Lecture 15

Wednesday, March 11, 2020 4:34 PM

- Hello everyone!
- Two important links for today's class:
  - The zoom workspace
    https://usc.zoom.us/j/660096062
    (For listening to me :))
  - The Google Doc
    https://docs.google.com/document/d/15uQew_IPK1CAwNBJ7olCaKXU3LdW1ecoWRApgRKl47Q/edit?usp=sharing
    (For talking back :))

- You are muted by default.
- Please unmute yourselves

- You are also being recorded
- Say Cheese!

- Finish discussion of Nelson-Oppen procedure
- Conclude Unit 2.

$$\underbrace{C_1}\ ;\ \underbrace{C_2}$$

$$\{P\}\ C_1\ \{x>0 \text{ and } y>x\} \Rightarrow \{y>0\}\ C_2\ \{x>5\}$$

$$\frac{\{P\}\ C_1\ \{Q\} \qquad \{Q\}\ C_2\ \{R\}}{\{P\}\ C_1 ; C_2\ \{R\}}$$

Sequencing

$$\frac{P' \Rightarrow P \qquad \{P\}\ C\ \{Q\} \qquad Q \Rightarrow Q'}{\{P'\}\ C\ \{Q'\}}$$

Consequencing rule

Sequencing

Consequencing rule

$$\frac{\{P\}\, C_1\, \{Q_1\} \qquad \boxed{Q_1 \Rightarrow Q_2} \qquad \{Q_2\}\, C_2\, \{R\}}{\{P\}\, C_1 ; C_2\, \{R\}}$$

<u>Question</u>: Is it true, for all integers x & y, that if $x > 0$ and $y > x$ then $y > 0$?

$$\forall x\ y \in \mathbb{Z}\ (x > 0 \text{ and } y > x) \Rightarrow y > 0 \quad \text{— ①}$$

<u>Question</u>: How do we use an SMT solvers to determine the $\underbrace{\text{truth}}_{\text{validity}}$ of ① ?

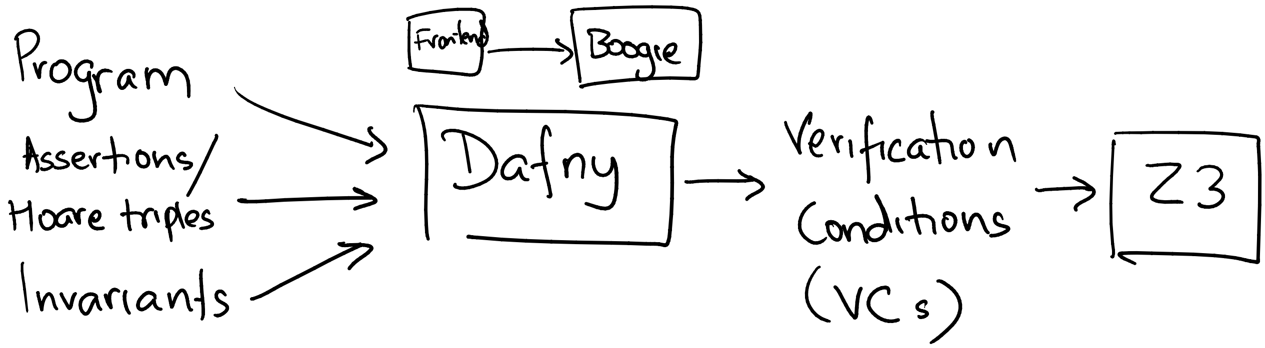
$$\underbrace{\exists x\ y\ \ (x>0 \text{ and } y>x)\ \underline{\text{but}}\ \underline{\text{not}}\ (y>0)}_{\text{② : Satisfiability}}$$

$$\exists x\ y \in \mathbb{Z}\ \ (x>0 \text{ and } y>x) \text{ and } (y \ngtr 0)$$

② witnesses every counterexample to ①.

(2) witnesses every counterexample to (1).

