Synthesizing Universally-Quantified Inductive Invariants

Sharon Shoham

Tel Aviv University
Synthesizing Universally-Quantified Inductive Invariants

Aleksandr Karbyshev  Nikolaj Bjorner  Shachar Itzhaky  Noam Rinetzky

Oded Padon  Kenneth McMillan  Aurojit Panda  Mooly Sagiv
System S is safe if all the reachable states satisfy the property $P = \neg \text{Bad}$

System S is safe iff there exists an inductive invariant $\text{Inv}$:

$\text{Inv} \Rightarrow P = \neg \text{Bad}$  
(Safety)

$\text{Init} \Rightarrow \text{Inv}$  
(Initiation)

if $\sigma \models \text{Inv}$ and $T(\sigma, \sigma')$ then $\sigma' \models \text{Inv}$  
(Consecution)
Safety Verification

System S is safe if all the reachable states satisfy the property $P = \neg \text{Bad}$.

System S is safe iff there exists an inductive invariant $\text{Inv}$:

\[
\begin{align*}
\text{Inv} \implies P &= \neg \text{Bad} & \text{(Safety)} \\
\text{Init} \implies \text{Inv} & & \text{(Initiation)} \\
\text{if } \sigma \models \text{Inv} \text{ and } T(\sigma, \sigma') & \text{ then } \sigma' \models \text{Inv} & \text{(Consecution)}
\end{align*}
\]
Challenges

Infer inductive invariants for safety verification

But also

- Specification: reasoning about infinite-state systems
  - Unbounded number of objects, threads, messages,...
  - Quantification is useful

- Deduction: reasoning about inductive invariants
  - Undecidability of implication checking
This talk

Specify systems and properties in decidable fragment of first-order logic
- Allows quantifiers to reason about unbounded sets
- Decidable to check inductiveness

Synthesize quantified inductive invariants
- Automatically by universal property directed reachability
- Interactively by providing graphical UI for gradually strengthening the inductive invariant
Effectively Propositional Logic – EPR
a.k.a. Bernays-Schönfinkel-Ramsey class

- Limited fragment of first-order logic
  - Restricted quantifier prefix: $\exists^* \forall^* \phi_{Q.F.}$
    - No $\forall^* \exists^*$
    - No recursive function symbols
    - No arithmetic

- Finite model property
  - A formula is satisfiable iff it is holds on models proportional to the number of existential variables

- Satisfiability is decidable

- Support from Z3, Iprover, Vampire
Example: Leader Election in a Ring

- Nodes are organized in a unidirectional ring
- Each node has a unique numeric id
- Protocol:
  - Each node sends its id to the next
  - A node that receives a message passes it (to the next) if the id in the message is higher than the node’s own id
  - A node that receives its own id becomes the leader
- Theorem:
  - The protocol selects at most one leader

Example: Leader Election in a Ring

- Nodes are organized in a unidirectional ring
- Each node has a unique numeric id
- Protocol:
  - Each node sends its id to the next
  - A node that receives a message passes it (to the next) if the id in the message is higher than the node’s own id
  - A node that receives its own id becomes the leader

Theorem:
- The protocol selects at most one leader

Proposition: This algorithm detects one and only one highest number.

Argument: By the circular nature of the configuration and the consistent direction of messages, any message must meet all other processes before it comes back to its initiator. Only one message, that with the highest number, will not encounter a higher number on its way around. Thus, the only process getting its own message back is the one with the highest number.

Modeling with EPR

- **State**: finite first-order structure over vocabulary $V$
  - $\leq (ID, ID)$ – total order on node id’s
  - $btw$ (Node, Node, Node) – the ring topology
  - $id$: Node $\rightarrow$ ID – relate a node to its id
  - $pending(ID, Node)$ – pending messages
  - $leader$(Node) – leader$(n)$ means $n$ is the leader

