Analysis of Electronic Commerce Protocols in Algebraic Specification Languages *

Kazuhiro Ogata[1]

In collaboration with *Kokichi Futatsugi*[1]

[1] School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)

*Presented at JAIST 21st Century COE Symposium on Verifiable and Evolvable e-Society, Sep 06-07, 2007, Tamachi

An Experience on Interactive Theorem Proving

- We tried verifying that an e-commerce protocol satisfied a property with CafeOBJ, an alg spec lang & system, which was used as an interactive theorem prover.
- In the course of the verification (in which a few different modles of the protocol were made), we happend to notice a counterexample showing that the protocol does not satisfy the property.

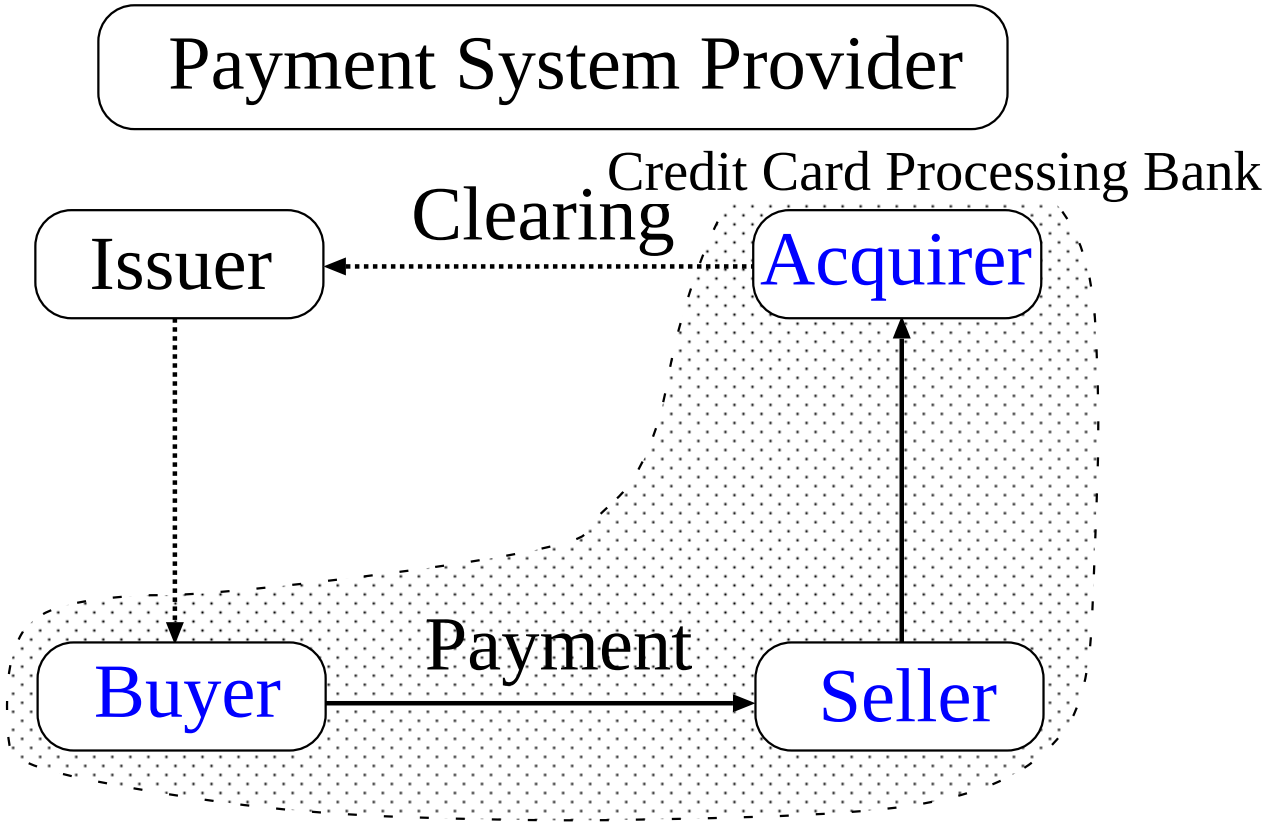
  It took *a couple of weeks* for us to happen to notice it!
- We had wondered whether such a counterexample is able to be found systematically and quickly.

  We conducted a case study in which the e-commerce protocol was *model checked* to find a counterexample showing that it does not satisfy the property.

