

Title	A Toolkit for Generating and Displaying Proof Scores in the OTS/CafeOBJ Method
Author(s)	Seino, Takahiro; Ogata, Kazuhiro; Futatsugi, Kokichi
Citation	
Issue Date	2009-09-22
Type	Presentation
Text version	publisher
URL	http://hdl.handle.net/10119/8331
Rights	
Description	1st VERITE : JAIST/TRUST-AIST/CVS joint workshop on VERification TEchnologyでの発表資料, 開催 : 2005年9月21日 ~ 22日, 開催場所 : 金沢市文化ホール 3F

AIST/JAIST joint workshop on verification technology

**A Toolkit for Generating and
Displaying Proof Scores
in the OTS/CafeOBJ Method**

Takahiro Seino, Kazuhiro Ogata and Kokichi Futatsugi

School of Information Science

Japan Advanced Institute of Science and Technology

Background

Formal Methods

- effective for systems are built safely and reliably.

The OTS/CafeOBJ method [Ogata 2003-]

- can model distributed systems as transition systems called OTS (Observational Transition Systems)
- can describe the system in CafeOBJ which is an algebraic specification language
- can verify that the system has invariant properties by induction on number of transition rules applied.
- easy to learn for ordinary engineers
 - based on (one-way) equational reasoning

Problem

Verification in the OTS/CafeOBJ method

Hundreds or thousands lines code

Base case

proof passage

Inductive step for Transition₁

Case splitting with pred. p_1

Case: p_1 holds

proof passage

Case: p_1 doesn't hold

proof passage

Inductive step for Transition_n

1. We must write proof score maintaining case splitting
2. We must check each reduced result is the expected term (= true)
 - 👉 human errors may occur.
 - 👉 disturb humans from concentrating on intellectual work.

```
open ISTEP
  op d1 : -> D1 .
  op d2 : -> D2 .
  ...
  eq p1 = true .
  ...
  eq s' = Transition1(s, ...) .
  red SIH implies istep(...) .
close
```

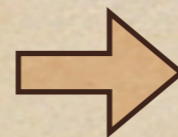
Our solution

Generating and checking proof scores

We must specify to generate proof scores:

1. predicate to be proven
2. list of predicates to be used in case splitting
3. list of lemmas to be strengthen induction hypothesis

```
mod PROOF-SCRIPT {  
  op  $d_1$  :  $\rightarrow D_1$  .  
  op  $d_2$  :  $\rightarrow D_2$  .  
  ...  
  eq basecase = inv(...) .  
  eq inductive = istep(...) .  
  trans predicates(Transition1(S)) =  $p_1$  .  
  ...  
  trans lemmas(Transition1(S)) =  $inv_1$  .  
  ...  
}
```



