CEGAR: Counterexample-guided Abstraction Refinement

Sayan Mitra
Slides from Pavithra Prabhakar

ECE/CS 584: Embedded System Verification

November 13, 2012

Outline

- Finite State Systems: Abstraction
- Refinement
- CEGAR
  - Validation
  - Refinment based on counterexample analysis
- Cegar for Hybrid Systems

Preliminaries

Finite state system (*FSS*) $\mathcal{T}$

- Q - finite set of states
- Σ - transition labels
- q^{init} - initial state
- $\rightarrow \subseteq Q \times \Sigma \times Q$ - transition function

Abstraction

Definition

Let $\mathcal{T}_1$ and $\mathcal{T}_2$ be two *FSS*s. $\mathcal{T}_2$ is an *abstraction* of $\mathcal{T}_1$ if there exists a function $\alpha : Q_1 \rightarrow Q_2$ such that $\alpha(q_1^{init}) = q_2^{init}$ and for every $q_1 \xrightarrow{a}_1 q_2$, $\alpha(q_1) \xrightarrow{a}_2 \alpha(q_2)$.

Notation: $\mathcal{T}_1 \prec \mathcal{T}_2$.

Given an equivalence relation $\sim \subseteq Q_1 \times Q_1$, define an abstraction $\mathcal{T}_2 = \mathcal{T}_1 / \sim$ of $\mathcal{T}_1$ as follows:

- $Q_2 = Q_1 / \sim$
- $q_2 \xrightarrow{a}_2 q_2'$ if there exists $q_1 \in q_2$ and $q_1' \in q_2'$ such that $q_1 \xrightarrow{a}_1 q_1'$.

Example

Abstraction

- Why do we want/need to abstract?

Abstraction

- Why do we want/need to abstract?
  To obtain "simpler" systems
- Does an abstraction preserve all properties?

Abstraction

- Why do we want/need to abstract?
  To obtain "simpler" systems
- Does an abstraction preserve all properties?
  - No. It preserves certain properties.
  - Example: If the abstraction is safe, then the original system is safe.
- Can an abstraction always prove the safety of a safe system?

Abstraction

- Why do we want/need to abstract?
  To obtain "simpler" systems
- Does an abstraction preserve all properties?
  - No. It preserves certain properties.
  - Example: If the abstraction is safe, then the original system is safe.
- Can an abstraction always prove the safety of a safe system?
  No. It might not be the right abstraction.
- How do we search for a "right" abstraction?

Abstraction

- Why do we want/need to abstract?
  To obtain "simpler" systems
- Does an abstraction preserve all properties?
  - No. It preserves certain properties.
  - Example: If the abstraction is safe, then the original system is safe.
- Can an abstraction always prove the safety of a safe system?
  No. It might not be the right abstraction.
- How do we search for a "right" abstraction?
  Refinement!

Refinement

Definition

Let $\mathcal{T}_1$ be a *FSS* and $\mathcal{T}_2$ its abstraction. A *refinement* of $\mathcal{T}_2$ is an *FSS* $\mathcal{T}_3$ such that $\mathcal{T}_1 \prec \mathcal{T}_3 \prec \mathcal{T}_2$.

Example

Definition

Let $\mathcal{T}_1$ be a *FSS* and $\mathcal{T}_2$ its abstraction. A *refinement* of $\mathcal{T}_2$ is an *FSS* $\mathcal{T}_3$ such that $\mathcal{T}_1 \prec \mathcal{T}_3 \prec \mathcal{T}_2$.

