Lecture 25

Wednesday, April 22, 2020 4:58 PM

- Hello!

- Last lecture: Building φ to CEGIS

- Today: Completing the description of the CEGIS framework

Some instantiations: Enumerative (ESolver)

Unification (EUSolver)

Constraint-based

Stochastic search.

- Not going to discuss: Sygus solver inside CVC4.

Quantifier elimination ⟶ unsat cores

↓

Proofs of unsatisfiability

↓

Function to be synthesized.

- Complexity aspects of program synthesis

From SAT solvers to TQBF solvers

From SAT solvers to TQBF solvers

↑ NP-Complete

↑ True Quantified Boolean Formulas

PSPACE-complete

- Building up to CEGIS

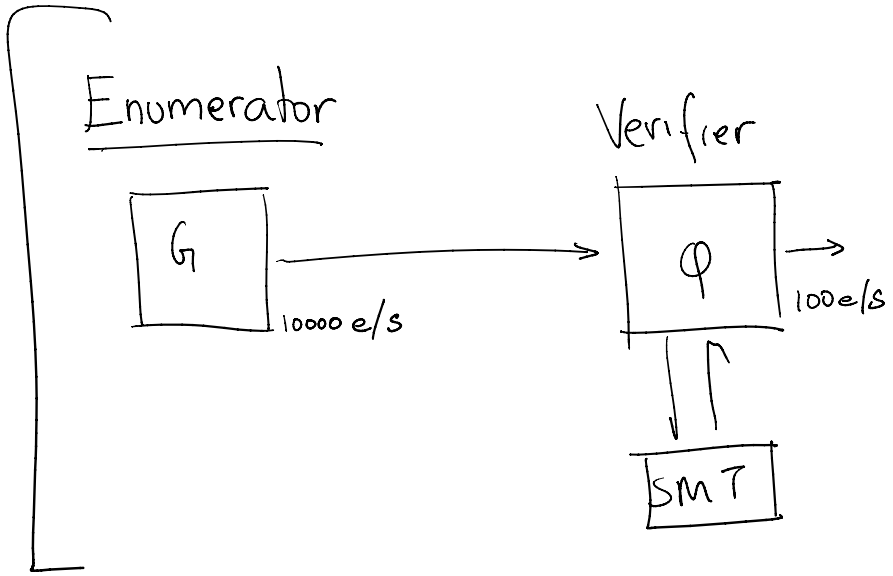
Spec V1.0

Given grammar G, specification $\varphi(\vec{x}\ f)$

find expression $f \in G$ s.t. $\forall \vec{x} . \varphi(\vec{x}\ f)$.

Factory Version 1.0

Farthest Expression # 10^6.



$\frac{10100\ s}{\sim 10000\ s} = 100\ s + 10000\ s$

(Dominated by

verification time).

Fix some finite set of $\vec{x}$-s : $C = \{\vec{x}_1, \vec{x}_2 \cdots \vec{x}_K\}$

Spec Version 2.0

Given G spec φ find $f \in G$ s.t.

① $\forall \vec{x} \in C$, $\varphi(\vec{x})$ holds

$\varphi(\vec{x}_1) \wedge \varphi(\vec{x}_2) \wedge \cdots \wedge \varphi(\vec{x}_n)$ ← Can be checked by evaluation.

② $\forall \vec{x}$ $\varphi(\vec{x})$ holds

Ex: $\varphi(x\ f) = f(x) \geq 0$ and $(f(x) = x$ or $f(x) = -x)$

Absolute value of x.

$C = \{-2, 1, 8\}$

$f(x) = x+2$.

Checking $\forall$-Spec: $\exists x.\ x+2 \geq 0$ and $(x+2 = x$ or $x+2 = -x)$

$\Leftrightarrow \exists x.\ x+2 \not\geq 0$ or $(x+2 \neq x$ and $x+2 \neq -x)$

$\neg\varphi(x, f)$

Checking C-Spec: $\exists x \in C.\ \neg\varphi(x\ f)$

Checking C-Spec: $\exists x \in C. \neg\varphi(x\ f)$

$\Leftrightarrow \neg\varphi(-2\ f)$ or $\neg\varphi(1\ f)$ or $\neg\varphi(8\ f)$