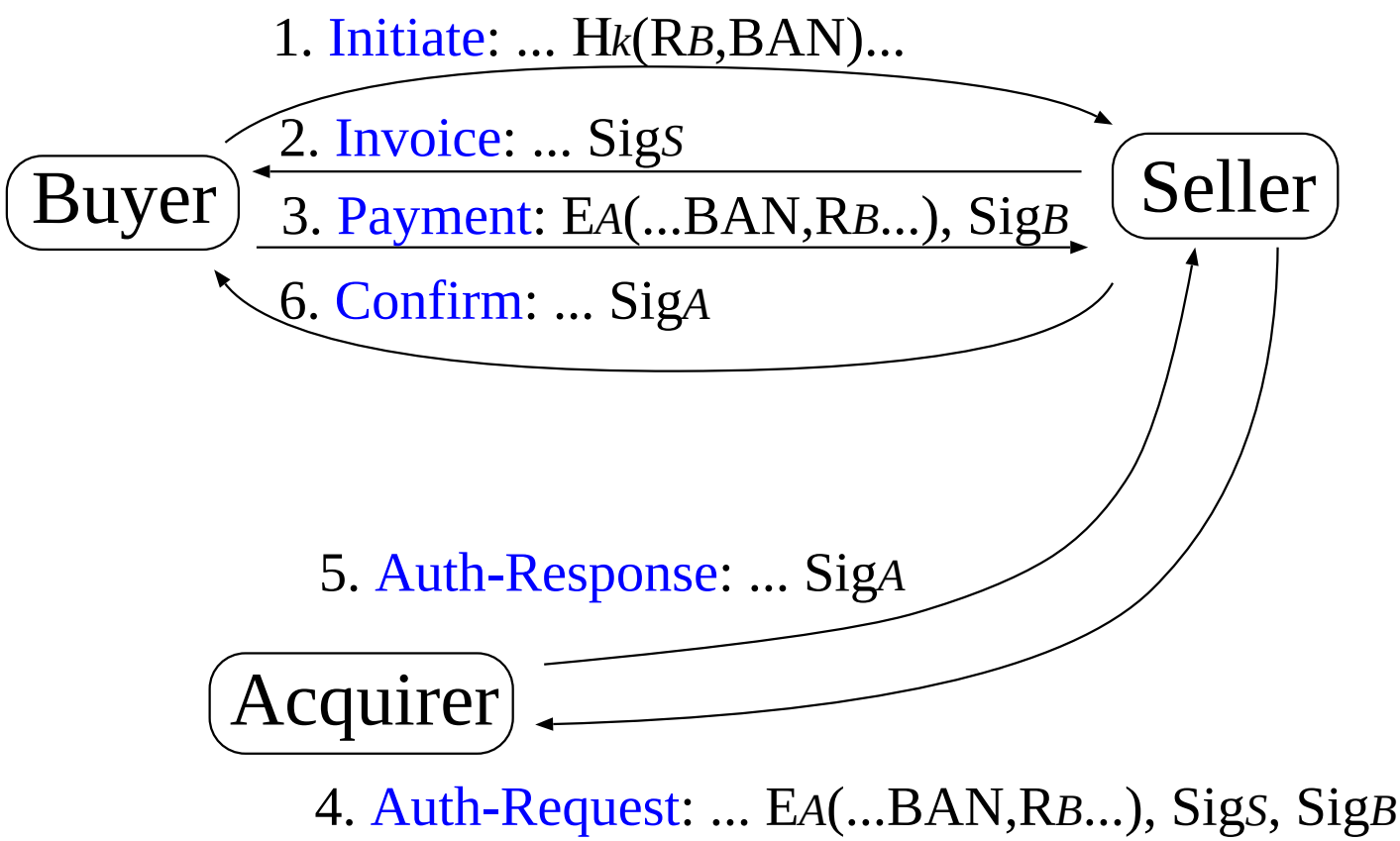
Roadmap

We report on the case study in which the e-commerce protocol was model checked to find a counterexample showing that it does not satisfy the property.

