CEGAR: Counterexample-guided Abstraction Refinement

Sayan Mitra
Slides from Pavithra Prabhakar

ECE/CS 584: Embedded System Verification

November 13, 2012
Finite State Systems: Abstraction
Refinement
CEGAR
  Validation
  Refinement based on counterexample analysis
Cegar for Hybrid Systems
Finite state system (FSS) $\mathcal{T}$

- $Q$ - finite set of states
- $\Sigma$ - 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 FSSs. $\mathcal{T}_2$ is an abstraction of $\mathcal{T}_1$ if there exists a function $\alpha : Q_1 \rightarrow Q_2$ such that $\alpha(q_1^{\text{init}}) = q_2^{\text{init}}$ 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?

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!

Slides from Pavithra Prabhakar
CEGAR: Counterexample-Guided Abstraction Refinement
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?
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?
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
Reachability problem: Given a state $q_f$, is it reachable from the initial state of $T_1$?

Construct an abstraction $T_2$.

- Model check $T_2$: Is $\alpha(q_f)$ is reachable from the initial state of $T_2$?
- Answer no $\Rightarrow T_1$ is safe.
- Answer yes $\Rightarrow$ don’t know.
- If yes, returns a path in $T_2$ 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

1. Initial Abstraction
2. Model Checker
   - Yes
   - No, C
3. Counterexample Analysis
   - Yes
   - No
4. Abstraction Refinement
   - new A

- H
Validation

\[ \sigma' = q_1' \xrightarrow{a_1} 2 q_2' \xrightarrow{a_2} 2 q_3' \xrightarrow{a_3} \cdots \xrightarrow{a_k} q_{k+1}' : \text{counter-example in } T_2. \]

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 \( T_1 \) such that \( q_1 = q_1^{\text{init}}, q_{k+1} = q_f \) and \( \alpha(q_i) = q_i' \) for all \( i \)?
Validation

\[ \sigma' = q'_1 \xrightarrow{a'_1} q'_2 \xrightarrow{a'_2} q'_3 \xrightarrow{a'_3} \cdots \xrightarrow{a'_k} q'_{k+1} : \text{counter-example in } T_2. \]

**Definition**

Validation: Does there exist \( \sigma = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q_{k+1} \) in \( T_1 \) such that \( q_1 = q_1^{\text{init}}, q_{k+1} = q_f \) and \( \alpha(q_i) = q'_i \) for all \( i \)?

**Validation procedure**

For \( 1 \leq i \leq k+1 \), compute \( \text{Reach}_i \): set of all states which can "mimic" \( q'_i \xrightarrow{a'_i} \cdots \xrightarrow{a'_k} q'_{k+1} \) and reach \( q_f \).

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

**Proposition**

*Concrete \( \sigma \) exists iff \( \text{Reach}_0 \neq \emptyset \).*
Let $j$ be the largest integer such that $\text{Reach}_j = \emptyset$, i.e., first set in the backward computation which becomes empty.

$\text{Post}(\alpha^{-1}(q'_j)) \cap \text{Reach}_{j+1} = \emptyset$.

Split $q'_{j+1}$ into two states, such that one contains $\text{Post}(\alpha^{-1}(q'_j))$ and the other contains $\text{Reach}_{j+1}$.

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

CEGAR: Counterexample-Guided Abstraction Refinement

Initial Abstraction

A

Model Checker

new A

Counterexample Analysis

Abstraction Refinement

H

Yes

No

H

Yes

No, C
\[ \mathcal{H} = (\text{Loc}, q_0, \text{edges}, \text{Cont}, \text{Cont}_0, \text{flow}, \text{invariant}, \text{guard}, \text{reset}): \]

