Falsification with Induction (Proof Scores) and Bounded Model Checking (Search)

School of Information Science
Japan Adv. Inst. of Sci. & Tech. (JAIST)

Kazuhiro Ogata
Gist

- We can systematically find a counterexample showing that an observational transition system (OTS) does not enjoy an invariant property with:
  - induction (proof scores),
  - bounded model checking (search), and
  - their combination (induction-guided falsification).

- A simple example is used to describe it.
Outline of Talk

- An example: a flawed mutual exclusion protocol (FMP)
- Specification of the protocol in CafeOBJ
- Falsification of FMP with induction (proof scores)
- Falsification of FMP with (bounded) model checking (search)
- Falsification of FMP with induction-guided falsification (IGF)
- Falsification of NSPK by IGF
- Conclusion
An example: a flawed mutual exclusion protocol (FMP)

- Specification of the protocol in CafeOBJ
- Falsification of FMP with induction (proof scores)
- Falsification of FMP with (bounded) model checking (search)
- Falsification of FMP with induction-guided falsification (IGF)
- Falsification of NSPK by IGF
- Conclusion
Mutual Exclusion Protocols

- Computer systems have resources that are shared by active entities such as processes. E.g. storages and printers.
- Many such resources should be exclusively used, namely that at most one process is allowed to use such resources. How to achieve this: the *mutual exclusion (mutex) problem*.
- *Mutex protocols* are a way of achieving this. E.g. spinlocks with atomic instructions such as test&set, Dijkstra’s semaphore and Hore’s monitor.
Flawed Mutex Protocol (FMP)

- The pseudo-code executed by all processes:

```plaintext
Loop: "Remainder Section (RS)"
  rs: wait until locked = false;
  es: locked := true;
  "Critical Section (CS)"
  cs: locked := false;
```

- `locked` is a Boolean variable shared by all processes, and is used in neither RS nor CS.
- Initially `locked` is false and all processes are at `rs`. 
One desired property a mutex protocol should enjoy is the mutex property:

*There exists at most one process in the critical section at any given moment.*

- **Good**
- **Bad**
An example: a flawed mutual exclusion protocol (FMP)

Specification of the protocol in CafeOBJ

Falsification of FMP with induction (proof scores)

Falsification of FMP with (bounded) model checking (search)

Falsification of FMP with induction-guided falsification (IGF)

Falsification of NSPK by IGF

Conclusion
Formalizing FMP as a State Machine (SM)

- A state:
  
  \[ \text{locked: false} \]
  
  \[ \text{pc}[p]: \text{rs} \]

- 3 transitions for each process \( p \):

  \[ \text{try}_p \]

  \[ \text{locked: false} \]
  
  \[ \text{pc}[p]: \text{rs} \]

  \[ \text{locked: false} \]
  
  \[ \text{pc}[p]: \text{es} \]

**Loop**: “Remainder Section (RS)"
- \( \text{rs} \): \textit{wait until} \( \text{locked} = \text{false} \);

  \[ \text{es: locked := true;} \]
  
  “Critical Section (CS)"

  \[ \text{cs: locked := false;} \]
Formalizing FMP as a SM (cont.)

- 3 transitions for each process $p$ (cont.):

  ![Diagram showing transitions](image)

  Loop: “Remainder Section (RS)”
  - $rs$: \texttt{wait until locked} = false;
  - $es$: $locked :=$ true;
    - “Critical Section (CS)”
      - $cs$: $locked :=$ false;
  - $cs$: $locked :=$ false;

  ![Diagram showing transitions](image)
The mutex property is violated at the state.
Specifying the SM in CafeOBJ

♦ Reachable states are specified by one constant denoting an arbitrary initial state and three transition (action) operators:

```
op init : -> Sys {constr}
op try : Sys Pid -> Sys {constr}
op enter : Sys Pid -> Sys {constr}
op exit : Sys Pid -> Sys {constr}
```

♦ States are characterized by two observation operators:

```
op locked : Sys -> Bool
op pc : Sys Pid -> Label
```
Specifying the SM in CafeOBJ (cont.)

