Counterexample-Guided Abstraction Refinement

Edmund Clarke Orna Grumberg Somesh Jha Yuan Lu
Helmut Veith

Seminal Papers in Verification (Reading Group)

June 2012

Motivations:

Apply model checking to industrial problems.

- Main challenge: State explosion.
- How to tackle that:
  - Use BDD (10^{20} states), Symmetry Reductions, POR, ...
  - Abstraction Techniques.

Abstraction Techniques:

Idea: Remove details, simplify components that are *irrelevant*.

Concrete model $\sim$ Abstract model.

The abstract model is *smaller* $\Rightarrow$ Easier to verify.

It comes with an information loss:

- Over-approximation comes with False negatives.
- Under-approximation comes with False positives.

CounterExample Guided Abstraction Refinement:

Integrates:

- Symbolic model checking.
- Over-approximation abstraction.

$\Rightarrow$ It comes with false negatives.

Fully automatic, including abstraction refinement.

CEGAR general scheme

- Model extraction: Initial abstraction.
- Model-Check the Abstract Model.
- If no bug is found:
  - The concrete Model is safe.
- If a counterexample is found: Is it a concrete one?
  - Yes. "Happy" end.
  - No. Refine the abstraction and model-check again.

Summary:

We talked so far about:

- Motivations behind CEGAR.
- General scheme of the approach.

Now we will talk about:

- Abstraction validity.
- Initial (abstract) model *extraction*.
- Algorithms used to:
  - Check validity of the counterexample.
  - Refine the abstraction.

Later we will present extensions and use of CEGAR approach.

Existential abstraction definition:

Formally, a program P can be modeled as a Kripke structure (S, I, R, L) where:

- S is the set of states.
- $I \in S$ is the set of initial states.
- $R \in S \times S$ is the transition relation.
- $L : S \mapsto 2^{Atoms(P)}$ is the state labeling function.

$Atoms(P)$ being the set of atomic propositions of the the program P.

Existential abstraction definition:

An abstraction is a surjection $h : D \mapsto \widehat{D}$ that induces an equivalence relation:

$$d \equiv e \text{ iff } h(d) = h(e)$$

The corresponding *abstract Kripke tructure* $(\widehat{S}, \widehat{I}, \widehat{R}, \widehat{L})$ is:

- $\widehat{S} = h(S)$.
- $\widehat{I}(\widehat{d})$ iff $\exists d \in I$ such that $h(d) = \widehat{d}$
- $\widehat{R}(\widehat{d_1}, \widehat{d_2})$ iff $\exists d_1, d_2 \in S$ such that $(h(d_1) = \widehat{d_1} \wedge h(d_2) = \widehat{d_2} \wedge R(d_1, d_2))$
- $\widehat{L}(\widehat{d}) = \bigcup_{h(d)=\widehat{d}} L(d)$

This is called the *Existential Abstraction*.

Abstraction Validity:

An atomic formula f *respects* an abstraction h iff:

$$\forall d_1, d_2 \in D \text{ we have: } d_1 \equiv d_2 \Rightarrow (d_1 \models f \Leftrightarrow d_2 \models f)$$

For an abstract state $\widehat{d}$ we say that $\widehat{L}(\widehat{d})$ is consistent iff:

$$\forall d \in D \text{ such that } h(d) = \widehat{d} \text{ we have: } d \models \bigwedge_{f \in \widehat{L}(\widehat{d})} f$$

Abstraction Validity:

$ACTL^*$ is the fragment of CTL^* that:

- Contains only A as path operator (no E).
- The only negations it contains concerns the atomic propositions.

Exmaple: A F p.

Theorem

Let h be an abstraction and φ be an $ACTL^*$ specification where the atomic subformulas respect h. then the following holds:

- $\widehat{L}(\widehat{d})$ *is consistent for all abstract state* $\widehat{d}$ *in* $\widehat{M}$.
- $\widehat{M} \models \varphi \Rightarrow M \models \varphi$

CEGAR general scheme

- Model extraction: Initial abstraction.
- Model-Check the Abstract Model.
- If no bug is found:
  - The concrete Model is safe.
- If a counterexample is found: Is it a concrete one?
  - Yes. "Happy" end.
  - No. Refine the abstraction and model-check again.