Axiomatized in first-order logic

```
protocol state

structure

$\sigma = ( \{ n_1, \ldots, n_6, id_1, \ldots, id_6 \}, I )$
$I (\leq) = \{ \langle id_1, id_1 \rangle, \langle id_1, id_2 \rangle, \langle id_1, id_3 \rangle, \langle id_1, id_4 \rangle \ldots \}$
$I (btw) = \{ \langle n_1, n_3, n_5 \rangle, \langle n_1, n_3, n_2 \rangle, \langle n_1, n_3, n_4 \rangle \ldots \}$
$I (id) = \{ n_1 \mapsto id_1, n_2 \mapsto id_6, n_3 \mapsto id_4, \ldots \}$
$I (pending) = \{ \}$
$I (leader) = \{ \}$
```
Modeling with EPR

- **State**: finite first-order structure over vocabulary V

- **Initial** states and **safety** property: EPR formulas over V
  - $\text{Init}(V)$ – initial states, e.g., $\forall \text{id}, n. \neg \text{pending}(\text{id}, n)$
  - $\text{Bad}(V)$ – bad states, e.g., $\exists n_1, n_2. \text{leader}(n_1) \land \text{leader}(n_2) \land n_1 \neq n_2$

- **Transition relation**:
  - EPR formula $\text{TR}(V, V')$
    - $V'$ is a copy of $V$ describing the next state
    - e.g. $\forall n. \text{leader}'(n) \leftrightarrow (\text{leader}(n) \lor \text{pending}(\text{id}[n], n))$
Modeling with EPR

- **State**: finite first-order structure over vocabulary V

- **Initial** states and **safety** property: EPR formulas over V
  - $\text{Init}(V)$ – initial states, e.g., $\forall \text{id}, n. \neg \text{pending}(\text{id}, n)$
  - $\text{Bad}(V)$ – bad states, e.g., $\exists n_1, n_2. \text{leader}(n_1) \land \text{leader}(n_2) \land n_1 \neq n_2$

Specify and verify the protocol for **any** number of nodes in the ring
Using EPR for Verification

- **System Model in EPR**
  
  $\text{Init}(V), \text{Bad}(V), \text{TR}(V, V')$

- **Inv(V)** is an **inductive invariant** if:
  
  - **Initiation** $\text{Init} \land \neg \text{Inv}$ unsat
  - **Consecution** $\text{Inv} \land \text{TR} \land \neg \text{Inv}'$ unsat
  - **Safety** $\text{Inv} \land \text{Bad}$ unsat

Decidable to check for $\text{Inv} \in \forall^*$

Useful for: linked lists, network routing, distributed protocols,...
Using EPR for Verification

- **System Model in EPR**
  \[ \text{Init}(V), \text{Bad}(V), \text{TR}(V, V') \]

- **Inv(V)** is an inductive invariant if:
  - **Initiation** \( \text{Init} \land \neg \text{Inv} \) unsat
  - **Consecution** \( \text{Inv} \land \text{TR} \land \neg \text{Inv}' \) unsat
  - **Safety** \( \text{Inv} \land \text{Bad} \) unsat

**Challenge:** find \( \text{Inv} \in \forall^* \)

Decidable to check for \( \text{Inv} \in \forall^* \)

Useful for: linked lists, network routing, distributed protocols, ...
Naïve algorithm

Iterative strengthening

Inv = \neg \text{Bad}

Check Inductiveness

Counterexample To Induction (CTI)

I can decide inductiveness!
Naïve algorithm

Iterative strengthening

\[ \text{Inv} = \neg \text{Bad} \land \text{"Avoid}(\sigma_1)" \]

Check Inductiveness

Counterexample To Induction (CTI)
Naïve algorithm

Key challenge for invariant inference: generalization

\[ \text{Inv} = \neg \text{Bad} \land \text{“Avoid}(\sigma_1)\text{”} \land \text{“Avoid}(\sigma_2)\text{”} \]
Generalization using Diagram

Use **diagrams** as abstract representation of states

- state $\sigma$ is a **finite** first-order structure

$$\text{Diag}(\sigma) = \exists x \ y. \ x \neq y \land L(x) \land \neg L(y)$$
$$\land \leq (x, y) \land \neg \leq (y, x)$$
$$\land \leq (x, x) \land \leq (y, y)$$

