Verification of Infinite-State Systems Using Decidable Logic

Sharon Shoham

Tel Aviv University

Supervised Verification of Infinite-State Systems
Model Checking [EC81,QS82]

Finite-State System $S$

Temporal Property $\varphi$

Model checker Does $S$ satisfy $\varphi$?

Counterexample

Proof

*Clarke, Emerson, and Sifakis won the 2007 Turing award for their contribution to Model Checking*
Model Checking [EC81, QS82]

Finite-State System $S$

Temporal Property $\varphi$

Model checker

Counterexample

Proof

State Explosion Problem

*Clarke, Emerson, and Sifakis won the 2007 Turing award for their contribution to Model Checking*
2009-2010 Stay at home mom

2011 Postdoc Technion

2011-2016 Senior Lecturer Academic college of Tel Aviv Yaffo & Visiting Researcher Tel Aviv University (2015-2016)
Infinite-State Systems

Programs:
• Dimensions of infinity:
  • unbounded number of dynamically allocated objects
  • unbounded number of threads
  • unbounded domain of variables (naturals, reals...)

Distributed systems:
• Dimensions of infinity:
  • unbounded number of hosts/switches
  • unbounded number of pending messages

Set of states is infinite
Automatic verification of infinite-state systems

Infinite-State System $S$

Property $\varphi$

Verification
Is there a behavior of $S$ that violates $\varphi$?

Counterexample
Unknown / Diverge
Proof

Rice’s Theorem
I can’t decide!
Safety Verification

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

Safety Property
“no two leaders are elected”

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

\[
\begin{align*}
\text{Init} & \subseteq \text{Inv} \quad \text{(Initiation)} \\
\text{if } \sigma \in \text{Inv} \text{ and } \sigma \rightarrow \sigma' \text{ then } \sigma' \in \text{Inv} \quad \text{(Consecution)} \\
\text{Inv} \cap \text{Bad} & = \emptyset \quad \text{(Safety)}
\end{align*}
\]
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 $Inv$:

1. $Init \subseteq Inv$ (Initiation)
2. If $\sigma \in Inv$ and $\sigma \rightarrow \sigma'$ then $\sigma' \in Inv$ (Consecution)
3. $Inv \cap Bad = \emptyset$ (Safety)
Example

• N pairs of players pass a ball:
  – $1^\uparrow$ will pass to $1^\downarrow$
  – $1^\downarrow$ will pass to $1^\uparrow$
  – $2^\uparrow$ will pass to $2^\downarrow$
  – $2^\downarrow$ will pass to $2^\uparrow$  ...
Example

• N pairs of players pass a ball:
  – 1↑ will pass to 1↓
  – 1↓ will pass to 1↑
  – 2↑ will pass to 2↓
  – 2↓ will pass to 2↑ ...
Example

• N pairs of players pass a ball:
  – $1^\uparrow$ will pass to $1^\downarrow$
  – $1^\downarrow$ will pass to $1^\uparrow$
  – $2^\uparrow$ will pass to $2^\downarrow$
  – $2^\downarrow$ will pass to $2^\uparrow$ ...

• The ball starts at player $1^\uparrow$

• Can the ball get to $2^\downarrow$?
Example

• N pairs of players pass a ball:
  – 1↑ will pass to 1↓
  – 1↓ will pass to 1↑
  – 2↑ will pass to 2↓
  – 2↓ will pass to 2↑ ...

• The ball starts at player 1↑

• Can the ball get to 2↓?

• Is “the ball is not at 2↓” an inductive invariant?
Example

- N pairs of players pass a ball:
  - $1^{\uparrow}$ will pass to $1^{\downarrow}$
  - $1^{\downarrow}$ will pass to $1^{\uparrow}$
  - $2^{\uparrow}$ will pass to $2^{\downarrow}$
  - $2^{\downarrow}$ will pass to $2^{\uparrow}$ ...

- The ball starts at player $1^{\uparrow}$
- Can the ball get to $2^{\downarrow}$?
- Is “the ball is not at $2^{\downarrow}$” an inductive invariant? **No!**
  - **Counterexample to induction**
Example

• N pairs of players pass a ball:
  – 1↑ will pass to 1↓
  – 1↓ will pass to 1↑
  – 2↑ will pass to 2↓
  – 2↓ will pass to 2↑ ...