$\Leftrightarrow (0 \not\geq 0$ or $(0 \neq -2$ and $0 \neq 2))$

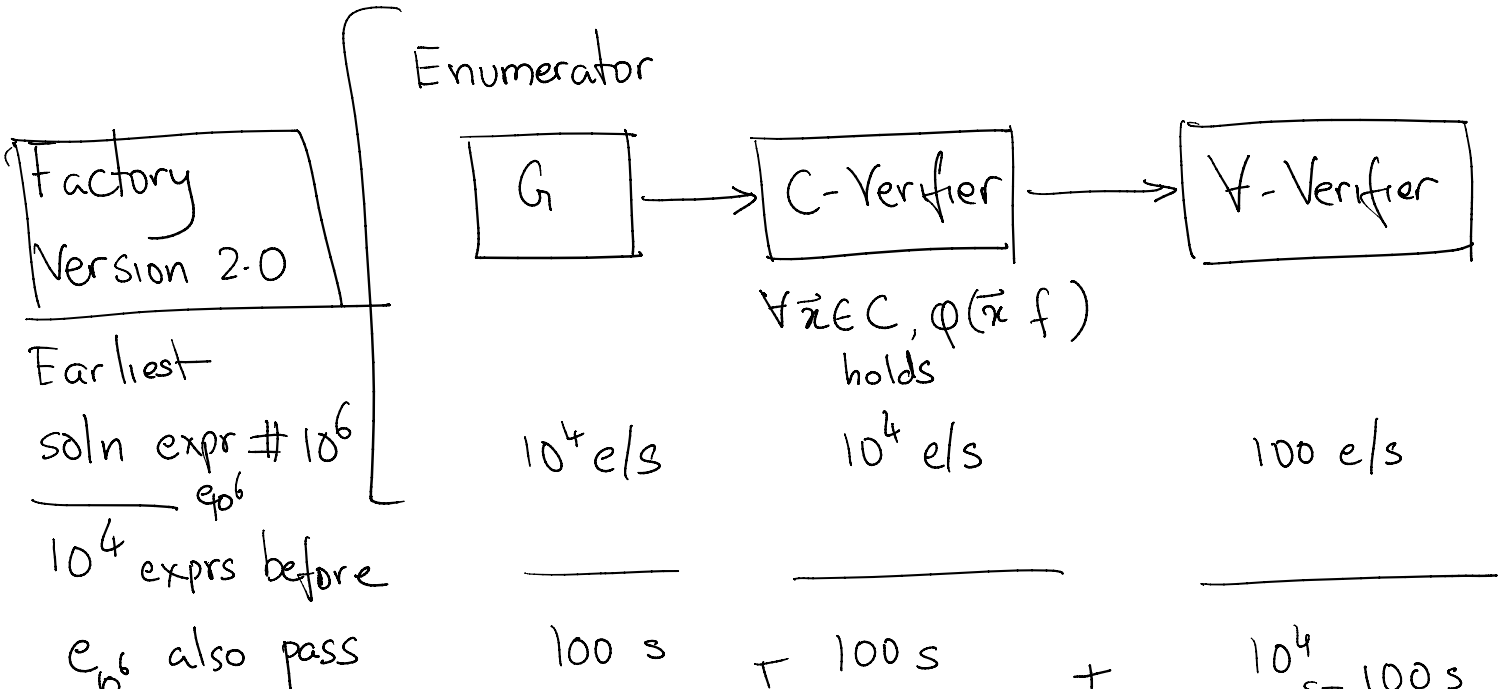
All 3 evaluate to true.

→ or $(3 \not\geq 0$ or $(3 \neq 1$ and $3 \neq -1))$

→ or $(10 \not\geq 0$ or $(10 \neq 8$ and $10 \neq -8))$

Can be checked simply by plugging in f into φ & subsequently evaluating.

(Can bypass SMT solver).



