Counter Example-Guided Abstraction-Refinement

Clarke, Grumberg, Jha, Lu, Veith CAV 2000.

- Used in Microsoft's SLAM tool for Device driver verification
- CPA Checker wins C program verification competitions
- Led to development of CEGIS Inductive synthesis tools
- Timed & Hybrid Variants HARE
- Automation of the Scientific Method!

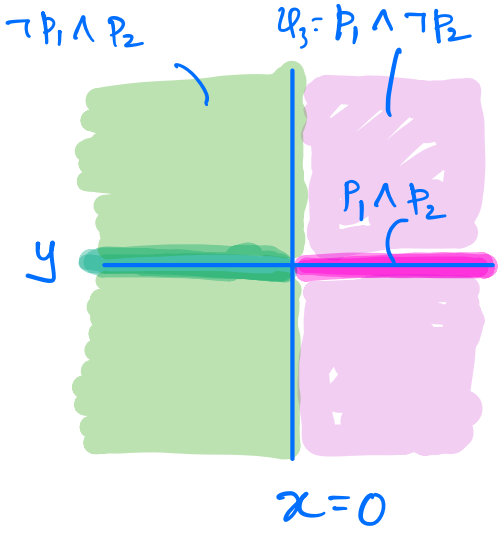
Predicate Abstraction

Given an automaton $\mathcal{A} = \langle Q, Q_0, A, \delta \rangle$ and a set of predicates $\mathcal{P} = \{P_1, \ldots, P_n\}$ $\quad P_i : Q \to \{0,1\}$ defines an abstract automaton over the state space $\{0,1\}^*$

Cubes($\mathcal{P}$): Set of formulas of the form $\bigwedge_i \ell_i$ where each $\ell_i \in \{P_i, \neg P_i\}$ is a literal

Ex. $Q = Val(\{x, y\})$

$\mathcal{P} = \{P_1, P_2\}$ $\quad P_1 : x > 0$ $\quad P_2 : y = 0$



These are the facts/propositions about the state variables that the abstraction will track.

partial cube $\varphi_1 : \{P_1\}, \{\neg P_2\}$

Complete cube $\varphi_3 : \{P_1 \wedge \neg P_2\}$

$\mathcal{P}$ partitions Q into equivalence classes or cells

$q_1 \equiv_{\mathcal{P}} q_2$ iff $\forall i \quad P_i(q_1) = P_i(q_2)$

Each complete cube corresponds to a particular cell

Each partial cube corresponds to an union of cells

$S, \leq$ $\quad \leq \subseteq S \times S$

reflexive $\quad \forall a \in S \quad a \leq a$

transitive $\quad \forall a, b, c \in S \quad a \leq b \wedge b \leq c \Rightarrow a \leq c$

anti-sym $\quad \forall a, b \quad a \leq b \Rightarrow b \not\leq a$

Predicate Abstraction Lattice

A Lattice $(S, \sqsubseteq)$ is a set S endowed with a partial order $\sqsubseteq$
(A partial order $\sqsubseteq \subseteq S \times S$ is a reflexive, antisym, transitive rel)

Such that for any pair $s_1, s_2 \in S$ has

- a least upper bound (join) $s_1 \sqcup s_2$
- a greatest lower bound (meet) $s_1 \sqcap s_2$

Complete lattice : All subsets have joins and meets (not just pairs)

Example Lattice for any set S $(\wp(S), \subseteq)$

- $\subseteq$ induces a partial order on $s_1, s_2 \subseteq S$

join $s_1 \sqcup s_2 := s_1 \cup s_2$

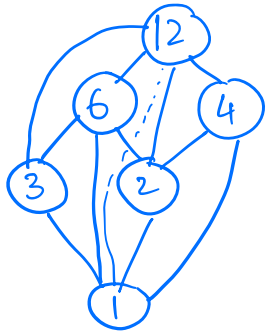
meet $s_1 \sqcap s_2 := s_1 \cap s_2$

- Divisors of a number, say 12

  $(\{1,2,3,4,6,12\}, \sqsubseteq)$ $a \sqsubseteq b$ iff $a \mid b$

join $a \sqcup b$: LCM

meet $a \sqcap b$: GCD



Non example. Open intervals in $\mathbb{R}$ ordered by inclusion $\subseteq$

$(0,1) \sqcap (2,3) = (0,1) \cap (2,3) = \phi$ not an open interval

Undefined

Claim $(Cubes(P), \Rightarrow)$ Forms a complete lattice

For any $\varphi_1, \varphi_2 \in Cubes(P)$ $\varphi_1 \sqsubseteq \varphi_2$ iff $\varphi_1 \Rightarrow \varphi_2$

