Interpolating Strong Induction

Hari Govind Vediramana Krishnan, Yakir Vizel, Vijay Ganesh, Arie Gurfinkel

Presented by Marek Chalupa

IA072 – spring 2021, May 21, 2021

The algorithm KAVY

Technique for safety verification of symbolic transition systems.

- Combines:
  - IC3/PDR
  - (bounded model checking with) iterpolation
  - k-induction
  - (AVY $\sim$ PDR-like algorithm with interpolation)
- Strong induction = k-induction

„KAVY uses *k*-induction to guide interpolation and PDR-style inductive generalization"

Symbolic transition systems

Symbolic transition system is a tuple $(\mathbf{x}, I, T)$ consisting of

- state (boolean) variables $\mathbf{x} = \{x_1, x_2, ..., x_n\}$,
- a propositional formula $I(\mathbf{x})$ describing initial states,
- a propositional formula $T(\mathbf{x}, \mathbf{x}')$ describing transition relation

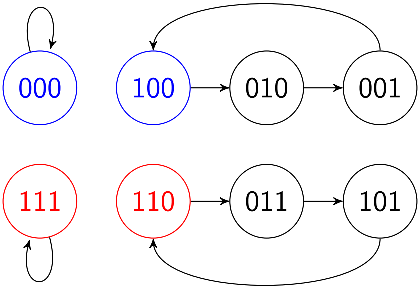
Symbolic transition systems

Symbolic transition system is a tuple $(\mathbf{x}, I, T)$ consisting of

- state (boolean) variables $\mathbf{x} = \{x_1, x_2, ..., x_n\}$,
- a propositional formula $I(\mathbf{x})$ describing initial states,
- a propositional formula $T(\mathbf{x}, \mathbf{x}')$ describing transition relation

Safety verification: given a set of *good* states P (the property), we want to decide whether all reachable states of the system are in P (P-states). If yes, we call the system *safe* (*unsafe* otherwise). $\neg P$-states are called *bad* states.

Symbolic transition system example



$$\mathbf{x} = \{x_1, x_2, x_3\}$$

$$I(\mathbf{x}) = \neg x_2 \wedge \neg x_3$$

$$T(\mathbf{x}, \mathbf{x}') = (x_1 \iff x_2') \wedge (x_2 \iff x_3') \wedge (x_3 \iff x_1')$$

$$\neg P(\mathbf{x}) = x_1 \wedge x_2$$

Inductive sets

Given a system $(\mathbf{x}, I, T)$, a set S of states is inductive invariant if

$$\begin{aligned} I(\mathbf{x}) &\implies S(\mathbf{x}) \\ S(\mathbf{x}) \wedge T(\mathbf{x}, \mathbf{x}') &\implies S(\mathbf{x}') \end{aligned}$$

Inductive sets

Given a system $(\mathbf{x}, I, T)$, a set S of states is <u>inductive invariant</u> if

$$\begin{aligned} I(\mathbf{x}) &\implies S(\mathbf{x}) \\ S(\mathbf{x}) \wedge T(\mathbf{x}, \mathbf{x}') &\implies S(\mathbf{x}') \end{aligned}$$

A set S of states is <u>k-inductive invariant</u> if

$$\begin{aligned} I(\mathbf{x}_0) \wedge T(\mathbf{x}_0, \mathbf{x}_1) \wedge \cdots \wedge T(\mathbf{x}_{k-2}, \mathbf{x}_{k-1}) &\implies \bigwedge_{0 \le i \le k-1} S(\mathbf{x}_i) \\ S(\mathbf{x}_0) \wedge T(\mathbf{x}_0, \mathbf{x}_1) \wedge S(\mathbf{x}_1) \wedge T(\mathbf{x}_1, \mathbf{x}_2) \wedge \cdots \wedge S(\mathbf{x}_{k-1}) \wedge T(\mathbf{x}_{k-1}, \mathbf{x}_k) &\implies S(\mathbf{x}_k) \end{aligned}$$

Bounded model checking (BMC)

Given a parameter k, we check the formula:

$$I(\mathbf{x_0}) \wedge T(\mathbf{x_0}, \mathbf{x_1}) \wedge \ldots T(\mathbf{x_{k-1}}, \mathbf{x_k}) \wedge (\neg P(\mathbf{x_0}) \vee \cdots \vee \neg P(\mathbf{x_k}))$$

Bounded model checking (BMC)

Incremental BMC: for parameter $k = 0, 1, 2, ...$, we check the formula:

$$I(\mathbf{x_0}) \wedge T(\mathbf{x_0}, \mathbf{x_1}) \wedge \ldots T(\mathbf{x_{k-1}}, \mathbf{x_k}) \wedge \neg P(\mathbf{x_k})$$

Bounded model checking with k-induction

For parameter $k = 0, 1, 2, \ldots$, we check the formulas:

base: $I(\mathbf{x_0}) \wedge T(\mathbf{x_0}, \mathbf{x_1}) \wedge \ldots T(\mathbf{x_{k-1}}, \mathbf{x_k}) \wedge \neg P(\mathbf{x_k})$

step: $P(\mathbf{x_0}) \wedge T(\mathbf{x_0}, \mathbf{x_1}) \wedge \ldots T(\mathbf{x_{k-1}}, \mathbf{x_k}) \wedge P(\mathbf{x_k}) \wedge T(\mathbf{x_k}, \mathbf{x_{k+1}}) \wedge \neg P(\mathbf{x_{k+1}})$