• The ball starts at player 1↑

• Can the ball get to 2↓?

• Is “the ball is not at 2↓” an inductive invariant? No!
  – Counterexample to induction

• Inductive invariant: “the ball is not at 2↑ or 2↓”
Logic-based verification

• Represent $Init, Tr, Bad, Inv$ by logical formulas
  • Formula $\Leftrightarrow$ Set of states

• Automated solvers for logical satisfiability made huge progress
  • Propositional logic (SAT) – industrial impact for hardware verification
  • First-order theorem provers
  • Satisfiability modulo theories (SMT) – major trend in software verification
  • Z3, CVC4, iProver, Vampire ....
How can we **check** an inductive invariant?

Inv(V) is an **inductive invariant** if the following **verification conditions** are valid:

- **Initiation**  
  \[ \text{Init}(V) \implies \text{Inv}(V) \]  
  \[ \text{unsat}( \text{Init}(V) \land \neg \text{Inv}(V) ) \]

- **Cons.**  
  \[ \text{Inv}(V) \land \text{TR}(V,V') \implies \text{Inv}(V') \]  
  \[ \text{unsat}( \text{Inv}(V) \land \text{TR}(V,V') \land \neg \text{Inv}(V') ) \]

- **Safety**  
  \[ \text{Inv}(V) \implies \neg \text{Bad}(V) \]  
  \[ \text{unsat}( \text{Inv}(V) \land \text{Bad}(V) ) \]

Are the logical VC’s valid?

- Counterexample
- Unknown / Diverge
- Proof

*Church’s Theorem*

I can’t decide!
What can we do about it?

Interactive theorem provers (Coq, Isabelle/HOL, LEAN)
  • Programmer gives inductive invariant and proves it
  • Huge programmer effort (~10-50 lines of proof per line of code)

SMT-based deductive verification (e.g. Dafny)
  • Programmer provides ind. invariant
  • VC’s generated and discharged automatically
  • SMT solver may diverge (matching loops, arithmetic)
Alternative: Restrict VC’s to decidable logic

\( \text{Inv}(V) \) is an **inductive invariant** if the following **verification conditions** are valid:

- **Initiation** \( \text{Init}(V) \Rightarrow \text{Inv}(V) \)
  \[
  \text{unsat}( \text{Init}(V) \land \neg \text{Inv}(V) )
  \]

- **Cons.** \( \text{Inv}(V) \land \text{TR}(V,V') \Rightarrow \text{Inv}(V') \)
  \[
  \text{unsat}( \text{Inv}(V) \land \text{TR}(V,V') \land \neg \text{Inv}(V') )
  \]

- **Safety** \( \text{Inv}(V) \Rightarrow \neg \text{Bad}(V) \)
  \[
  \text{unsat}( \text{Inv}(V) \land \text{Bad}(V) )
  \]

Are the logical VC’s valid? \( \in \) Decidable logic

Counterexample

Proof

I can decide!
Challenges for verification with decidable logic

Formal specification
Modeling in a decidable logic

Deduction
Checking validity of the VC’s

Invariant inference
Finding an inductive invariant
This talk

Formal specification

• Use EPR – decidable fragment of first order logic
• Surprisingly expressive

Invariant inference

• Automatic (based on PDR)
  - Semi-algorithm: may diverge
• Interactive
  - Based on graphically displayed counterexamples to induction
Effectively Propositional Logic – EPR

Decidable fragment of first order logic

+ Quantification (∃*∀*) - Theories (e.g., arithmetic)

😊 Allows quantifiers to reason about unbounded sets
  - ∀x,y. leader(x) ∧ leader(y) → x = y

😊 Satisfiability is decidable => Deduction is decidable

😊 Small model property => Finite cex to induction

😊 Turing complete modeling language

😊 Limited language for safety and inductive invariants
  ➢ Suffices for many infinite-state systems
Successful verification with EPR

- Shape Analysis
  [Itzhaky et al. CAV’13, POPL’14, CAV’14, CAV’15]
- Software-Defined Networks
  [Ball et al. PLDI’14]
- Distributed protocols
  [Padon et al. PLDI’16, OOPSLA’17, POPL’18, PLDI’18]
- Concurrent Modification Errors in Java programs
  [Frumkin et al. VMCAI’17]
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

