Verifying Specifications with Proof Scores in CafeOBJ

FUTATSUGI, Kokichi
二木 厚吉

Chair of Language Design
Graduate School of Information Science
Japan Advanced Institute of Science and Technology (JAIST)
Japan

(this talk is based on our group's research results with many persons' contributions)

I am going to talk about…

- Our perception of current situation of formal methods and specification verifications
- Introduction to how to model/specify and verify in CafeOBJ with simple examples
- An overview of specifications and proof scores in CafeOBJ including what kinds of formal models are used for writing formal specifications and proof scores
- Current status and future issues of the proof score approach for verifying specifications

CafeOBJ

Our perception about Formal Methods and Specification Verifications

Application areas of formal methods (FM)

1. **Analysis and verification of developed program codes (post-coding)**

2. **Analysis and verification of (models/specs of) domains, requirements, and designs before/ without coding (pre-coding or without coding)**

Successful application of formal methods to the area of (models/specifications of) domains, requirements, designs can bring drastic good effects for systems developments, but it is not well exploited and/or practiced yet.

specification = description of model

CafeOBJ

The current situation of FM

- Verification with formal specifications still have a potential to improve the practices in upstream (pre-coding) of systems development processes
- Model checking has brought a big success but still has limitations
  - It is basically "model checking" for program codes
    - Still mainly for post-coding
  - Infinite state to finite state transformation can be unnatural and difficult
- Established interactive theorem provers (Isabelle/ HOL, Coq, PVS, etc.) are not necessary well accepted to software/systems engineers
  - especially in upstream (pre-coding) phase

CafeOBJ

Difficulties in domains, requirements, designs

- High level specifications of domains, requirements, designs are inherently partial and evolutional
- Usually there is no established formal (mathematical) model for the problems
- It is not easy to be convinced that some important property holds for domains, requirements, designs

Interactive developments with analyses/verifications are needed

Developments of domain theories can help

CafeOBJ

Our approach

- Reasonable blend of user and machine capabilities, intuition and rigor, high-level planning and tedious formal calculation
  - fully automated proofs are not necessary good for human beings to perceive logical structures of real problems/systems
  - interactive understanding of real problem domains and/or requirements is necessary

Proof Score Approach

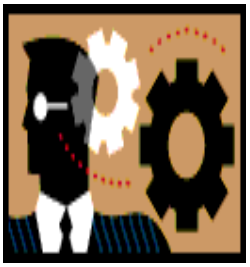
CafeOBJ

Proof Score Approach

- Domain/requirement/design engineers are expected to construct proof scores together with formal specifications
- Proof scores are instructions such that when executed (or "played") and everything evaluates as expected, then the desired property is convinced to be hold
  - Proof by construction/development
  - Proof by reduction/computation/rewriting

CafeOBJ

Modeling/Specifying and Verifying in CafeOBJ



Modeling/Specifying and Verifying in CafeOBJ

1. By understanding a problem to be modeled/specified, determine several sorts of objects (entities, data, agents, states) and operations (functions, actions, events) over them for describing the problem
2. Define the meanings/functions of the operations by declaring equations over expressions composed of the operations
3. Write proof scores for properties to be verified

CafeOBJ

Natural Numbers -- Signature --

0 0+1 0+1+1 0+1+1+1 0+1+1+1+1 …

0 s(0) s(s(0)) s(s(s(0))) s(s(s(s(0)))) …

objects: Nat
operations: 0 : returns zero without arguments
s : given a natural number *n* returns the next natural number (s *n*) of *n*

```
-- sort
[ Nat ]
-- operations
op 0 :  -> Nat
op s_:  Nat -> Nat
```

CafeOBJ

Natural Number

-- Expressions/terms composed of operations --

```
mod! BASIC-NAT
  { [ Nat ] op 0: -> Nat op s_: Nat -> Nat }
```

1. 0 is a natural number
2. If *n* is natural number then（s *n*）is a natural number
3. An object which is to be a natural number by 1 and 2 is only a natural number

Peano's definition of natural numbers（1889）, Giuseppe Peano (1858-1932)

Nat = {0, s(0), s(s(0)), s(s(s(0))), s(s(s(s(0)))) … }

Nat = {0, s 0, s s 0, s s s 0, s s s s 0, … }

Describe a problem in expressions/terms!