- The values returned by the observation operators for each state (and each process ID) are defined in equations.

\[
\begin{align*}
eq \text{locked}(\text{init}) &= \text{false} . \\
eq \text{pc}(\text{init},I) &= \text{rs} .
\end{align*}
\]
Specifying the SM in CafeOBJ (cont.)

\[
eq \text{locked}(\text{try}(S,I)) = \text{locked}(S) .
\]
\[
\begin{align*}
\text{ceq } & \text{pc}(\text{try}(S,I),J) \\
& = (\text{if } I = J \text{ then } \text{es} \text{ else } \text{pc}(S,J) \text{ fi}) \\
& \text{if } \text{c-try}(S,I) .
\end{align*}
\]
\[
\begin{align*}
\text{ceq } & \text{try}(S,I) = S \text{ if not } \text{c-try}(S,I) .
\end{align*}
\]

where \( \text{c-try}(S,I) \)
\[
= (\text{pc}(S,I) = \text{rs} \text{ and not } \text{locked}(S))
\]

\( \text{locked: false} \)

\( \ldots \)

\( \text{pc}[I]: \text{rs} \)

\( \ldots \)

\( \text{try}_I \)

\( \text{locked: false} \)

\( \ldots \)

\( \text{pc}[I]: \text{es} \)

\( \ldots \)
Specifying the SM in CafeOBJ (cont.)

\[
\begin{align*}
\text{ceq } & \text{locked(enter}(S,I)) = \text{true} \\
& \text{if } \text{c-enter}(S,I) . \\
\text{ceq } & \text{pc(enter}(S,I),J) \\
& = (\text{if } I = J \text{ then } cs \text{ else } \text{pc}(S,J) \text{ fi}) \text{ if } \text{c-enter}(S,I) . \\
\text{ceq } & \text{enter}(S,I) = S \text{ if } \text{not } \text{c-enter}(S,I) . \\
\text{where } & \text{c-enter}(S,I) = (\text{pc}(S,I) = \text{es}) \\
\end{align*}
\]
Specifying the SM in CafeOBJ (cont.)

\[
\begin{align*}
\text{ceq} & \ \text{locked} (\text{exit}(S,I)) = \text{false} \\
\text{if} & \ \text{c-exit}(S,I) . \\
\text{ceq} & \ \text{pc}(\text{exit}(S,I),J) \\
& = (\text{if} \ I = J \ \text{then} \ rs \ \text{else} \ \text{pc}(S,J) \ \text{fi}) \\
\text{if} & \ \text{c-exit}(S,I) . \\
\text{ceq} & \ \text{exit}(S,I) = S \ \text{if not} \ \text{c-exit}(S,I) . \\
\text{where} & \ \text{c-exit}(S,I) = (\text{pc}(S,I) = \text{cs})
\end{align*}
\]
An example: a flawed mutual exclusion protocol (FMP)
Specification of the protocol in CafeOBJ
Falsification of FMP with induction (proof scores)
Falsification of FMP with (bounded) model checking (search)
Falsification of FMP with induction-guided falsification (IGF)
Falsification of NSPK by IGF
Conclusion
Proof Attempt of the MP for the SM

- The MP (that there does not exist more than one process in the CS at the same time) can be rephrased as follows:

  *If there are processes in the CS, then those processes are the same.*

- The MP is expressed as the state predicate:

  \[
  \text{eq inv1}(S, I, J) = (\text{pc}(S, I) = \text{cs} \text{ and } \text{pc}(S, J) = \text{cs} \text{ implies } I = J).
  \]

- What to do is to try to prove that the state predicate is a theorem wrt the spec (or invariant wrt the SM).
Proof Attempt of the MP for the SM (cont.)

- The proof attempt is conducted by writing \textit{proof scores}, which consist of \textit{proof passages} (PPs).
- A typical proof passage looks like

```plaintext
open AModule
  -- fresh constants
ops s s' -> Sys . ... 
  -- assumptions
eq e_1 . ... eq e_n .
  -- successor state
eq s' = a(s,...) .
  -- check
red p(s,...) implies p(s',...) .
close
```

✓ The PP corresponds to a sub-case of an induction case.
✓ The sub-case is characterized by the \( n \) equations \( e_1, \ldots, e_n \).
✓ The equations are obtained by case analysis.
Proof Attempt of the MP for the SM (cont.)

- The proof attempt that inv1 is invariant wrt the SM by structural induction on $S$ conjectures the necessary lemma:

\[
eq \text{inv2}(S,I,J) = \neg (pc(S,I) = es \text{ and } pc(S,J) = cs \text{ and } \neg (I = J)).
\]

This says that there does not exist more than one process at $es$ or $cs$ at the same time.

Loop: “Remainder Section (RS)”
- rs: \textbf{wait until} locked = false;
- es: locked := true;

“Critical Section (CS)”
- cs: locked := false;
A necessary lemma of a state predicate $p$ is a state predicate $q$ such that if $q$ has a counterexample, then so does $p$, or equivalently if $p$ is invariant wrt a state machine concerned, then so is $q$.