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

\[
\begin{align*}
\langle n_5, n_1, n_3 \rangle & \in I(bt w) \\
\langle n_5, n_1, n_3 \rangle & \in I(bt w)
\end{align*}
\]
Modeling with EPR

• **State**: finite first-order structure over vocabulary $V$ (+ axioms)

• **Initial** states and **safety** property: EPR formulas over $V$
  - $\text{Init}(V)$ – initial states, e.g., $\forall \text{id, n. } \neg \text{pending(id, n)}$
  - $\text{Bad}(V)$ – 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**: expressed as EPR formula $\text{TR}(V, V')$, e.g.:
  - $\exists n,s. \text{ “s = next(n)”} \land \forall x,y. \text{ pending’}(x,y) \leftrightarrow (\text{pending}(x,y) \lor (x=\text{id[n]} \land y=s))$
  - $\lor \exists n. \text{ pending (id[n],n)} \land \forall x. \text{ leader’}(x) \leftrightarrow (\text{leader}(x) \lor x=n)$

...
Modeling with EPR

- **State**: finite first-order structure over vocabulary $V$ (+ axioms)

  - 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**: expressed as EPR formula $\text{TR}(V, V')$, e.g.:
  
  - $\exists n, s. \text{“s = next}(n)\text{”} \land \forall x, y. \text{pending}'(x, y) \leftrightarrow (\text{pending}(x, y) \lor (x = \text{id}[n] \land y = s))$

  - $\lor \exists n. \text{pending} (\text{id}[n], n) \land \forall x. \text{leader}'(x) \leftrightarrow (\text{leader}(x) \lor x = n)$

  ...
Modeling with EPR

- **State**: finite first-order structure over vocabulary $V$ (+ axioms)

- **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**: expressed as EPR formula $\text{TR}(V, V')$, e.g.:
  - $\exists n, s. "s = \text{next}(n)" \land \forall x, y. \text{pending}'(x, y) \leftrightarrow (\text{pending}(x, y) \lor (x = \text{id}[n] \land y = s))$
  - $\lor \exists n. \text{pending} (\text{id}[n], n) \land \forall x. \text{leader}'(x) \leftrightarrow (\text{leader}(x) \lor x = n)$ ...
Modeling with EPR

- **State**: finite first-order structure over vocabulary $V$ (+ axioms)

- **Initial** states and **safety** property: EPR formulas over $V$
  - $\text{Init}(V)$ – initial states, e.g., $\forall \text{id}, n. \neg \text{pending}($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 $\text{Init}(V), \, \text{Bad}(V), \, \text{TR}(V, \, V') \in \text{EPR}$

- Inductive invariant $\text{Inv}(V) \in \forall^*$

- Verification conditions
  - Initiation $\text{Init}(V) \implies \text{Inv}(V)$ $\quad \text{unsat}( \text{Init}(V) \land \neg \text{Inv}(V) )$
  - Cons. $\text{Inv}(V) \land \text{TR}(V, V') \implies \text{Inv}(V')$ $\quad \text{unsat}( \text{Inv}(V) \land \text{TR}(V, V') \land \neg \text{Inv}(V') )$
  - Safety $\text{Inv}(V) \implies \neg \text{Bad}(V)$ $\quad \text{unsat}( \text{Inv}(V) \land \text{Bad}(V) )$

Verification conditions $\in \text{EPR}$

⇒ Decidable to check
Inductive Invariant for Leader Election

Safety property:

\[ I_0 = \neg \text{Bad} = \forall x, y: \text{Node.} \; \text{leader}(x) \land \text{leader}(y) \rightarrow x = y \]

Inductive?

No!