$\sigma' \models \text{Diag}(\sigma)$ iff $\sigma$ is a substructure of $\sigma'$

$\sigma$ is obtained from $\sigma'$ by removing elements and projecting relations on remaining elements

Use **diagrams** as abstract representation of states

- state $\sigma$ is a **finite** first-order structure

$$\text{Diag}(\sigma) = \exists x \ y. \ x \neq y \land L(x) \land \neg L(y) \land \leq (x, y) \land \neg \leq (y, x) \land \leq (x, x) \land \leq (y, y)$$

$\sigma' \models \text{Diag}(\sigma)$ iff $\sigma$ is a substructure of $\sigma'$

$\sigma$ is obtained from $\sigma'$ by removing elements and projecting relations on remaining elements

$$\text{Avoid}(\sigma) = \neg \text{Diag}(\sigma)$$

---

Generalization using Diagram

Can generalize more
- remove facts/conjuncts

\[ \text{Diag}(\sigma) = \exists x \ y. \ x \neq y \land L(x) \land \lnot L(y) \land \leq (x, y) \land \lnot \leq (y, x) \land \leq (x, x) \land \leq (y, y) \]

\[ \text{gen(Diag}(\sigma)) = \exists x \ y. \ x \neq y \land \leq (x, y) \land \lnot \leq (y, x) \land \leq (x, x) \land \leq (y, y) \]

\[ \text{Avoid}(\sigma) = \lnot \text{gen(Diag}(\sigma)) \]

Universally-Quantified Invariant

\[ \text{Inv} \equiv \forall \overline{x}. \ (l_{1,1} (\overline{x}) \lor \ldots \lor l_{1,m} (\overline{x})) \land \ldots \land \forall \overline{x}. \ (l_{n,1} (\overline{x}) \lor \ldots \lor l_{n,m} (\overline{x})) \]

\[ \text{Inv} \equiv \neg \exists \overline{x}. \ (\neg l_{1,1} (\overline{x}) \land \ldots \land \neg l_{1,m} (\overline{x})) \land \ldots \land \neg \exists \overline{x}. \ (\neg l_{n,1} (\overline{x}) \land \ldots \land \neg l_{n,m} (\overline{x})) \]
Universally-Quantified Invariant

\[ \text{Inv} \equiv \forall \bar{x}. (l_{1,1}(\bar{x}) \lor \ldots \lor l_{1,m}(\bar{x})) \land \ldots \land \forall \bar{x}. (l_{n,1}(\bar{x}) \lor \ldots \lor l_{n,m}(\bar{x})) \]

Questions:
- How to find the states to generalize from?
- How to select which facts to remove in the generalization?
Next

- UPDR: **Semi-algorithm** for inference of universal inductive invariants

- IVy: **Interactive** approach for inferring universal inductive invariants
Automatic Synthesis of Universal Invariants

Universal Property Directed Reachability (UPDR)

- Performs automatic generalization
- Based on Bradley’s IC3/PDR \([\text{VMCAI}11, \text{FMCAD}11]\)
Property Directed Reachability

- $F_i$ over-approximates the states that are reachable in at most $i$ steps
- If $F_{k+1} \equiv F_k$ then $F_k$ is an inductive invariant
- Computation of $F_i$ is guided by the property $P=\neg \text{Bad}$

Initial states (Init)

Bad states

Formal expression:

- $F_0 = \text{Init}$
- $F_i \Rightarrow F_{i+1}$
- $F_i \wedge \text{TR} \Rightarrow (F_{i+1})'$
- $F_i \Rightarrow \neg \text{Bad}$
How is $F_{i+1}$ computed in (U)PDR?

If $\text{Diag}(\sigma_{i+1})$ is reachable from $F_i$: continue backwards until $\text{Init}$

$$F_{i+1} = \text{true}$$

$$\text{SAT}(F_{i+1} \land \text{Bad})?$$

$$\text{SAT}(F_i \land \text{TR} \land \text{Diag}(\sigma_{i+1}'))?$$

$$\sigma_i$$

$$\sigma_{i-1}$$

$$\sigma_{i+1}$$

$F_1$

$F_{i-1}$

......
How is $F_{i+1}$ computed in (U)PDR?

