Motivation

Contents

Overview

1. Introduction

Overview

2.1 Preliminaries

- A Boolean Transition System $S = (X, I, T)$ consists of
  - Set of boolean variables X
  - A conjunction representing the initial state I
  - A propositional formula over variables in X and $X' = \{x \in X \mid x' \in X'\}$, called Transition Relation

- States in S are cubes containing each variable from X with a boolean valuation of it

  ➔ Finite number of states: $2^{|X|}$

2.1 Preliminaries

- Given a formula φ over X, we get a primed formula φ' by replacing each variable with its corresponding variable in X'

- A literal is a variable or ist negation
- A cube is a conjunction of literals
- A clause is a disjunction of literals
  - ➔ Negation of a cube is a clause and vice versa

- A Safety Property P is a formula over X that should be satisfiable by every state reachable from I
  - ➔ $\bar{P}$ being a set of bad states

2.2 Algorithm

- PDR on hardware checks if states in $\bar{P}$ are reachable from I

- For that it uses cubes of clauses, called Frames
  - Frame F_i represents an over-approximation of reachable states in at most i transitions from I

- PDR maintains sequence of frames $[F_0, F_1, \ldots, F_k]$, called trace

- Algorithm repeats three phases until termination
  - Next Transition
  - Blocking-Phase
  - Propagation-Phase

2.2 Algorithm

```
1:  procedure PDR-PROVE(I, T, P)
2:      check for 0-counter-example
3:      trace.push(new frame(1))

4:      loop
5:          while ∃ cube c, s.t. trace.last() ∧ T ∧ c' is SAT and c ⇒ P̄ do
6:              recursively block proof-obligation(c, trace.size() - 1)
7:              and strengthen the frames of the trace.
8:              if a proof-obligation(p, 0) is generated then
9:                  return false

10:         F_{k+1} = new frame(P)
11:         for all clause c ∈ trace.last() do
12:             if trace.last() ∧ T ∧ c̄' is UNSAT then
13:                 F_{k+1} = F_{k+1} ∧ c
14:         if trace.last() == F_{k+1} then
15:             return true
16:         trace.push(F_{k+1})
```

2.3 Example

2.4 Possible Improvements

- Blocking one state at a time is ineffective.

  ➔ Generalize blocked states by removing cubes not used in the proof, delivered by Unsat-cores

- Ternary simulation to generalize proof-obligations
  - Extend binary variables with a new unknown value and check variables for importance

Overview

3.1 Preliminaries

3.2 Lifted Algorithm

- Algorithm Pseudocode here

3.3 Example

3.4 Possible Improvements

Overview

4.1 Implementation

4.2 Implemented Improvements

Overview

5.1 Data Comparison

5.2 Discussion

Overview

6. Related Work

Overview

7.1 Further Improvements

Overview

8. Conclusion