If all lemmas used are necessary ones in the course of the proof attempt and one necessary lemmas has a counterexample, then so does the main goal (state predicate).
Proof Attempt of the MP for the SM (cont.)

How to conjecture necessary lemmas

1. A case (typically each induction case) is split into multiple sub-cases such that CafeOBJ returns either true or false for each sub-case.

2. A necessary lemma is conjectured from each sub-case such that CafeOBJ returns false. Let $e_1, \ldots, e_n$ be all equations characterizing such a sub-case.

3. The equations are conjoined, the formula is negated, and fresh constants are replaced with variables.

$$\neg(e_1 \wedge \ldots \wedge e_n)[c \rightarrow X, \ldots]$$

Note that if $e_i$ is $p = \text{true}$, $p$ is used, if $e_i$ is $p = \text{false}$, not $p$ is used, and otherwise, $e_i$ is used.
Proof Attempt of the MP for the SM (cont.)

How to conjecture \( \text{inv2} \):

\[
\text{eq inv2}(S,I,J) = \neg (\text{pc}(S,I) = es \land \\
\text{pc}(S,J) = cs \land \neg (I = J)) .
\]

open MUTEX-ISTEP

-- assumptions
  eq \text{pc}(s,k) = es .
  eq i = k .
  eq (j = k) = false .
  eq \text{pc}(s,j) = cs .
-- successor state
  eq s' = \text{enter}(s,k) .
-- check
  red inv1(s,i,j)
  implies inv1(s',i,j) .

✓ CafeOBJ returns false for the proof passage.
✓ Note that fresh constants \( s, s', k, i, j \) are declared in MUTEX-ISTEP.
Proof Attempt of the MP for the SM (cont.)

How to conjecture \( \text{inv2} \) (cont.):

\[
eq \text{inv2}(S, I, J) = \neg(\text{pc}(S, I) = \text{es} \quad \text{and} \quad \text{pc}(S, J) = \text{cs} \quad \text{and} \quad \neg(I = J))
\]

- The 4 equations are conjoined, the formula is negated, and the fresh constants are replaced with variables.

\[
\neg(\text{pc}(S, K) = \text{es} \quad \text{and} \quad I = K \quad \text{and} \quad \neg(J = K) \quad \text{and} \quad \text{pc}(S, J) = \text{cs})
\]

- This is equivalent to

\[
\neg(\text{pc}(S, I) = \text{es} \quad \text{and} \quad \text{pc}(S, J) = \text{cs} \quad \text{and} \quad \neg(I = J))
\]
Proof Attempt of the MP for the SM (cont.)

- In the course of the proof attempt, 4 more necessary lemmas are conjectured. One of them is:

\[ \text{eq inv6}(S,I,J) = \text{not}(pc(S,I) = \text{rs} \text{ and } pc(S,J) = \text{rs} \text{ and not}(I = J) \text{ and not}(\text{locked}(S))) . \]

This says that if there exist processes in the RS, then all processes are the same (there exists only one process) or \text{locked} is true.

- \text{inv6}(\text{init},i,j) \text{ reduces to false if } i \text{ is different from } j.
- Hence, the lemma does not hold for the SM.
Proof Attempt of the MP for the SM (cont.)

- Since all lemmas conjectured are necessary wrt the MP, we conclude that the SM does not enjoy the MP.
- A counterexample can be constructed by looking at the chain of lemma conjectures up to inv6.

\[ \neg \text{inv6} \]
\[ \text{locked: false} \]
\[ \text{pc}[p]: \text{rs} \]
\[ \text{pc}[q]: \text{rs} \]

\[ \text{try}_p \]

\[ \neg \text{inv5} \]
\[ \text{locked: false} \]
\[ \text{pc}[p]: \text{es} \]
\[ \text{pc}[q]: \text{rs} \]

\[ \text{enter}_p \]

\[ \neg \text{inv4} \]
\[ \text{locked: false} \]
\[ \text{pc}[p]: \text{es} \]
\[ \text{pc}[q]: \text{es} \]

\[ \text{try}_q \]

\[ \neg \text{inv2} \]
\[ \text{locked: true} \]
\[ \text{pc}[p]: \text{cs} \]
\[ \text{pc}[q]: \text{es} \]

\[ \text{enter}_q \]

\[ \neg \text{inv1} \]
\[ \text{locked: true} \]
\[ \text{pc}[p]: \text{cs} \]
\[ \text{pc}[q]: \text{cs} \]
An example: a flawed mutual exclusion protocol (FMP)
Specification of the protocol in CafeOBJ
Falsification of FMP with induction (proof scores)
Falsification of FMP with (bounded) model checking (search)
Falsification of FMP with induction-guided falsification (IGF)
Falsification of NSPK by IGF
Conclusion
Bounded Model Checking (BMC)

- The bounded reachable state space (BRSS) up to some depth $d$ from an initial state $init$ is checked for a state predicate $p$.

