Current Issues on Algebraic Specification and Verification in CafeOBJ

Kokichi FUTATSUGI
RCSV-JAIST

Kaken Kiban(S) Workshop
Hotel Arrowle, Kaga, Ishikawa, Japan
25 June 2015
An Important Key Technology in SE

Software has became an important infrastructure of our society. Reliability and Security are most important attributes.

- middle ware software/protocol (electric commerce, cloud computing/services)
- International standards for important software systems (standards for on-board computers, etc.)

Verification of Problem Specifications is a Key Technology
Problem verification with Proof Scores

Technical Issues

1. Development of specs in appropriate abstraction level
2. Verification methods combining inference and search

CafeOBJ Proof Scores

Innovate Spec Verification and Realize Practical Problem Spec Verifications
Concurrent development of cases and methods

Requirements from cases

Practical cases
- on-board OS standard
- middleware protocol

Problem spec Verification system

Innovative problem spec verification methods
- appropriate abstraction
- inference*search veri.

Verification with PS
Specification Verification

- Constructing specifications and verifying them in the upstream of system/software development are still one of the most important challenges in system/software development. It is because many critical defects are caused at the phases of domains, requirements, and design specifications.

- Proof scores are still intended to meet this challenge.
A system and the system’s properties are specified in an executable algebraic specification language (CafeOBJ in our case).

Proof scores are described also in the same executable specification language for checking whether the system specifications imply the property specifications.

Specifications and proof scores are expressed in equations, and the checks are done by reduction (i.e. rewriting from left to right) with the equations.

The logical soundness of the checks is guaranteed by the correctness of proof scores and the fact that the reduction are consistent with the equational reasoning with the equations.
Verification methods with proof scores includes OTS/CafeOBJ method, CITP/CafeOBJ method, and Generate&Check method. They are methods for verifying transition systems.

**OTS**: Observational Transition System  
**CITP**: Constructor-based Inductive Theorem Prover


OTS Spec in Equations: OTS, CITP

-- CafeOBJ variables
var S : Sys
vars I J : Pid
-- for want transition
op c-want : Sys Pid -> Bool {memo strat: (0 1 2)}
eq c-want(S,I) = (pc(S,I) = rm) .
--
ceq pc(want(S,I),J)
    = (if I = J then wt else pc(S,J) fi) if c-want(S,I) .
ceq queue(want(S,I)) = put(I,queue(S)) if c-want(S,I) .
ceq want(S,I) = S if not c-want(S,I) .
OTS Spec in Trans Rules: Gen&Check

-- wt: want transition
mod! WT {pr(STATE) tr[wt]:
  [Q:Aq  r (A:Aid AS1:As) w AS2:As  c AS3:As] => [(Q | A) r AS1  w (A AS2) c AS3] .}

-- ty: try transition
mod! TY {pr(STATE) tr[ty]:
  [(A:Aid | Q:Aq) r AS1:As w (A:Aid AS2:As) c AS3:As] => [(A | Q) r AS1  w AS2 c (A AS3)] .}

-- exc: exit transition with a condition
mod! EXc {pr(STATE) ctr[exc]:
  [(A1:Aid | Q:Aq) r AS1:As w AS2:As c (A2:Aid AS3:As)] => [Q r (A2 AS1) w AS2 c AS3] if (A1 = A2) .}
OTS Proof Score in 'open...close’s

-- Proof: By induction on X.
--> I Base
open PRED-PNAT+
-- check
red +rz(0) .
close
--> II Step
open PRED-PNAT+
-- arbitrary value
op x : -> Nat .
-- induction hypothesis
eq x + 0 = x .
-- check
red +rz(s(x)) .
close
--> QED
OTS Scheme for Proof Scores in 'open..close’s

open ISTEP
Declare constants denoting arbitrary objects.
Declare equations denoting case^i_j.
Declare equations denoting facts if necessary.

eq s’ = a(s,y) .
red istepi(xi) .
close

Proof Scores in CITP Commands: CITP

open PNAT+.
:goal { eq [+rz]: M:Nat + 0 = M:Nat .
    eq [+rs]: M:Nat + s N:Nat = s(M:Nat + N:Nat). }
-- Proof: By induction on M.
:ind on (M:Nat)
:auto
--> QED
close
Proof Scores in Geneate & Check

```haskell
mod! Q-INV-2-genCheck { ex(Q-INV-2 + GENstTerm + CONSTandLITL) op ck : -> IndTr . eq ck = check 
    ([(g("[r_w_c_]"))
      [(empQ;(a1 | q)),(empS;(a2 s1)),
       (empS;(a3 s2)),(empS;(a4 s3))]]).
--
open Q-INV-2-genCheck . pr(INIT + INV) eq p-init = init . eq p-iinv = mx hq=c q=wc . red ck . close
```
[OTS] An algebraic/equational programming scheme for writing proof scripts. All proof planning is described and transparent. Flexible and versatile but could be sloppy. Easy to learn and use.
[**CITP**] Automation of a part of OTS proof scores based on sound and quasi-complete "specification calculus". Formal enough and can automate (1) induction and (2) the case splitting induced by conditional equations, but proof planning is not transparent. CITP/CafeOBJ has a potential to be incorporated into OTS proof scores.
[Gen&Check] Realized as a library in CafeOBJ for automatically generating covering state patterns and checking verification conditions over them. Can be used for covering patterns of any sort. Covering patterns represent split cases and unproved cases are explicit for analyses, which may help to find lemmas. CafeOBJ’s built-in search predicates and the concept of literal constants play important roles. Formal meta-theory for the method is provided.
Properties to be Verified

\[ \text{OTS} \] invariant, unless, stable, ensures
\[ \text{CITP} \] invariant
\[ \text{Gen&Check} \] invariant, leads-to

OTS/TransitionSystem Spec

\[ \text{Intuitiveness} \] Equations (OTS,CITP) \(<\) TransRules (Gen&Check)
Proof Scores

[Flexibility] CITP < Gen&Check < OTS

[Automation] OTS < Gen&Check < CITP

[Formality] OTS < Gen&Check < CITP
Key-Secrecy of PACE with OTS/CafeOBJ, Dominik Klein (dominik.klein@bsi.bund.de) — Key-secrecy of PACE (The ICAO-standardized Password Authenticated Connection Establishment) is proven by first modeling it as an Observational Transition System (OTS) in CafeOBJ, and then proving invariant properties by induction.
A Formal Description of the OSEK/VDX Specification
Hirokazu Yatsu (hirokazu.yatsu@f.ait.kyushu-u.ac.jp) [with OTS+alpha] — A formal description of the OSEK/VDX specification has been developed with an algebraic formalization with right abstraction. Verification of dead lock freeness has been done with the formal description, and the developed formal description is precise enough to verify important properties.
Formalization and Verification of Declarative Cloud Orchestration in CafeOBJ

Hiroyuki Yoshida (yuki.yoshida@jaist.ac.jp) [with Gen&Check+OTS] — The behavior of TOSCA (Topology and Orchestration Specification for Cloud Applications) topologies is specified in CafeOBJ as state transition systems. It is verified that orchestrated operations always successfully complete by proving the transition systems enjoys leads-to (a class of liveness) properties.
Current Issues

- To make explicit what good is each proof score method, and establish techniques/methods to coordinate the three methods in an effective way.

- To make the message "verification is the best way to get high-quality formal specs" more persuasive. To make it clear what can be obtained out of developments of formal specs and proof-scores.

- Make more effective use of versatile CafeOBJ’s module structuring mechanism for giving good structures to specs and proof-scores.
  - Generic specs and proof-scores
  - Domain specific frameworks for specs and proof-scores