(2) can be automatically checked using a solver for difference logic.



$$\frac{P \Rightarrow I \quad \{I \wedge b\}\ C\ \{I\} \quad I \wedge \bar{b} \Rightarrow Q}{\{P\}\ \text{while } (b) \text{ do } C\ \{Q\}}$$

Combining Theories Nelson Oppen

We have seen:

- How SAT solvers work.
- How two specific theory solvers work.

EUF, Difference logic

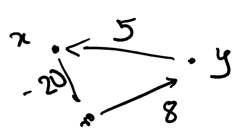
$t \quad t'$

$f(t) - f(t')$

$f(t) \neq f(t')$

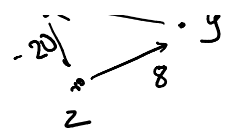
$\exists x\ y\ z \quad x - y \geq 5 \quad y - z \geq 8 \quad x - z \leq 20$

$z - x \geq -20$



$f(t) \neq f(t')$

Congruence
closure



- Know about the existence of solvers for other theories

  LIA LRA Arrays Bitvectors Strings

- DPLL (T)

  Theory solvers take a <u>conjunction</u> of literals

  DPLL (T) allows us to combine formulas with <u>one</u> theory T and an interesting Boolean skeleton.

SAT solver $\xrightarrow{\text{Conjunction of literals}}$ Theory solver

SAT solver $\xleftarrow{\text{Conflict clause}}$ Theory solver

① Finds satisfying assignment to Boolean skeleton

<u>Ex:</u> $l_1 \wedge \overline{l_2} \wedge l_3 \wedge \ldots$

② $x - y > 0$ and $y - z \not> 5$ and $z - x \leq 3$

③ If sat, great! Done!

④ If unsat, report conflict.

$\overline{l_1 \wedge \overline{l_2} \wedge l_3} = (\overline{l_1} \vee l_2 \vee \overline{l_3})$

$$\overline{l_1 \wedge \overline{l_2} \wedge l_3} = \underbrace{(\overline{l_1} \vee l_2 \vee \overline{l_3})}_{\text{Clause}}$$

- <u>Nelson Oppen</u> : Provides technique to combine theories.

	T_1	T_2
	Diff. logic	EUF
Signature	"$-$" "$<$" "$=$"	$f(\_)$ "$=$"
	Σ_1	Σ_2

Signature of combination : "$-$" "$<$" "$=$" "$f(\_)$"

$$\Sigma_1 \cup \Sigma_2$$

Nelson-Oppen requires that $\Sigma_1 \cap \Sigma_2 = \{=\}$.

Equality should be the only symbol common to both theories.

Nelson-Oppen (Conjunction of literals φ over two theories T_1 and T_2)

① Purify φ into ($\varphi_1 \wedge \varphi_2$)

Only over T Only over T

Only over T_1 Only over T_2

Ex: Consider $\boxed{x \le f(x) + 1}$ φ

LIA EUF

$\Sigma_{LIA} = \{ <, =, + \}$ $\Sigma_{EUF} = \{ =, f(-) \}$

Create new variable y.

$\varphi \iff (x \le y+1$ and $y = f(x))$

Can be consumed by an LIA solver

Can be consumed by an EUF solver

(2) Submit φ_1 to a solver for T_1

Submit φ_2 to a solver for T_2

Case 1: Either theory solver reports unsat.

$$\varphi = \underbrace{\varphi_1}_{\text{Is unsat}} \wedge \varphi_2$$

Therefore φ is unsat.

Case 2: If both solvers report SAT.

$$\varphi = \exists x\, y\, z\, (\varphi_1 \wedge \varphi_2)$$

$$\varphi = \underbrace{\exists x\, y\, z\, (\varphi_1}_{} \wedge \underbrace{\varphi_2}_{})$$

$$\frac{\exists x\, y\, z \cdot \varphi_1}{\text{Is sat}} \qquad \frac{\exists x\, y\, z\ \varphi_2}{\text{Is sat}}.$$

<u>Question</u>: Does this mean that φ as a whole is sat?

<u>No</u>.

<u>Ex</u>: $\exists\, x\, y \in \mathbb{Z}\ \ 1 \le x \le y \le 2$ and
$x + y = 2$ and
$\Rightarrow$ LIA solver says: $x = y = 1$

$f(x) = f(1)$ and
$f(y) \ne f(1)$.