How to compute joins and meets of Cubes?

$$\varphi_1 \sqcup \varphi_2 \triangleq \bigwedge_{\ell \in Lit(\varphi_1) \cap Lit(\varphi_2)} \ell$$

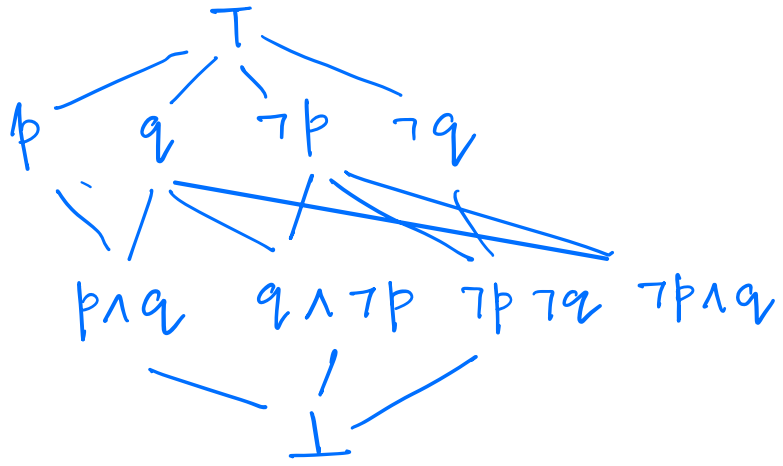
Keep identical literals
$\top$ if no common literal
Smallest Cell containing both

$$\varphi_1 \sqcap \varphi_2 \triangleq \begin{cases} \bot & \text{if } \exists \ell \in Lit(\varphi_1) \quad \neg\ell \in Lit(\varphi_2) \\ \bigwedge_{\ell \in Lit(\varphi_1) \cup Lit(\varphi_2)} \ell & \text{otherwise} \end{cases}$$

intersection

Example $P = \{p, q\}$ $p: x \le 100,\ y > 10$ $q: x = y$

$Cubes(P) = \{\top, \bot, p, q, \neg p, \neg q, \neg p \wedge \neg q, \neg p \wedge q, p \wedge q, p \wedge \neg q\}$



$(p \wedge q) \sqcup (p \wedge \neg q) = p$

$(p \wedge q) \sqcap (p \wedge \neg q) = \bot$

$p \sqcap q = p \wedge q$

$p \sqcup q = \top$

The predicates P define an abstract statespace $Cubes(P)$ with the lattice structure

We now define abstract transitions

Recall $post(q, a) = \{q' \mid (q, a, q') \in \delta\}$

Define $post(\varphi, a) = \{q' \mid \exists q \in [\![\varphi]\!] \ (q, a, q') \in \delta\}$

Abstract Post

$post^{\#}(\varphi, a) =$ **Smallest** Strongest cube φ' over P such that

$post(\varphi, a) \Rightarrow \varphi'$

Example

$P: \{x = y, \ x \neq y, \ x \geq y\}$

$Post^{\#}(\varphi: \{x = y\}, \ a = x := x+1)$

we have the post $x' = y' + 1$

both $x \neq y$ and $x \geq y$

therefore the strongest cube φ' over P

such that $post(\{x = y\}, \ x := x+1) \Rightarrow x \neq y, x \geq y$

$Post^{\#}(\varphi, a) = \{x \neq y, \ x \geq y\}$

Predicate Abstraction of Automaton A with given P

is the automaton $A^{\#} = \langle Cubes(P), Q_0^{\#}, Post^{\#} \rangle$

Example Automaton

$P = \{ \underbrace{x \leq 100}_{p_1}, \underbrace{x = y}_{p_2}, \underbrace{y = 100}_{p_3} \}$