CafeOBJ

Mathematical Induction over Natural Numbers

Goal: Prove that for any natural number $n \in \{0, s\ 0, s\ s\ 0, \cdots\}$
$P(n)$ is true

Induction Scheme:

$$\frac{P(0) \qquad \forall n \in \mathbb{N}.\,[P(n) \supset P(s\ n)]}{\forall n \in \mathbb{N}.\,P(n)}$$

Concrete Procedure: (induction with respect to n)

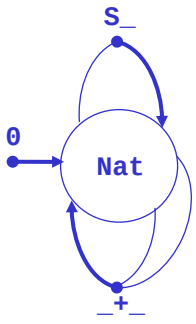
1. Prove $P(0)$ is true
2. Assume that $P(n)$ holds, and prove that $P(s\ n)$ is true

CafeOBJ

Natural numbers with addition operation

-- signature and expressions/terms --

```
-- sort
[ Nat ]
-- operations
op 0 :  -> Nat
op s_:  Nat -> Nat
op _+_: Nat Nat -> Nat
```



$$\mathrm{Nat} = \{\ 0\ \} \cup \{\ s\ n \mid n \in \mathrm{Nat}\ \} \cup \{\ n1 + n2 \mid n1 \in \mathrm{Nat} \wedge n2 \in \mathrm{Nat}\ \}$$

CafeOBJ

Natural numbers with addition

-- expressions/terms composed by operations --

```
op 0: -> Nat . op s_:  Nat -> Nat . op _+_: Nat Nat -> Nat .
```

```
Nat = {
0, s 0, s s 0, s s s 0, ... ,
0 + 0, 0 + (s 0), 0 + (s s 0), 0 + (s s s 0), ...,
(s 0) + 0, (s 0) + (s 0), (s 0) + (s s 0),
                          (s 0) + (s s s 0), ...,
(s s 0) + 0, (s s 0) + (s 0), (s s 0) + (s s 0),
                        (s s 0) + (s s s 0), ...,
... ...
0 + (0 + 0), 0 + (0 + (s 0)), ...
...
(0 + 0) + 0, (0 + (s 0)) + 0, ...
...
. }
```

CafeOBJ

Natural numbers with addition

-- equations defining meaning/function --

CafeOBJ code describing
Natural numbers with addition

```
mod! NAT+ {
-- sort
[ Nat ]
-- operations
op 0 :  -> Nat
op s_:  Nat -> Nat
op _+_: Nat Nat -> Nat
-- equations
eq N:Nat + 0 = N .
eq N:Nat + (s M:Nat) = s (N + M) .
}
```

Inference/Computation
with the equations

```
 (s 0) + (s s 0)
= s((s 0) + (s 0))
= s s((s 0) + 0)
= s s s 0
```

CafeOBJ

Proof Score
for the proof of associativity of addition (_+_)

```
-- opening module NAT+ and EQL
-- EQL is a built-in meta-module
-- for making a predicate (_ = _) available
open (NAT+ + EQL)
--> declaring constants as arbitrary values
ops i j k : -> Nat .

--> Prove associativity: (i + j) + k = i +(j + k)
--> by induction on k

--> base case proof for 0:
red i + (j + 0) = (i + j) + 0 .
--> induction hypothesis:
eq (i + j) + k = i + (j + k) .
--> proof of induction step for (s k):
red (i + j) + (s k) = i + (j + (s k)) .
--> QED {end of proof for associativity of (_+_)}
close
```

CafeOBJ

Specification and Proof Score
for two definitions of factorial (1)

```
mod! NAT*ac {
  [ Nat ]
  op 0 : -> Nat
  op s_ : Nat -> Nat
  op _+_ : Nat Nat -> Nat {assoc comm}
  eq M:Nat + 0 = M .
  eq M:Nat + s N:Nat = s(M + N) .
  op _*_ : Nat Nat -> Nat {assoc comm}
  eq M:Nat * 0 = 0 .
  eq M:Nat * s N:Nat = (M * N) + M . }

mod! NAT*dist { protecting(NAT*ac)
  eq L:Nat * (M:Nat + N:Nat) = (L * M) + (L * N) . }
```

CafeOBJ

Specification and Proof Score for two definitions of factorial (2)

```
mod! FACT { protecting(NAT*ac)
  op fact : Nat -> Nat
  eq fact(0) = s 0 .
  eq fact(s N:Nat) = (s N) * fact(N) .
}

mod! FACT2 { protecting(NAT*ac)
  op fact2 : Nat Nat -> Nat
  eq fact2(0, A:Nat) = A .
  eq fact2((s N:Nat), A:Nat) = fact2(N, (s N) * A) .
}
```

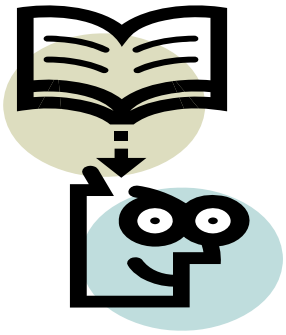
CafeOBJ

Specification and Proof Score for two definitions of factorial (3)

```
open (FACT + FACT2 + NAT*dist + EQL)
--> i,j,k stand for any natural numbers
ops i j k : -> Nat .
--> proving: fact2(i, j)  = j * fact(i)
--> by induction on i
--> proof of induction base for 0:
red fact2(0,j) = j * fact(0) .
--> induction hypothesis:
eq fact2(i,J:Nat) = J * fact(i) .
--> proof of induction step for (s i):
red fact2(s i, j) = j * fact(s i) .
--> QED (quod erat demonstrandum:
-->      which was to be demonstrated)
close
```

