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:

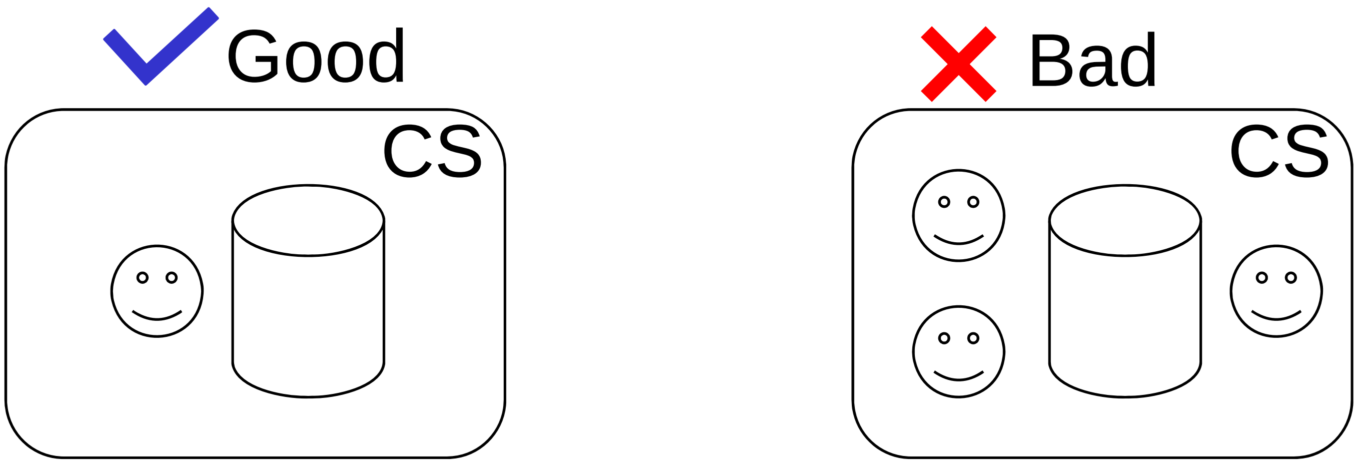
Loop: "Remainder Section (RS)"
  rs: **wait until** $locked$ = false;
  es: $locked$:= true;
    "Critical Section (CS)"
  cs: $locked$:= false;

✓ $locked$ 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.

Mutex Property

- 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.



- 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:

  $locked$: false
  …
  $pc[p]$: rs
  …

- 3 transitions for each process p:

$$\begin{array}{l} locked\text{: false} \\ \dots \\ pc[p]\text{: rs} \\ \dots \end{array} \xrightarrow{\quad try_p \quad} \begin{array}{l} locked\text{: false} \\ \dots \\ pc[p]\text{: es} \\ \dots \end{array}$$

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

Formalizing FMP as a SM (cont.)