What is

Variables $x, y := 0$

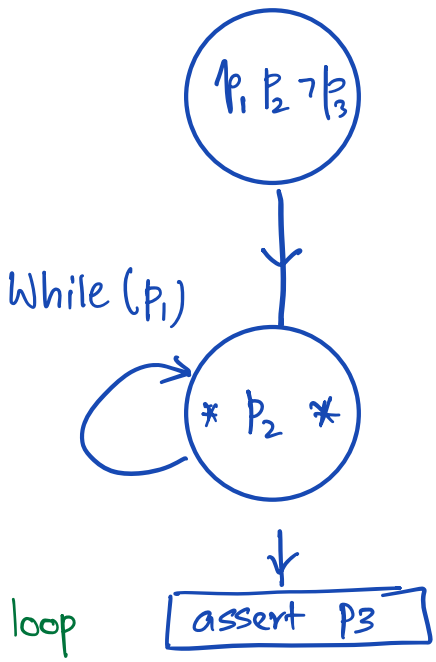
actions update

update

Pre $x < 100$

eff $x := x+1$

$y := y+1$



assert $y = 100$

Safety must hold at the end loop

$Post^{\#}(\{p_1, p_2, p_3\}, update)$

p_1 in prestate $x \leq 100 \Rightarrow x < 100$ or $x = 100$

in post state $x \leq 100$ or $x > 100$ p_1 or $\neg p_1$ *

p_2 in prestate $p_2 \Rightarrow x = y$

in post state p_2 is preserved

p_3 also unknown in general

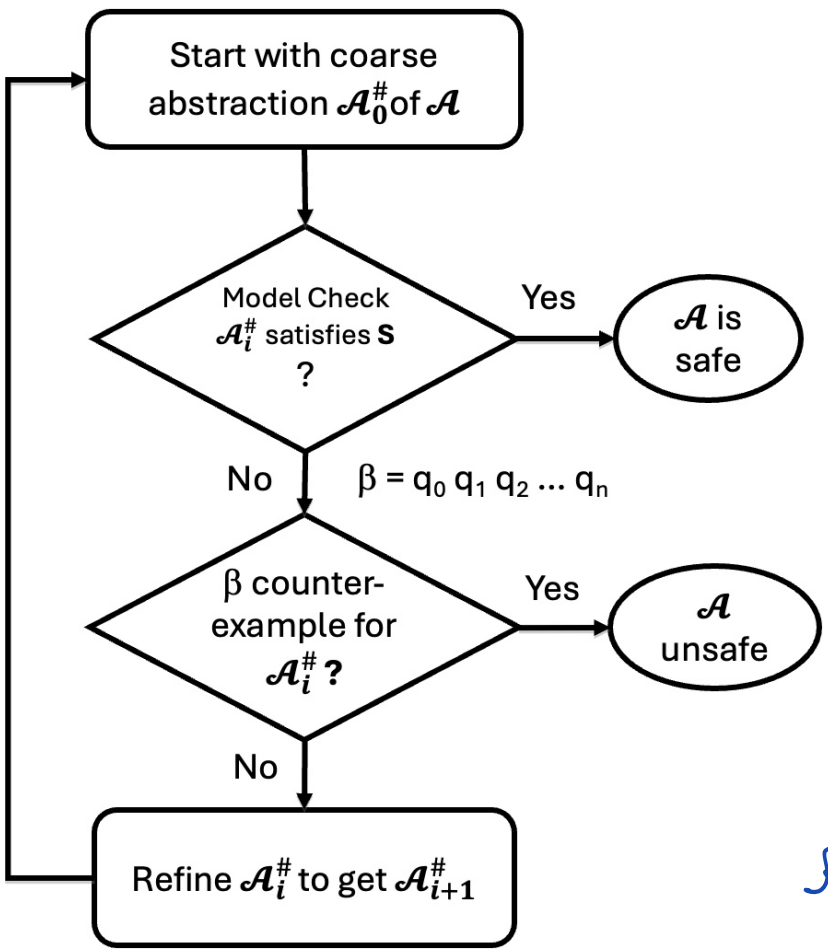
$\alpha^{\#} = (P_1, P_2, \neg P_3) \xrightarrow{update^{\#}} (*, P_2, *)$

We cannot prove $y=100, P3$ using this predicate abstraction of the program

Counter example generated by model checking $A^{\#}$ $\alpha^{\#}$

Observations: Predicate abstraction is very sensitive to the choice of the predicates P

CEGAR paradigm allows automatic and iterative learning of predicates that are sufficient for verification



Construct boolean abstraction of A converting each literal in the cube as a boolean variable

