Abstraction & Symbolic Model Checking without BDDs

Sanjit A. Seshia
EECS, UC Berkeley

Acknowledgments: Kenneth McMillan

Key Optimizations in (Symbolic) Model Checking

• Abstraction
  – Compute a smaller state graph by “merging states” s.t. if the property holds on the smaller system model, it holds on the larger one

• Symmetry Reduction
  – Group states into equivalence classes by exploiting symmetries in the model

• Compositional Reasoning
  – Compose proofs of correctness of modules to prove the overall system correct
Today’s Lecture

• Abstraction
  – Counter-example guided abstraction refinement (CEGAR)

• Symbolic Model Checking without BDDs
  – Uses SAT instead of BDDs
  – Started with Bounded Model Checking
  – Extended to Unbounded Model Checking
    • Abstraction + BMC
    • Interpolation-based model checking

Abstraction
Abstraction

• Extracting information from a system description that is relevant to proving a property
• Goal: Reduce size of system model

• Terminology:
  – Original model = Concrete system/model

Abstraction (2)

• Reduce the size of the system model by throwing out information / grouping states
  – If this information is irrelevant to the property of interest (i.e., the property is true on the original model iff it is true on the abstract model) then it is a precise abstraction
  – If the property is true on the original model if it is true on the abstract model, it is a safe abstraction
Example

• Abstractions exhibit more behaviors
• Consider the following two properties on the original model and abstraction:
  \[
  G(go \rightarrow X \text{ stop}) \quad \quad G F \text{ go}
  \]

A Simple Form of Abstraction

• Suppose the temporal logic property mentions only a subset of variable \( V' \) of the entire set \( V \)
• Can I use this information to construct a precise abstraction of the original model?
A Simple Form of Abstraction

• Suppose the temporal logic property mentions only a subset of variable \( V' \) of the entire set \( V \)

• Can I use this information to construct a precise abstraction of the original model?
  – YES. One such method is the “cone of influence” reduction.
    • Transitively propagate syntactic dependences on variables and “delete” all variables not in the transitive closure

Formal Definition

• Abstraction is defined by an abstraction function

• Abstraction function \( \alpha : S \to \hat{S} \)
  – \( S \) – set of concrete states
  – \( \hat{S} \) – set of abstract states

• An abstraction induces an equivalence relation over the concrete states
  – Two concrete states are equivalent if they are mapped to the same abstract state
Formal Definition

• Suppose concrete system is \((S, S_0, R, L)\), and abstract system \((\hat{S}, \hat{S}_0, \hat{R}, \hat{L})\)
• Abstraction function \(\alpha : S \rightarrow \hat{S}\)
  – \(S\) – set of concrete states
  – \(\hat{S}\) – set of abstract states
• \(\hat{S}_0 = \{ t \mid \exists s . S_0(s) \land \alpha(s) = t \}\)
• \(\hat{R} = ?\)
  – How do we algorithmically construct \(\hat{S}_0\) and \(\hat{R}\)?
  – How are labels assigned to abstract states?

Example of Abstraction

• Our examples in this lecture will be abstractions that extract a subset of state variables
  – State variables partitioned into: visible and invisible
  – An abstract state is an evaluation of visible variables
  – What is \(\alpha\)?
  – Two concrete states that agree on values of visible variables are grouped together
Example

• Abstractions exhibit more behaviors

Abstraction and Properties

• If an LTL property is true on the abstract model, is it necessarily true on the concrete model?

• If an LTL property is false on the abstract model, is it necessarily false on the concrete model?
## Cone-of-influence

- Suppose the property $\phi$ mentions a subset of variables $V'$ of the total set $V$
  - Track variables that $V'$ syntactically depend on, add them to $V'$, and iterate until no new variable dependencies generated
  - Resulting $V'$ is the cone-of-influence and its elements are the visible variables
- Problem: Final $V'$ might be as big as $V$ because it only tracks syntactic dependencies
  - But resulting abstraction is precise $\Rightarrow$ if $\phi$ is false in abstract model it is false in concrete model

---

### Example: Cone-of-influence can be conservative

![Diagram]

Let $a$, $b$, $c$, $g$ be state variables

What are the expressions for next state variables $c'$ and $g'$?