- \( \leq \) (ID, ID) – total order on node id’s
- \( \text{btw} \) (Node, Node, Node) – the ring topology
- \( \text{id} \): Node \( \rightarrow \) ID – relate a node to its id
- \( \text{pending} \) (ID, Node) – pending messages
- \( \text{leader} \) (Node) – leader(n) means n is the leader
Inductive Invariant for Leader Election

Safety property:
\[ I_0 = \neg \text{Bad} = \forall x, y: \text{Node. leader}(x) \land \text{leader}(y) \rightarrow x = y \]

Inductive invariant: \( \text{Inv} = I_0 \land I_1 \land I_2 \land I_3 \)

\[ I_1 = \forall n_1, n_2: \text{Node. leader}(n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1] \]

\[ I_2 = \forall n_1, n_2: \text{Node.} \ pnd(\text{id}[n_1], n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1] \]

\[ I_3 = \forall n_1, n_2, n_3: \text{Node.} \ btw(n_1, n_2, n_3) \land pnd(\text{id}[n_2], n_1) \rightarrow \text{id}[n_3] \leq \text{id}[n_2] \]

- \( \leq (\text{ID, ID}) \) – total order on node id’s
- \( btw (\text{Node, Node, Node}) \) – the ring topology
- \( \text{id}: \text{Node} \rightarrow \text{ID} \) – relate a node to its id
- \( \text{pending}(\text{ID, Node}) \) – pending messages
- \( \text{leader}(\text{Node}) \) – leader(n) means n is the leader

The reason for using “\( btw \)” instead of “\( next \)”

The leader has the highest id

Only highest id can be self-pnd

Cannot bypass higher nodes
Inductive Invariant for Leader Election

Safety property:
\[ I_0 = \neg \text{Bad} = \forall x, y: \text{Node}. \text{leader}(x) \land \text{leader}(y) \rightarrow x = y \]

**Inductive invariant:**
\[ \text{Inv} = I_0 \land I_1 \land I_2 \land I_3 \]

\[ I_1 = \forall n_1, n_2: \text{Node}. \text{leader}(n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1] \]

\[ I_2 = \forall n_1, n_2: \text{Node}. \text{pnd}(\text{id}[n_1], n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1] \]

\[ I_3 = \forall n_1, n_2, n_3: \text{Node}. \text{btw}(n_1, n_2, n_3) \land \text{pnd}(\text{id}[n_1], n_1) \rightarrow \text{id}[n_3] \]

The leader has the highest id

Only highest id can be self-pnd

Cannot bypass higher nodes

Init(\(V\)) \land \neg \text{Inv}(\(V\))

\(\text{Inv}(\(V\)) \land \text{TR}(\(V, V'\)) \land \neg \text{Inv}(\(V'\))\)

\(\text{Inv}(\(V\)) \land \text{Bad}(\(V\))\)

[Proof]

\[
\text{EPR Solver}
\]

I can decide EPR!
So far

Formal specification

• Use EPR – decidable fragment of first order logic
• Surprisingly expressive
  • Integers: numeric id’s expressed with ≤
  • Transitive closure: ring topology expressed with btw
  • Network semantics: pending messages
• Sets and cardinalities (for consensus protocols)
Invariant inference: finding inductive invariants

• Automatically
  – Adapt techniques from finite-state model checking (PDR)
• Interactively
  – Based on graphically displayed counterexamples to induction
How can we find a universally quantified inductive invariant?
### Inductive Invariant for Leader Election

<table>
<thead>
<tr>
<th>$I_0$</th>
<th>$\forall n_1, n_2 : \text{Node. } \text{leader}(n_1) \land \text{leader}(n_2) \rightarrow n_1 = n_2$</th>
<th>At most one leader elected</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\exists n_1, n_2 : \text{Node. } \text{leader}(n_1) \land n_1 \neq n_2$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$I_1$</th>
<th>$\forall n_1, n_2 : \text{Node. } \text{leader}(n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1]$</th>
<th>The leader has the highest id</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\exists n_1, n_2 : \text{Node. } \text{leader}(n_1) \land \text{id}[n_2] &gt; \text{id}[n_1]$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$I_2$</th>
<th>$\forall n_1, n_2 : \text{Node. } \text{pnd}(\text{id}[n_1], n_1) \rightarrow \text{id}[n_2] \leq \text{id}[n_1]$</th>
<th>Only highest id can be self-pnd</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\exists n_1, n_2 : \text{Node. } \text{pnd}(\text{id}[n_1], n_1) \land \text{id}[n_2] &gt; \text{id}[n_1]$</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$I_3$</th>
<th>$\forall n_1, n_2, n_3 : \text{Node. } \text{btw}(n_1, n_2, n_3) \land \text{pnd}(\text{id}[n_2], n_1)$</th>
<th>Cannot bypass higher nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\rightarrow \text{id}[n_3] \leq \text{id}[n_2]$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\neg \exists n_1, n_2, n_3 : \text{Node. } \text{btw}(n_1, n_2, n_3) \land \text{pnd}(\text{id}[n_2], n_1)$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$\land \text{id}[n_3] &gt; \text{id}[n_2]$</td>
<td></td>
</tr>
</tbody>
</table>
Construct Inv by excluding “bad” states