Base case

proof passage

Inductive step for $Transition_1$

Case splitting with pred. p_1

Case: p_1 holds

proof passage

Case: p_1 doesn't hold

proof passage

Inductive step for $Transition_n$

Generated, Checked
Display hierarchically

CASE tool platform

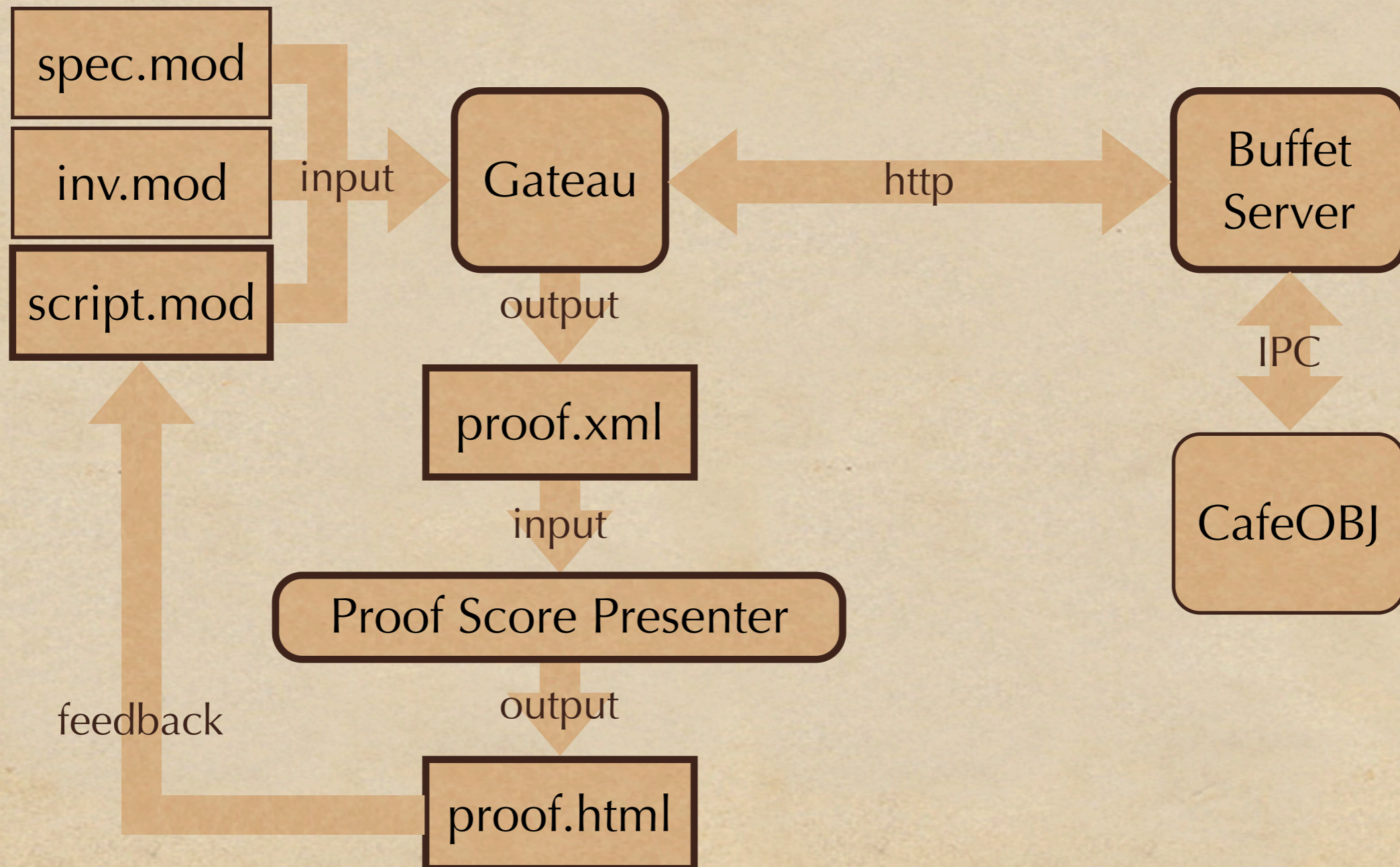
We propose a CASE tool platform CafeOBJ/XML

- based on XML technology
- has a syntax corresponding to abstract syntax of CafeOBJ
- also represents proofs

Design policy of CafeOBJ/XML

- scope: describing specifications and proofs.
- makes implementing CASE tools easier.
- doesn't depend a specific programming language.

Overview of Buffet toolkit



Ex. A Mutual Exclusion

We verify that

```
var lock := false
l1: Remainder Section
l2: repeat until  $\neg$ (fetch&store(lock, true))
    Critical Section
cs: lock := false
```

has the mutual exclusion property.

Modeling with OTS

Data types:

- $B = \{ \text{true}, \text{false} \}$ Boolean values
- $P = \{ p_1, p_2, \dots \}$ Set of process IDs
- $L = \{ l_1, l_2, \text{cs} \}$ Set of location labels

Note that equivalence relation denoted by '=' for each data type have been defined.