Generic Model of Payment System



Overview of 3KP



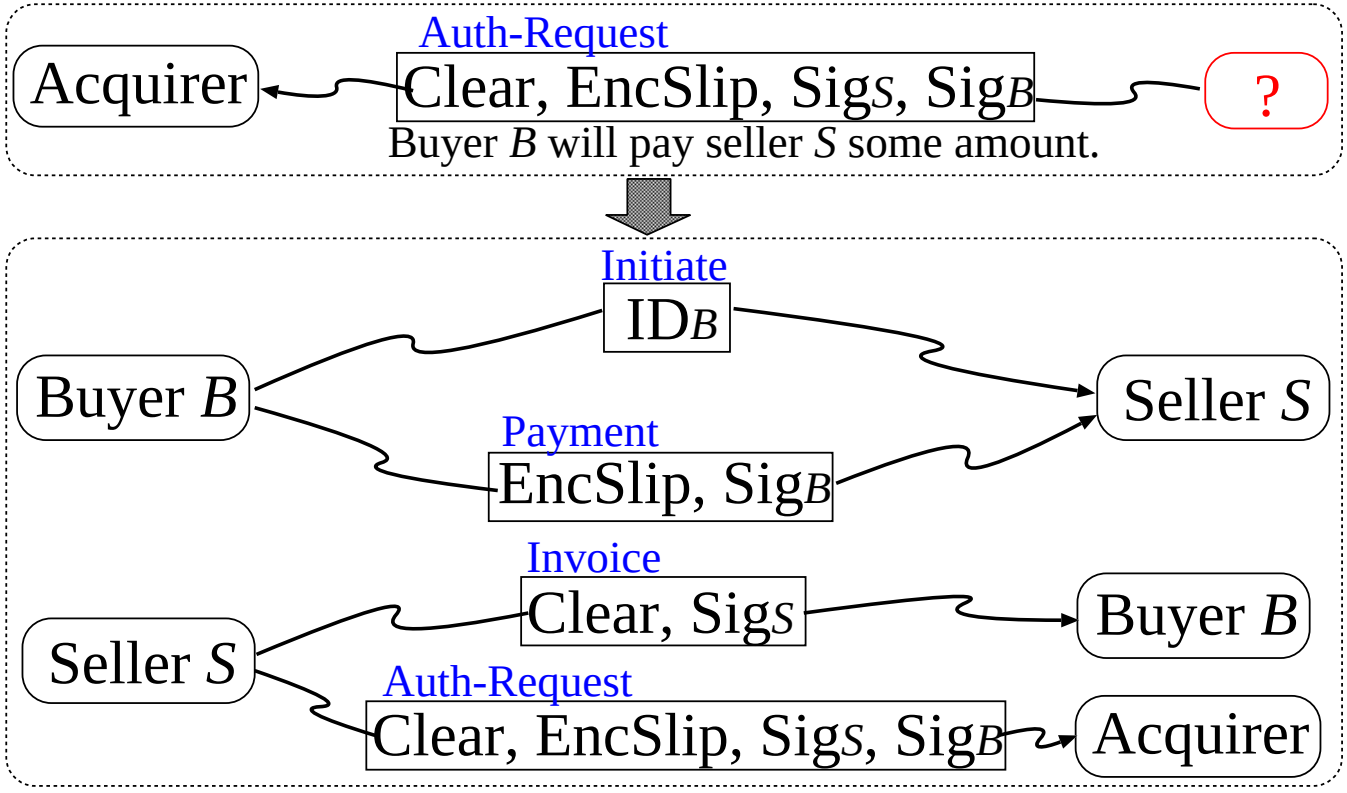
Traditional Description of 3KP

Initiate	$B \rightarrow S : \ \mathrm{ID}_B$
Invoice	$S \rightarrow B : \ \mathrm{Clear}, \mathrm{Sig}_S$
Payment	$B \rightarrow S : \ \mathrm{EncSlip}, \mathrm{Sig}_B$
Auth-Request	$S \rightarrow A : \ \mathrm{Clear}, \mathrm{EncSlip}, \mathrm{Sig}_S, \mathrm{Sig}_B$
Auth-Response	$A \rightarrow S : \ \mathrm{RESPCODE}, \mathrm{Sig}_A$
Confirm	$S \rightarrow B : \ \mathrm{RESPCODE}, \mathrm{Sig}_A$

- $\mathrm{ID}_B : \ \mathcal{H}_k(\mathrm{R}_B, \mathrm{BAN})$
- $\mathrm{Common} : \ \mathrm{PRICE}, \mathrm{ID}_S, \mathrm{NONCE}_S, \mathrm{ID}_B$
- $\mathrm{Clear} : \ \mathrm{ID}_S, \mathrm{NONCE}_S, \mathcal{H}(\mathrm{Common})$
- $\mathrm{Slip} : \ \mathrm{PRICE}, \mathcal{H}(\mathrm{Common}), \mathrm{BAN}, \mathrm{R}_B$
- $\mathrm{EncSlip} : \ \mathcal{E}_A(\mathrm{SLIP})$
- $\mathrm{Sig}_A : \ \mathcal{S}_A(\mathrm{RESPCODE}, \mathcal{H}(\mathrm{Common}))$
- $\mathrm{Sig}_S : \ \mathcal{S}_S(\mathcal{H}(\mathrm{Common}))$
- $\mathrm{Sig}_B : \ \mathcal{S}_B(\mathrm{EncSlip}, \mathcal{H}(\mathrm{Common})))$

