_2 = f(x)$

$\wedge$

$e_3 = f(y)$

$\wedge$

$f(e_4) = e_5$

$\wedge$

$e_4 = 0$

$\wedge$

$e_5 = a + 2$

$\wedge$

$x = y$

$\wedge$

$f(x) = f(y)$

Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$ ⇐
$\wedge$
$e_3 = f(y)$ ⇐
$\wedge$
$f(e_4) = e_5$
$\wedge$
$e_4 = 0$
$\wedge$
$e_5 = a + 2$
$\wedge$
$x = y$
$\wedge$
$f(x) = f(y)$ ⇐
$\wedge$
$e_2 = e_3$

Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(e_4) = e_5$
$\wedge$
$e_4 = 0$
$\wedge$
$e_5 = a + 2$
$\wedge$
$x = y$
$\wedge$
$f(x) = f(y)$
$\wedge$
$e_2 = e_3$

$\wedge$
$e_2 - e_3 = 0$

Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

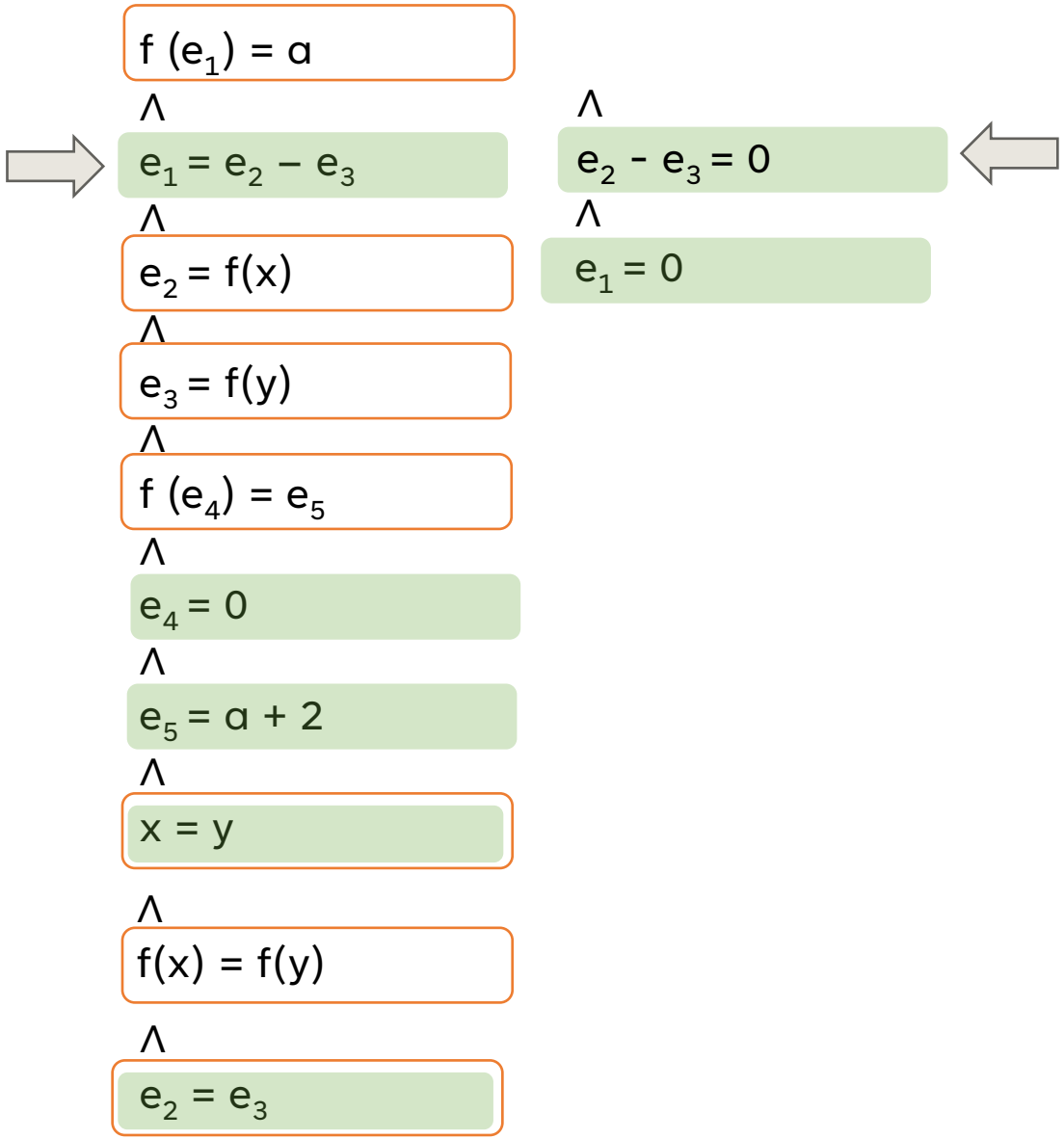
$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING



Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(e_4) = e_5$
$\wedge$
$e_4 = 0$
$\wedge$
$e_5 = a + 2$
$\wedge$
$x = y$
$\wedge$
$f(x) = f(y)$
$\wedge$
$e_2 = e_3$

$\wedge$
$e_2 - e_3 = 0$
$\wedge$
$e_1 = 0$
$\wedge$
$e_1 = e_4$

Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

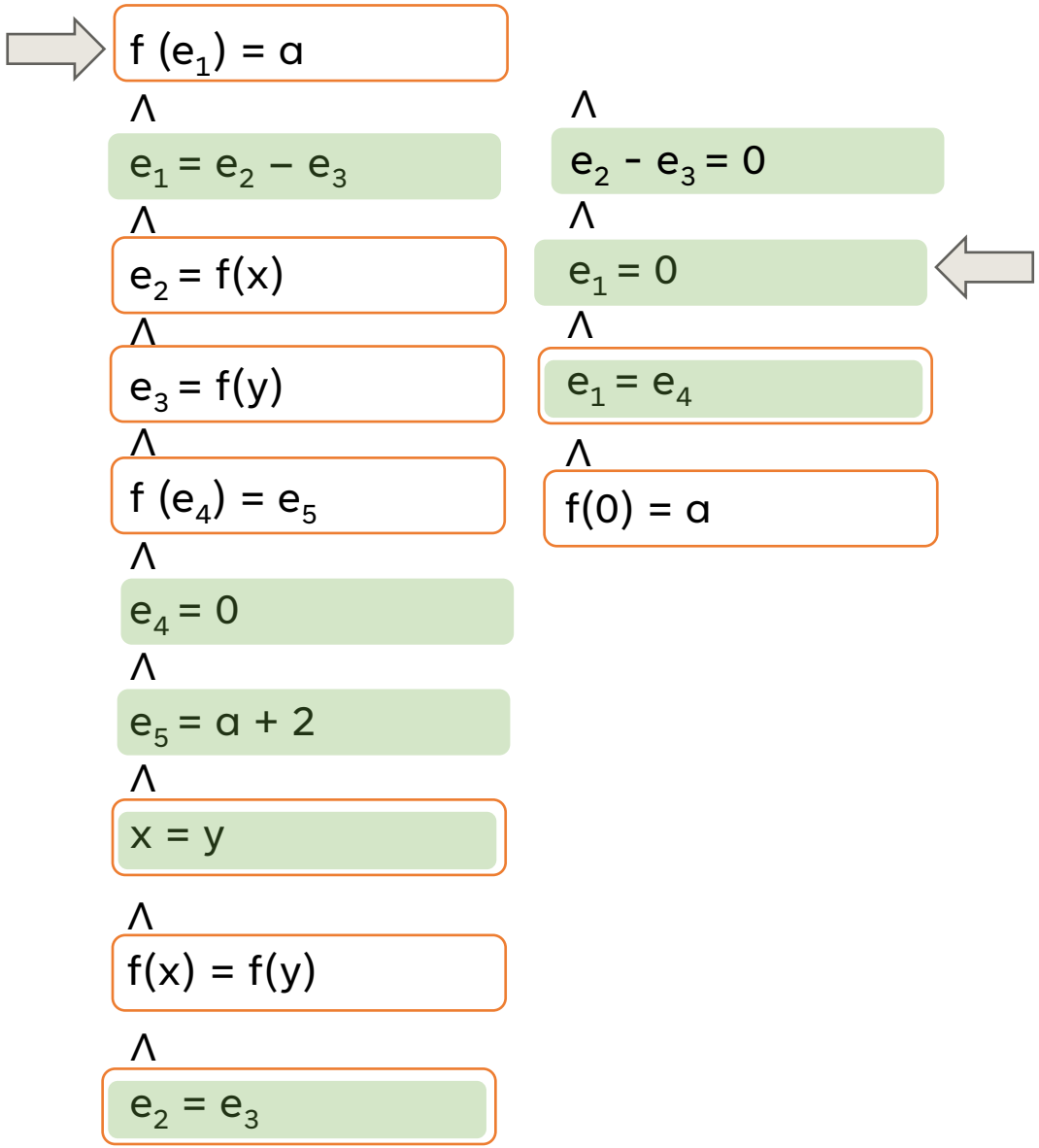
$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING



Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

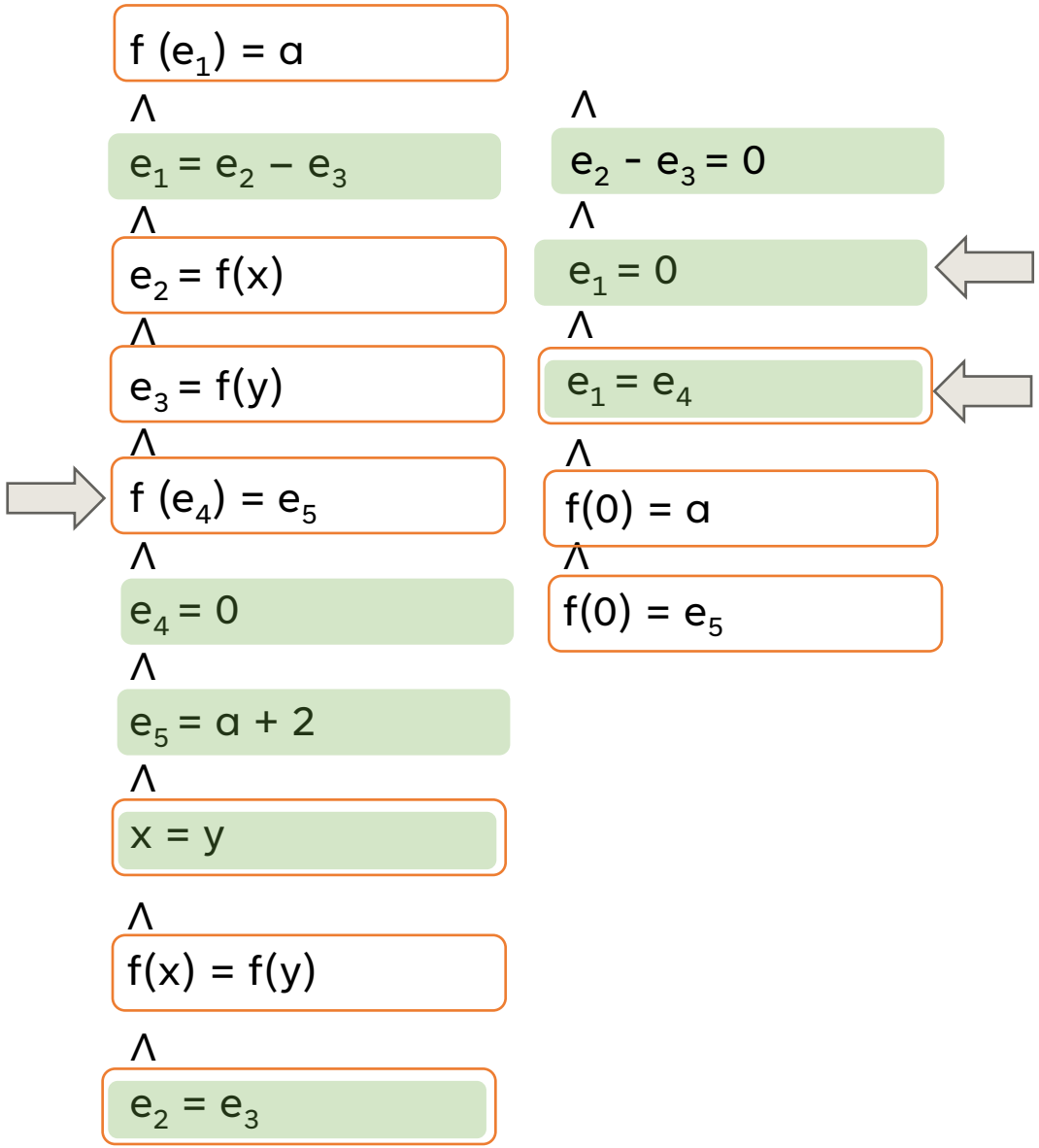
$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING



Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

EXAMPLE
SMT SOLVING

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(e_4) = e_5$
$\wedge$
$e_4 = 0$
$\wedge$
$e_5 = a + 2$
$\wedge$
$x = y$
$\wedge$
$f(x) = f(y)$
$\wedge$
$e_2 = e_3$

$\wedge$
$e_2 - e_3 = 0$
$\wedge$
$e_1 = 0$
$\wedge$
$e_1 = e_4$
$\wedge$
$f(0) = a$
$\wedge$
$f(0) = e_5$
$\wedge$
$e_5 = a$

Arithmetic Contradiction

Some EUF Axioms

Congruence:

$x = y \Rightarrow f(x) = f(y)$

Symmetry

$x = y \Rightarrow y = x$

Transitivity:

$x = y \wedge y = z \Rightarrow x = z$

...

“CONVENIENT” EQUALITIES

SMT SOLVING

$f(e_1) = a$
$\wedge$
$e_1 = e_2 - e_3$
$\wedge$
$e_2 = f(x)$
$\wedge$
$e_3 = f(y)$
$\wedge$
$f(e_4) = e_5$
$\wedge$
$e_4 = 0$
$\wedge$
$e_5 = a + 2$
$\wedge$
$x = y$
$\wedge$
$f(x) = f(y)$
$\wedge$
$e_2 = e_3$

$\wedge$
$e_2 - e_3 = 0$
$\wedge$
$e_1 = 0$
$\wedge$
$e_1 = e_4$
$\wedge$
$f(0) = a$
$\wedge$
$f(0) = e_5$
$\wedge$
$e_5 = a$

The lynchpin of our success was the existence of some useful equalities. What if they aren’t in the original constraints?

Case split!

Can add logical predicates for all possible equalities…

$(e_1 = e_2 \vee e_1 \neq e_2)$
$\wedge$
$(e_2 = e_3 \vee e_2 \neq e_3)$
$\wedge$
$(e_1 = e_3 \vee e_1 \neq e_3)$
$\wedge$
…

and start making (hopefully smart) guesses

TESTING GLOBAL SOLUTIONS

SMT SOLVING



Abstract all non-logical clauses

$P_1 \wedge P_2 \wedge (P_3 \vee P_4)$

Run DPLL on sub-formulae

P_3:true P_4: true

Run Linear Solver

We'll only use the theory solvers in service of a global satisfying assignment

It may happen that the individual theories cannot deliver a requested satisfying assignment from DPLL

TESTING GLOBAL SOLUTIONS

SMT SOLVING

P_1 P_2 P_3 P_4

$x \geq 0 \wedge y = x + 1 \wedge (y > 2 \vee y < 1)$

Abstract all non-logical clauses

$P_1 \wedge P_2 \wedge (P_3 \vee P_4)$

Run DPLL on sub-formulae

P_3:true P_4: true

Run Linear Solver

contradiction!

Add information and start over

$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge \neg(P_3 \wedge P_4)$

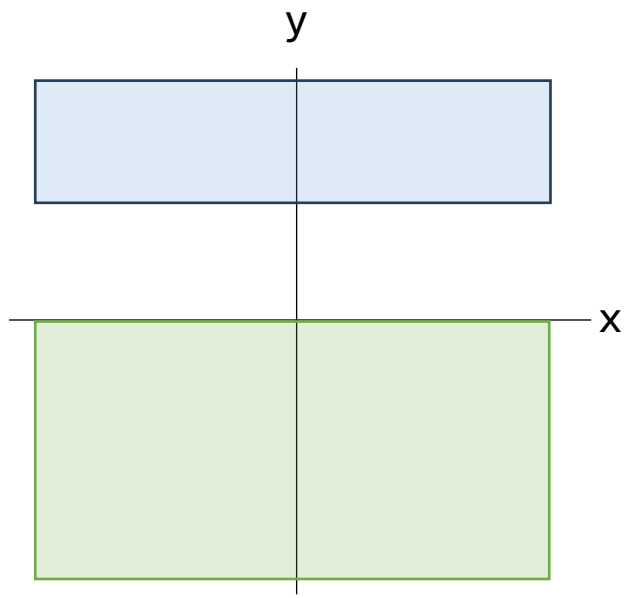
⇩

$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge (\neg P_3 \vee \neg P_4)$