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

locked: B
…
$pc[p]$: es
…

$\xrightarrow{enter_p}$

locked: true
…
$pc[p]$: cs
…

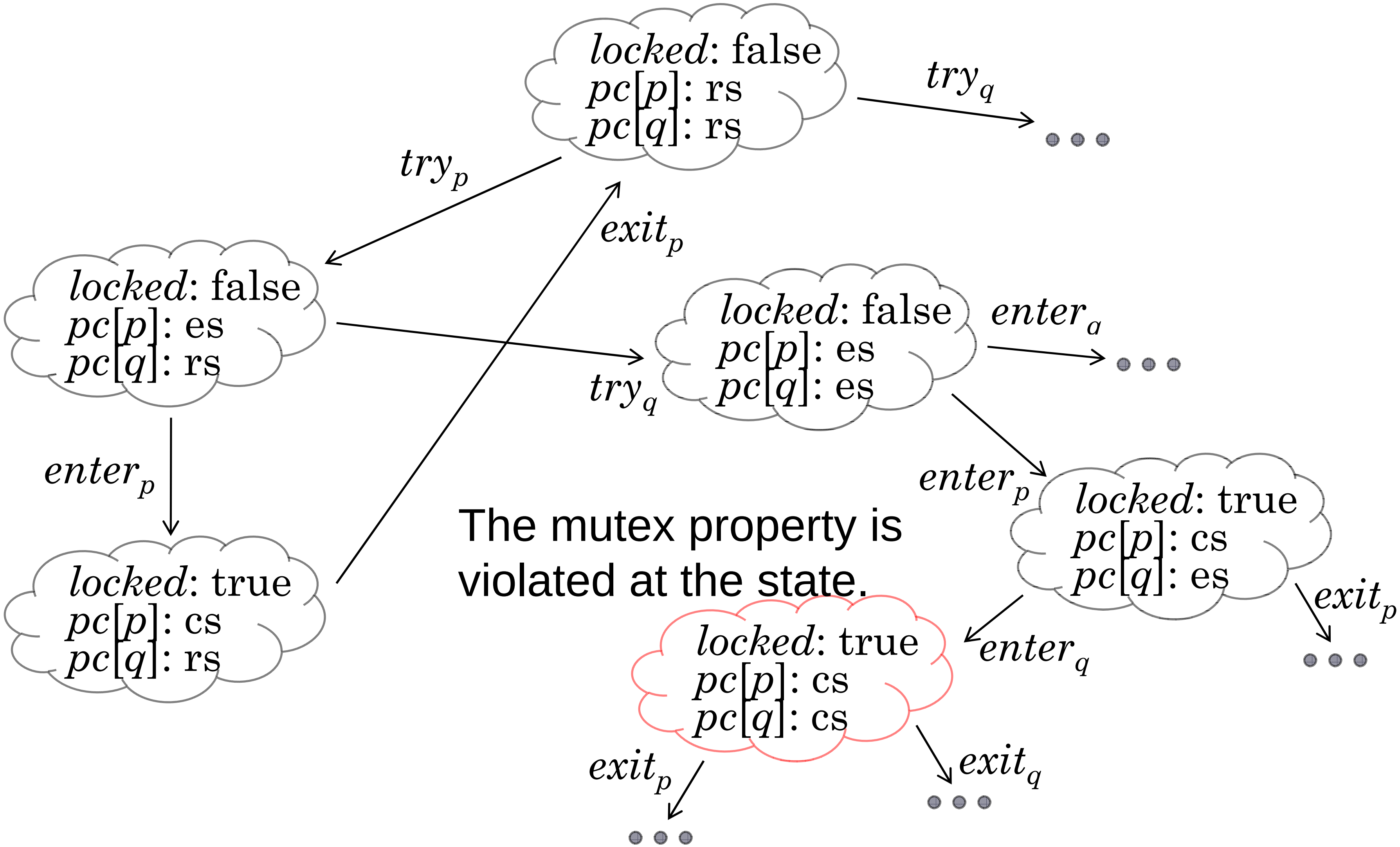
Loop: "Remainder Section (RS)"
rs: **wait until** *locked* = false;
es: *locked* := true;
"Critical Section (CS)"
cs: *locked* := false;

locked: B
…
$pc[p]$: cs
…

$\xrightarrow{exit_p}$

locked: false
…
$pc[p]$: rs
…

State Transition Diagram



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}
```

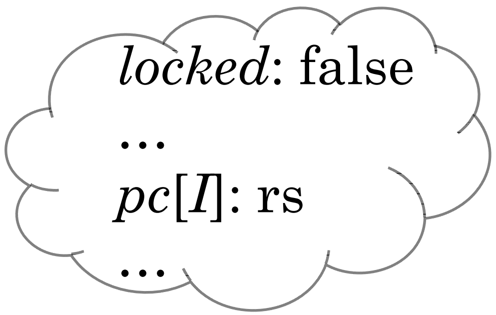
- States are characterized by two observation operators:

```
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.