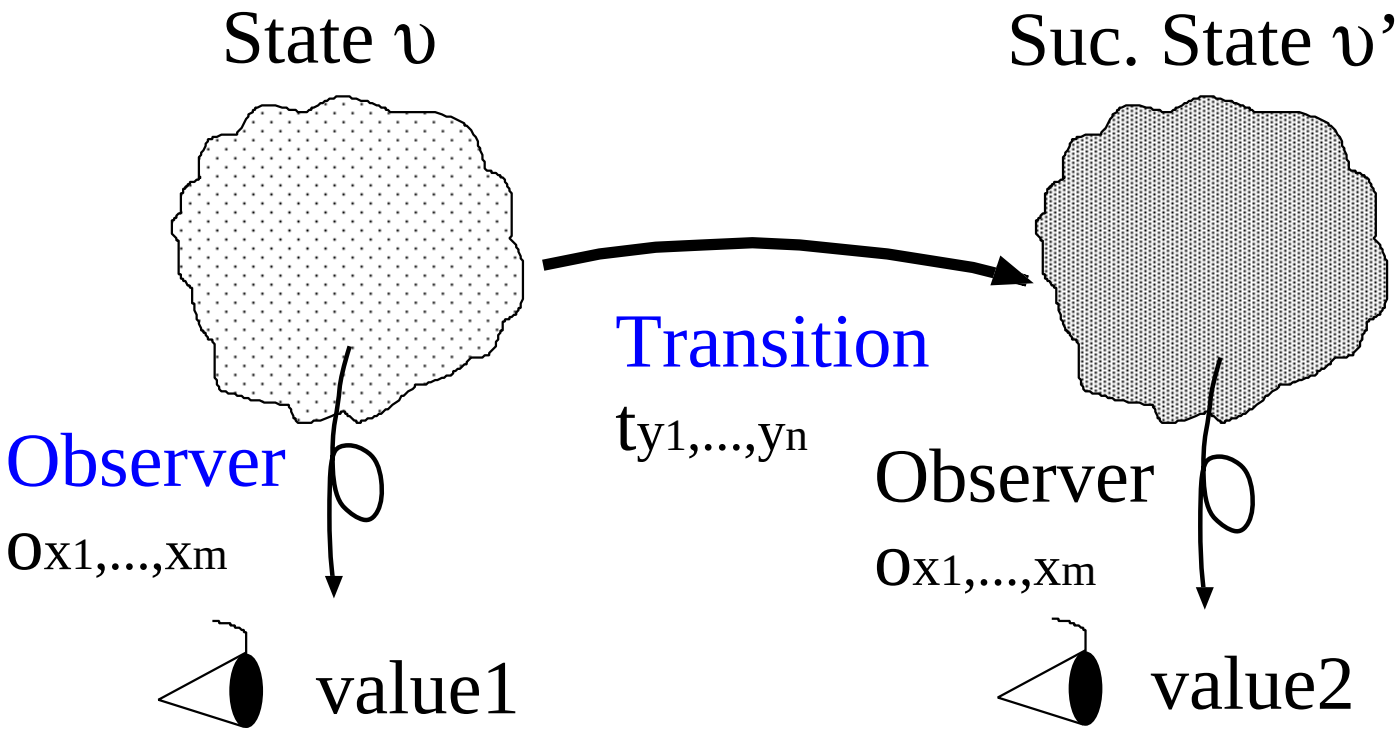
Payment Agreement Property

Whenever an acquirer authorizes a payment,
both the buyer and seller concerned agree on it.



Informal Description of OTSs

OTSs are transition systems.



Definition of OTSs

Suppose a *universal state space* Υ and data types D_* used in OTSs.

An OTS $\mathcal{S}$ is $\langle \mathcal{O}, \mathcal{I}, \mathcal{T} \rangle$ such that

- $\mathcal{O}$: A finite set of observers.

  Each *observer* is an indexed function $o_{x_1:D_{o1},\ldots,x_m:D_{om}} : \Upsilon \rightarrow D_o$.

- $\mathcal{I}$: The set of initial states such that $\mathcal{I} \subseteq \Upsilon$.

- $\mathcal{T}$: A finite set of transitions.

  Each *transition* is an indexed function $t_{y_1:D_{t1},\ldots,y_n:D_{tn}} : \Upsilon \rightarrow \Upsilon$.

  Each t has the condition $c\text{-}t$ called the *effective condition*.

  If $c\text{-}t_{y_1,\ldots,y_n}(\upsilon)$ does not hold, then $t_{y_1,\ldots,y_n}(\upsilon) =_{\mathcal{S}} \upsilon$.

Assumptions

- There exists one genuine acquirer (ga) who is known by every principal.
- Ciphertexts such as $\mathcal{E}_A(\text{SLIP})$ can be never decrypted and signatures such as $\mathcal{S}_A(\text{RESPCODE}, \mathcal{H}(\text{Common}))$ can be never made unless the corresponding private keys such as A's private key are known.
- Secret information such as BANs is never guessable.
- There exists the general intruder who acts as a buyer (ib), a seller (is) and an acquirer (ia). What the intruder can do is
  - To look at every message in the network.
  - To glean the quantities obtained from such messages.
  - To fake messages based on the gleaned quantities.

Formalizing Quantities & Composite Fields

For example,

- Hban : Keyed hashed BANs. hban(r, bn) denotes $\mathcal{H}_k(r, bn)$.
- Common : Commons. com(p, s, n, hbn) denotes the Common that consists of p, s, n and hbn.
- Hcom : Hashed Commons. hcom(c) denotes $\mathcal{H}(c)$.
- Eslip : EncSlips. esl(a, sl) denotes $\mathcal{E}_a(sl)$.
- SigA : Acquirers' signatures. siga(a, rc, hc) denotes $\mathcal{S}_a(rc, hc)$.

Formalizing Messages

Initiate – $\mathrm{im}(b1, b, s, hbn)$

Invoice – $\mathrm{vm}(s1, s, b, cl, gs)$

Invoice (cl, gs) that has been sent by $s1$ to b, but seems to have been sent by s, which may be different from $s1$.