Example:

We will model a program P using transition blocks associated to its variables.

```
init(x) := 0;
next(x) := case
  reset=TRUE : 0;
  x<y        : x+1;
  x=y        : 0;
  else       : x;
esac;
```

```
init(y) := 1;
next(y) := case
 reset=TRUE            : 0;
 (x=y) && !(y=2)       : x+1;
 x=y                   : 0;
 else                  : y;
esac;
```

Model Extraction:

We construct the initial abstraction such that:

$$\forall d_1, d_2 \text{ such that } d_1 \equiv d_2 \text{ we have } \bigwedge_{\forall f \in Atoms(P)} d_1 \models f \Leftrightarrow d_2 \models f$$

How to:

- Define formula clusters FC_i of formulas dealing with the same variable set.
- Define the corresponding variable clusters VC_i.
- For each variable cluster, define an abstraction whose abstract states are consistent with the cluster formulas.
- Cross-product all the cluster abstractions to obtain The abstraction.

Example:

We will model a program P using transition blocks associated to its variables.

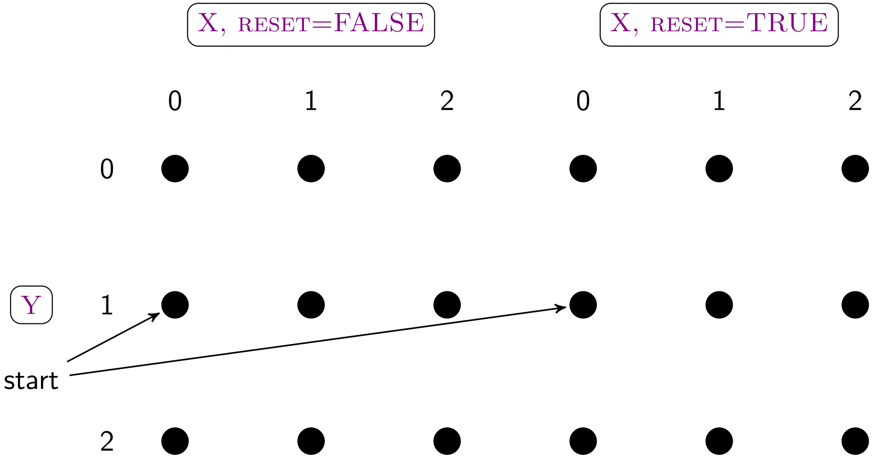
```
init(x) := 0;
next(x) := case
  reset=TRUE : 0;
  x<y        : x+1;
  x=y        : 0;
  else       : x;
esac;
```

```
init(y) := 1;
next(y) := case
 reset=TRUE           : 0;
 (x=y) && !(y=2)      : x+1;
 x=y                  : 0;
 else                 : y;
esac;
```

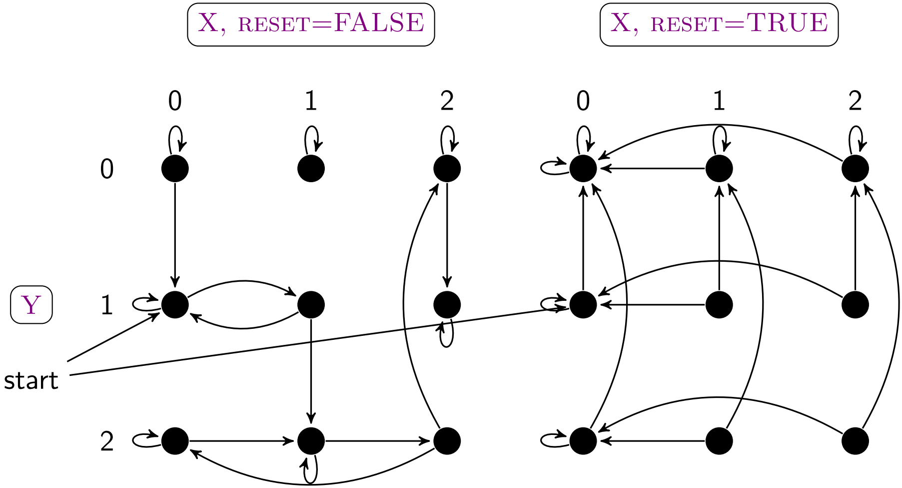
Model Extraction:

- Atoms(P) = {(reset=TRUE), (x=y), (x<y), (y=2)} = $FC_1 \cup FC_2$.
- VC_1 = {reset}, VC_2={x,y}.
- FC_1 equivalence classes: $\begin{array}{ll} \{0\} & \{\} \\ \{1\} & \{reset\} \end{array}$
- FC_2 equivalence classes: $\begin{array}{ll} \{(0,0),(1,1)\} & \{(x=y)\} \\ \{(0,1)\} & \{(x<y),\} \\ \{(0,2),(1,2)\} & \{(x<y),(y=2)\} \\ \{(1,0),(2,0),(2,1)\} & \{\} \\ \{(2,2)\} & \{(x=y),(y=2)\} \end{array}$

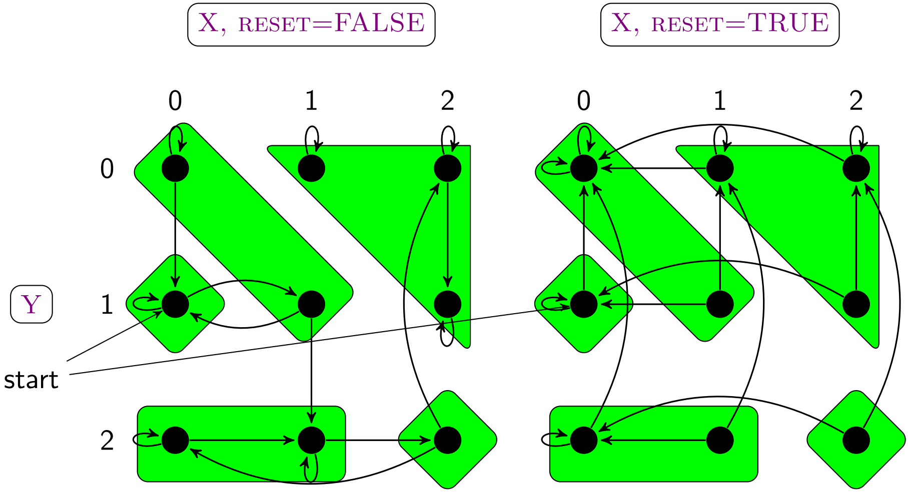
Model Extraction: Concrete State Space



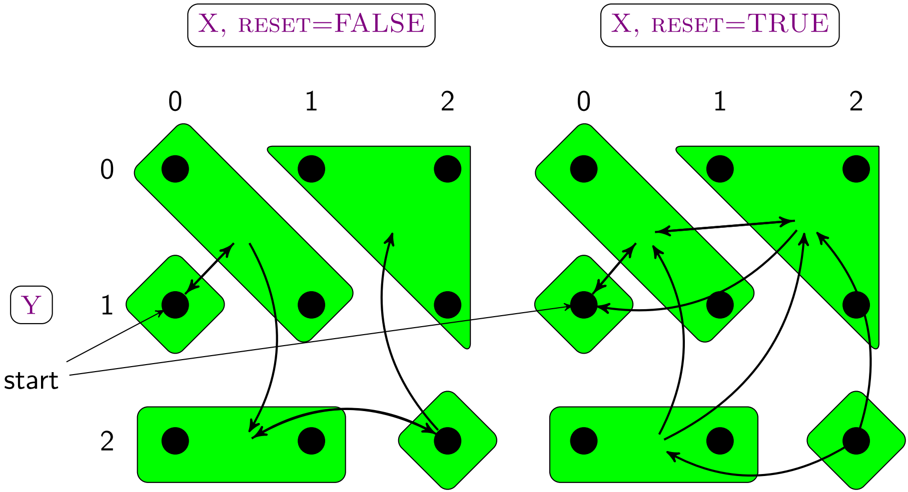
Model Extraction: Concrete Transition Relation