$\varphi_1 = \exists\, x\, y \in \mathbb{Z}$ s.t. $1 \le x \le y \le 2$ and $x + y = 2$

Sat $(x = y = 1)$

$\varphi_2 = \exists\, x\, y \in \mathbb{Z}$ s.t. $f(x) = f(1)$ and $f(y) \ne f(1)$

Sat $\left(\begin{array}{ll} x = 1 & y = 0 \\ f(1) = 1 & f(0) = 0 \end{array}\right)$

<u>Essential problem</u>: Theories may cause variables to be equal.

Nelson-Oppen step 2

For each pair of variables x, y
which appear in both φ_1 and φ_2

① Check if $\underline{\varphi_1 \Rightarrow (x = y)}$

Then update $\varphi_2 := \varphi_2 \wedge (x = y)$

② Check if $\varphi_2 \Rightarrow (x = y)$

If so, update $\varphi_1 := \varphi_1 \wedge (x = y)$

③ Fork into two cases:

$$\varphi_1 \wedge \varphi_2 \wedge (x = y) \quad \Big| \quad \varphi_1 \wedge \varphi_2 \wedge (x \neq y)$$

T_1 T_2 | T_1 T_2

$$\varphi = \varphi_1 \wedge \varphi_2 \equiv \Big(\overbrace{(\varphi_1 \wedge x = y)}^{T_1} \wedge \overbrace{(\varphi_2 \wedge x = y)}^{T_2}\Big) \text{ or}$$
$$\Big(\underbrace{(\varphi_1 \wedge x \neq y)}_{T_1} \wedge \underbrace{(\varphi_2 \wedge x \neq y)}_{T_2}\Big).$$

Convex and non-convex theories

Convex: Don't require case splitting.

splitting.

1. Purify literals
2. Insert equality /inequality constraints over shared variables
3. Solve resulting formulas using DPLL (T) solvers.

Recall: Theories were stably infinite.

- Will release HW2 today. Please turn it in by April 1.

Unit 3: Abstract Interpretation

- Question: Why is program verification hard?

P

```
x := 0 ; y = 200; z := input()
while (x < z) {
    if (x + y < 0) Error
    x := x+1
}
```