Payment – $\mathrm{pm}(b1, b, s, esl, gb)$

Auth-Request – $\mathrm{qm}(s1, s, a, esl, gb)$

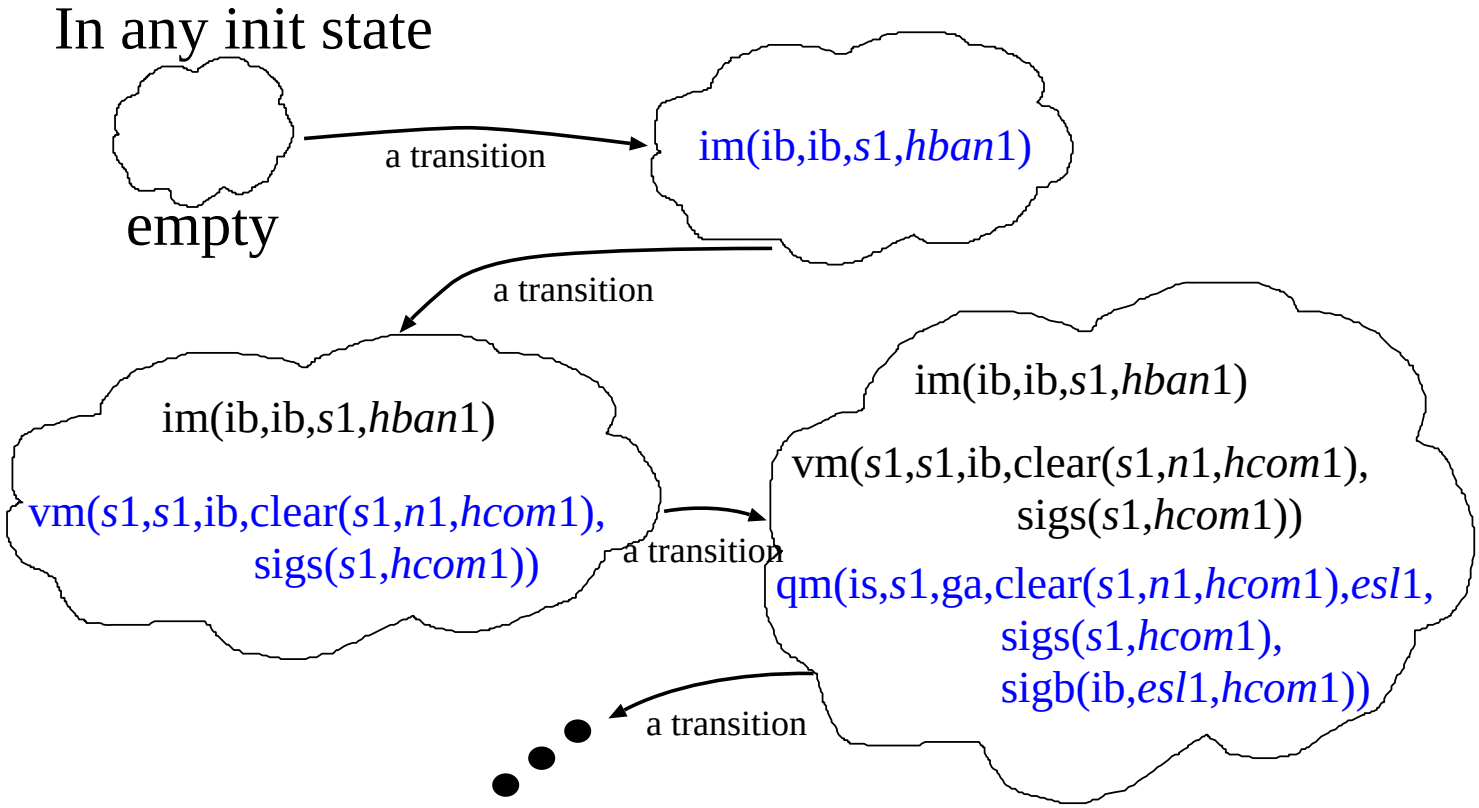
Auth-Response – $\mathrm{sm}(a1, a, s, cl, esl, gs, gb)$

Confirm – $\mathrm{cm}(s1, s, b, rc, ga)$

- 1st arg: *The principal who has actually sent the message.*
- 2nd arg: The principal who seems to have sent the message.
- 3rd arg: The principal who is supposed to receive the message.
- 4th arg and subsequent ones: The body of the message.

Formalizing Networks

The network is denoted by a collection of messages that has been sent.



$\mathcal{S}_{3KP}$ (1)

$\mathcal{O}_{3KP}$ contains

- rand : $\Upsilon \rightarrow$ Rand
- nonce : $\Upsilon \rightarrow$ Nonce
- nw : $\Upsilon \rightarrow$ Network
- nonces : $\Upsilon \rightarrow$ NonceBag
- hbans : $\Upsilon \rightarrow$ HbanBag
- hcoms : $\Upsilon \rightarrow$ HcomBag
- bans : $\Upsilon \rightarrow$ BanBag
- rands : $\Upsilon \rightarrow$ RandBag
- eslips : $\Upsilon \rightarrow$ EslipBag
- sigas : $\Upsilon \rightarrow$ SigABag
- sigss : $\Upsilon \rightarrow$ SigSBag
- sigbs : $\Upsilon \rightarrow$ SigSBag

The collections of gleaned quantities.