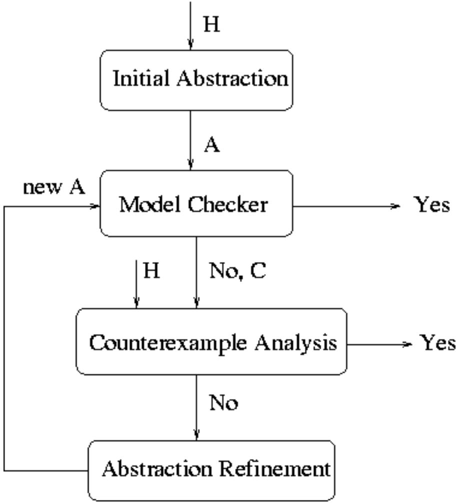
Example

How do we refine? One approach - CEGAR

Counter-example guided abstraction refinement

- Reachability problem: Given a state q_f, is it reachable from the initial state of $\mathcal{T}_1$?
- Construct an abstraction $\mathcal{T}_2$.
  - Model check $\mathcal{T}_2$: Is $\alpha(q_f)$ is reachable from the initial state of $\mathcal{T}_2$?
  - Answer no $\Rightarrow$ $\mathcal{T}_1$ is safe.
  - Answer yes $\Rightarrow$ don't know.
  - If yes, returns a path in $\mathcal{T}_2$ which reaches $\alpha(q_2)$ - abstract counter-example.
  - Check if the abstract counter-example has a corresponding concrete counter-example - validation
  - If yes, you have found a counter-example, and can conclude that the system is unsafe.
  - Otherwise, abstract counter-example is "spurious".
  - Use it to refine the abstraction.

CEGAR loop


H
Initial Abstraction
A
new A
Model Checker
Yes
H
No, C
Counterexample Analysis
Yes
No
Abstraction Refinement

Validation

$\sigma' = q'_1 \xrightarrow{a_1}_2 q'_2 \xrightarrow{a_2}_2 q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1}$: counter-example in $\mathcal{T}_2$.

Definition

Validation: Does there exist $\sigma = q_1 \xrightarrow{a_1}_2 q_2 \xrightarrow{a_2}_2 q_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q_{k+1}$ in $\mathcal{T}_1$ such that $q_1 = q_1^{init}$, $q_{k+1} = q_f$ and $\alpha(q_i) = q'_i$ for all i?

Validation

$\sigma' = q'_1 \xrightarrow{a_1}_2 q'_2 \xrightarrow{a_2}_2 q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1}$: counter-example in $\mathcal{T}_2$.

Definition

Validation: Does there exist $\sigma = q_1 \xrightarrow{a_1}_2 q_2 \xrightarrow{a_2}_2 q_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q_{k+1}$ in $\mathcal{T}_1$ such that $q_1 = q_1^{init}$, $q_{k+1} = q_f$ and $\alpha(q_i) = q'_i$ for all i?

Validation procedure

For $1 \leq i \leq k+1$, compute $Reach_i$: set of all states which can "mimic" $q'_i \xrightarrow{a_i} \cdots \xrightarrow{a_k} q'_{k+1}$ and reach q_f.

- $Reach_{k+1} = \{q_f\}$.
- $Reach_i = \alpha^{-1}(q'_i) \cap Pre(Reach_{i+1}, a_i)$ for $1 \leq i \leq k$.

Proposition

Concrete σ exists iff $Reach_0 \neq \emptyset$.

Refinement using the counterexample analysis

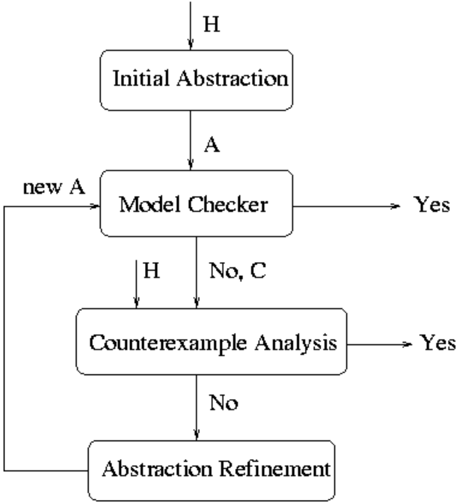
- Let j be the largest integer such that $Reach_j = \emptyset$, i.e., first set in the backward computation which becomes empty.
- $Post(\alpha^{-1}(q'_j)) \cap Reach_{j+1} = \emptyset$.
- Split q'_{j+1} into two states, such that one contains $Post(\alpha^{-1}(q'_j))$ and the other contains $Reach_{j+1}$.
- This is the new abstraction $\mathcal{T}_3$.