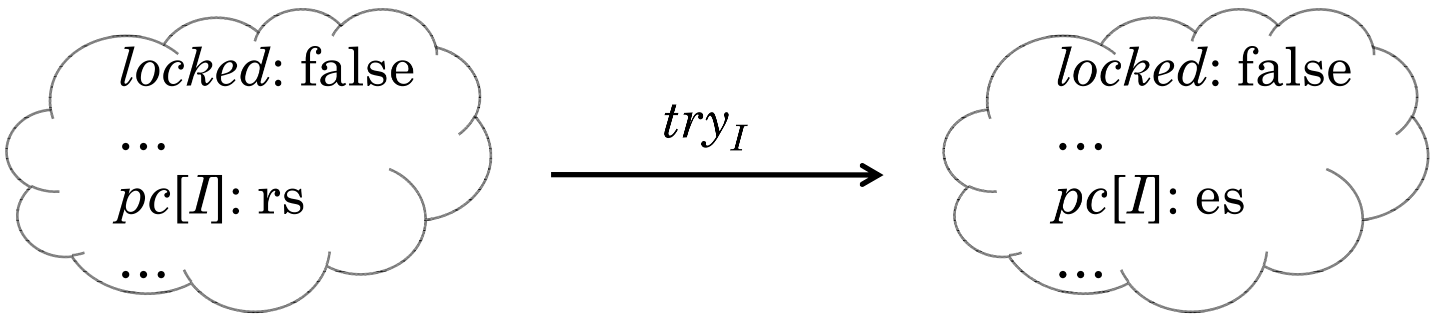
```
eq locked(init) = false .
eq pc(init,I) = rs .
```



Specifying the SM in CafeOBJ (cont.)

```
eq locked(try(S,I)) = locked(S) .
ceq pc(try(S,I),J)
 = (if I = J then es else pc(S,J) fi)
 if c-try(S,I) .
ceq try(S,I) = S if not c-try(S,I) .
```

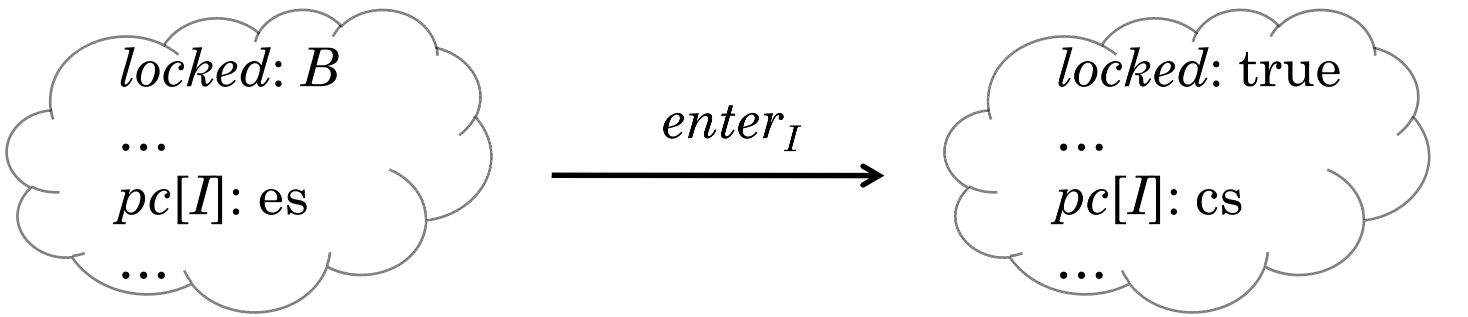
where `c-try(S,I)`
`= (pc(S,I) = rs and not locked(S))`



Specifying the SM in CafeOBJ (cont.)

```
ceq locked(enter(S,I)) = true
 if c-enter(S,I) .
ceq pc(enter(S,I),J)
 = (if I = J then cs else pc(S,J) fi)
 if c-enter(S,I) .
ceq enter(S,I) = S if not c-enter(S,I) .
```