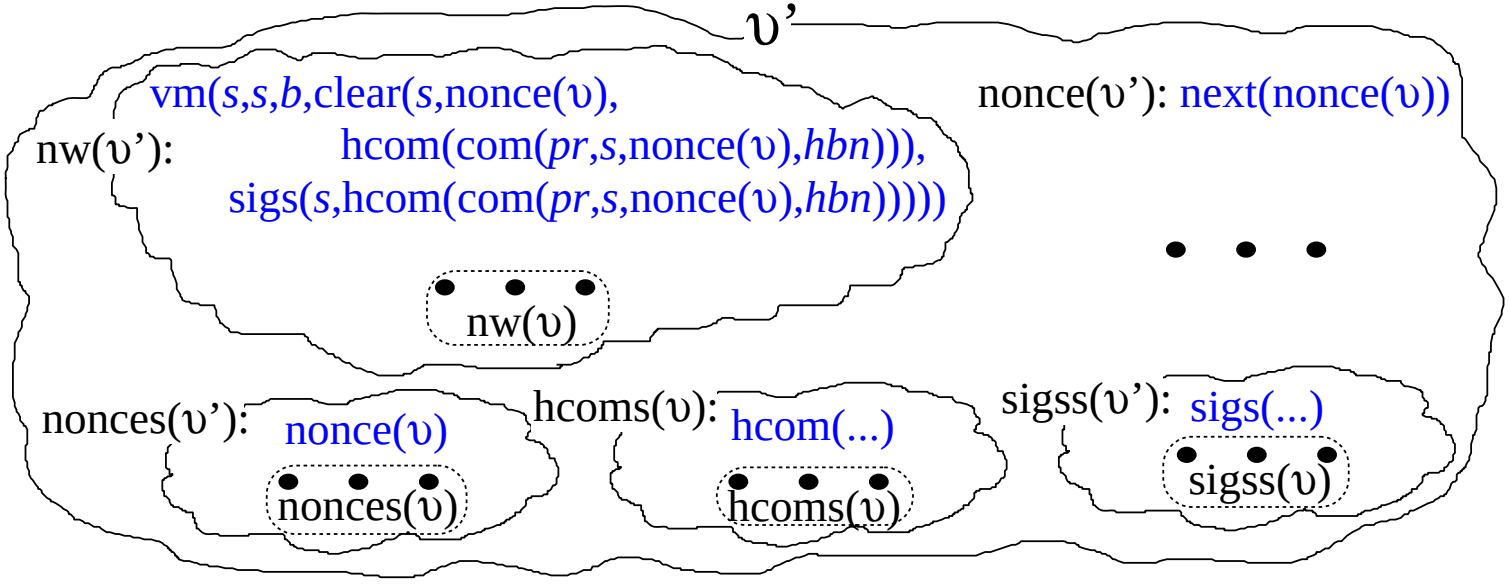
For each $\upsilon_0 \in \mathcal{I}_{3KP}$,

- rand(υ_0) = seed
- nonce(υ_0) = in
- nw(υ_0) = empty
- nonces(υ_0) = empty
- hbans(υ_0) = empty
- hcoms(υ_0) = empty
- bans(υ_0) = empty
- rands(υ_0) = empty
- eslips(υ_0) = empty
- sigas(υ_0) = empty
- sigss(υ_0) = empty
- sigbs(υ_0) = empty

$\mathcal{S}_{3KP}$ (2)

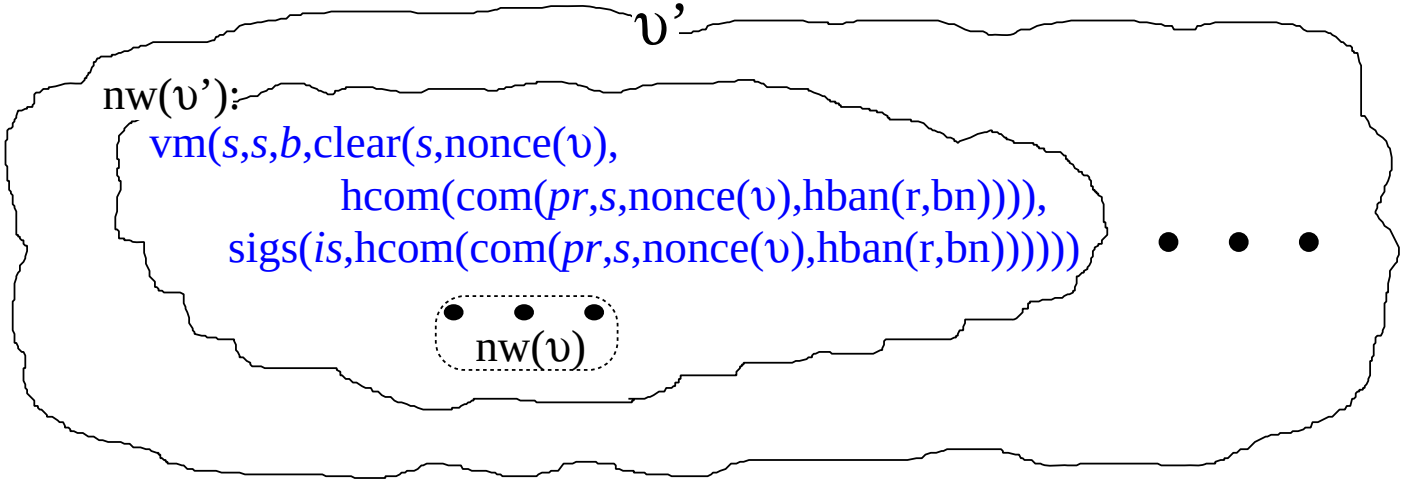
$\mathcal{T}_{3KP}$ contains 43 transitions: 6 for sending messages exactly following the protocol and 37 for faking messages based on the gleaned information.