Related Work

CEGAR used in software verification

- Clarke et. al
- Microsoft research: SLAM, boolean programs
- Abstraction mechanisms: Predicate abstraction

Review of CEGAR algorithm

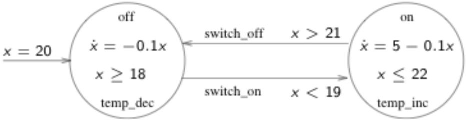


Hybrid system

$\mathcal{H} = (Loc, q_0, edges, Cont, Cont_0, flow, invariant, guard, reset)$:

- Loc - finite set of locations,
- $\ell_0 \in Loc$ - initial location,
- $edges \subseteq Loc \times Loc$,
- $Cont = \mathbb{R}^n$ - set of continuous states,
- $Cont_0 \subseteq Cont$ - initial continuous states,
- $flow : Loc \times Cont \rightarrow (\mathbb{R}_+ \rightarrow Cont)$,
- $invariant : Loc \rightarrow 2^{Cont}$,
- $guard : edges \rightarrow 2^{Cont}$, and
- $reset : edges \rightarrow 2^{Cont \times Cont}$.

Example: thermostat



- $Loc = \{off, on\}$,
- $\ell_0 = off$,
- $edges = \{(off, on), (on, off)\}$,
- $Cont = \mathbb{R}$,
- $Cont_0 = \{20\}$,
- $flow(off, x, t) = xe^{-0.1t}$ and $flow(on, x, t) = \cdots$,
- $invariant(off) = \{x \mid x \geq 18\}$ and $invariant(on) = \cdots$,
- $guard(off, on) = \{x \mid x < 19\}$ and $guard(on, off) = \cdots$, and
- $reset(off, on) = reset(on, off) = \{(x, x) \mid x \in \mathbb{R}\}$.

Semantics of a Hybrid Automaton

$[\![\mathcal{H}]\!] = (Q, \Sigma, q^{init}, \rightarrow)$:

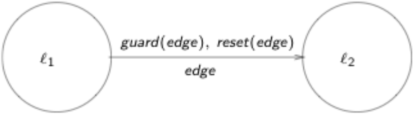
- $Q = Loc \times Cont$,
- $\Sigma = Actions \cup \{\tau\}$,
- $q^{init} = \{q_0\} \times Cont_0$.
- $(\ell, x) \rightarrow_a (\ell', x')$:
  - Discrete transitions: $a \in Actions$,
  - Continuous transitions: $a = \tau$.

Semantics of HA

Discrete transitions

$(\ell_1, x_1) \rightarrow_a (\ell_2, x_2)$ iff $\exists edge = (\ell_1, a, \ell_2)$:

- $x_1 \in guard(edge)$, and
- $(x_1, x_2) \in reset(edge)$.

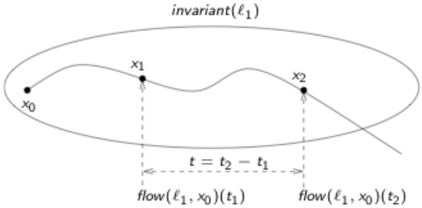


Semantics of HA

Continuous transitions

$(\ell_1, x_1) \rightarrow_\tau (\ell_2, x_2)$ iff

- $\ell_1 = \ell_2$,
- $\exists t$ such that $flow(\ell_1, x_1)(t) = x_2$, and for all $t' \in [0, t]$, $flow(\ell_1, x_1)(t') \in invariant(\ell_1)$.



Discrete abstraction

Definition

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two *HS*s. $\mathcal{H}_2$ is an *abstraction* of $\mathcal{H}_1$ if there exists a function $\alpha : Q_1 \rightarrow Q_2$ such that $\alpha(q_1^{init}) = q_2^{init}$ and for every $q_1 \xrightarrow{a}_1 q_2$, $\alpha(q_1) \xrightarrow{a}_2 \alpha(q_2)$.

Discrete abstraction

Definition

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two *HS*s. $\mathcal{H}_2$ is an *abstraction* of $\mathcal{H}_1$ if there exists a function $\alpha : Q_1 \to Q_2$ such that $\alpha(q_1^{init}) = q_2^{init}$ and for every $q_1 \xrightarrow{a}_1 q_2$, $\alpha(q_1) \xrightarrow{a}_2 \alpha(q_2)$.