If $\text{Diag}(\sigma_{i+1})$ is reachable from $F_i$: continue backwards until $\text{Init}$

If $\text{Diag}(\sigma_j)$ is unreachable from $F_{j-1}$: strengthen $F_j$ to exclude $\text{UnsatCore}(\text{Diag}(\sigma_j))$
UPDR: Possible Outcomes

- Fixpoint: *universal inductive invariant found*
  - System is safe

- Abstract counterexample:

```
\sigma_0 \ldots \sigma_{i-1} \sigma_i \sigma_{i+1}
```

```
Init F_1 F_{i-1} F_i F_{i+1}
```

```
\sigma_0 \sigma_1 \sigma_{i-1} \sigma_i \sigma_{i+1}
```
UPDR: Possible Outcomes

• Fixpoint: universal inductive invariant found
  • System is safe

• Abstract counterexample:
  • Safety not determined*
    • But no universal inductive invariant exists!

* can use Bounded Model Checking to find real counterexamples
Proving the absence of universal Invariant

Suppose that a safety universal invariant $I$ exists. Then:

$I$ satisfies safety:\n\[
\sigma_{i+1} \models \text{Bad} \Rightarrow \sigma_{i+1} \not\models I
\]

$I$ is universal:\n\[
\sigma'_{i+1} \models \text{Diag}(\sigma_{i+1}) \Rightarrow \sigma'_{i+1} \not\models I
\]

$I$ satisfies consecution:\n\[
\sigma'_{i+1} \not\models I \land \text{TR}(\sigma_i, \sigma'_{i+1}) \Rightarrow \sigma_i \not\models I
\]

$I$ satisfies initiation:\n\[
\sigma_0 \not\models I \Rightarrow \sigma_0 \not\models \text{Init}
\]

If there is $I \in \forall^*$, then any relaxed trace does not reach Init.

$\Rightarrow$ A relaxed trace from Init to Bad implies no $I \in \forall^*$ exists.
Experiments

Used to infer **inductive invariants / procedure summaries** of:

- **Heap-manipulating programs**, e.g.
  - Singly-linked list
  - Doubly-linked list
  - Nested lists
  - Iterators in Java - Concurrent modification error (CME)

- **Distributed protocols**
  - Spanning tree
  - Learning switch

---


Termination?

Is it decidable to infer universal inductive invariants? [POPL’16]

- No, in the general case
  - if the vocabulary contains at least one binary relation which is unrestricted

- Yes, for linked lists
  - if the vocabulary contains only one "transitive closure" binary relation, but as many constants and unary predicates as desired
  - UPDR will also terminate
  - proof uses well-quasi-order and Kruskal’s tree theorem

Interactive Synthesis of Universal Invariants

https://github.com/Microsoft/ivy

Invariant Inference in IVy

Iterative strengthening

Inv = \neg Bad \land \text{"Avoid}(\sigma_1)\text{"} \land \text{"Avoid}(\sigma_2)\text{"}...

Key challenge for invariant inference: generalization

UPDR: diagram + unsat core

IVy’s approach: put the user in the loop

interactive generalization

Generalize from CTI

User \leftrightarrow Automation
Interactive Generalization from CTI

1. Generalize by removing “irrelevant” facts to form a conjecture
   - User graphically selects which facts to remove
2. Check if the conjecture is true up to K: BMC(K)
   - User determines the right K to use
   - IVy uses a SAT solver
3. Automatically remove more facts: Interpolate(K)
   - IVy uses the SAT solver to discover more facts to remove
   - User examines the result – it could be wrong
Summary

• Decidable deduction using EPR
  – EPR transition system
  – Inductive invariant Inv ∈ ∀*

• Synthesis of Inv ∈ ∀* by generalization
  – Automatically: UPDR
  – Interactively: IVy

• Key idea: use diagram to generalize from counterexamples to induction
  – Can sometimes prove absence of Inv ∈ ∀*