Modeling with OTS

Universal state space: Υ

set of **Observers** = $\{ o : \Upsilon \rightarrow D \}$

- $lock : \Upsilon \rightarrow B$
- $loc_p : \Upsilon \rightarrow L$ for $p \in P$

set of **Initial states**

- $\{ s_0 \mid lock(s_0) = \text{false} \wedge \forall p \in P. loc_p(s_0) = |1 \}$

set of **Transitions** = $\{ t : \Upsilon \rightarrow \Upsilon \}$

- $try_p : \Upsilon \rightarrow \Upsilon$ for $p \in P$
- $enter_p : \Upsilon \rightarrow \Upsilon$ for $p \in P$
- $leave_p : \Upsilon \rightarrow \Upsilon$ for $p \in P$

Modeling with OTS

Definition of try_p :

```
var lock := false
l1: Remainder Section
l2: repeat until
    ¬(fetch&store(lock, true))
    Critical Section
cs: lock := false
```

$$c_{try_p}(s) \equiv loc_p(s) = l1$$

$try_p(s')$ where $c_{try_p}(s)$ holds

$$lock(s') = lock(s)$$

$$loc_p(s') = l2$$

$$loc_q(s') = loc_p(s) \quad \text{if } p \neq q$$

where $c_{try_p}(s)$ doesn't holds
nothing changes

Invariants

Execution sequence $\{s_0, s_1, \dots\}$ satisfies:

- s_0 is in the set of initial states
- there exists a transition for each pair of (s_i, s_{i+1})

Reachability

- State s is **reachable**: there exists an execution sequence of an OTS in which s appears.

Invariants

- A predicate p such that $p(s)$ holds for every reachable state s .
- In the ex., $\forall i, j \in P. loc(s, i) = cs \wedge loc(s, j) = cs \Rightarrow i = j$

Describing invariant

Invariant candidates are described:

```
mod INV { pr(OTS-SPEC)
```

```
op inv1 :  $\Upsilon$  ...  $\rightarrow$  Bool  
op inv2 :  $\Upsilon$  ...  $\rightarrow$  Bool  
...
```

Signatures of invariants

```
eq inv1(s :  $\Upsilon$ , ...) = ... .  
eq inv2(s :  $\Upsilon$ , ...) = ... .  
...
```

Invariants denoted by CafeOBJ term

```
mod ISTEP { pr(INV)
```

```
ops s s' :  $\rightarrow$   $\Upsilon$   
op istep1 : ...  $\rightarrow$  Bool  
op istep2 : ...  $\rightarrow$  Bool
```

```
eq istep1(...) = inv1(s, ...) implies inv1(s', ...) .  
eq istep2(...) = inv2(s, ...) implies inv2(s', ...) .
```

Terms denoting reasonings
in the inductive step

Buffet server

Buffet server relays requests/responses between a client to the CafeOBJ system

• we can get the information of already defined/loaded CafeOBJ specification from the CafeOBJ system

inv.mod
script.mod

Gateau

http

Buffet Server

IPC

CafeOBJ

but, it's fragmentary

proof.xml

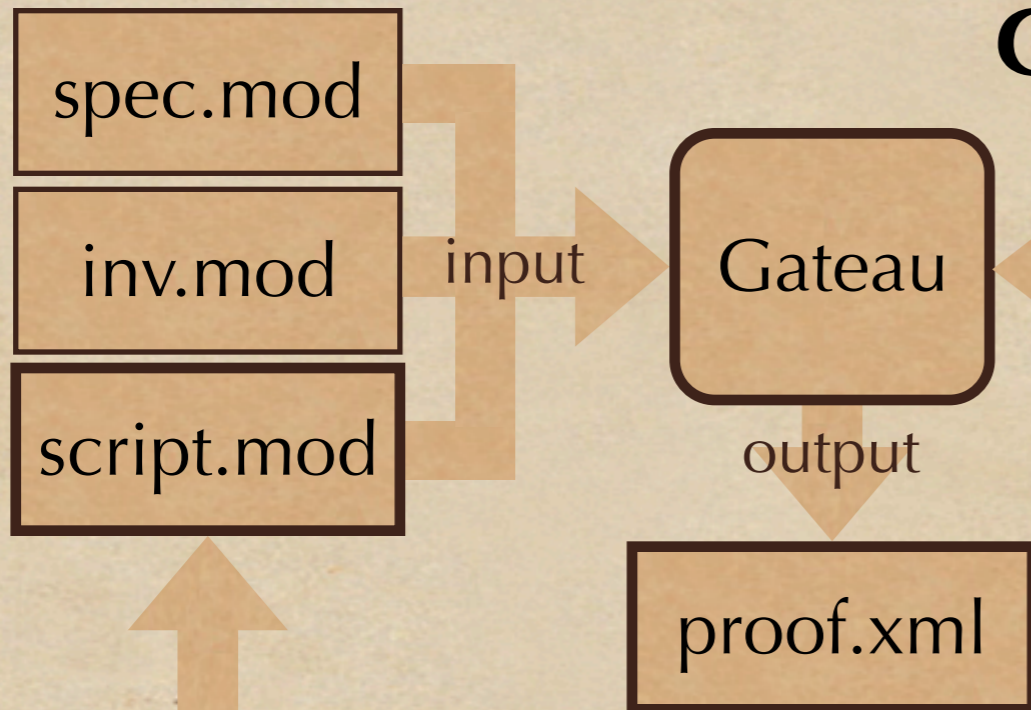
• Buffet server reconstructs the information in an XML document