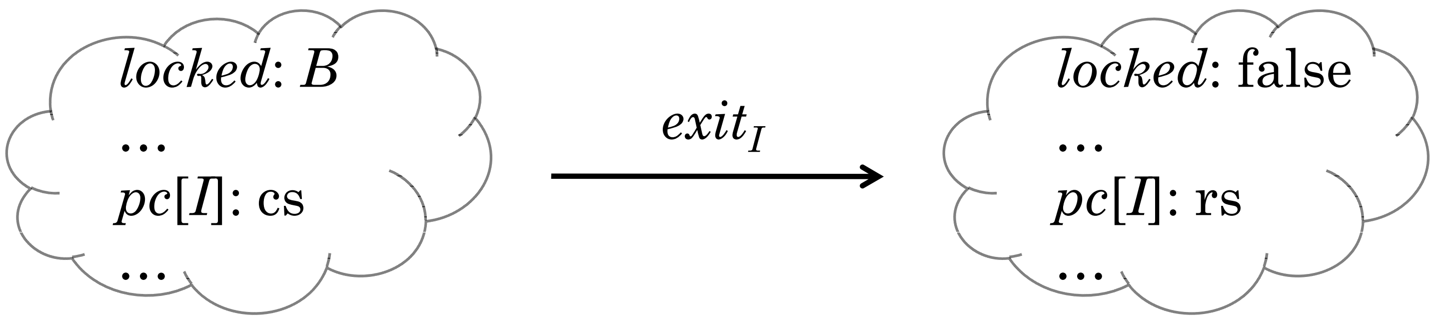
where `c-enter(S,I) = (pc(S,I) = es)`



Specifying the SM in CafeOBJ (cont.)

```
ceq locked(exit(S,I)) = false
 if c-exit(S,I) .
ceq pc(exit(S,I),J)
 = (if I = J then rs else pc(S,J) fi)
 if c-exit(S,I) .
ceq exit(S,I) = S if not c-exit(S,I) .
```

where `c-exit(S,I) = (pc(S,I) = 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

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:

```
eq inv1(S,I,J)
   = (pc(S,I) = cs and pc(S,J) = cs
      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 *proof scores*, which consist of *proof passages* (PPs).
- A typical proof passage looks like

```
open AModule
 -- fresh constants
 ops s s' -> Sys . …
 -- assumptions
 eq e1 . … eq en .
 -- 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 inv2(S,I,J)
   = not(pc(S,I) = es and pc(S,J) = cs
         and not(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: **wait until** *locked* = false;
  es: *locked* := true;
     "Critical Section (CS)"
  cs: *locked* := false;

Proof Attempt of the MP for the SM (cont.)

- 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 `= true`, p is used, if e_i is p `= false`, `not` p is used, and otherwise, e_i is used.

Proof Attempt of the MP for the SM (cont.)

- How to conjecture `inv2`:

```
eq inv2(S,I,J) = not(pc(S,I) = es and
                      pc(S,J) = cs and not(I = J)) .
```

```
open MUTEX-ISTEP
-- assumptions
  eq pc(s,k) = es .
  eq i = k .
  eq (j = k) = false .
  eq pc(s,j) = cs .
-- successor state
  eq s' = enter(s,k) .
-- check
  red inv1(s,i,j)
   implies inv1(s',i,j) .
close
```

✓ 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 `inv2` (cont.):

```
eq inv2(S,I,J) = not(pc(S,I) = es and
                     pc(S,J) = cs and not(I = J)) .
```

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

```
not(pc(S,K) = es and I = K and not(J = K)
and pc(S,J) = cs)
```