- Suppose we want to prove $G(c \Rightarrow Xc)$. What’s the cone of influence?
- If we make $g$ invisible, can we still prove the property?
  - what about $a$ and $b$?
Another approach to Abstraction

- Start with an arbitrary subset of variables as visible
  - An option: the ones mentioned in the property
- Construct abstract model, model check it
  - If property passes, we’re done
  - If we get a counterexample, check whether it is a counterexample for the concrete model
    - If yes, we’re done
    - If not (spurious counterex.) we must make more variables visible and repeat (REFINEMENT)

Counter-Example Guided Abstraction-Refinement (CEGAR)

[R. Kurshan, E. Clarke et al.]

- Start with a choice of $\alpha$
- Construct abstract model, model check it
  - If property passes, we’re done
  - If we get a counterexample, check whether it’s is a counterexample for the concrete model (How do we do this?)
    - If yes, we’re done
    - If not (spurious counterex.), we must refine $\alpha$ and repeat
Intuition about Refinement

- Remember that $\alpha$ partitions the concrete states into equivalence classes
  - $C_1, C_2, \ldots, C_k$
- A refinement $\alpha'$ can further break up the $C_i$'s
  - States that are equivalent under $\alpha'$ should also be equivalent under $\alpha$

Formal Definition of Refinement

- $\alpha'$ refines $\alpha$ if
  - $\forall s, t . \alpha'(s) = \alpha'(t) \Rightarrow \alpha(s) = \alpha(t)$
  - $\exists s, t . \alpha'(s) \neq \alpha'(t) \wedge \alpha(s) = \alpha(t)$

- Given above definition, why will the CEGAR iteration terminate?
Visible/Invisible Abstraction

• The set of variables is partitioned into visible $V$ and invisible $I$

• Questions:
  – How do we construct the abstract model?
    • Given an arbitrary set of visible variables
  – How do we refine the abstraction?
    • i.e., how do we pick new variables to make visible?
    • We want to pick those that will remove the current spurious counterexample

Constructing Abstract Model

• Simply make all invisible variables take arbitrary values
  – Non-deterministically assigned 0 or 1 on each step
• How does this make model checking more efficient?
Constructing Abstract Model

• Simply make all invisible variables take arbitrary values
  – Non-deterministically assigned 0 or 1 on each step
• How does this make model checking more efficient?
  – Avoids some existential quantification, simplifies transition relation

Refining the Abstraction

• The CEGAR approach is most often used today in conjunction with a technique called Bounded Model Checking
• We will study abstraction-refinement in that context
Bounded Model Checking (BMC)

- **Given**
  - A FSM M described by $S_0, R$
  - A property $G p$ and a integer $k \geq 1$
- **Determine**
  - Does M generate a counterexample to $G p$ of length $k$ transitions or fewer?

This problem can be translated to a SAT problem. How?

Unfolding in BMC

- Unfold the model $k$ times:
  \[ U_k = R_0 \land R_1 \land \ldots \land R_{k-1} \]
- Use SAT solver to check satisfiability of $S_0 \land U_k \land E_k$
- A satisfying assignment is a counterexample of $k$ steps
Old view on BMC

• Originally introduced as a debugging tool
  – By finding counterexamples
• Proving properties:
  – Only possible if a bound on the diameter of the state graph is known
    • The diameter is the maximum over shortest path lengths between any two states.
  – Worst case is exponential in system description.

New perspectives: BMC + CEGAR

• BMC + Abstraction can prove properties too!
• Here’s how it works:
  Why does this terminate?

  Create abstraction A
  Perform (unbounded) model checking on A

  Property true
  OK

  Counterexample of length k

  Extract information for refinement from refutation
  Proof fails

  Prove that this abstract counterexample of length k is a concrete counterex.
  using k-step BMC on M
  Proof succeeds

  Counterexample
Steps

1. Create abstraction A
2. Model check A
3. Prove that abstract counterexample is a concrete counterexample using BMC
4. Use refutation of abstract counterexample to do refinement

Checking Abstract Counterex.

• Recall: BMC for length k
  – Use SAT solver to check satisfiability of $S_0 \land U_k \land E_k$
• How do we use this to prove the abstract counterexample of length k also holds for concrete model?
Checking Abstract Counterex.