If there exists a state such that $p$ does not hold and the state is in the BRSS, then BMC can find the state or the path to the state from $init$, namely a counterexample of $\Box p$.

Note that $\Box p$ means that $p$ is invariant wrt a state machine.
The search functionality can be used to conduct BMC:

\[ \text{red } \text{init } = (n, d) \Rightarrow * \text{ pattern } \text{ suchThat } \text{cond} \].

By setting \text{init} to an initial state of a state machine and expressing \( \neg p \) in \text{pattern} \& \text{cond}.

To use this functionality, (state) transitions should be described in transition rules.
Transitions in Transition Rules (cont.)

❄ Configuration of states:

\[
\begin{align*}
\text{op } \text{void} & : \to \text{Sys} \{\text{constr}\} \\
\text{op } _\_ & : \text{Sys Sys} \to \text{Sys} \\
& \{\text{constr assoc comm id: void}\}
\end{align*}
\]

❄ Operators that hold values characterizing states:

\[
\begin{align*}
\text{op } (\text{pc}[\_]:\_) & : \text{Pid Label} \to \text{Obs} \{\text{constr}\} \\
\text{op } \text{locked:}_\_ & : \text{Bool} \to \text{Obs} \{\text{constr}\}
\end{align*}
\]

❄ If two processes \(p_1\) & \(p_2\) participate in the protocol, the initial state is expressed as

\[
(\text{pc}[p_1]: \text{rs}) (\text{pc}[p_2]: \text{rs}) (\text{locked}: \text{false})
\]
Transitions in Transition Rules (cont.)

\[
\text{trans [try]} : \ (pc[I] : rs) \ (\text{locked}: \ false) \\
\hspace{1em} \Rightarrow \ (pc[I] : es) \ (\text{locked}: \ false) .
\]

\[
\text{trans [enter]} : \ (pc[I] : es) \ (\text{locked}: \ B) \\
\hspace{1em} \Rightarrow \ (pc[I] : cs) \ (\text{locked}: \ true) .
\]

\[
\text{trans [exit]} : \ (pc[I] : cs) \ (\text{locked}: \ B) \\
\hspace{1em} \Rightarrow \ (pc[I] : rs) \ (\text{locked}: \ false) .
\]

Loop: “Remainder Section (RS)”
\[
\begin{align*}
\text{rs: } & \text{wait until } locked = false; \\
\text{es: } & locked := true; \\
\text{“Critical Section (CS)”} & \\
\text{cs: } & locked := false;
\end{align*}
\]
Falsification of the MP for FMP by BMC

- When we have two processes, a counterexample (CX) for MP is found with the search functionality.
  
  \[
  \text{red init } = (1,*) \\
  \quad \Rightarrow^* (pc[I]: cs) (pc[J]: cs) S .
  \]

- The CX is also found by exhaustively traversing the bounded reachable state space (BRSS) up to depth 4.
  
  \[
  \text{red init } = (1,4) \\
  \quad \Rightarrow^* (pc[I]: cs) (pc[J]: cs) S .
  \]

- But, it is not found by exhaustively traversing the BRSS up to depth 3.
  
  \[
  \text{red init } = (1,3) \\
  \quad \Rightarrow^* (pc[I]: cs) (pc[J]: cs) S .
  \]
An example: a flawed mutual exclusion protocol (FMP)

Specification of the protocol in CafeOBJ

Falsification of FMP with induction (proof scores)

Falsification of FMP with (bounded) model checking (search)

Falsification of FMP with induction-guided falsification (IGF)

Falsification of NSPK by IGF

Conclusion
Induction Guided Falsification (IGF)

What if a counterexample (CX) exists outside of the bounded reachable state space (BRSS) ?

One option is to increase \( d \).
But, the BRSS up to \( d+1 \) may not be exhaustively traversed due to the state explosion problem.

A CX that exists outside of the BRSS that can be exhaustively traversed is called a *deep CX* in the talk.
Another option is to try to prove $\Box p$ by induction, conjecturing lemmas $\Box q_1, ..., \Box q_n$, and check the bounded reachable state space for each $\Box q_i$ instead of $\Box p$.

If there exists a state $s_1$ s.t. $\neg q_k$ and there exists a path from $s_1$ to a state $s_2$ s.t. $\neg p$, then we find a counterexample of $\Box p$.

IGF alternately uses BMC and induction to find deep counterexamples.