- \text{Loc} - finite set of locations,
- \( \ell_0 \in \text{Loc} \) - initial location,
- \text{edges} \subseteq \text{Loc} \times \text{Loc},
- \text{Cont} = \mathbb{R}^n - set of continuous states,
- \text{Cont}_0 \subseteq \text{Cont} - initial continuous states,
- \text{flow} : \text{Loc} \times \text{Cont} \to (\mathbb{R}_+ \to \text{Cont}),
- \text{invariant} : \text{Loc} \to 2^{\text{Cont}},
- \text{guard} : \text{edges} \to 2^{\text{Cont}}, \text{ and }
- \text{reset} : \text{edges} \to 2^{\text{Cont} \times \text{Cont}}.\]
Example: thermostat

- \( \text{Loc} = \{ \text{off}, \text{on} \} \),
- \( \ell_0 = \text{off} \),
- \( \text{edges} = \{(\text{off}, \text{on}), (\text{on}, \text{off})\} \),
- \( \text{Cont} = \mathbb{R} \),
- \( \text{Cont}_0 = \{20\} \),
- \( \text{flow}(\text{off}, x, t) = xe^{-0.1t} \) and \( \text{flow}(\text{on}, x, t) = \cdots \),
- \( \text{invariant}(\text{off}) = \{x | x \geq 18\} \) and \( \text{invariant}(\text{on}) = \cdots \),
- \( \text{guard}(\text{off}, \text{on}) = \{x | x < 19\} \) and \( \text{guard}(\text{on}, \text{off}) = \cdots \), and
- \( \text{reset}(\text{off}, \text{on}) = \text{reset}(\text{on}, \text{off}) = \{(x, x) | x \in \mathbb{R}\} \).
Semantics of a Hybrid Automaton

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

- \( Q = \text{Loc} \times \text{Cont} \),
- \( \Sigma = \text{Actions} \cup \{ \tau \} \),
- \( q^{\text{init}} = \{ q_0 \} \times \text{Cont}_0 \).
- \((\ell, x) \rightarrow_a (\ell', x') \):
  - Discrete transitions: \( a \in \text{Actions} \),
  - Continuous transitions: \( a = \tau \).
Semantics of HA

Discrete transitions

\((\ell_1, x_1) \rightarrow_a (\ell_2, x_2)\) iff \(\exists \text{edge} = (\ell_1, a, \ell_2):\)

- \(x_1 \in \text{guard}(\text{edge})\), and
- \((x_1, x_2) \in \text{reset}(\text{edge})\).
Continuous transitions

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

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

**Definition**

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two HSs. $\mathcal{H}_2$ is an abstraction of $\mathcal{H}_1$ if there exists a function $\alpha : Q_1 \to Q_2$ such that $\alpha(q_{1}^{\text{init}}) = q_{2}^{\text{init}}$ and for every $q_1 \xrightarrow{a} q_2$, $\alpha(q_1) \xrightarrow{a} \alpha(q_2)$. 
Definition

Let $\mathcal{H}_1$ and $\mathcal{H}_2$ be two HSs. $\mathcal{H}_2$ 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 $q_1 \xrightarrow{a} q_2$, $\alpha(q_1) \xrightarrow{a} \alpha(q_2)$.

- 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 $\text{Pre}(P', a) \cap P \neq \emptyset$.
- Computing time transitions - computing time predecessors, intersections.
  - $P \tau \rightarrow P'$ iff $\bigcup_{t} \text{Pre}(P', t) \cap P \neq \emptyset$.
  - Often can only compute an approximation of $\bigcup_{t} \text{Pre}(P', t)$.
  - But still an abstraction!
Challenges

- Defining the partition - by linear constraints, polynomial constraints, ellipsoids etc.
- Computing discrete transitions - computing weakest preconditions, intersections.
  \[ P \xrightarrow{a} P' \iff \text{Pre}(P', a) \cap P \neq \emptyset. \]
- Computing time transitions - computing time predecessors, intersections.
  \[ P \xrightarrow{\tau} P' \iff \bigcup_t \text{Pre}(P', t) \cap P \neq \emptyset. \]

Often can only compute an approximation of \( \bigcup_t \text{Pre}(P', t) \). But still an abstraction!
Challenges

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

Slides from Pavithra Prabhakar

CEGAR: Counterexample-Guided Abstraction Refinement
CEGAR loop

Initial Abstraction

Model Checker

Counterexample Analysis

Abstraction Refinement
\( \sigma' = q'_1 \xrightarrow{a_1} q'_2 \xrightarrow{a_2} q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1} \): counter-example in \( T_2 \).