- Finite partition of the state space
- Construct the transitions - time transitions and discrete transitions

Challenges

- Defining the partition - by linear constraints, polynomial constraints, ellipsoids etc.

Challenges

- Defining the partition - by linear constraints, polynomial constraints, ellipsoids etc.
- Computing discrete transitions - computing weakest preconditions, intersections.
  - $P \xrightarrow{a} P'$ iff $Pre(P', a) \cap P \neq \emptyset$.

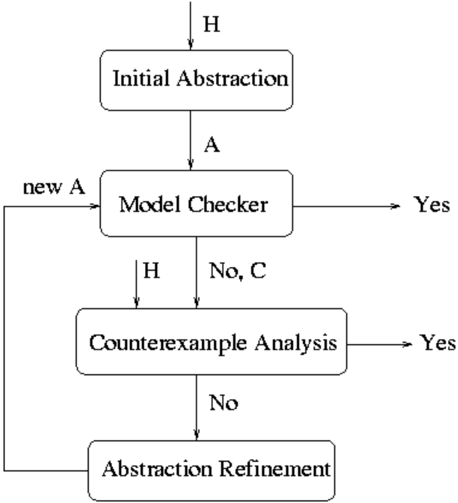
Challenges

- Defining the partition - by linear constraints, polynomial constraints, ellipsoids etc.
- Computing discrete transitions - computing weakest preconditions, intersections.
    - $P \xrightarrow{a} P'$ iff $Pre(P', a) \cap P \neq \emptyset$.
- Computing time transitions - computing time predecessors, intersections.
    - Depends on the continuous dynamics
    - $P \xrightarrow{\tau} P'$ iff $\cup_t Pre(P', t) \cap P \neq \emptyset$.

Challenges

- Defining the partition - by linear constraints, polynomial constraints, ellipsoids etc.
- Computing discrete transitions - computing weakest preconditions, intersections.
  - $P \xrightarrow{a} P'$ iff $Pre(P', a) \cap P \neq \emptyset$.
- Computing time transitions - computing time predecessors, intersections.
  - Depends on the continuous dynamics
  - $P \xrightarrow{\tau} P'$ iff $\cup_t Pre(P', t) \cap P \neq \emptyset$.
  - Often can only compute an approximation of $\cup_t Pre(P', t)$.
  - But still an abstraction!

CEGAR loop


H
Initial Abstraction
A
new A
Model Checker
Yes
H
No, C
Counterexample Analysis
Yes
No
Abstraction Refinement

Validation

$\sigma' = q'_1 \xrightarrow{a_1}_2 q'_2 \xrightarrow{a_2}_2 q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1}$: counter-example in $\mathcal{T}_2$.

Validation procedure

For $1 \leq i \leq k+1$, compute $Reach_i$: set of all states which can "mimic" $q'_i \xrightarrow{a_i} \cdots \xrightarrow{a_k} q'_{k+1}$ and reach q_f.

- $Reach_{k+1} = \{q_f\} \times inv(q_f)$.
- $Reach_i = \alpha^{-1}(q'_i) \cap Pre(Reach_{i+1}, a_i)$ for $1 \leq i \leq k$.

Validation

$\sigma' = q'_1 \xrightarrow{a_1}_2 q'_2 \xrightarrow{a_2}_2 q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1}$: counter-example in $\mathcal{T}_2$.

Validation procedure

For $1 \leq i \leq k+1$, compute $Reach_i$: set of all states which can "mimic" $q'_i \xrightarrow{a_i} \cdots \xrightarrow{a_k} q'_{k+1}$ and reach q_f.

- $Reach_{k+1} = \{q_f\} \times inv(q_f)$.
- $Reach_i = \alpha^{-1}(q'_i) \cap Pre(Reach_{i+1}, a_i)$ for $1 \leq i \leq k$.

- Cannot compute $Reach_i$ exactly.
- What do we do?