e_{6} also pass C-verifier 100 s + 100 s + $\frac{10^4}{100}$ s = 100 s

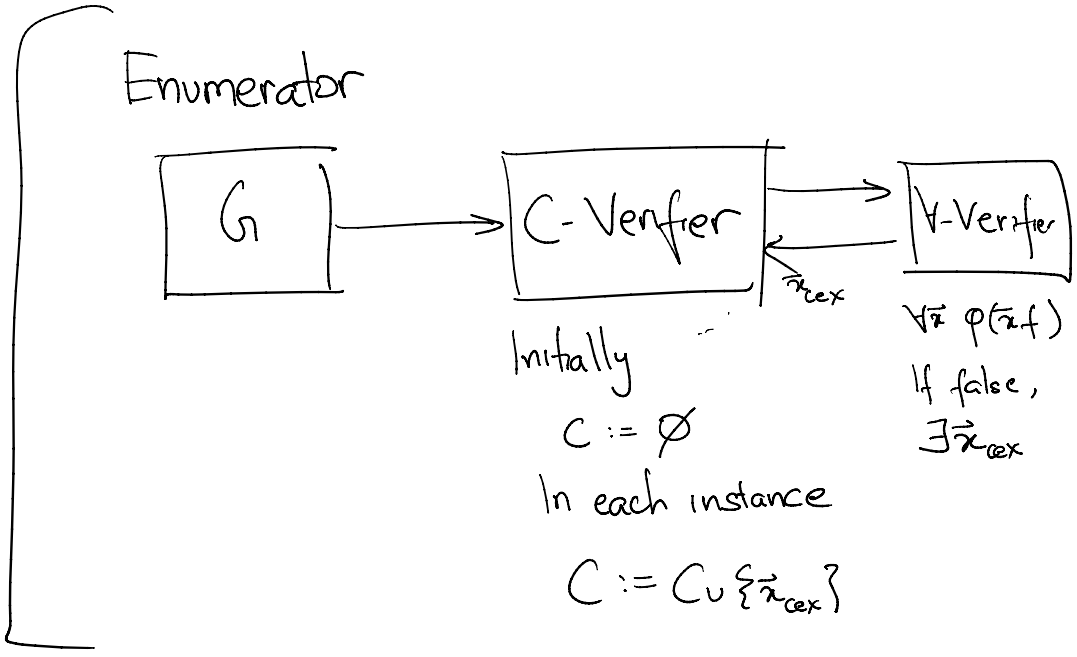
300 s

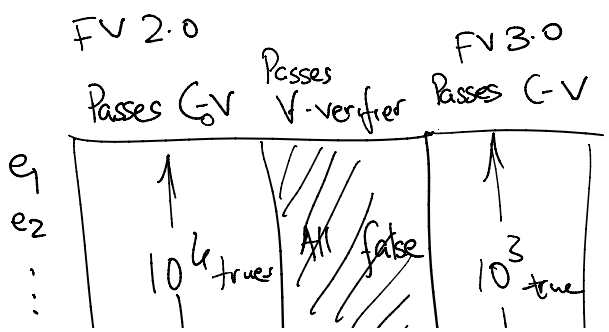
How to pick C?

- Use counterexample points from ∀-Verifier as elements of C.

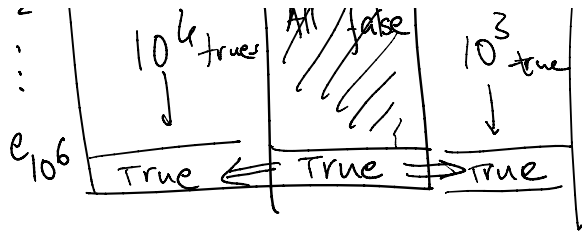
Factory Version 3.0

Earliest soln expr # 10^6



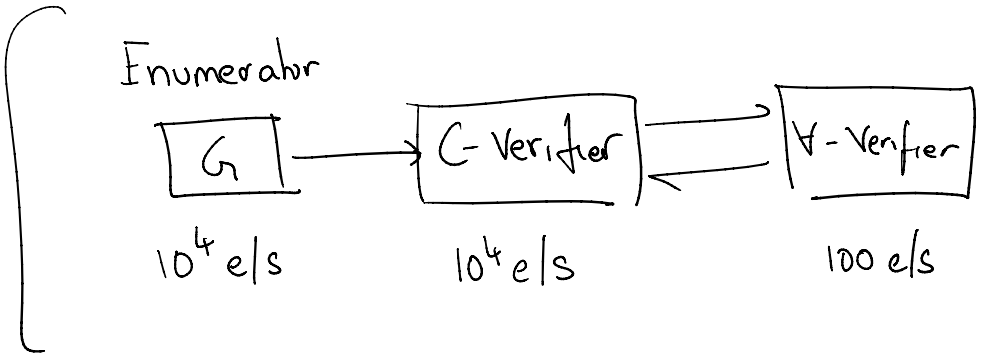


C-Verifier becomes increasingly strict over time. Passes



Over time. Passes fewer & fewer expressions to V-verifier.