Validation procedure

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

- \( \text{Reach}_{k+1} = \{ q_f \} \times \text{inv}(q_f) \).
- \( \text{Reach}_i = \alpha^{-1}(q'_i) \cap \text{Pre}(\text{Reach}_{i+1}, a_i) \) for \( 1 \leq i \leq k \).
Validation

\[ \sigma' = q'_1 \xrightarrow{a_1} q'_2 \xrightarrow{a_2} q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1} : \text{counter-example in } T_2. \]

**Validation procedure**

For \(1 \leq i \leq k+1\), compute \(\text{Reach}_i\): set of all states which can “mimic” \(q'_i \xrightarrow{a_i} \cdots \xrightarrow{a_k} q'_{k+1}\) and reach \(q_f\).

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

- Cannot compute \(\text{Reach}_i\) exactly.

- What do we do?
Validation

\[ \sigma' = q'_1 \xrightarrow{a_1} q'_2 \xrightarrow{a_2} q'_3 \xrightarrow{a_3} \cdots \xrightarrow{a_k} q'_{k+1} : \text{counter-example in } T_2. \]

Validation procedure

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

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

- Cannot compute \( \text{Reach}_i \) exactly.
- What do we do?
  - If validation fails, i.e., \( \text{Reach}_k = \emptyset \) for some \( k \), continue to the refinement step.
  - Otherwise, use better \( \text{Pre} \) computation. (After sometime, give up!)
  - If \( \text{Reach}_0 \neq \emptyset \), cannot conclude anything. But can only conjecture that the design is probably not good.
Let $k$ be the largest integer such that $\text{Reach}_k = \emptyset$, i.e., first set in the backward computation which becomes empty.

$Post(\alpha^{-1}(q'_i)) \cap \text{Reach}_{k+1} = \emptyset$.

Split $q'_{k+1}$ into two states, such that one contains $Post(\alpha^{-1}(q'_i))$ and the other contains $\text{Reach}_{k+1}$.

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.
Hybrid Automata based CEGAR

Main concept

Abstract a hybrid automaton by another “simpler” hybrid automaton.
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

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

Example: 
- \( x_1, x_2, x_3 \) - variables in \( H_1 \).
- \( z_1, z_2 \) - variables in \( H_2 \).
- \( \alpha_{Var}(z_1) = x_1 + 3x_2 \)
- \( \alpha_{Var}(z_2) = x_1 - x_3 \)

Slides from Pavithra Prabhakar
CEGAR: Counterexample-Guided Abstraction Refinement
Abstraction

\[ \alpha_{\text{Loc}} : \text{Loc}_1 \rightarrow \text{Loc}_2. \]
\[ \alpha_{\text{edges}} : \text{edges}_1 \rightarrow \text{edges}_2. \]
\[ \alpha_{\text{Var}} : \mathbb{R}\mid\text{Var}_1\mid \rightarrow \mathbb{R}\mid\text{Var}_2\mid. \]

Example:

\[ x_1, x_2, x_3 - \text{variables in } \mathcal{H}_1. \]
\[ z_1, z_2 - \text{variables in } \mathcal{H}_2. \]
  \[ \alpha_{\text{Var}}(z_1) = x_1 + 3x_2 \]
  \[ \alpha_{\text{Var}}(z_2) = x_1 - x_3 \]
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 bisimulation” (when converted to multirate automata).
CEGAR for discrete systems.
CEGAR for hybrid systems.
  - Discrete abstractions
    Alur et. al - TACAS 2003 Clarke et. al - TACAS 2003
  - Hybrid abstractions
  - CEGAR for stability
    Duggirala & Mitra ICCPS 2011, HSCC 2012