Search / BFS

Validation . SMT check

Introducing new predicates in A_{i+1} based on validation

Start with $P = \{ \underbrace{x \leq 100}_{p_1}, \underbrace{y = 100}_{p_2} \}$

Abstraction of the original program

$Q_0 = \{x = 0, y = 0\}$ $\quad$ $Q_0^{\#} = \langle p_1 = \text{true}, p_2 = \text{false} \rangle$

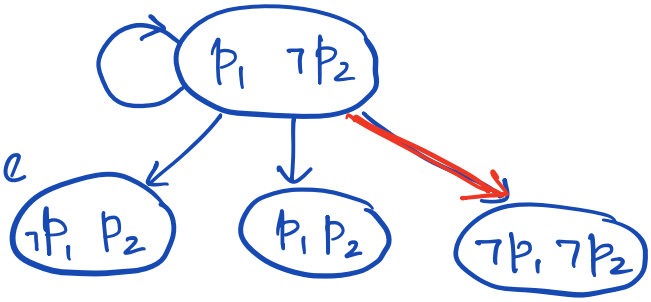
Guard $x < 100$ $\quad$ Guard$^{\#}$ p_1 because $x < 100 \Rightarrow x \leq 100$

transition $x := x+1$, $y := y+1$ $\quad$ trans$^{\#}$ $p_1 := *$, $p_2 := *$

assert (p_2)

Model checking

quickly finds a counter-example in $A^{\#}$ from $\langle p_1, \neg p_2 \rangle$ to $\langle \neg p_1, \neg p_2 \rangle$



Counter-example Validation checks whether the found cex $\alpha^\#$ for $A^\#$ does indeed correspond to a cex for A.

This involves running A while resolving nondeterministic choices according to $\alpha^\#$.

Replay $\alpha^\# \langle p_1 \neg p_2 \rangle \langle \neg p_1 \neg p_2 \rangle$

Step 1 Concrete $q_0 : (x=0, y=0)$ matches $p_1 : x \leq 100$, $\neg p_2 : y \neq 100$

Step 2 (enter loop) Pre $(x < 100)$ satisfied at q_0

Step 3 after 1 iteration $x := x+1$, $y := y+1$ $q_1 = (1, 1)$

Step 4 exit loop $\Rightarrow$ guard $(x < 100)$ must be false that is $x \geq 100$ but in q_1 $x = 1$

Contradiction

$\alpha^\#$ does not correspond to a valid cex of A. Spurious / fake cex

More generally $A^\#$ did not track 1. important relation between x and y $(x=y)$ and 2. Status of guard $(x<100)$ accurately $\Rightarrow$ early exit.

CEX-guided Refinement

Add predicates in P to eliminate spurious cex $\alpha^{\#}$

For the CEGAR loop to terminate refinement should eliminate a family of CEXs.

Basic idea compute strongest post conditions for each transition in $\alpha^{\#}$ and add these to the P.

Claim Consider a spurious cex $\alpha^{\#} = q_0 a_1 q_1 a_2 \cdots q_n$

Let $p_0 = \text{True}$ $p_i = SP(a_i, p_{i-1})$

Adding $p_1 \cdots p_n$ to P gives new abstract automaton $A^{\#}_{i+1}$ with $P' = P \cup \{p_1 \cdots p_n\}$ and $\alpha^{\#} \notin Execs_{A^{\#}_{i+1}}$

Proof by induction on n

Idea p_i is exactly the concrete set of reachable states following the prefix $q_0 a_1 q_1 a_2 \cdots q_i$

Refine with predicates $P_2 = \{\underbrace{x<100}_{p_3}, \underbrace{x \leq 100}_{p_1}, \underbrace{x=y}_{p_4}, \underbrace{y=100}_{p_2}\}$

$\mathcal{A}_2^\#$

$Q_0^\# = \langle p_1, p_3, p_4, \neg p_2 \rangle$

guard$^\#$: while p_3

p_1 = true

p_2 = *

p_3 = *

p_4 = true

assert (p_4)

$\neg p_3 \Rightarrow p_4$ model checking succeeds for $\mathcal{A}_2^\#$

any path that exits the loop must satisfy p_2

Soundness of abstraction

Implies that $\mathcal{A}$ satisfies the assertion.

Summary

- Start with coarse abstractions and refine only along counter-examples
- Property / Requirement driven refinement
- Right predicates / approximations matter
  e.g. linear constraints, box predicates
  approx linear dynamics with IRHA hybridization