How to check if there exists a path from $s_1$ to $s_2$.

- One option is to use BMC to find a state $s_2$ s.t. $\neg p$ in the bounded reachable state space from $s_1$ instead of $init$.
- Another option is to use necessary lemmas, namely that if a lemma $\Box q_k$ has a counterexample, then so does its main goal ($\Box p$).
IGF (cont.)

- IGF can be regarded as a combination of forward & backward reachability analysis methods.

- BMC is a typical forward reachability analysis method.
- Induction can be regarded as a backward reachability analysis method.

In the induction case, it is checked that each transition \( t \) preserves a state predicate \( p \).

\[
\begin{array}{c}
\bullet & \xrightarrow{t} & \bullet \\
p & & p' \\
s & & s' \\
\end{array}
\]

If \( p \) does not hold in \( s' \), the concern is whether \( s \) is reachable. This can be checked by conjecturing \( q \) that does not hold in \( s \) and proving \( \Box q \).

So, one state transition is taken back by induction.

Falsification of FMP by IGF

- We suppose that the bounded reachable state space (BRSS) up to depth 4 is too large to be exhaustively traversed.
- Only BMC cannot find any counterexamples for the MP in the BRSS up to depth 3.
- Then, we try to prove the MP by induction, conjecturing the necessary lemma $\text{inv}2$.

\[
eq \text{inv}2(S, I, J) = \neg (\text{pc}(S, I) = \text{es} \text{ and } \text{pc}(S, J) = \text{cs} \text{ and } \neg (I = J)).
\]
Falsification of FMP by IGF (cont.)

- BMC finds a counterexample for $\text{inv}_2$ in the BRSS up to depth 3.

$\text{red init } = (1, 3)$

$\Rightarrow^* (\text{pc}[I]: \text{es}) (\text{pc}[J]: \text{cs}) S$.

- Since $\text{inv}_2$ is a necessary lemma of the MP, we conclude that the SM does not enjoy the MP.
An example: a flawed mutual exclusion protocol (FMP)

Specification of the protocol in CafeOBJ

Falsification of FMP with induction (proof scores)

Falsification of FMP with (bounded) model checking (search)

Falsification of FMP with induction-guided falsification (IGF)

Falsification of NSPK by IGF

Conclusion
NSPK & Agreement Property

- **NSPK ([Needham & Schroeder 1978]):**
  - **Init:** \( \{ n_p, p \}_k(q) \)
  - **Resp:** \( \{ n_p, n_q \}_k(p) \)
  - **Ack:** \( \{ n_q \}_k(q) \)

- **Agreement Property (AP):** Whenever a protocol run is successfully completed by \( p \) and \( q \),
  - AP1: the principal with which \( p \) is communicating is really \( q \), and
  - AP2: the principal with which \( q \) is communicating is really \( p \).
Model Checking AP1 & AP2

- The bounded reachable state space (BRSS) up to depth 5 can be exhaustively traversed on a laptop with 2.33GHz CPU and 3GB RAM, but the BRSS up to depth 6 cannot.

- No counterexample of AP1 is found in the BRSS up to depth 5.

- No counterexample of AP2 is found in the BRSS up to depth 5.
Lemmas for AP1 & AP2

- A proof attempt of AP1 & AP2 conjectures 5 lemmas.
- One of them is what is called Nonce Secrecy Property (NSP) which is as follows:

The 2 nonces $n_p, n_q$ generated in a protocol run conducted by two non-intruder principals $p, q$ cannot be obtained by the intruder.
Model Checking NSP

A counterexample of NSP is found in the bounded reachable state space up to depth 5.

Since NSP is not a necessary lemma of AP1 & AP2, however, we cannot conclude that NSPK does not enjoy AP immediately.

Then, we need to find a path from a state in which NSP is violated to a state in which AP (precisely AP2) is violated.

Such a path is found and then we conclude that NSPK does not enjoy AP (precisely AP2).

✓ Note that this case study used Maude as a model checker.

Outline of Talk

- An example: a flawed mutual exclusion protocol (FMP)
- Specification of the protocol in CafeOBJ
- Falsification of FMP with induction (proof scores)
- Falsification of FMP with (bounded) model checking (search)
- Falsification of FMP with induction-guided falsification (IGF)
- Falsification of NSPK by IGF
- Conclusion
Conclusion

♦ Summary

- We have described 3 ways to systematically find a counterexample showing that an OTS does not enjoy an invariant property using a small example: induction, BMC, and IGF.
- A case study on falsification of NSPK by IGF has been briefly reported.

♦ Effect

- IGF may alleviate the notorious state explosion problem.
Thank you very much!