Model Extraction: Abstract State Space



Model Extraction: Abstract Transition Relation



CEGAR general scheme

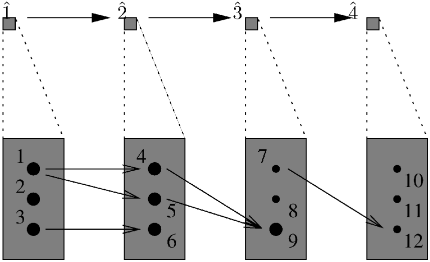
- Model extraction: Initial abstraction.
- Model-Check the Abstract Model.
- If no bug is found:
  - The concrete Model is safe.
- If a counterexample is found: Is it a concrete one?
  - Yes. "Happy" end.
  - No. Refine the abstraction and model-check again.

Checking finite counterexample:

Counterexample $\widehat{T} = \langle \widehat{s_1}, \ldots, \widehat{s_n} \rangle$.

Concrete traces are given by:

$$\{ \langle s_1, \ldots, s_n \rangle \mid \bigwedge_{i=1}^{n} h(s_i) = \widehat{s_i} \wedge I(s_1) \wedge \bigwedge_{i=1}^{n-1} R(s_i, s_{i+1}) \}$$



Checking finite counterexample:

SplitPATH: Symbolic algorithm to compute concrete paths:

$S := h^{-1}(\widehat{s_1}) \cap I$

$j := 1$

while ($S \neq \emptyset$ and $j < n$ **)** {

  $j := j + 1;$

  $S_{prev} := S$

  $S := Img(S, R) \cap h^{-1}(\widehat{s_j})$

}

if $S \neq \emptyset$ **then** output counterexample // −> Happy end.

else output j,S_{prev} // −> Move to the refinement step.

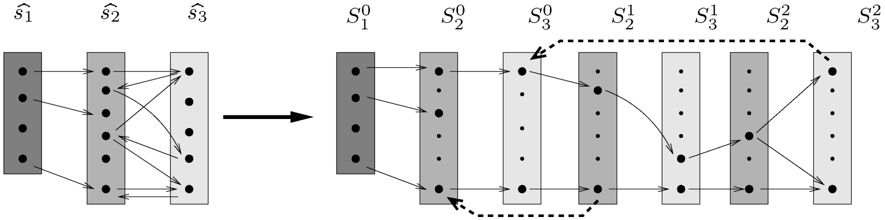
Checking infinite counterexample:

Counterexample $\widehat{T} = \langle \widehat{s_1}, \ldots, \widehat{s_i} \rangle \langle \widehat{s_{i+1}}, \ldots, \widehat{s_n} \rangle^\omega$.

Example:

$\langle \widehat{s_1} \rangle \langle \widehat{s_2}, \widehat{s_3} \rangle^\omega$

Unwinding:



For one abstract loop we get:

- Many concrete loops with different sizes.
- Different start points.

Checking infinite counterexample:

Also, the unwinding become eventually periodic.

Question: How many unwindings are necessary to check the abstract loop?

Theorem

The following are equivalent:

- $\widehat{T}$ *corresponds to a concrete counterexample.*
- $h_{path}^{-1}(\widehat{T}_{unwind})$ *is not empty.*

Where:

- $\widehat{T} = \langle \widehat{s_1}, \ldots, \widehat{s_i} \rangle \langle \widehat{s_{i+1}, \ldots, \widehat{s_n}} \rangle^{\omega}$
- $\widehat{T}_{unwind} = \langle \widehat{s_1}, \ldots, \widehat{s_i} \rangle \langle \widehat{s_{i+1}, \ldots, \widehat{s_n}} \rangle^{min}$
- $min = min_{i+1 \leq j \leq n} |h^{-1}(\widehat{s_j})|$

We can use SplitPATH to check $\widehat{T}_{unwind}$

CEGAR general scheme

- Model extraction: Initial abstraction.
- Model-Check the Abstract Model.
- If no bug is found:
  - The concrete Model is safe.
- If a counterexample is found: Is it a concrete one?
  - Yes. "Happy" end.
  - No. Refine the abstraction and model-check again.

Abstraction Refinement:

We consider here only finite path counterexample.