✓ This is equivalent to

```
not(pc(S,I) = es and pc(S,J) = cs and
not(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:

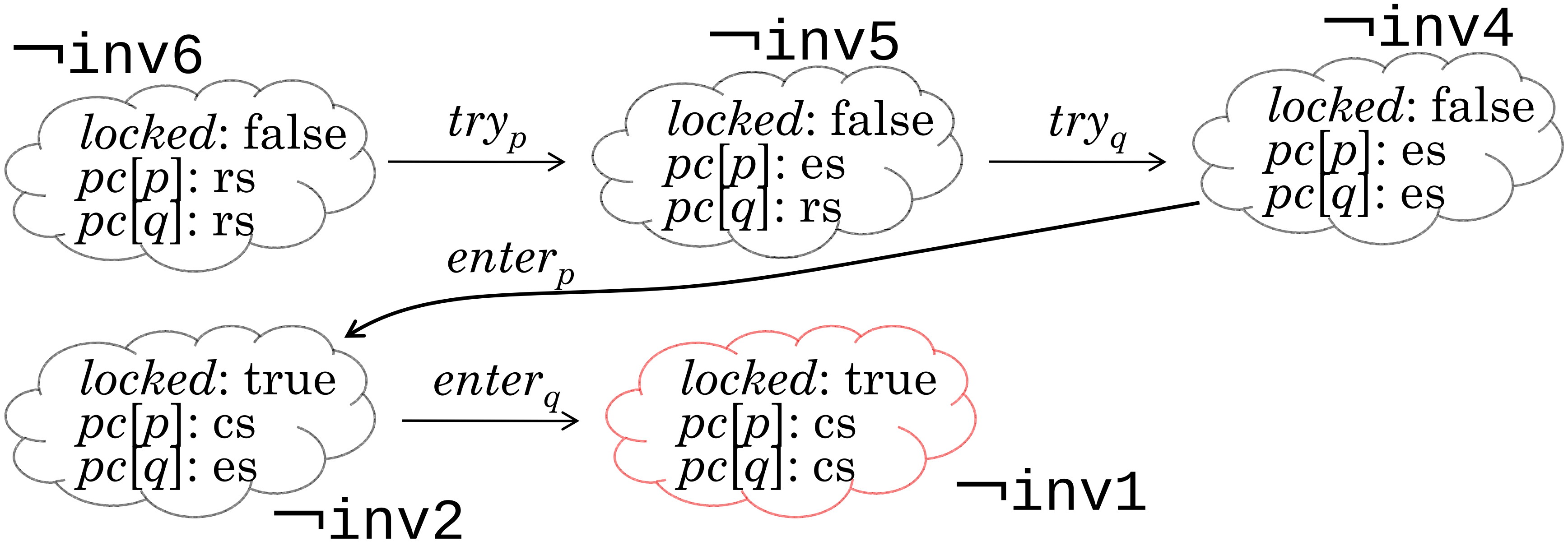
```
eq inv6(S,I,J)
 = not(pc(S,I) = rs and pc(S,J) = rs
       and not(I = J) and not(locked(S))) .
```

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

- `inv6(init,i,j)` reduces to `false` if `i` is different from `j`.
- Hence, the lemma does not hold for the SM.

Proof Attempt of the MP for the SM (cont.)

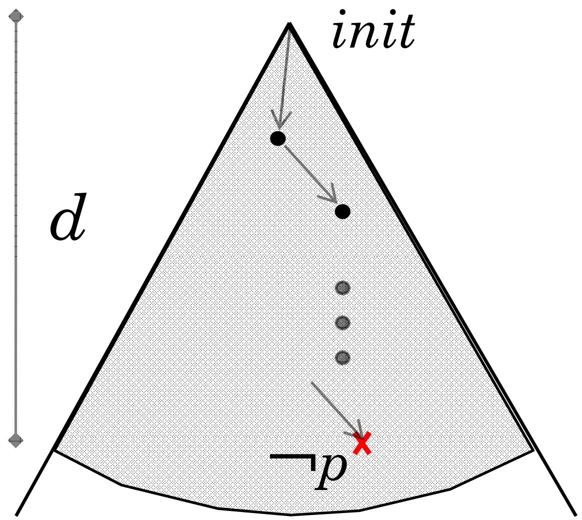
- Since all lemmas conjectured are necessary wrt the MP, we conclude that the SM does not enjoy the MP.
- A counterexample can be constructed by looking at the chain of lemma conjectures up to `inv6`.



- 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.

BMC (cont.)

- The search functionality can be used to conduct BMC:

  `red` $init$ `=(`n`,`d`)=>*` $pattern$ `suchThat` $cond$ `.`

- By setting $init$ to an initial state of a state machine and expressing $\neg p$ in $pattern$ & $cond$.
- To use this functionality, (state) transitions should be described in transition rules.

Transitions in Transition Rules (cont.)

- Configuration of states:

```
op void : -> Sys {constr}
op _ _ : Sys Sys -> Sys
         {constr assoc comm id: void}
```

- Operators that hold values characterizing states:

```
op (pc[_]:_) : Pid Label -> Obs {constr}
op locked:_ : Bool -> Obs {constr}
```

- If two processes p1 & p2 participate in the protocol, the initial state is expressed as

```
(pc[p1]: rs) (pc[p2]: rs) (locked: false)
```

Transitions in Transition Rules (cont.)

```
trans [try] : (pc[I]: rs) (locked: false)
  => (pc[I]: es) (locked: false) .

trans [enter] : (pc[I]: es) (locked: B)
  => (pc[I]: cs) (locked: true) .

trans [exit] : (pc[I]: cs) (locked: B)
  => (pc[I]: rs) (locked: false) .
```

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

Falsification of the MP for FMP by BMC

- When we have two processes, a counterexample (CX) for MP is found with the search functionality.

