

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 $A \models B$ then exists such formula C that:

- $A \models C$
- $C \models B$
- C only consists of non-logical $A \wedge B$

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 .

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

□ Example

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

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

□ Step 1

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

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

Semantic Proof of theorem [2]

□ Step 2

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

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

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

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

□ Step 3

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

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

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

Then our **interpolant** is $C = \varphi_1 \wedge \varphi_2 \wedge \dots \varphi_n$

Semantic Proof of theorem [3]

□ Checking of C

1. $A \models B$ because every formula in G is a consequence of A
2. $C \models B$ 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

Definition 1 - Interpolant

- Let A and B be quantifier-free propositional formulas whose conjunction $A \wedge B$ is unsatisfiable. An interpolant is a quantifier-free propositional formula I such that $A \rightarrow I$ is valid, $I \wedge B$ is unsatisfiable, and the variables occurring in I are a subset of the variables common to A and B .

Intuitively, if A represents reachable states and B unsafe or bad states, then the interpolant I safely overapproximates A , a property frequently used in the context of fixed-point detection. (More about it later)

Definition 2 - Interpolant sequence

Let $\langle A_1, A_2, \dots, A_n \rangle$ be an ordered sequence of propositional formulas such that the conjunction $\bigwedge_{i=1}^n A_i$ is unsatisfiable. An interpolation sequence is a sequence of formulas $\langle I_0, I_1, \dots, I_n \rangle$ such that all of the following conditions hold.

1. $I_0 = T$ and $I_n = F$.
2. For every $0 \leq j < n$, $I_j \wedge A_{j+1} \rightarrow I_{j+1}$ is valid.
3. For every $0 < j < n$, it holds that the variables in I_j are a subset of the common variables of $\langle A_1, \dots, A_j \rangle$ and $\langle A_{j+1}, \dots, A_n \rangle$.

Definiton 3 - Finite-state Transition System

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

- \mathbf{V} - a set if boolean variables. V induces a set of states $S \stackrel{\text{def}}{=} \mathbb{B}^{|V|}$, and a state $s \in S$ is an assignment to V and can be represented as a conjunction of literals that are satisfied in s . More generally, a formula over V represents the set of states in which it is satisfiable.

Given a formula F over V , we use F' to denote the corresponding formula in which all variables $v \in V$ have been replaced with their counterparts $v' \in V'$. In the context of multiple steps of the transition system, we use V_i instead of V' to denote the variables in V after i steps. Given a formula F over V^i , the formula $F[V^i \leftarrow V^j]$ is identical to F except that for each variable $v \in V$, each occurrence of v^i in F is replaced with v^j . This substitution allows us to change the execution step to which a formula refers.

Definiton 3 - Finite-state Transition System [2]

- Initial states (I)** - A formula over V describing all possible starting configurations of the system.
- Transition relation (T)** - A formula defining how the system moves from one state to the next.
($T(V, V')$) is a formula over the variables V and their primed counterparts $V' = \{v' | v \in V\}$, representing starting states and successor states of the transition.
- Safe states (P)** - A formula over V describing all possible safe states of the system.

Definition 4 - Forward reachability sequence

- **FRS** - forward reachability sequence, denoted as such $\overline{F}_{[k]}$ is a sequence $\langle F_0, \dots, F_k \rangle$ of propositional formulas over V such that the following holds:
- $F_0 = I$
- $F_i \wedge T \rightarrow F'_{i+1}$ for $0 \leq i < k$
- A reachability sequence $\overline{F}_{[k]}$ is monotonic if $F_i \rightarrow F_{i+1}$ for $0 \leq i < k$ and safe if $F_i \rightarrow P$ for $0 \leq i \leq k$.
- The individual propositional formulas F_i are called elements or frames of the sequence.

An element F_i in an FRS $\overline{F}_{[k]}$ represents an overapproximation of states reachable in i steps of the transition system. If the FRS *is monotonic*, then F_i is an overapproximation of all states reachable in at most i steps.

- **Fixpoint** - \overline{F} is a fixpoint, if there is $0 < i \leq k$ and $F_i \rightarrow \bigvee_{i=0}^{k-1} F_i$

Monotonic FRS arise in the context of IC3 algorithm.

Definition 5 - Inductive invariant, Consecution

A set of states characterized by the formula F is inductive (satisfies consecution, respectively) if $F \wedge T \rightarrow F'$ holds. F is inductive relative to a formula G if $G \wedge F \wedge T \rightarrow F'$ holds. F satisfies initiation if $I \rightarrow F$, i.e., if F comprises all initial states. F is safe with respect to P if $F \rightarrow P$.

We say that F is an inductive invariant if it satisfies initiation and is inductive.

□ Lemma 1

Let $M = \langle V, I, T, P \rangle$ be a transition system and let F be a propositional formula representing a set of states. If F is an inductive invariant that implies P (i.e., $F \rightarrow P$ is valid), then P holds in M and M is said to be safe.

The following lemma highlights the relationship between inductive invariants and FRS.

□ Lemma 2

Let M be a transition system $\langle V, I, T, P \rangle$ and let $\overline{F}_{[k]}$ be an FRS. Let us define $F \stackrel{\text{def}}{=} \bigvee_{j=0}^i F_j$ where $0 \leq i < k$. Then, F is an inductive invariant if $F_{i+1} \rightarrow P$. In addition, if $\overline{F}_{[k]}$ is safe, then $F \rightarrow P$ holds, and thus M is safe.

Model checking algorithms

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 states.

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]

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

□ Easy formula

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

□ Hard one

- $\text{path}^{i,j} \stackrel{\text{def}}{=} T(V^i, V^{i+1}) \wedge \dots \wedge T(V^{j-1}, V^j)$, where $0 \leq i < j$ and $j - i = k$. An initial path of length k is defined using the formula $I(V^0) \wedge \text{path}^{0,k}$.

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

- $\varphi^k \stackrel{\text{def}}{=} I(V^0) \wedge \text{path}^{0,k} \wedge (\neg P(V^k))$

Into k-step BMC

If BMC is **unsatisfiable**, then it may be splitted into two formulas:

1. $A = I(s_0) \wedge T(s_0, s_1)$
2. $T(s_{k-1}, s_k) \wedge (\psi(s_0) \vee \psi(s_1) \vee \dots \vee \psi(s_k))$

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

1. $A \rightarrow A'$
2. $A' \wedge B$ 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 computed 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.

To make it short, ITP replaces accurate calculation of FRS with interpolants' approximation, which makes algorithm faster.

There's a Lemma that states: A FRS of length n exists iff there is no counterexample of length $\leq n$.

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