Formal Methods in Software Engineering

Propositional Logic — Spring 2025

Konstantin Chukharev

§1 Craig Interpolation in Module checking

Introduction into Craig Interpolation Theorem

Craig interpolation theorem states: If there's such a formulas A and B and $A \vDash B$ then exists such formula C that:

- $A \vDash C$
- $C \vDash B$
- C only consists of non-logical $A \land B$

Real-world examples:

- K-step BMC.
- TODO.
- TODO.

Semantic Proof of theorem

Let L_a be a language of A and L_b a language of B.

$L_{\mathrm{AB}} = L_A \wedge L_B$

▫ Example

Let $A = P(X) \wedge Q(X)$, $B = Q(Y) \vee R(Y)$, then

- $L_A = \{P, Q\}$
- $L_B = \{Q, R\}$
- $L_{\mathrm{AB}} = \{Q\}$

▫ Step 1

$G = \{\varphi \in L_{\mathrm{AB}} \mid A \vDash \varphi\}.$

G - a set of formulas that are true if A is true.

Semantic Proof of theorem

□ Step 2

We neeed to show, that if some model M satisfies all formulas from G, then it implies B automatically.

Let M be a model of G. $(\forall \gamma \in G : M \vDash \gamma)$

We are expanding M up to a M' by adding interpretation of symbols from $L_A \setminus L_{\text{AB}}$ so $M' \vDash A$

It is possible because G already contains all consequences of A in L_{AB}

□ Step 3

If $A \vDash B$, then $M' \vDash B$ because B depends only on symbols from L_B and M' equals with M on $L_{\text{AB}} \in L_B$, then $M \vDash B$

In result, **any model** of G satisfies B, means $G \vDash B$

By Theorem of compactness, if $G \vDash B$, then exists a finite set of $\varphi_1, \varphi_2, ..., \varphi_n$ from G such as that $(\varphi_1 \land \varphi_2 \land ...\varphi_n) \vDash B$

Then our **interpolant** is $C = \varphi_1 \land \varphi_2 \land ...\varphi_n$

Semantic Proof of theorem [2]

□ Checking of C

1. $A \vDash B$ because every formula in G is a consequence of A
2. $C \vDash B$ from proof
3. C only uses symbols from L_{AB}

Example: $A = P(x) \wedge Q(x), B = Q(y) \vee R(y)$

$C -?$

§2 SAT and usage Module checking

Introduction

SAT-solver - checks satisfiability of a certain boolean formula.

Module checking - verification if model of a system is correct.

Module checking and SAT are connected by methods of **Symbolic verification and symbolic model checking**.

Symbolic model checking - an important hardware model checking technique. In symbolic model checking, sets of states and transition relations of circuits are represented as formulas of explicit sates.

Current design blocks with well-defined functionality have many thousands of state elements. To handle such a scale, researchers deployed SAT solvers, starting with the invention of SAT-based BMC.

Bounded model checking (BMC) relies on SAT solvers to exhaustively check hardware designs up to a limited depth.

BMC, Bounded model checking

A SAT solver either finds a satisfying assignment for a propositional formula or proves its absence. Using this terminology, BMC determines whether a transition system has a counterexample of a given length k or proves its absence.

BMC uses a SAT solver to achieve this goal. Given a transition system M, BMC translates the question "Does M have a counterexample of length k?" into a propositional formula and uses a SAT solver to determine if the formula is satisfiable or not.

If the solver finds a satisfying assignment, a counterexample exists and is represented by the assignment. If the SAT solver proves that no satisfying assignment exists, then BMC concludes that no counterexample of length k exists.

BMC, Bounded model checking [2]

Given a transition system $M = \langle I, T \rangle$, BMC is an iterative process for checking P in all initial paths up to a given bound on the length.

-**Initial states (I)** - A formula describing all possible starting configurations of the system.

-**Transition relation (T)** - A formula defining how the system moves from one state to the next.

In order to search for a counterexample of length k, the following propositional formula is built:

- $\mathrm{BMC}(k) = I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{\text{k-1}}, s_k) \wedge (\psi(s_0) \vee \psi(s_1) \vee \ldots \vee \psi(s_k))$
  - $if\ satisfiable$: there's a counterexample
  - $if\ unsatisfiable$:: there's no errors that are reachable in k steps

Into k-step BMC

If BMC is **unsatisfiable**, then it may be spliited into two formulas^

1. $A = I(s_0) \land T(s_0, s_1)$
2. $T(s_{\text{k-1}}, s_k) \land (\psi(s_0) \lor \psi(s_1) \lor \ldots \lor \psi(s_k))$

Since $A \land B$ is **false**, then (by Craig Theorem) there's an interpolant A' such as:

1. $A \to A'$
2. $A' \land B$ is unsatisfiable
3. A' only uses common symbols from A and B (e.g. state variables s_i)

Into k-step BMC [2]

□ Interpolant

Extracted interpolant represents an overapproximation of reachable states from I after one transition. In addition, no counterexample can be reached from I in $k - 1$ transitions or less.

When k is increased, the precision of the computated interpolant is also increased. For a sufficiently large k, the approximation obtained through interpolation becomes precise enough such that algorithm is guaranteed to find an inductive invariant if the system is safe.

ITP, Interpolation-Based Model Checking

ITP is a complete SAT-based model checking algorithm that relies on interpolation to compute the FRS.

FRS - forward reachability sequence, denoted as such $\overline{F}_{[\mathrm{k}]}$ is a sequence $\langle F_0, \ldots, F_k \rangle$ of propositional formulas over V such that the following holds:

- $F_0 = I$
- $F_i \land T \to F'_{\mathrm{i+1}}$ for $0 \leq i < k$

(F' is an F but with all variables $v \in V$ replaced with counterparts $v' \in V'$, or starting and successor states before and after transition)

FRS are in the context of IC3 algorithm.

TODO(IC3) TODO(FBA) TODO(ITP) TODO(FRS)