FV 3.0



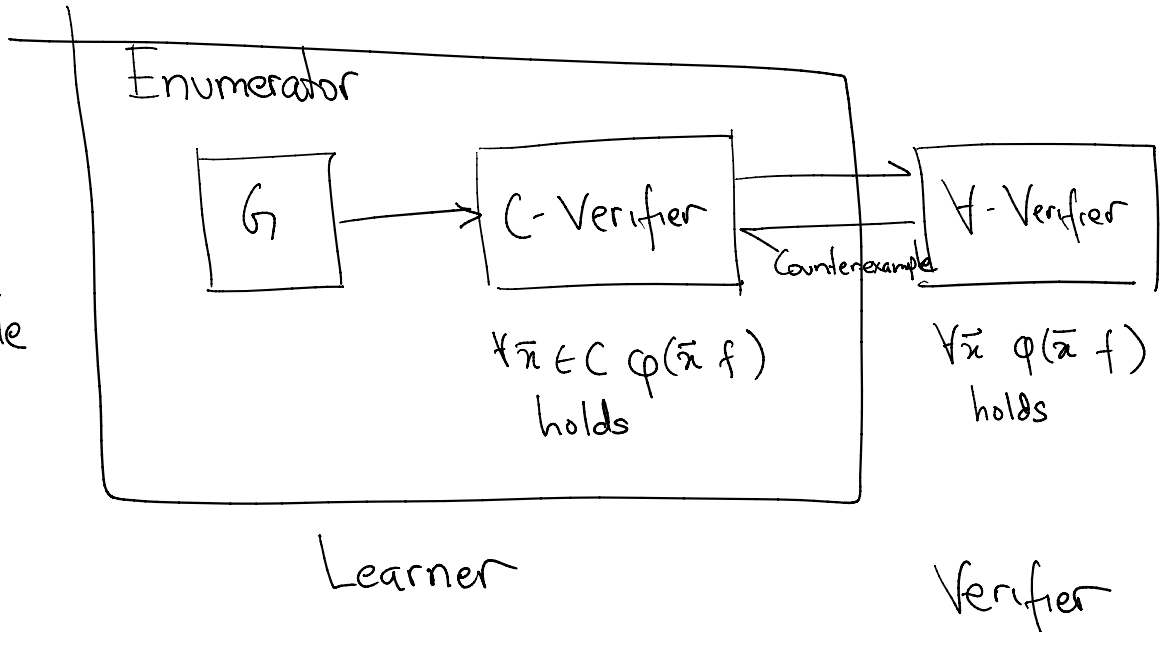
210 s = 100s + 100s + 10s

	FV 1.0	FV 2.0	FV 3.0	...
Time	10100 s	300 s	210 s	$\geq$100s
Bottleneck	V-Verifier	C-Verifier +Enumerator	C-Verifier +Enumerator	Enumeration bottleneck

---

CEGIS

Counterexample-Guided Inductive Synthesis



Synthesis

Learner

$\forall \vec{x} \in C \quad \varphi(\vec{x}\ f)$ holds

Verifier

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

Induction

Induction / Deduction / Abduction

↑ Induction

I saw a crow today. It was black.

I saw a crow yesterday. It was black.

∴ All crows are black.

— Machine learning

— Natural science

↑ Deduction

$A \Rightarrow B$. A.
Therefore B

If it is raining, then I must be wet.
I am not wet.
Therefore it is not raining.

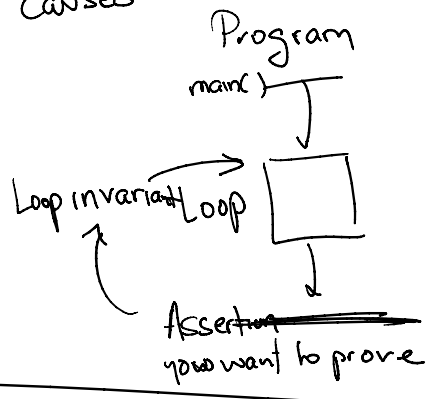
↑ Abduction

Most likely explanations

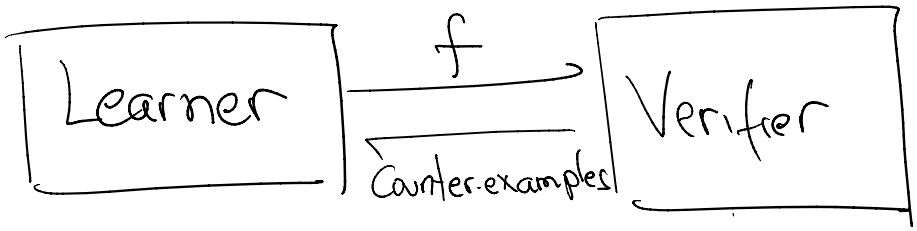
Caricature of a detective

- Policeman saw a dead body.
- He saw a knife next to the body
- The victim must have been killed with the knife.

Reverse engineering of causes



CEGIS



$\forall \vec{x} \in C \; \varphi(\vec{x}, f)$ holds

$\forall \vec{x} \; \varphi(\vec{x}, f)$ holds

Q1: Is there a universal learner?

Is there a learner s.t. $\forall \varphi$ $\forall$ verifiers, the learner eventually converges to the target fn?