Proof Score Presenter

feedback

proof.html

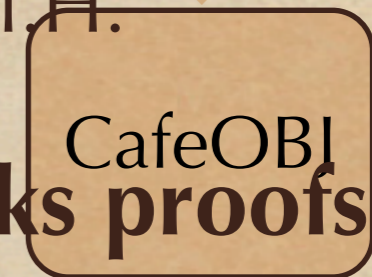
Gateau



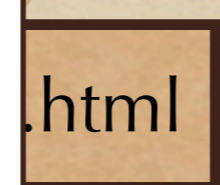
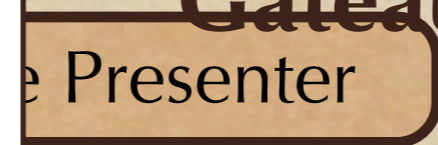
Gateau generates proof scores

• according to

- `http`
- given predicates to use for case splitting
- given lemmas to strengthen I.H.

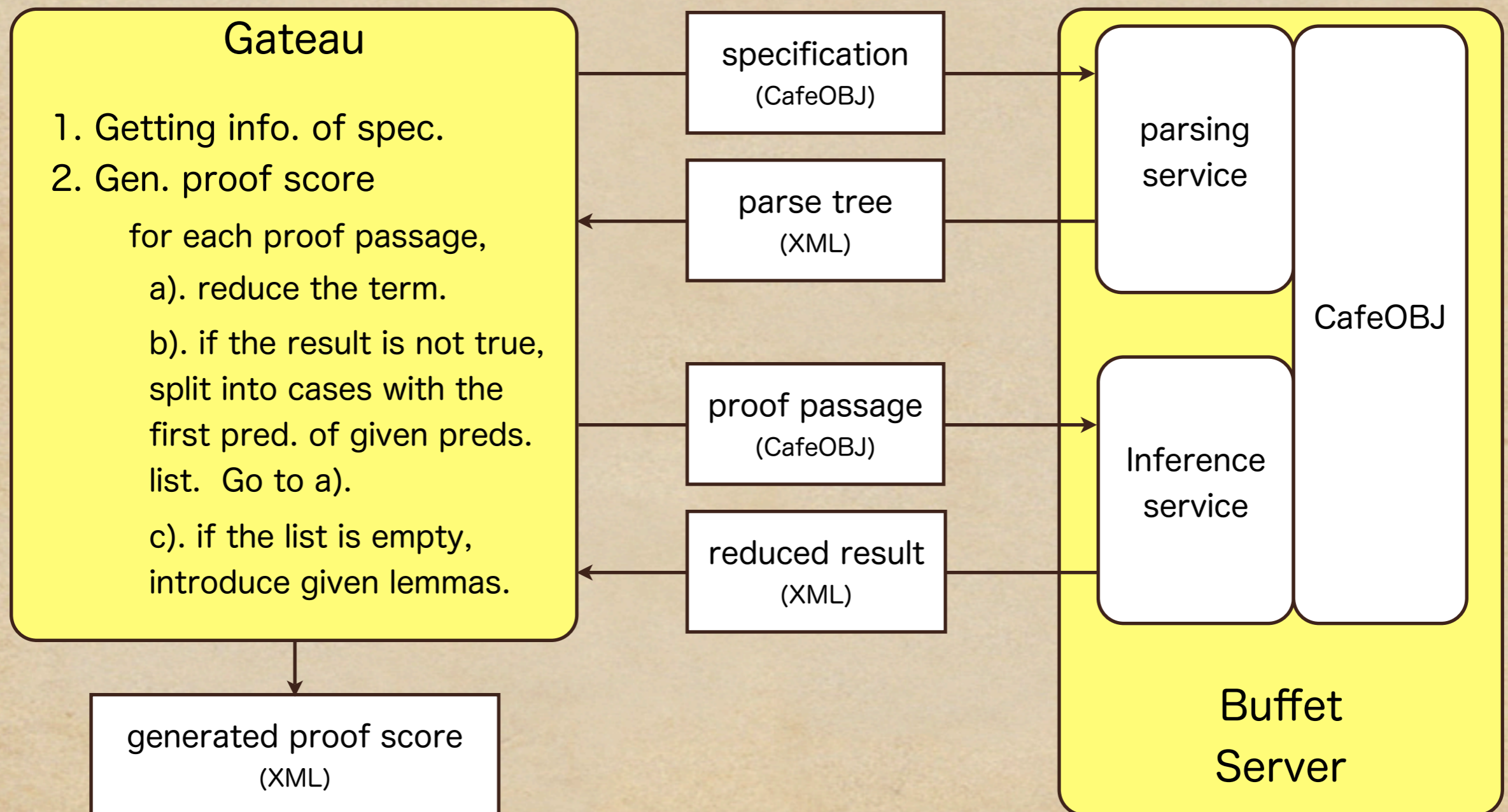


Gateau also checks proofs

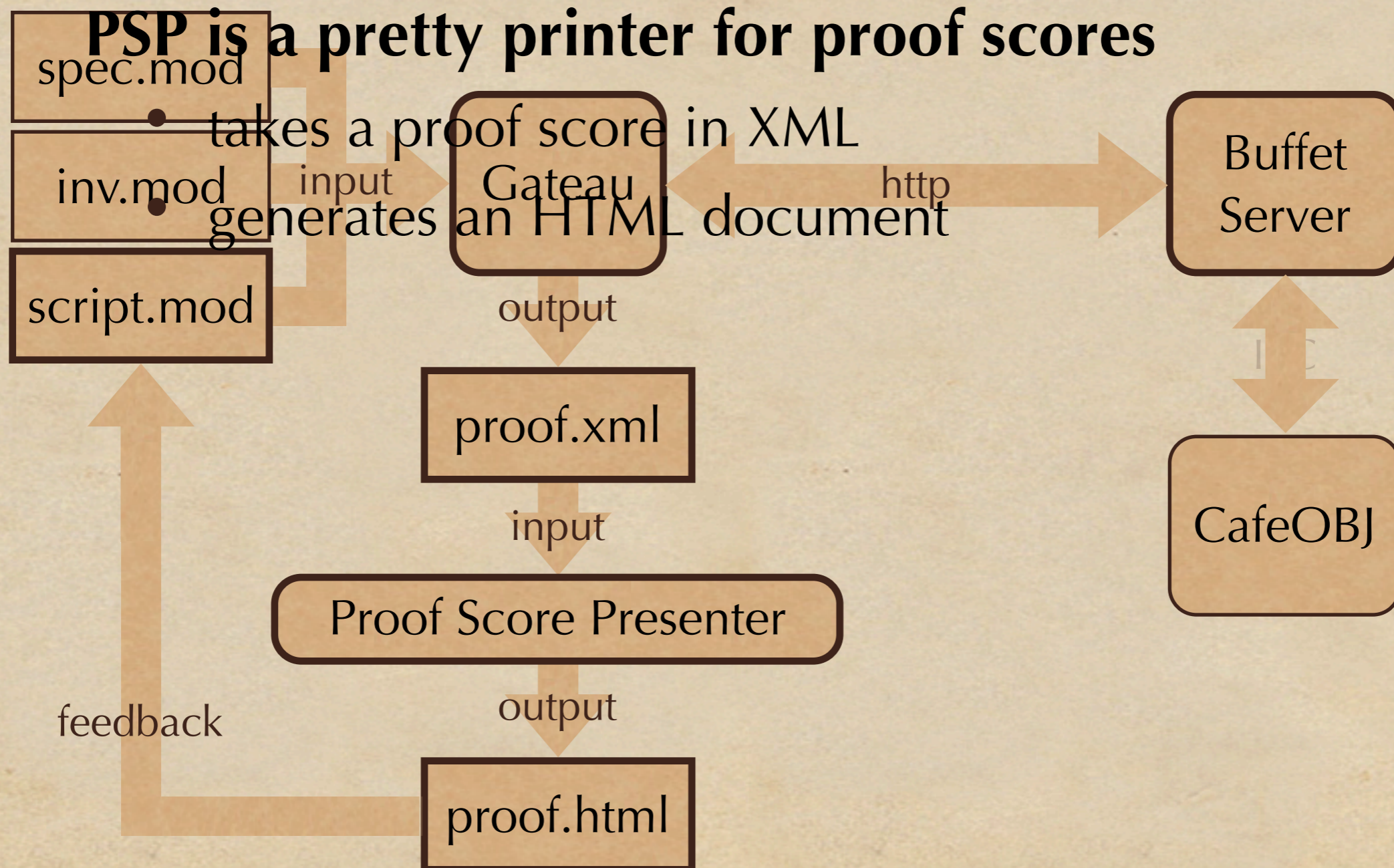


```
open ISTEP
-- arbitrary objects:
op pid1 : -> Pid .
-- assumptions:
eq (loc(s,pid1)) = (l1) .
eq (s') = (try(s,pid1)) .
-- reduce the following term:
red istep1(i, j) .
close
```