• Recall: we use BMC for the length k of the abstract counterexample
  – Use SAT solver to check satisfiability of
    \( S_0 \land U_k \land E_k \)
    under the partial assignment corresponding to values of the visible variables
  – If SAT solver reports “SAT” we have a concrete counterexample
    • What is a satisfying assignment?
  – If not, we have a refutation \( \Leftarrow \) proof of unsatisfiability

Refinement

• Given proof of unsatisfiability of
  \( S_0 \land U_k \land E_k \)
  under the partial assignment corresponding to values of the visible variables
• Look at unsatisfiable core of proof
  – Invisible variables that appear in the core indicate why the abstract counterexample is spurious
  – Make those variables visible
Modifying the Abstraction-Refinement Loop

- **Insight:** Why pick an abstraction to start with?
  - Initial abstraction may not be the best start point
  - Why not do BMC initially and use its results to generate abstractions?

---

**Proof-based Abstraction (PBA)**

- **BMC on M at depth k**
  - Cex?
  - No Cex?
  - Use refutation to choose abstraction
  - Increase k to k'
  - MC on abstraction
  - Property true?
    - OK
    - False, counterexample of length k'?

[McMillan, Amla, 2003]
Termination of PBA

- Depth $k$ increases at each iteration
- Eventually $k > \text{diameter } d$
- If $k > d$, no counterexample is possible

CEGAR vs. PBA

- Refutation via $k$-step BMC
  - PBA refutes all concrete counterexamples of up to length $k$
  - CEGAR refutes only the abstract counterexample of length $k$
- So PBA does more work in the refutation, but usually results in fewer iterations of the loop
Abstract/Concrete Error Trace

Abstract trace OK

Abstract trace spurious

Abstraction and Reachability

• An abstraction expands the set of states reachable from the initial state
  – OVER-APPROXIMATION
• Instead of starting by abstracting states, one can directly abstract the transition relation
  – Each time you compute the set of next states, you get an over-approximation of the actual set of next states
  – Gives a way of computing an over-approximation of the set of reachable states
Abstraction using Interpolation

- Abstraction is extracting sufficient/relevant information from a system to prove a given property.
- This notion is in some sense closely related to a notion of “interpolant” and a lemma called “Craig's interpolation lemma”

Interpolation Lemma \(^{(Craig, 57)}\)

- If $A \land B = \text{false}$, there exists an interpolant $A'$ for $(A,B)$ such that:
  
  $A \implies A'$
  $A' \land B = \text{false}$
  $A'$ refers only to common variables of $A,B$

- Example:
  $A = p \land q$, $B = \neg q \land r$, $A' = q$
Interpolants from Proofs

(Pudlak, Krajicek, 97)

• Interpolant A’ for A \land B:
  \[ A \Rightarrow A' \]
  \[ A' \land B = \text{false} \]
  A’ refers only to common variables of A, B

• Interpolants can be obtained from proofs
  – given a resolution-based refutation (proof of unsatisfiability) of A \land B,
  \[ A' \text{ can be derived in time linear in the proof} \]

Interpolation based Model Checking

(McMillan, 2003)

• Main Idea: Pose the problem of over-approximating the set of next states as finding an interpolant

\[
S_0(v_0) \land R(v_0, v_1) \land R(v_1, v_2) \land \ldots \land R(v_{k-1}, v_k) \land E_k(v_k)
\]
Interpolation based Model Checking

For a fixed $k$:

1. Set $Z$ initially to $S_0$
2. Do BMC starting from $Z$ for $k$ steps
   - If SAT: have we found a counterexample?
   - If UNSAT, continue
3. Use interpolation to compute over-approximation of next states of $Z$ and add them back into $Z$
   - Can newly added states lead to error states in $k-1$ steps? In $k$ steps?
4. If $Z$ does not increase
   - We’ve reached a fixed point. Is the property true?
5. Otherwise, back to step 2

What set of states does $A'$ represent?

$S_0(v_0) \land R(v_0, v_1) \land R(v_1, v_2) \land \ldots \land R(v_{k-1}, v_k) \land E_k(v_k)$

$A = S_0(v_0) \land R(v_0, v_1)$

$B = R(v_1, v_2) \land \ldots \land R(v_{k-1}, v_k) \land E_k(v_k)$

$A'$ is a function of $v_1$ s.t.

1. $A \Rightarrow A'$
2. $A' \land B$ is unsat
Intuition

- $A'$ tells us everything the prover deduced about the image of $S_0$ in proving it can't reach an error in $k$ steps.
- Hence, $A'$ is in some sense an abstraction of the image relative to the property and the bound $k$

Refinement

- Model checking may fail for a fixed $k$
  - May add a state that reaches error in $k$ steps (getting SAT in step 2 with $Z \neq S_0$)
- Refinement is just increasing $k$
  - How big can $k$ get?