- $\text{sdvm}_{s,b,b1,hbn,pr} : \Upsilon \rightarrow \Upsilon$: Let υ' be $\text{sdvm}_{s,b,b1,hbn,pr}(\upsilon)$.
  $\text{c-sdvm}_{s,b,b1,hbn,pr}(\upsilon) \triangleq \text{im}(b1, b, s, hbn) \in \text{nw}(\upsilon)$



$\mathcal{S}_{3KP}$ (3)

- $\text{fkvm5}_{s,b,n,pr,r,bn} : \Upsilon \rightarrow \Upsilon$: Let υ' be $\text{sdvm}_{s,b,b1,hbn,pr}(\upsilon)$.
  $\text{c-fkvm5}_{s,b,n,pr,r,bn} \triangleq n \in \text{nonces}(\upsilon) \wedge r \in \text{rands}(\upsilon) \wedge bn \in \text{bans}(\upsilon)$



Maude

- Data & state machines.
  - Data are specified in membership equational logic.
  - State machines are specified in rewriting logic.
- Fast rewrite engine & flexible meta-programming environment.
- Model checking facilities.

  Inductive types can be used. Entire state spaces do not have to be finite.
  - On-the-fly explicit state LTL model checker.
  - Search command.

    Even reachable state spaces do not have to be finite; BMC can be performed.

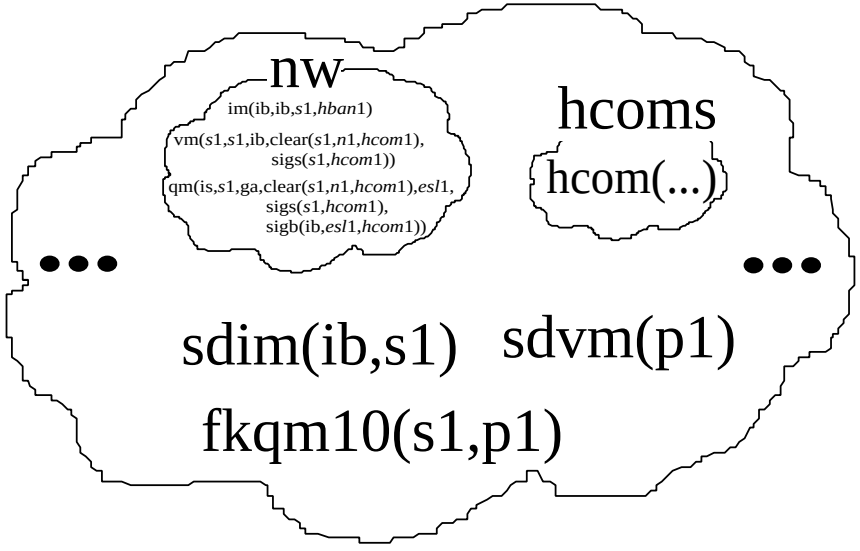
    It can be used to find counterexamples showing that state machines do not satisfy invariant properties.

Overview of Specification

Observers & transitions are denoted by operators. For example,

```
op nw :_ : Network -> Observer .
op sdvm : Price -> Transition .
```

States are denoted by collections of terms whose sorts are `Observer` or `Transition`.



Definitions of Transitions

Transitions are defined in rewriting rules that change collections of terms whose sorts are `Observers` or `Transitions`.

For example, the rule defining $\mathrm{sdvm}_{s,b,b1,hbn,pr} : \Upsilon \rightarrow \Upsilon$:

```
rl [rule-sdvm] : sdvm(PR) (nonce : N) (nw : (im(B1,B,S,HBN) NW))
   (nonces : Ns) (hcoms : HCs) (sigss : GSs)
   => sdvm(PR) (nonce : next(N))
      (nw : (vm(S,S,B,clear(S,N,hcom(com(PR,S,N,HBN))),
                        sigs(S,hcom(com(PR,S,N,HBN))))
              im(B1,B,S,HBN) NW))
      (nonces : (N Ns)) (hcoms : (hcom(com(PR,S,N,HBN)) HCs))
      (sigss : (sigs(S,hcom(com(PR,S,N,HBN))) GSs)) .
```

