

SAT-Based Model Checking Without Unrolling

Author: Aaron R. Bradley
Reporter: Xie Li

April 6, 2022

IC3/PDR: Related Papers

- Checking Safety by Inductive Generalization of Counterexamples to Induction
- SAT-Based Model Checking Without Unrolling
- Understand IC3
- Efficient implementation of property directed reachability

Overview

- Preliminaries
- Introduction to a Naive Model Checker FSIS:
 - The Idea
 - Algorithm: LIC and MIC
 - Observations.
- Introduction to IC3:
 - The Idea.
 - Description of the algorithm.

Preliminaries: Propositional Logic

- **Literal:** A literal l is a propositional variable or its negation: $x, \neg y$.
- **Clause:** A clause c is a disjunction of literals. We use $|c|$ to denote the size of a clause, i.e. the number of literals in the clause.
- **Subclause:** A subclause d of a clause c is a disjunction of a subset of literals of c .
abbr. $d \sqsubseteq c$
 $c : x \vee y \vee \neg z, d : x \vee \neg z$.

Preliminaries: Transition System

Definition (Boolean Transition System)

A boolean transition system $\mathcal{S} = \langle \bar{x}, \theta, \rho \rangle$ has three component where:

- $\bar{x} = \{x_1, x_2, \dots, x_n\}$ is the set of propositional variables that are assigned true.
- $\theta(\bar{x})$ is a propositional formula stating the initial condition.
- $\rho(\bar{x}, \bar{x}')$ is a propositional formula describing the transition relation.

The semantic of a transition system is given by its computations:

Definition (State and Computation)

- A *state* s of a boolean transition system S is an assignment of the variables \bar{x} .
- A *computation* $\sigma : s_0, s_1, s_2, \dots$ is a sequence of states satisfying initial condition and transition relations:
$$\theta(s_0) \wedge \forall i \geq 0. \rho(s_i, s_{i+1}) \equiv T.$$

Preliminaries: Subclause Lattice

Consider an clause c and its induced subclause lattice $L_c = \langle 2^c, \sqcap, \sqcup, \sqsubseteq \rangle$ where

- Elements of 2^c are subclauses of c ,
- Elements are ordered by the subclause relation \sqsubseteq ,
- Join operator \sqcup is just disjunction, and
- Meet operation \sqcap is the disjunction of common literals.

By Tarski theorem, every monotone function on L_c has a least fixpoint and greatest fixpoint.

Preliminaries: Inductive Invariant

Definition (Inductive Invariant)

A formula φ is an inductive invariant on \mathcal{S} if

- it holds initially: $\theta \Rightarrow \varphi$
- and it is preserved under transition: $\varphi \wedge \rho \Rightarrow \varphi'$

A formula φ is **inductive related to** an inductive formula ψ if

- it holds initially: $\theta \Rightarrow \varphi$
- and $\psi \wedge \varphi \wedge \rho \Rightarrow \varphi'$

FSIS: Finite-State Inductive Strengthening

Problem:

Given a transition system \mathcal{S} and specification formula Π , is Π an invariant on \mathcal{S} ?

FAQ:

- Why does this problem matter?
- Where does “strengthening” comes from?
 Π usually is an invariant not inductive, we wish to find a *strengthening assertion* χ such that $\Pi \wedge \chi$ is inductive.

Basic Idea: generating many clauses, each of which is inductive related to previous-generated clause. Later use them to construct $\Pi \wedge \chi$

Introduction to FSIS

Introduction to FSIS

Introduction to FSIS

Introduction to FSIS

LIC and MIC: Compute Minimal Inductive Subclause

$\text{down}(L_c, d)$:

Given a subclause lattice L_c and the clause d , return the unique largest subclause $e \sqsubseteq d$ such that the implication $\psi \wedge e \wedge \rho \Rightarrow d'$ holds: use counterexample to shrink the space.

$\text{LIC}(L_c, c)$: applies down several times.

- If the implication $\psi \wedge c \wedge \rho \Rightarrow c'$ and $\theta \Rightarrow c$ holds, then return c .
- If it does not hold, $\psi \wedge c \wedge \rho \wedge \neg c'$ is satisfied by some assignment (s, s') .
- Let $\neg t$ be the best over-approximation of $\neg s$ in L_c and compute a new clause $d = c \sqcap \neg t$.
- Recurse on d .