```
red init =(1,*)
  =>* (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.

```
red init =(1,4)
  =>* (pc[I]: cs) (pc[J]: cs) S .
```

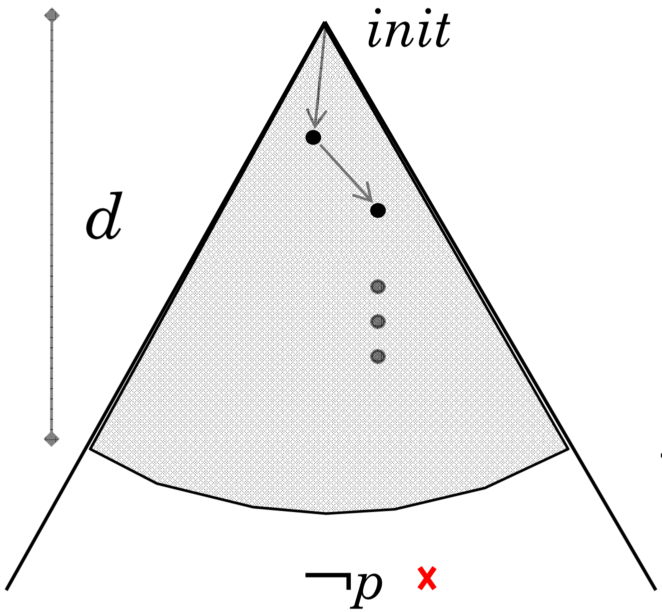
- But, it is not found by exhaustively traversing the BRSS up to depth 3.

```
red init =(1,3)
  =>* (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) ?

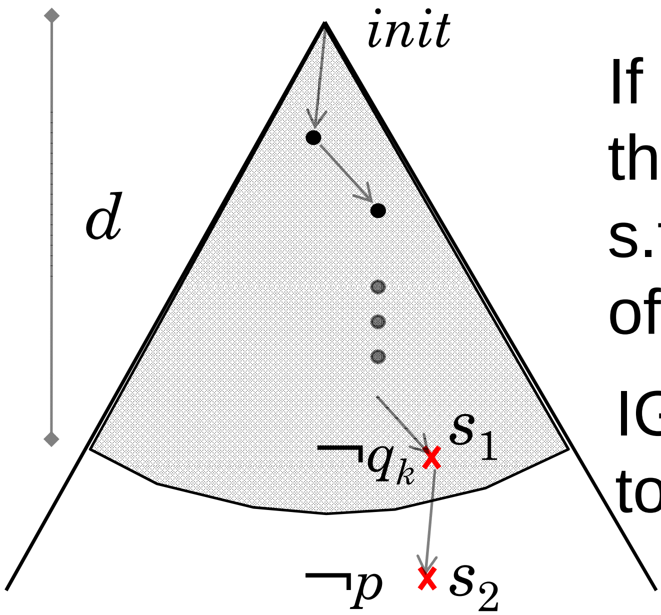


One option is to increase d.
But, the BRSS up to d+1 may not be exhaustively traversed due to the state explosion problem.

A CX that exists outside of the BRSS that can be exhaustively traversed is called a *deep CX* in the talk.

IGF (cont.)

- Another option is to try to prove $\Box p$ by induction, conjecturing lemmas $\Box q_1,\ldots,\Box q_n$, and check the bounded reachable state space for each $\Box q_i$ instead of $\Box p$.



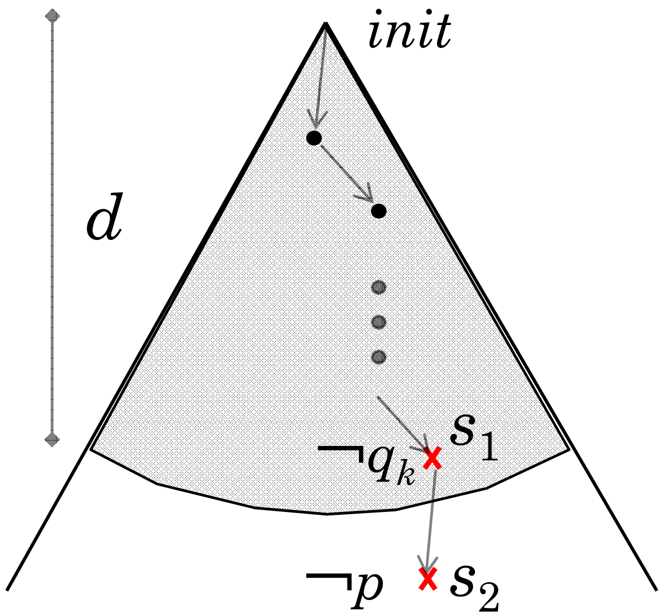
If there exists a state s_1 s.t. $\neg q_k$ and there exists a path from s_1 to a state s_2 s.t. $\neg p$, then we find a counterexample of $\Box p$.

IGF alternately uses BMC and induction to find deep counterexamples.

K. Ogata, M. Nakano, W. Kong, K. Futatsugi: Induction-Guided Falsification, 8th ICFEM, LNCS 4260, Springer, pp.114-131 (2006).

IGF (cont.)

- 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 reachabiity 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.

$$\underset{p}{s} \xrightarrow{t} \underset{p\ (?)}{s'}$$

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.

K. Ogata, K. Futatsugi: A combination of Forward & Backward Reachability Analysis Methods, 12th ICFEM, LNCS 6447, Springer, pp.501-517 (2010).

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 `inv2`.