How to gen. proof score



Proof Score Presenter



There are 7 cases, 3 cases need human's help.

► base

Hierarchical view
with disclosing triangles

▼ action: try

case splitting: c-try(s, pid1)

▼ case: true

open ISTEP

-- arbitrary objects:

op pid1 : -> Pid .

-- assumptions:

eq (loc(s, pid1)) = (l1) .

eq (s') = (try(s, pid1)) .

-- reduce the following term:

red istep1(i, j) .

close

result:

(((((if (pid1 = i) then l2 else loc(s,i) fi) = cs) and ((if (pid1 = j) then l2
else loc(s,j) fi) = cs)) and ((loc(s,i) = cs) and (loc(s,j) = cs)))

xor ((((((if (pid1 = i) then l2 else loc(s,i) fi) = cs) and ((if (pid1 = j) then l2
else loc(s,j) fi) = cs)) and (((loc(s,i) = cs) and (loc(s,j) = cs)) and (i =
j))))

xor (((if (pid1 = i) then l2 else loc(s,i) fi) = cs) and ((if (pid1 = j) then l2
else loc(s,j) fi) = cs))

xor (((if (pid1 = i) then l2 else loc(s,i) fi) = cs) and (((if (pid1 = j) then l2
else loc(s,j) fi) = cs) and (i = j)))

xor true

Displaying the part of proof scores
for which further case analysis
should be done and/or
lemmas should be used

► case: false

A hidden part of proof scores

▼ action: enter

case splitting: c-enter(s, pid1)

▼ case: true

Other case studies

Otway-Rees authentication protocol

- 1 secrecy property (48 cases)
- 3 lemmas (36-37 cases)

NSLPK authentication protocol

- 1 secrecy property (37 cases)
- 6 lemmas (24-65 cases)

Conclusion

We have implemented the Buffet toolkit

- can generate & check proof scores automatically
 - generated proof scores cover all cases
 - success of proofs depends on given predicates and lemmas
- can display proof scores hierarchically
 - provided views helps the verification
- can be applied including non-trivial problems
 - Simple mutual exclusion
 - NSLPK, and Otway-Rees authentication protocols

Implemented tools

Buffet Server (1,200 lines, in Perl)

Gateau (800 lines, in Perl)

Proof Score Presenter (600 lines, in XSLT)

Eclipse plug-ins (working)

- CafeOBJ Editor (300 lines, in Java)
- Proof Score Viewer (400 lines, in Java)
 - the final goal will be an Interactive Editor for Proof Score

Cafe2Maude (by Kong-san, in Java)

Future plan

Integrating Eclipse

- GUI based implementation (Gateau & PSP)
 - more interactive

More tightly integrating Eclipse

- Test Driven Development
 - Test case generation from proof scores

Demonstration