LIC and MIC: Compute Minimal Inductive Subclause

Theorem (Large Inductive Subclause)

The fixpoint of the iteration sequence computed by $\text{LIC}(L_c, c)$ is the largest subclause of c that satisfies consecution. If it also satisfies initiation, then it is the largest inductive subclause of c . Finding it require at most $O(|c|)$ SAT queries.

Let the computed sequence be $c_0 = c, c_1, \dots, c_k$.

Suppose $e \sqsubseteq c$ also satisfies consecution but is not a subclause of c_k :

Let i be the position that

$$e \sqsubseteq c_i \wedge e \not\sqsubseteq c_{i+1}$$

Partition c_i into $e \vee f$ where f contains only literals from c_i .

The consecution is not satisfied: $\psi \wedge (e \vee f) \wedge \rho \wedge \neg(e' \vee f')$ is satisfied by (s, s')

- $\psi \wedge e \wedge \rho \wedge \neg e' \wedge \neg f'$
- $\psi \wedge \neg e \wedge f \wedge \rho \wedge \neg e' \wedge f'$

$\neg e$ is true under the assignment s , hence $e \sqsubseteq \neg s$. Contradiction.

LIC and MIC: Compute Minimal Inductive Subclause

$\text{MIC}(\mathcal{S}, \psi, c)$: returns a minimal subclause of c that is inductive related to ψ .

```
let rec min  $p$   $S_0$  = function
| []      →  $S_0$ 
|  $h :: t$  → if  $p(S_0 \cup t)$ 
             then min  $p$   $S_0$   $t$ 
             else min  $p$  ( $h :: S_0$ )  $t$ 
let minimal  $p$   $S$  = min  $p$  []  $S$ 
```

Fig. 1. Linear-time minimal

Theorem (Correct)

The algorithm terminates and returns the minimal \bar{S} s.t. $p(\bar{S})$ is true.

Obsevation of FSIS

- The algorithm can be implemented to find inductive clauses in parallel.
- The algorithm enters long searches for the next relatively inductive clause
- An unreachable state may not have a inductive generalization.
- Not making good use of stepwise information.

IC3: Incremental Construction of Inductive Clauses for Indubitable Correctness.

Data structure: A sequence of formulas

$$F_0, F_1, F_2, \dots, F_k$$

- $\theta \Rightarrow F_0$
- $F_i \Rightarrow F_{i+1}$ for $0 \leq i < k$
- $F_i \Rightarrow \Pi$ for $0 \leq i \leq k$
- $F_i \wedge \rho \Rightarrow F'_{i+1}$ for $0 \leq i < k$

We use $\text{clause}(F_i)$ to denote the set of clauses that comprises F_i :

$$F_i = \Pi \wedge \bigwedge \text{clause}(F_i).$$

Introduction to IC3

Introduction to IC3

Introduction to IC3

Introduction to IC3

Description of the Algorithm

- First check whether $\theta \wedge \neg \Pi$ or $\theta \wedge \rho \wedge \Pi'$ is satisfiable to detect counterexample.
- Major iteration:
 - If $F_k \wedge \rho \Rightarrow \Pi'$, then enter major iteration $k + 1$ and let $F_{k+1} = \Pi$.
 - For any clause $c \in F_i$, $0 \leq i \leq k$, if $F_i \wedge \rho \Rightarrow c'$ and $c \notin \text{clauses}(F_{i+1})$, then c is conjoined to F_{i+1} .
 - If ever $F_i = F_{i+1}$, the proof is complete and Π is an invariant.
- Minor iteration: Suppose $F_k \wedge \rho \not\Rightarrow \Pi'$:
 - Let the counterexample be s and find the greatest F_i s.t. $\neg s$ is inductive related to F_i . Then a strengthening $c \sqsubseteq \neg s$ will be conjoined to F_0, \dots, F_{i+1} .
 - If $i = k$ or $i = k - 1$, then after conjoining $F_k \wedge \rho \Rightarrow \Pi'$.
 - Otherwise there is a state $t \in F_{i+1}$ but $t \notin F_i$.