```
eq inv2(S,I,J)
   = not(pc(S,I) = es and pc(S,J) = cs
         and not(I = J)) .
```

Falsification of FMP by IGF (cont.)

- BMC finds a counterexample for `inv2` in the BRSS up to depth 3.

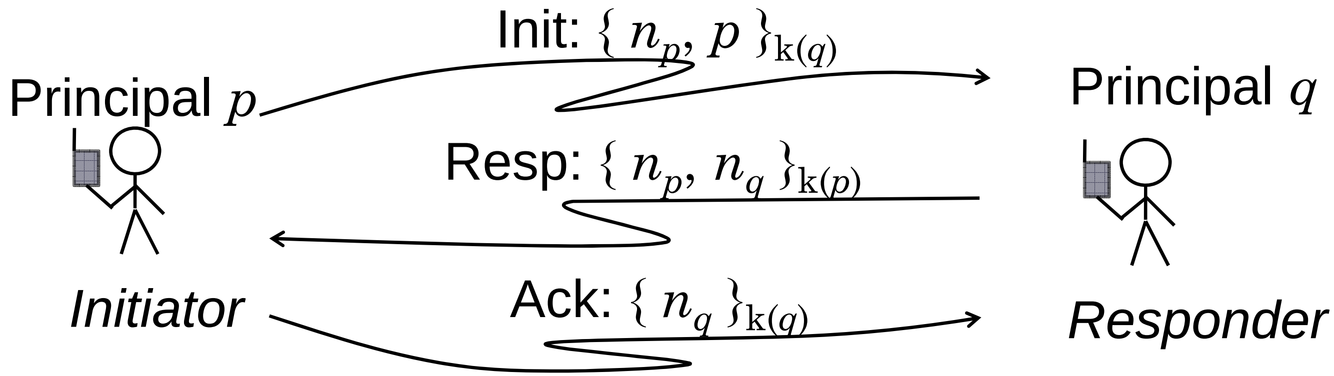
```
red init =(1,3)
  =>* (pc[I]: es) (pc[J]: cs) S .
```

- Since `inv2` is a necessary lemma of the MP, we conclude that the SM does not enjoy the MP.

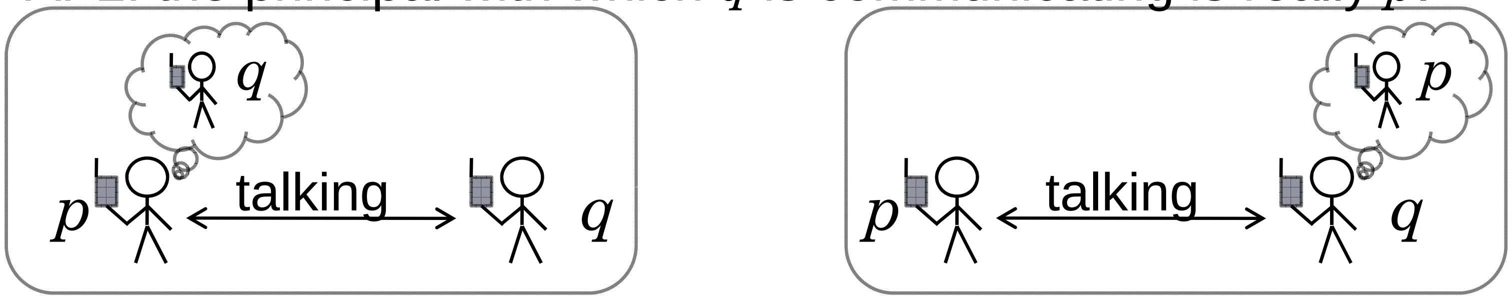
- An example: a flawed mutual exclusion protocol (FMP)
- Specification of the protocol in CafeOBJ
- Falsification of FMP with induction (proof scores)
- Falsification of FMP with (bounded) model checking (search)
- Falsification of FMP with induction-guided falsification (IGF)
- Falsification of NSPK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

NSPK & Agreement Property

- NSPK ([Needham&Schroeder 1978]):



- 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.

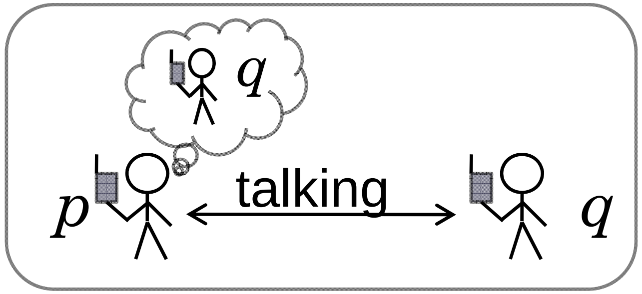


Model Checking AP1 & AP2

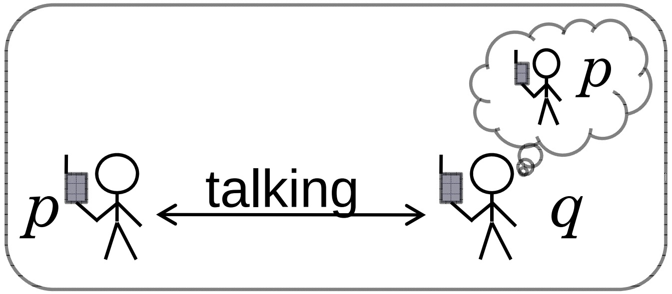
