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 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

- Base case
  - Inductive step for Transition^{1}
    - Case splitting with pred. p_1
      - Case: p_1 holds
      - Case: p_1 doesn’t hold
  - 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.

```plaintext
open ISTEP
op d_1 : -> D_1 .
op d_2 : -> D_2 .
...
eq p_1 = true .
...
eq s' = Transition^{1}(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 d1 : -> D1 .
  op d2 : -> D2 .
  ...
  eq basecase = inv(...) .
  eq inductive = istep(...) .
  trans predicates(Transition1(S)) = p1 .
  ...
  trans lemmas(Transition1(S)) = inv1 .
  ...
}
```
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

- spec.mod
- inv.mod
- script.mod
- proof.xml
- proof.html
- Gateau
- Buffet Server
- CafeOBJ
- input
- output
- feedback
- http
- IPC
Ex. A Mutual Exclusion

We verify that

\begin{align*}
\text{var } lock & := \text{false} \\
\text{l1: Remainder Section} \\
\text{l2: repeat until } \neg (\text{fetch\&store}(lock, \text{true})) \\
\text{Critical Section} \\
\text{cs: } lock & := \text{false}
\end{align*}

has the mutual exclusion property.
Modeling with OTS

Data types:

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

Note that equivalence relation denoted by ‘=’ for each data type have been defined.
Modeling with OTS

Universal state space: \( \mathcal{T} \)

set of Observers = \( \{ o : \mathcal{T} \to D \} \)

- \( \text{lock} : \mathcal{T} \to B \)
- \( \text{loc}_p : \mathcal{T} \to L \) for \( p \in P \)

set of Initial states

- \( \{ s_0 \mid \text{lock}(s_0) = \text{false} \land \forall p \in P.\text{loc}_p(s_0) = 11 \} \)

set of Transitions = \( \{ t : \mathcal{T} \to \mathcal{T} \} \)

- \( \text{try}_p : \mathcal{T} \to \mathcal{T} \) for \( p \in P \)
- \( \text{enter}_p : \mathcal{T} \to \mathcal{T} \) for \( p \in P \)
- \( \text{leave}_p : \mathcal{T} \to \mathcal{T} \) for \( p \in P \)
Modeling with OTS

Definition of \( \text{try}_p \) :

\[ c_{\text{try}_p}(s) \equiv \text{loc}_p(s) = l1 \]

\( \text{try}_p(s') \) where \( c_{\text{try}_p}(s) \) holds

\[ \text{lock}(s') = \text{lock}(s) \]
\[ \text{loc}_p(s') = l2 \]
\[ \text{loc}_q(s') = \text{loc}_p(s) \text{ if } p \neq q \]

where \( c_{\text{try}_p}(s) \) doesn’t hold

nothing changes

\begin{verbatim}
var lock := false
l1: Remainder Section
l2: repeat until ¬(fetch&store(lock, true))
    Critical Section
cs: lock := false
\end{verbatim}
Invariants

Execution sequence \( \{s_0, s_1, \ldots \} \) 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 \textit{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., \( \forall i, j \in P. \text{loc}(s, i) = \text{cs} \land \text{loc}(s, j) = \text{cs} \Rightarrow i = j \)
Describing invariant

Invariant candidates are described:

mod INV { pr(OTS-SPEC)
  op inv₁ : Y ... -> Bool
  op inv₂ : Y ... -> Bool
  ...
  eq inv₁(S: Y,...) = ... .
  eq inv₂(S: Y,...) = ... .
  ...
}

mod ISTEP { pr(INV)
  ops s s' : -> Y
  op istep₁ : ... -> Bool
  op istep₂ : ... -> Bool
  ...
  eq istep₁(...) = inv₁(s,...) implies inv₁(s',...) .
  eq istep₂(...) = inv₂(s,...) implies inv₂(s',...) .
}

Signatures of invariants

Invariants denoted by CafeOBJ term

Terms denoting reasonings in the inductive step
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.
  - But, it’s fragmentary.
- Buffet server reconstructs the information in an XML document.

Diagram:

- Buffet Server
- CaféOBJ
- Gateau
- Proof Score Presenter
- proof.html
- proof.xml
- spec.mod
- inv.mod
- script.mod

Feedback:
- input
Gateau generates proof scores

- according to
  - 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)) = (11) .
eq (s') = (try(s,pid1)) .
-- reduce the following term:
red istep1(i, j) .
close
How to gen. proof score

1. Getting info. of spec.
2. Gen. proof score
   for each proof passage,
   a). reduce the term.
   b). if the result is not true, split into cases with the first pred. of given preds. list. Go to a).
   c). if the list is empty, introduce given lemmas.

 generated proof score (XML)
Proof Score Presenter

**PSP** is a pretty printer for proof scores
- takes a proof score in XML
- generates an HTML document

- **spec.mod**
- **inv.mod**
- **script.mod**
There are 7 cases, 3 cases need human's help.

```plaintext
base

action: try
case splitting: c-try(s, pid1)

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

  result:

  (((if (pid1 = i) then 12 else loc(s,i) fi) = cs) and ((if (pid1 = j) then 12 else loc(s,j) fi) = cs)) and ((loc(s,i) = cs) and (loc(s,j) = cs))
  xor (((if (pid1 = i) then 12 else loc(s,i) fi) = cs) and ((if (pid1 = j) then 12 else loc(s,j) fi) = cs)) and ((loc(s,i) = cs) and (loc(s,j) = cs)) and (i = j))
  xor (((if (pid1 = i) then 12 else loc(s,i) fi) = cs) and ((if (pid1 = j) then 12 else loc(s,j) fi) = cs))
  xor (((if (pid1 = i) then 12 else loc(s,i) fi) = cs) and ((if (pid1 = j) then 12 else loc(s,j) fi) = cs)) and (i = j))
  xor true

case: false

action: enter
case splitting: c-enter(s, pid1)
  case: true
```

Hierarchical view with disclosing triangles

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