Let's recall SplitPATH algorithm:

$S := h^{-1}(\widehat{s_1}) \cap I$

$j := 1$

while ($S \neq \emptyset$ and $j < n$ **)** {

  $j := j + 1;$

  $S_{prev} := S$

  $S := Img(S, R) \cap h^{-1}(\widehat{s_j})$

}

if $S \neq \emptyset$ **then** output counterexample // −> Happy end.

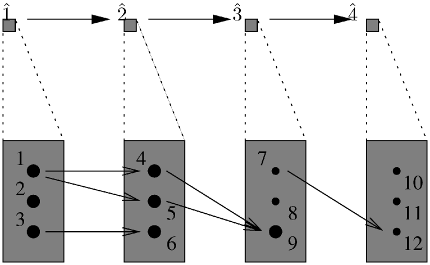
else output j,S_{prev} // −> Move to the refinement step.

Abstraction Refinement:

There exist i, such that $S_i \subset h^{-1}(\widehat{s_i})$, $Img(S_i, R) \cap h^{-1}(\widehat{s_{i+1}}) = \emptyset$ andS_i reachable from $h^{-1}(\widehat{s_1}) \cap I$

We partition $h^{-1}(\widehat{s_i})$ into three subsets:

- $S_{i,0} = S_i$
- $S_{i,1} = \{s \in h^{-1}(\widehat{s_i}) | \exists s' \in h^{-1}(\widehat{s_{i+1}}).R(s, s')\}$
- $S_{i,x} = h^{-1}(\widehat{s_i}) \backslash (S_{i,0} \cup S_{i,1})$



Abstraction Refinement:

Abstraction defined by $h^{-1}(\widehat{s}) = E_1 \times \dots E_m$, m being the number of variable clusters.

We need to separate $S_{i,o}$ and $S_{i,1}$ by refining our abstraction, i.e. refining the equivalence classes $\equiv_j, 1 \leq j \leq m$.

Objective: Maintain the smallest possible abstraction.

Theorem

The problem of finding the coarsest refinement is NP-hard.
When $S_{i,x} = \emptyset$, the problem can be solved in polynomial time.

Abstraction Refinement:

Abstraction refining algorithm:

```
for j := 1 to m {
    ≡'_j := ≡_j
    for every a, b ∈ E_j {
        if proj(S_{i,0}, j, a) ≠ proj(S_{i,0}, j, b)
         then ≡'_j := ≡'_j \{(a, b)}
    }
}
```

Where $proj(S_{i,0}, j, a) \neq proj(S_{i,0}, j, b)$ means that:

$$\exists(d_1, \ldots, d_j, d_{j+1}, \ldots, d_m) \text{ such that: }$$
$$(d_1, \ldots, d_j, a, d_{j+1}, \ldots, d_m) \in S_{i,0}$$
$$(d_1, \ldots, d_j, b, d_{j+1}, \ldots, d_m) \notin S_{i,0}$$

Abstraction Refinement:

Abstraction refining algorithm:

for $j := 1$ **to** m $\{$

$\equiv'_j := \equiv_j$

for every $a, b \in E_j$ $\{$

if $proj(S_{i,0}, j, a) \neq proj(S_{i,0}, j, b)$

then $\equiv'_j := \equiv'_j \setminus \{(a, b)\}$

$\}$

$\}$

Lemma

When $S_{i,x} = \emptyset$ the relation $\equiv'_j$ computed by PolyRefine is an equivalence relation which refines $\equiv_j$ and separates $S_{i.0}$ and $S_{i,1}$. Furthermore, the equivalence relation $\equiv'_j$ is the coarsest refinement of $\equiv_j$

Abstraction Refinement:

Theorem

Given a model M and an ACTL* specification φ whose counterexample is either path or loop, CEGAR will find a model $\widehat{M}$ such that
$\widehat{M} \models \varphi \Leftrightarrow M \models \varphi$

Extensions and tools:

CEGAR has been implemented in many tools such as Blast, Moped ..

It has been also enriched with:

- Use of SAT Solvers instead of OBDD.
- Use of Inerpolants in order to refine the abstraction.

Has been also applied for infinite state systems . . .

Thanks for your attention.