A: FV 3.0 is a universal learner

Q2: Is there a universal teacher?

Is there a verifier s.t. $\forall \varphi$ $\forall$ learners, the learner eventually converges to the target fn?

A: In general, no.

No defense against stubborn students. ☺

- If grammar G is finite, then every teacher is universal

then every teacher is universal.

	FV 1.0	FV 2.0	FV 3.0	...	
Time	10100 s	300 s	210 s		$\geq$ 100 s
Bottleneck	V. Verifier	C-Verifier + Enumerator	C-Verifier + Enumerator		Enumerator bottleneck

Building a simple bottom-up enumerator. G_1:

Start ::= x | y | 0 | 1
| Start + Start

All Expressions of G_1 in order = All Exprs of G_1 of size 1 in order + All Exprs of G_1 of size 2 in order +

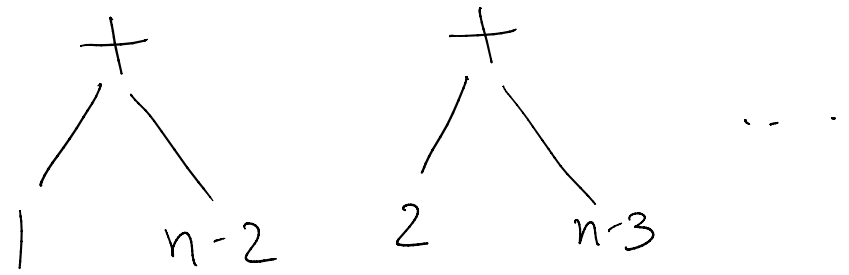
$$G^* = G^1 ++ G^2 ++ G^3 ++ \cdots$$

How to build G^n ?

How to build G^n:

If $n=1$: simply $[x, y, 0, 1]$

Otherwise:



Ex: $0 + (x+y)$

For each n_1 n_2 s.t. $n_1+n_2 = n-1$

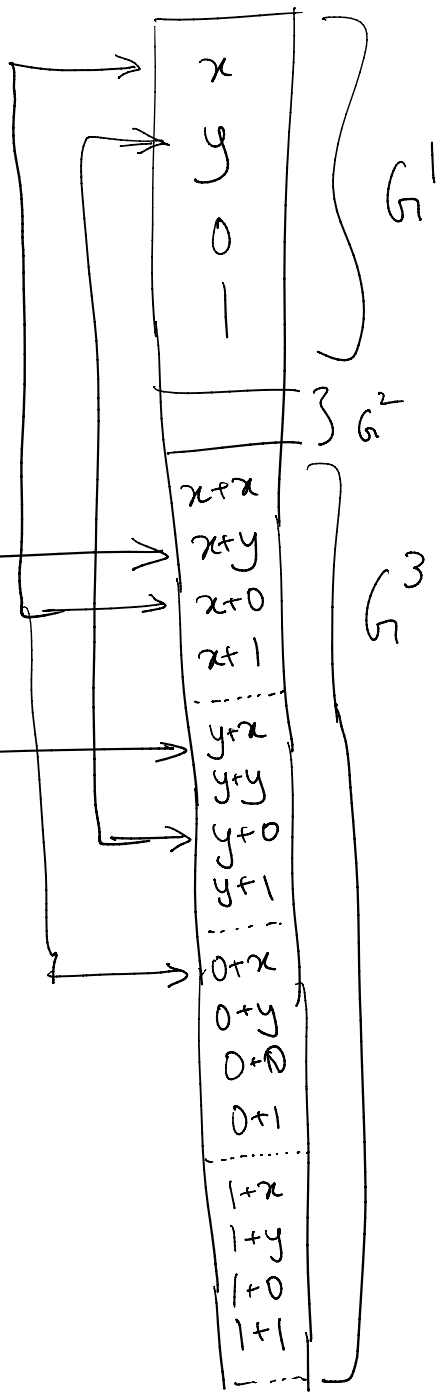
pick e_1 in G^{n_1} $e_2 \in G^{n_2}$

produce e_1+e_2.

Accelerating the Enumerator

G_1:

Start ::= $x \mid y \mid 0 \mid 1$
| Start + Start



Enumeration of Many Algebraically Equivalent Expressions

$\forall\ e_1\ e_2 \quad e_1 + e_2 = e_2 + e_1$

$\forall\ e_1\ e_2\ e_3 \quad (e_1 + e_2) + e_3 = e_1 + (e_2 + e_3)$

$\forall\ e \quad e + 0 = 0 + e = e$

Question: How to prevent enumeration of algebraically equivalent expressions?