Simplex algorithm Phase I: no feasible region

$y > 2$

$y < 1$



TESTING GLOBAL SOLUTIONS

SMT SOLVING

P_1 P_2 P_3 P_4

$x \geq 0 \wedge y = x + 1 \wedge (y > 2 \vee y < 1)$

$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge (\neg P_3 \vee \neg P_4)$

Run DPLL on sub-formulae, good

Run DPLL on formula

P_1:true P_2:true P_3:false P_4: true

Run Linear Solver

Contradiction

Add information and start over

$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge (\neg P_3 \vee \neg P_4) \wedge \neg(P_1 \wedge P_2 \wedge \neg P_3 \wedge P_4)$

⇩

$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge (\neg P_3 \vee \neg P_4) \wedge (\neg P_1 \vee \neg P_2 \vee P_3 \vee \neg P_4)$

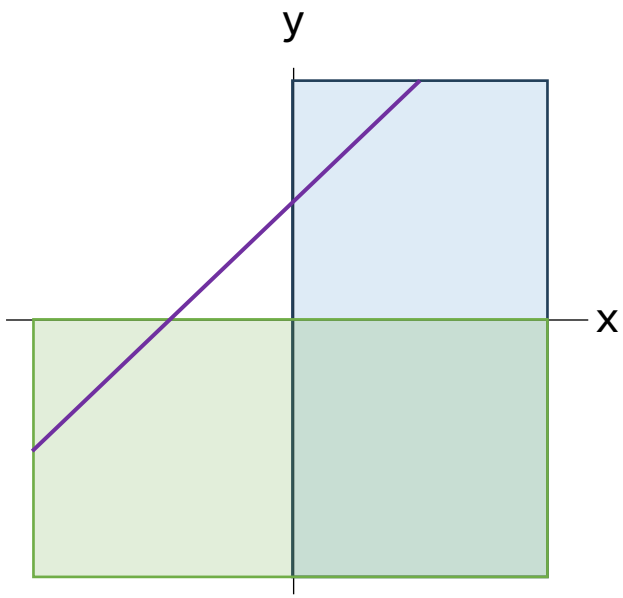
Simplex algorithm Phase I: no feasible region

$x \geq 0$

$y = x + 1$

$y \leq 2$

$y < 1$



TESTING GLOBAL SOLUTIONS

SMT SOLVING

P_1 P_2 P_3 P_4

$x \geq 0 \wedge y = x + 1 \wedge (y > 2 \vee y < 1)$

Simplex algorithm Phase I: feasible region!

$P_1 \wedge P_2 \wedge (P_3 \vee P_4) \wedge (\neg P_3 \vee \neg P_4) \wedge (\neg P_1 \vee \neg P_2 \vee P_3 \vee \neg P_4)$

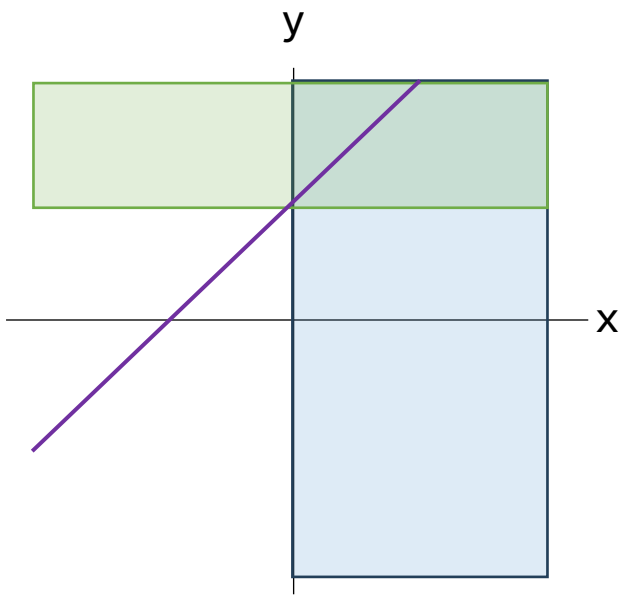
$(P_3 \vee P_4) \wedge (\neg P_3 \vee \neg P_4) \wedge (P_3 \vee \neg P_4)$

$x \geq 0$

$y = x + 1$

$y > 2$

$y \geq 1$



WRAP-UP
SMT SOLVERS

Hopefully I've convinced you that Solvers can be implemented

Not strictly magic, but they do employ some very clever techniques



NEXT TIME

WEB SECURITY

ENOUGH THEORY!

By “popular” request: Web security