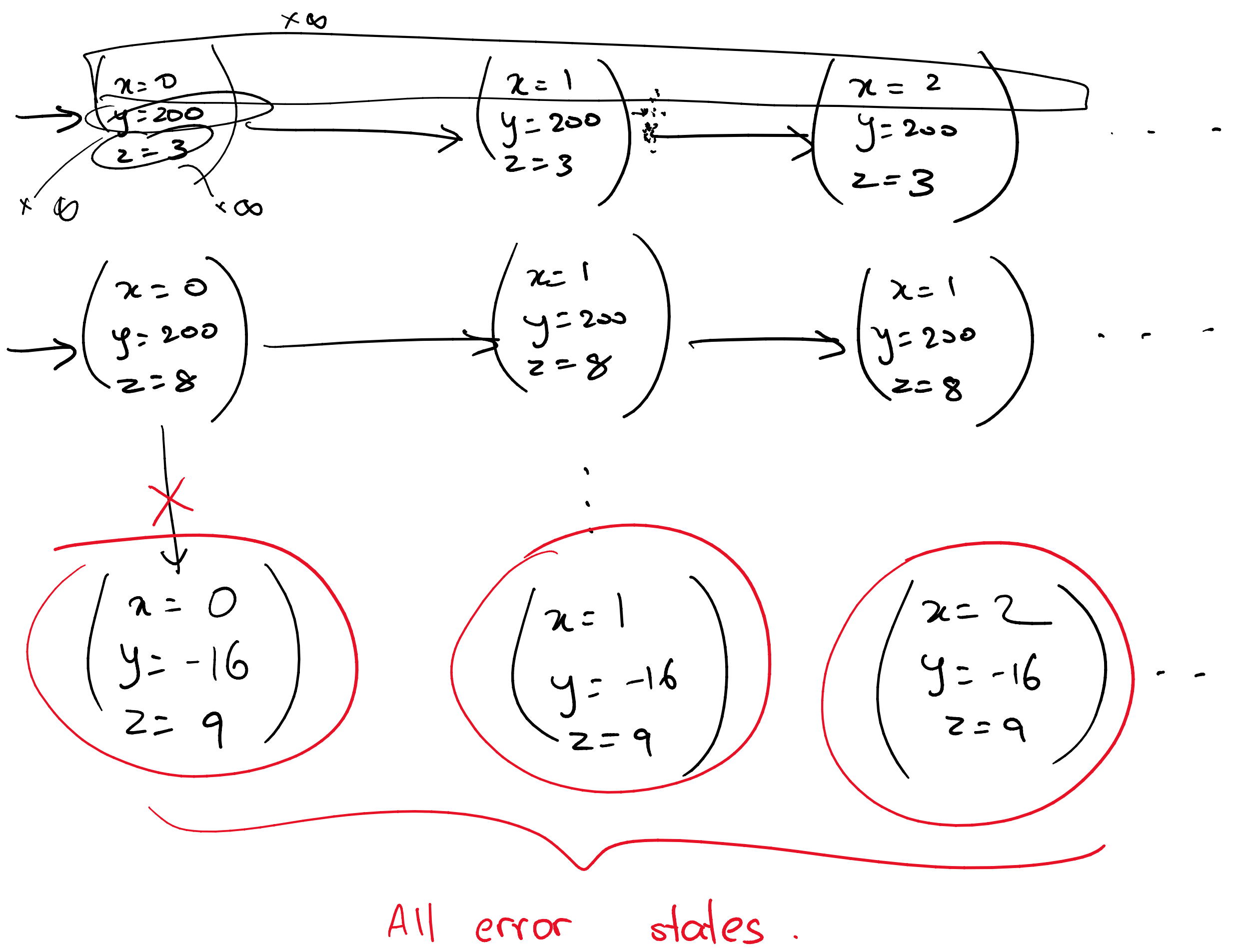
Not reachable!

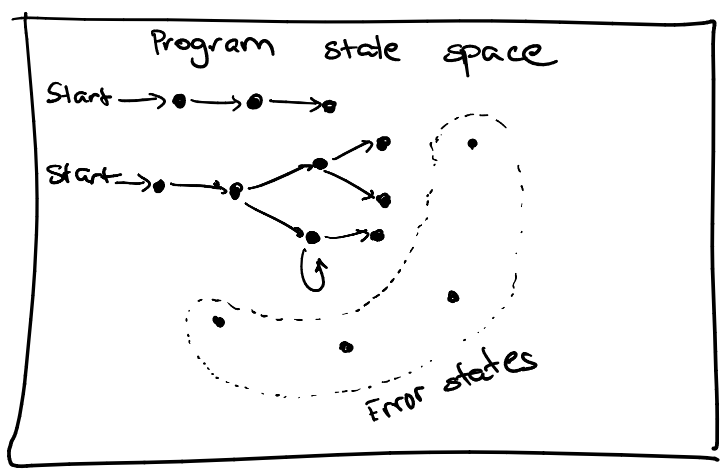
Because x is always positive.

So is y.

$x > 0$ and $y > 0 \Rightarrow x + y \not< 0$.

Imagine state space of P (x, y, z)





Automatic Verification, Approach 1: Exhaustively explore reachable states in the transition graph.

Doesn't work. — State space is infinite

— State space is exponentially large (Even when finite)