Validation

$\sigma' = q'_1 \xrightarrow{a_1}_2 q'_2 \xrightarrow{a_2}_2 q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1}$: counter-example in $\mathcal{T}_2$.

Validation procedure

For $1 \leq i \leq k+1$, compute $Reach_i$: set of all states which can "mimic" $q'_i \xrightarrow{a_i} \cdots \xrightarrow{a_k} q'_{k+1}$ and reach q_f.

- $Reach_{k+1} = \{q_f\} \times inv(q_f)$.
- $Reach_i = \alpha^{-1}(q'_i) \cap Pre(Reach_{i+1}, a_i)$ for $1 \leq i \leq k$.

- Cannot compute $Reach_i$ exactly.
- What do we do?
  - If validation fails, i.e., $Reach_k = \emptyset$ for some k, continue to the refinement step.
  - Otherwise, use better *Pre* computation. (After sometime, give up!)
  - If $Reach_0 \neq \emptyset$, cannot conclude anything. But can only conjecture that the design is probably not good.

Refinement

- Let k be the largest integer such that $Reach_k = \emptyset$, i.e., first set in the backward computation which becomes empty.
- $Post(\alpha^{-1}(q'_i)) \cap Reach_{k+1} = \emptyset$.
- Split q'_{k+1} into two states, such that one contains $Post(\alpha^{-1}(q'_i))$ and the other contains $Reach_{k+1}$.
- This is the new abstraction $\mathcal{T}_3$.

- Add some constraint that splits the two sets of states.
- Alur et. al - find separating predicates for polyhedra.

Some remarks

- Need not terminate, unlike for the finite state case.
- Even upon termination due to validation of a counter-example, cannot conclude that the system is erroneous (due to overapproximations).
- In general, computing abstractions is expensive.

Main concept

Abstract a hybrid automaton by another "simpler" hybrid automaton.

Hybrid Automata based CEGAR

Main concept

Abstract a hybrid automaton by another "simpler" hybrid automaton.

- Constructing abstractions may be easier, example, approximating a linear system by a rectangular system.
- Need different refinement methods.
- Hybrid Automata based CEGAR for rectangular hybrid automata - complete for the initialized fragment.

Abstraction

- $\alpha_{Loc} : Loc_1 \to Loc_2$.
- $\alpha_{edges} : edges_1 \to edges_2$.
- $\alpha_{Var} : \mathbb{R}^{|Var_1|} \to \mathbb{R}^{|Var_2|}$.

Abstraction

- $\alpha_{Loc} : Loc_1 \rightarrow Loc_2$.
- $\alpha_{edges} : edges_1 \rightarrow edges_2$.
- $\alpha_{Var} : \mathbb{R}^{|Var_1|} \rightarrow \mathbb{R}^{|Var_2|}$.

Example:

- x_1, x_2, x_3 - variables in $\mathcal{H}_1$.
- z_1, z_2 - variables in $\mathcal{H}_2$.
    - $\alpha_{Var}(z_1) = x_1 + 3x_2$
    - $\alpha_{Var}(z_2) = x_1 - x_3$

CEGAR for rectangular hybrid automata

- Abstract a rectangular hybrid automata by another hybrid automata.
- Example

CEGAR for rectangular hybrid automata

- Abstract a rectangular hybrid automata by another hybrid automata.
- Example
- Focus on scaling/omitting variable abstractions.
- Refinement in rectangular hybrid automaton by adding or scaling variables.

Completeness

- Can compute reach exactly.
- When the variable abstraction function corresponds to scaling/omitting variables, and the rectangular hybrid automaton is initialized rectangular, the CEGAR loop terminates always with the right answer.
  - Abstracts initialized RHA to initialized RHA.
  - IRHA have "finite bisimilation" (when converted to multirate automata).

Summary

- CEGAR for discrete systems.
- CEGAR for hybrid systems.
  - Discrete abstractions
    Alur et. al - TACAS 2003 Clarke et. al - TACAS 2003
  - Hybrid abstractions
    Larsen et. al - FORMATS 2007, Prabhakar et. al - VMCAI 2012.
  - CEGAR for stability
    Duggirala & Mitra ICCPS 2011, HSCC 2012