Lecture 27

Wednesday, April 29, 2020 4:35 PM

- Hello!
- Today: Last class of the semester!
- Goals
  - Concluding our discussion of program synthesis
    1. What is the complexity of program synthesis.
    2. ⇒ Using program synthesizers for invariant generation.
  - Concluding the course
    1. Recap of ideas
    2. Future research directions.

Using Program Synthesis for Invariant Gen.

Ex: Assume LIA.

$\exists f.\ \forall [x\ y]\ [f(x\ y)] \geq x$

and $[f(x\ y)] \geq y$.

Satisfied by $f(x\ y) = \max(x\ y)$

Satisfied by $f(x\ y) = \max(x\ y)$

$f(x\ y) = \max(x\ y) + 2$

any $f(x\ y) \geq \max(x\ y)$.

SyGuS: Given G $\exists f \in G$ s.t

$\forall \vec{x}$ $\varphi(\vec{x}\ f)$ — No single invocation property.

Alternate Defn: Given G $\exists f \in G$ s.t

$\forall \vec{x}$ $\varphi(\vec{x}\ f(\vec{x}))$

↑ Single invocation property.

Ex: Assume LIA.

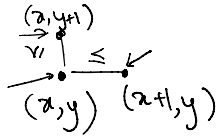
$\exists f\ \forall x\ y\ \ f(x+1\ y) \geq f(x\ y)$

Function f is monotonically increasing: $f(x\ y+1) \geq f(x\ y)$.

Ex: Assume LIA.

$\exists f.\ \forall x\ y\ \ f(x\ y) \geq x$

and $f(x\ y) \geq y$.



To recap: Invariant generation. (Inductive)

Find a predicate I s.t. $(x \leq 5$ and $y + x \geq 100)$.

① All initial states satisfy the predicate

① All initial states satisfy the predicate

② For all transitions $q \to q'$

If q satisfies I then $q' \underbrace{\models}_{\text{satisfies}} I$

③ All states satisfying I are safe.

$\forall q$ if $q \models I$ then q satisfies the assertion.

Predicate: A function $I : Q \to Bool$.

(Q: State)

$$q \models I \equiv I(q) = true \equiv I(q)$$

Question: Can we use a program synthesizer to come up with I?

Fix a suitable grammar G.

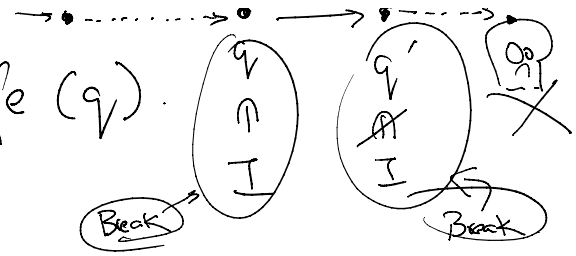
$\exists I : Q \to Bool$ s.t. $\forall q\ q'$

① $initial(q) = true \implies I(q) = true$, and

② $I(q) = true$ and $next(q\ q') \implies I(q') = true$ and

③ $I(q) = true \implies safe(q)$.

(3) $I(q) = \text{true} \Rightarrow \text{safe}(q)$.



Single Invocation Problems

① $\exists f \; \forall \vec{x} \quad \varphi(\vec{x} \; f(\vec{x}))$: Does this admit a solution?

② $\forall \vec{x} \; \exists y \quad \varphi(\vec{x} \; y)$: Is this formula true?

$f : \mathbb{D} \times \mathbb{D}'$ s.t.

$f = \{(d_1, d_1'), (d_2, d_2'), \ldots, \ldots\}$

$(d \; d')(d \; d'') \in f \Rightarrow d' = d''$:

Claim : ① $\Longleftrightarrow$ ②

Proof of $\Rightarrow$ Observe that setting $y = f(\vec{x})$ works.

$$\forall x, \; y = f(\vec{x}) \Rightarrow \varphi(\vec{x} \; y).$$

Proof of $\Leftarrow$: Let's work on Booleans (for simplicity).

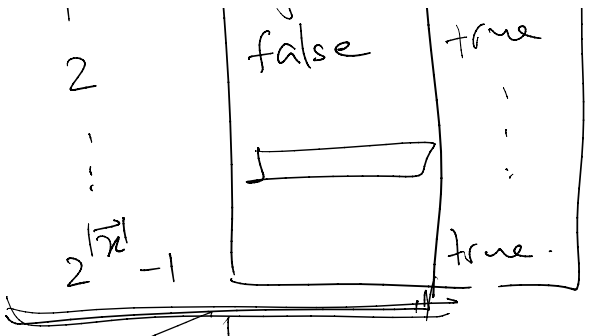
- $\vec{x}$ can assume $2^{|\vec{x}|}$ values

- Draw a table:

$\forall \vec{x}, \exists y \cdot \varphi(\vec{x} \; y)$

↑ Vector of Booleans ↑ Scalar

$\vec{x}$	y	$\varphi(\vec{x} \; y)$
0	true	true
1	false	true
2	false	true



To construct: f s.t.