Search Command to Find a Counterexample

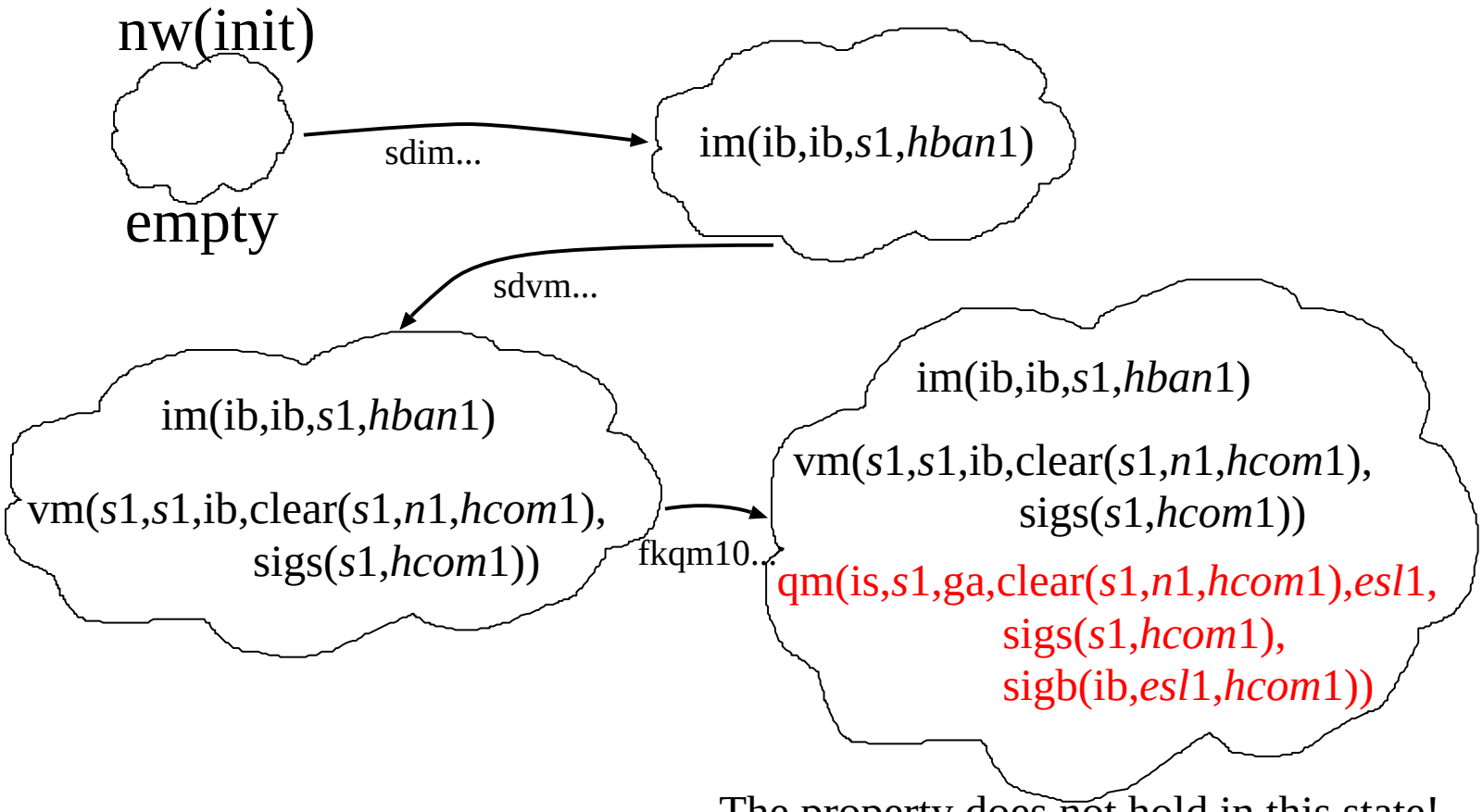
The search command to find a counterexample showing that $\mathcal{S}_{3KP}$ does not satisfy the payment agreement property looks like

```
search [1,10] init =>* (nw : (qm(S1,S,ga,...) NW)) Prot
  such that not (not(S == is and B == ib) implies
    im(B,B,S,...) \in NW and vm(S,S,B,...) \in NW and
    pm(B,B,S,...) \in NW and qm(S,S,ga,...) \in NW) .
```

The constant `init` represents an initial state where there are one seller, one buyer, the genuine acquirer and the intruder.

Watch a demo!

Counterexample



The property does not hold in this state!

Summary

- We reported on the case study showing that 3KP does not satisfy the payment agreement property by model checking $\mathcal{S}_{\mathrm{3KP}}$ with Maude.
- It took about 170ms for the search command to find the counterexample, while it took a couple of weeks for us to happen to find it.

Future & Ongoing Work

- What if the bounded reachable state space of $\mathcal{S}_{3KP}$ whose depths are at most 3 was too large to be traversed within a reasonable time?

  We have shown that mathematical induction can alleviate the problem – *Induction-Guided Falsification* (*IGF*).

- We have been developing a methodology that uses a model checker to support interactive theorem proving.

  The methodology needs some tools, one of which is a translator from CafeOBJ specifications of OTSs into Maude specifications of OTSs.

  We have been redesigning the translator for larger examples.

- *Creme*, an automatic invariant prover, will be used to verify that the modified 3KP satisfies the property.