STATE EXPLOSION PROBLEM

Cf. curse of dimensionality

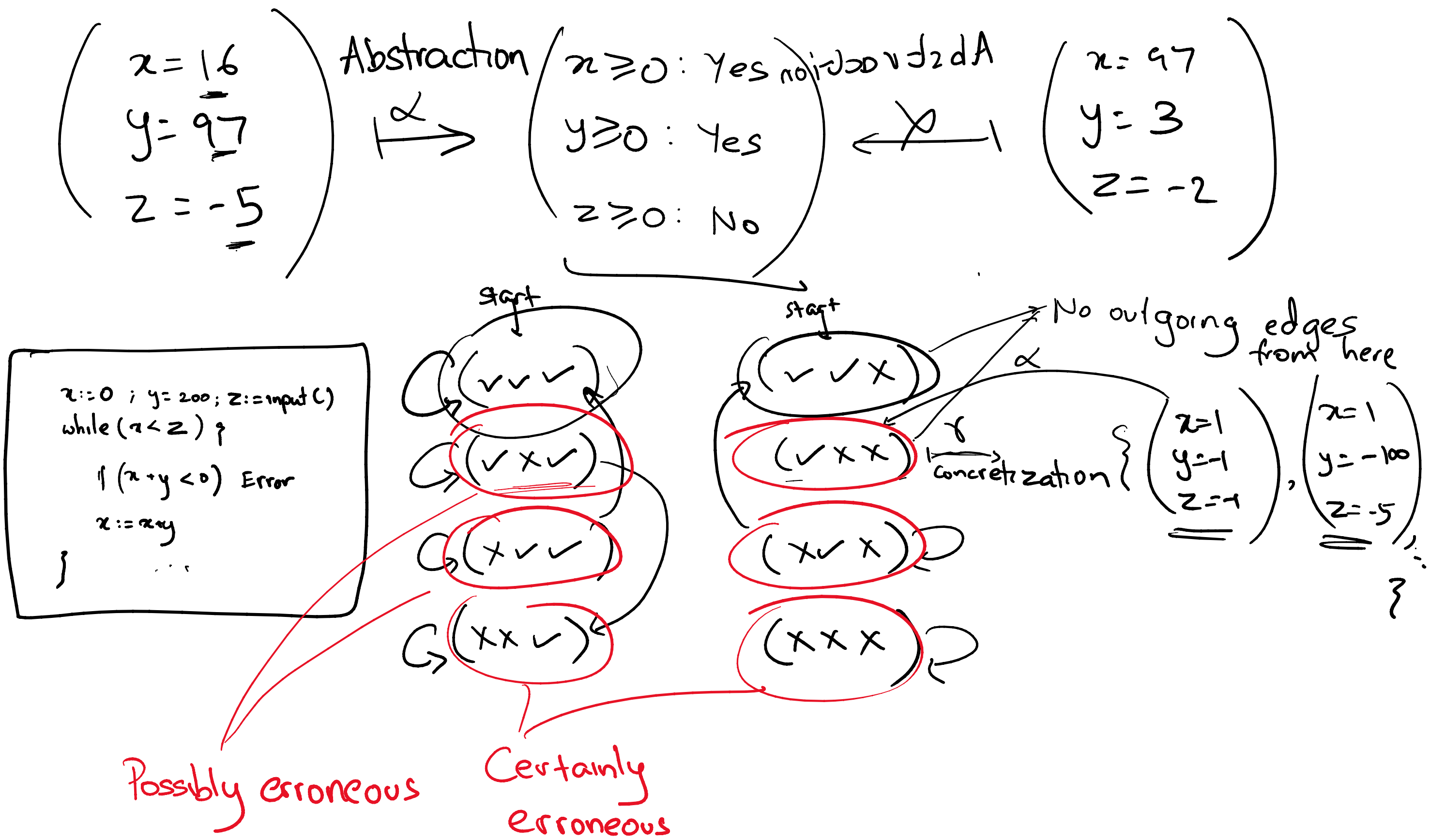
Vote: Which field has a cooler sounding problem?

FM / PL / CPS / SE	Statistics / theory / ML / ...
State explosion problem	Curse of dimensionality
|||| ✓ Wins.	|||

Predicate abstraction: Collapse infinite state space into something finite.

Essential idea: Instead of tracking concrete value of every variable, only track value of every variable, only track the values of some previously chosen predicates.

Predicates: $\{x \geq 0,\ y \geq 0,\ z \geq 0\}$



- Started with a program P with an infinite (concrete) state space.

- Select a finite set of predicates V

- Construct Boolean abstraction of P. $\alpha(P) \subseteq 2^V \times 2^V$

- Exhaustively explore states of $\alpha(P)$ for possible errors.
  - If no error state is reachable, excellent! Program is safe!
  - If some possibly erroneous state is reached, we may have discovered a bug.