$\forall \vec{x} \quad \varphi(\vec{x}, f(\vec{x}))$.

Translate into a Boolean fn $f: 2^{|\vec{x}|} \to \text{Bool}$.

Table already represents a mathematical fn.

So our proof is technically complete.

Missing: Succinct description as an expression of G.

Compression algorithm is missing.

Single Invocation Problems

$\exists f \quad \forall \vec{x} \quad \varphi(\vec{x}, f(\vec{x}))$ ← Program synthesis problem

Complex object "function" $\forall \vec{x}$

Scalar Boolean value.

Skolemization

f: Skolem function

$\exists y \quad \varphi(\vec{x}, y)$

Not a fixed unchanging thing.

"Quantified Boolean Formula".

Complexity Classes

PSPACE-complete

P, NP, co-NP
PSPACE-complete

Decision problems
Answer: Y/N.

#P: Model counting

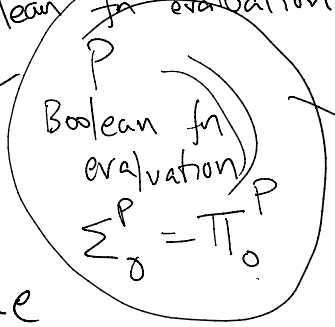
NP ⊇ NP-complete
sat

co-NP-C ⊆ co-NP
validity

#P

Counting problem
Answer: # of models.
$\varphi(\vec{x})$ true?

P: 2-SAT, unit propagation
graph reachability
Boolean ~~fn~~ evaluation



Satisfiability

$\exists \vec{x}\ \varphi(\vec{x})$ is true

$NP = \Sigma_1^P$

Validity

$\forall \vec{x}\ \varphi(\vec{x})$ is true

Π_1^P co-NP

$\exists \vec{x}\ \forall \vec{y}\ \varphi(\vec{x}, \vec{y})$

Σ_2^P

$\forall \vec{x}\ \exists \vec{y}\ \varphi(\vec{x}, \vec{y})$

Π_2^P

$\exists x\ \forall y\ \exists z\ \varphi(x\ y\ z)$

Σ_2^P

$\forall x\ \exists y\ \forall z\ \varphi(x\ y\ z)$

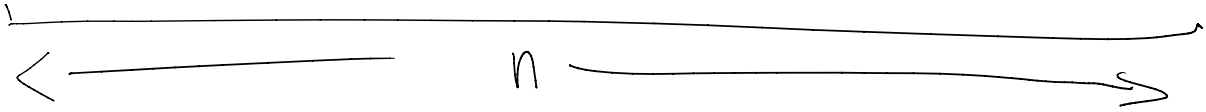
Π_3^P

Limiting case

PSPACE-complete: TQBF: True Quantified Boolean formula

$$\forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \forall x_k \underbrace{\varphi(x_1 x_2 \cdots x_k)}_{\text{Boolean formula}}$$