Init:	$p \to q$	$\{n_p, p\}_{\mathrm{k}(q)}$
Resp:	$q \to p$	$\{n_p, n_q, q\}_{\mathrm{k}(p)}$
Ack:	$p \to q$	$\{n_q\}_{\mathrm{k}(q)}$

- The bounded reachable state space (BRSS) up to depth 5 can be exhaustively traversed on a laptop with 2.33GH 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.

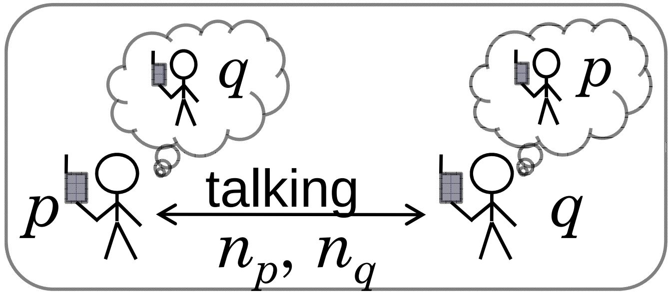


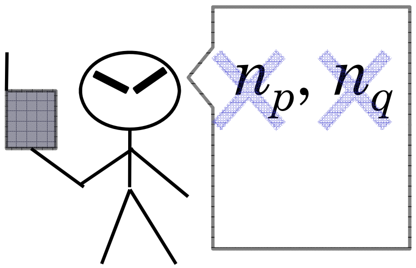
Lemmas for AP1 & AP2

Init:	$p \rightarrow q$	$\{n_p,p\}_{\mathrm{k}(q)}$
Resp:	$q \rightarrow p$	$\{n_p,n_q,q\}_{\mathrm{k}(p)}$
Ack:	$p \rightarrow q$	$\{n_q\}_{\mathrm{k}(q)}$

- 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

Init:	$p \rightarrow q$	$\{n_p, p\}_{\mathrm{k}(q)}$
Resp:	$q \rightarrow p$	$\{n_p, n_q, q\}_{\mathrm{k}(p)}$
Ack:	$p \rightarrow q$	$\{n_q\}_{\mathrm{k}(q)}$

- 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.

K. Ogata, K. Futatsugi: A combination of Forward & Backward Reachability Analysis Methods, 12th ICFEM, LNCS 6447, Springer, pp.501-517 (2010).

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!