Interpolants

Given two formulas A, B such that $A \wedge B$ is unsat, a Craig's interpolant is a formula R, such that:

- $A \implies R$
- $R \wedge B$ is unsat
- R uses only variables common to A and B

BMC with interpolants

$$\overbrace{I(\mathbf{x_0}) \wedge T(\mathbf{x_0}, \mathbf{x_1})}^{A} \;\wedge\; \overbrace{T(\mathbf{x_1}, \mathbf{x_2}) \dots T(\mathbf{x_{k-1}}, \mathbf{x_k}) \wedge \neg P(\mathbf{x_k})}^{\mathrm{B}}$$

- If BMC query is unsat, obtain the interpolant R of A and B
- R is a formula over the variables $\mathbf{x_1}$
- R over-approximates the set of states reachable in one transition
- No bad state is reachable from R in $k-1$ steps

PDR

- (Inductive) trace is a sequence $F = [F_0, \ldots, F_n]$ of states where
  - $F_0 = I$
  - $F_i(\mathbf{x}) \wedge T(\mathbf{x}, \mathbf{x}') \implies F_{i+1}(\mathbf{x}')$ for all $0 \leq i < n$
- A trace is monotone if $F_i \implies F_{i+1}$ for all $0 \leq i < n$
- Two phases: block bad states, push forward good states

Important pieces

- BMC searches for counter-examples (reachable $\neg$P-states)
- k-induction uses multiple transitions to get more information about system
- Interpolation can over-approximate states reachable in one (or more) transitions
- PDR takes a set of good states and find its inductive subset (in the form of a monotonic inductive trace)

(K)AVY - intuition

KAVY Algorithm (main loop)

$F, N \leftarrow [I], 0$

```
# Do BMC constrained to F
while True:
    let U ≡ F_0(x_0) ∧ T(x_0, x_1) ∧ ... ∧ F_N(x_n) ∧ T(x_n, x_{n+1}) ∧ ¬P(x_{n+1})
    if sat(U): return unsafe (+ cex)

    (i, k) ← frame_to_extend(F)
    [F_0, ..., F_{N+1}] ← extend(F, (i, k))
    [F_0, ..., F_{N+1}] ← pdr_push(F)

    # some frame got inductive
    if ∃i ≤ N : F_i ⟹ (⋁_{j<i} F_j): return safe

    N ← N + 1
```

KAVY – frame_to_extend

```
def frame_to_extend(F):
```

$$\texttt{let}\ \ S_i(i,k) \equiv \overbrace{F_i(\mathbf{x}_0) \land T(\mathbf{x}_0,\mathbf{x}_1) \land F_i(\mathbf{x}_1) \land T(\mathbf{x}_1,\mathbf{x}_2) \land \ldots F_i(\mathbf{x}_{k-1}) \land T(\mathbf{x}_{k-1},\mathbf{x}_k)}^{\texttt{k steps in } F_i}$$

$$\texttt{let}\ \ S_r(i,k) \equiv \overbrace{F_{i+1}(\mathbf{x}_k) \land T(\mathbf{x}_k,\mathbf{x}_{k+1}) \land \cdots \land F_N(\mathbf{x}_{k+(N-i)}) \land T(\mathbf{x}_{k+(N-i)},\mathbf{x}_B)}^{\texttt{step through the rest of } F}$$

$$\texttt{let}\ \ S(i,k) \equiv \begin{cases} S_i(i,k) \land S_r(i,k) \land \neg P(\mathbf{x}_B) & \text{if } i < N \\ S_i(i,k) \land \neg P(\mathbf{x}_x) & \text{if } i = N \end{cases}$$

$$i \leftarrow max\{j \mid 0 \leq j \leq N : S(j, j+1)\ \texttt{is unsat}\}$$
$$k \leftarrow min\{l \mid 1 \leq l \leq (i+1) : S(i,k)\ \texttt{is unsat}\}$$

```
return (i, k)
```

KAVY – extending trace

```
def extend(F, (i, k)):
    R_{i-k+2}, ..., R_{N+1} ← interpolants(S(i, k))
    G ← [F_0, ..., F_N, ⊤]

    # k-prefix in F_i
    for j in i − k + 1, ..., i:
        pdr_block(G, G_{i+1}, ¬(G_j ∨ (G_{i+1} ∧ I_{j+1}))

    # frame F_i
    pdr_block(G, G_{i+1}, ¬(G_i ∨ (G_{i+1} ∧ I_{i+1})))

    # the rest of the trace
    for j in i + 1, ..., N + 1:
        pdr_block(G, G_{j+1}, ¬(G_j ∨ (G_{j+1} ∧ Ij + 1)))
        pdr_push(G)
    return G
```

MUNI
FACULTY
OF INFORMATICS