Alternation



$$(x_1 \wedge \neg x_2) \vee (x_1 \wedge \neg x_3)$$

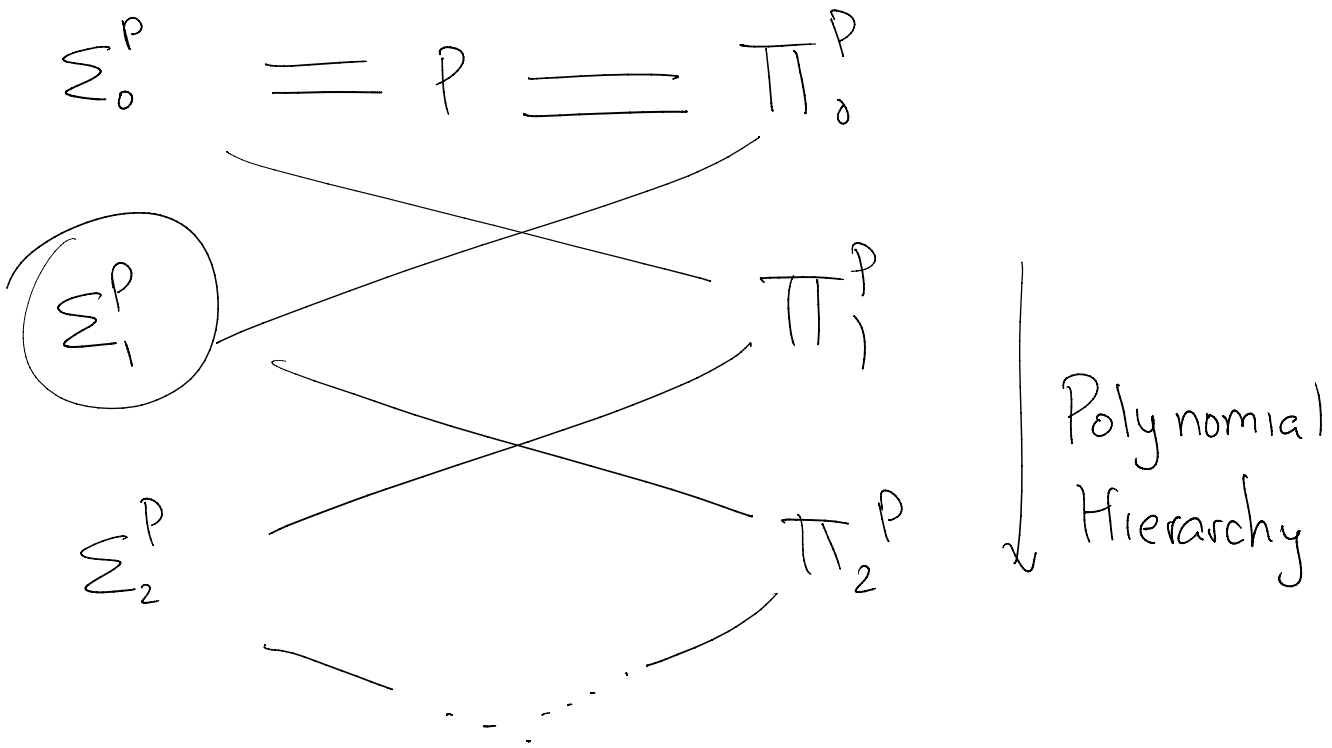
$\forall x \exists y \ \varphi(x\ y)$ is true

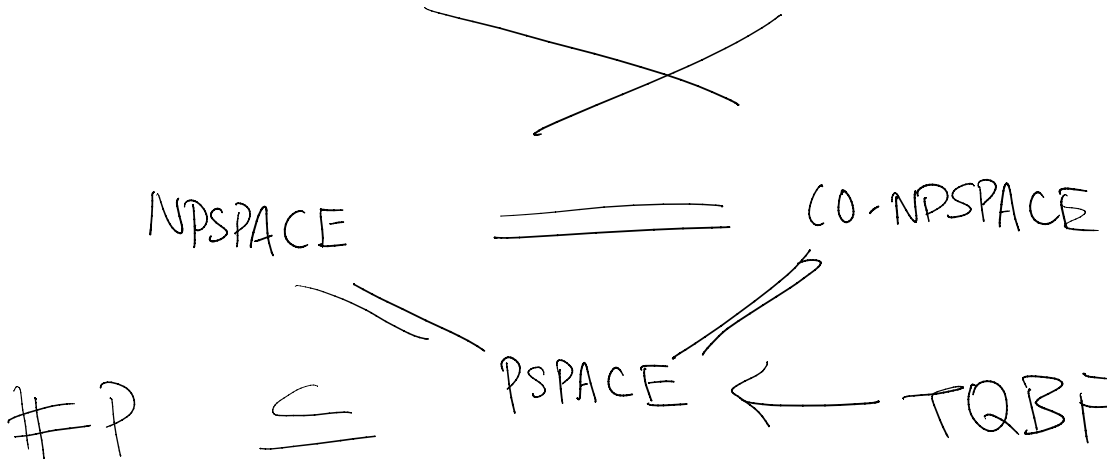
x	y
1	1
1	0
0	1
0	0

If $\exists y \varphi(\text{true}\ y)$ is true &

$\exists y \varphi(\text{false}\ y)$ is true

Iterate over all assignments to the variables.





Hard in practice.

PSPACE is the new NP.

Bonus Question: Is [illegible] TQBF?

Bonus Question 1: Can we use a TQBF solver to solve SyGuS?

Unit 1: Inductive Invariants & Hoare Logic

Unit 2: Algorithms for satisfiability & theorem proving

SAT SMT

Unit 3: Predicate abstraction & Static analysis / Datalog

Unit 4: Program synthesis

SyGuS problem & how to solve it

CEGIS - Enumerative

Unification

Unification
Stochastic search
Constraint-based learners

Connections to program verification
Complexity theory.

Bigger Question: How to help programmers write better code?

Human Machine Interfaces

How does the programmer specify their intent?

- How to build better solvers?

SAT, SMT
QBF, model counters,
first-order theorem provers,
...

L

Abstract domains.

Type system

- Machine Learning $\rightleftharpoons$ PL.
  - Can ML help build better verification engines?
  - Anomaly detection in programs?
  - Can we verify neural networks & other learning algorithms?

- Application: Program repair.

  Program engineering