1. How to find these states?
2. How to generalize into conjectures?
Use \textit{diagrams} as generalization.

- 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

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

Generalization using Diagram:

Generalize even more if \( \sigma \) is a partial structure

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

\[
\text{exclude}(\sigma) = \neg \text{Diag}(\sigma)
\]

∀* Invariant - excluded substructures

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}))

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}))
Leader election example

At most one leader

The leader has the highest ID

Only the leader can be self-pending

Cannot bypass higher nodes

How to find the (partial) states to generalize from?
(1) UPDR: Automatic inference

- Based on Bradley’s IC3/PDR [VMCAI11,FMCAD11]
  - SAT-based verification of finite-state systems
- Abstracts concrete states using their logical diagram
- Backward traversal performed over diagrams
- Blocking of CTI excludes a generalization of its diagram \(\Rightarrow\) generates universally quantified lemmas

UPDR: Possible outcomes

- Universal inductive invariant found
  - System is safe

Used to infer inductive invariants / procedure summaries of:
- Heap-manipulating programs, e.g.
  - Singly/Doubly/Nested linked list
  - Iterators in Java - Concurrent modification error (CME)
- Distributed protocols
  - Spanning tree
  - Learning switch
  - ...
UPDR: Possible outcomes

• 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 universally quantified inductive invariant $I$ exists. Then:

$I$ satisfies initiation: $\sigma_0 \models I \implies \sigma_0 \models I$

$I$ satisfies consecution: $\sigma_{i-1} \models I \land \text{TR}(\sigma_{i-1}, \sigma_i') \implies \sigma_i' \models I$

$I$ is universal: $\sigma_i' \models \text{Diag}(\sigma_i) \implies \sigma_i \models I$

Contradiction to safety!

If there is a universal inductive invariant $I \in \forall^*$, then any abstract trace does not reach Bad

An abstract trace to Bad implies no universal inductive invariant exists
Termination?
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

• More decidable classes

Automatic Verification (e.g., UPDR)

Ultimately limited by undecidability
Interactive Verification

- Divide the problem between the human and the machine
- Find a suitable way to conduct the interaction

Supervised Verification of Infinite-State Systems
Interactive Verification

Proof Assistants

Ultimately limited by human

proof/code:
Verdi: ~10
IronFleet: ~4

Ultimately limited by undecidability

Model Checking
Static Analysis

Supervised Verification of Infinite-State Systems
(2) Ivy: Interactive inference

Model

Inductive?

Yes

Inductive Invariant

No

Find “minimal” counterexample to induction (CTI)

EPR

Candidate Inductive Invariant

Inv = I_0 ∧ I_1 ∧ ... ∧ I_k

Modify candidate invariant

Inductive Invariant Found

Generalize from CTI

User

Diagram

Automation

EPR

Summary 1

Verification with decidable logic

• EPR - decidable fragment of FOL
  • Deduction is decidable
  • Finite counterexamples

• Can be made surprisingly powerful
  • Transitive closure: linked lists, ring topology [PLDI’16]
  • Paxos, Multi-Paxos, [OOPSLA’17]
  • Liveness and Temporal Properties [POPL’18]
  • Developing verified implementations [PLDI’18]
Summary 2

Invariant Inference

- Automatic inference: UPDR
- Interactive inference: Ivy

- Use logical diagram to infer \( \text{Inv} \in \forall^* \)
- Can also prove absence of \( \text{Inv} \in \forall^* \)
Decidable logic is useful!

• Other logics
• Theoretical understanding of limitations and tradeoffs
• Interactive verification
  • Dividing the problem between human and machine
  • Inference schemes
  • Forms of interaction

Seeking postdocs and students

Supervised Verification of Infinite-State Systems