CafeOBJ

Specifications and Proof Scores in CafeOBJ



Specifications and Proof scores in CafeOBJ

- In the proof score approach, specifications are only algebraic equational specifications
- Proof score is a sequence of reduction (simplification) commands for reducing expressions (usually boolean) to its normal form in some appropriate situations
  - situation: a set of equations (axioms) with some bindings (a set of `name->object` relationships)
  - proof score also contains CafeOBJ codes which build appropriate situations in which reductions take place

CafeOBJ

Development of proof scores in CafeOBJ

- Many simple proof scores are written in OBJ language from 1980's; some of them are not trivial
- From around 1997 CafeOBJ group at JAIST use proof scores seriously for verifying specifications for various examples
  - From static to dynamic/reactive system
  - From ad hoc to more systematic proof scores
  - Introduction of OTS (Observational Transition System) was a most important step

CafeOBJ

Introducing CafeOBJ

- CafeOBJ is an algebraic formal specification language; it is a successor of the OBJ language
- CafeOBJ is a formal language for writing a formal model and reasoning about the model
  - It is not a programming language for writing program codes for a system which is supposed to run on machines
  - However, CafeOBJ codes (specifications) are executable on machines for simulating, analyzing, and/or reasoning about the models described

CafeOBJ

A little bit of CafeOBJ history

- KF thought of the basic ideas of CafeOBJ after he participated OBJ project at SRI in 1983-1984, and several design and implementation attempts were done during 1985-1995
- The CafeOBJ development project is fully supported by IPA/MITI of Japanese Government from 1996.4 to 1998.3
  - Six Japanese Companies, Five Japanese Universities, Three Foreign Research Group participate CAFÉ project
  - A book entitled "CafeOBJ Report" was published in 1998 which defines the syntax and semantics of the CafeOBJ language
- Sufficiently reliable and usable CafeOBJ system was available at around the beginning of 1999.
- Several groups including KF's group at JAIST are using CafeOBJ for developing formal methods for various application areas and/or for education of FM

CafeOBJ

Related on going Language System Development Projects

- "Maude" Language of SRI/UIUC is another project for following up the OBJ language
  - LNCS entitled "All about Maude" (almost 800 pages) is published recently (2007)
- "CASL" language of European researchers is an attempt of developing a common algebraic specification language
  - Two volumes of LNCS are published

CafeOBJ

VDM/Z versus OBJ/CafeOBJ/Maude

- VDM/Z
  - non-executable; good formal communication languages between specifiers/designers/implementers, but not so powerful for automatic checkings/verifications
- OBJ/CafeOBJ/Maude
  - executable; a powerful tool for rapid prototyping, automatic checking, formal reasoning, and verifications

CafeOBJ

Main features of CafeOBJ

- Equational Specification -- OBJ, CafeOBJ, Maude
  - equational logic
- Behavioral Specification -- CafeOBJ
  CHA: coherent hidden algebra; OTS
- Modular Specification -- OBJ, CafeOBJ, Maude
  - parameterized modules and module expressions
  - institutions based semantics
- Typed Specification -- OBJ, CafeOBJ, Maude
  - order-sorted algebra
- Rewriting Specification -- Maude, CafeOBJ
  - rewriting logic

CafeOBJ

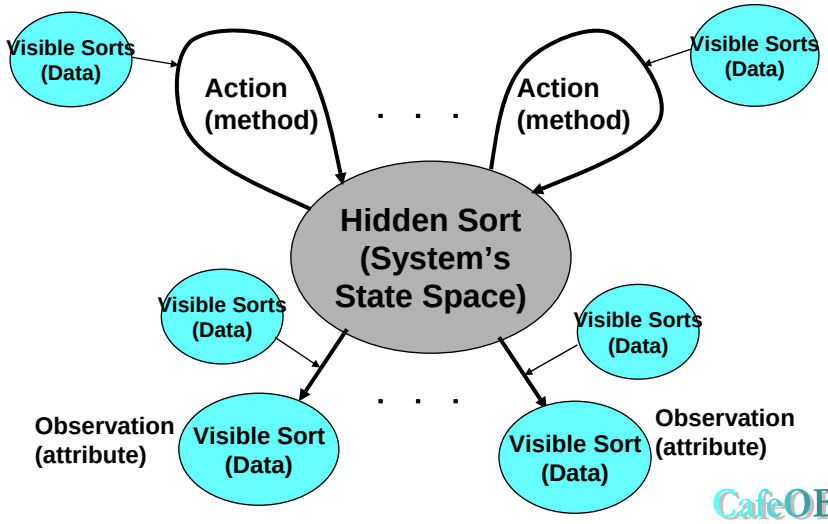
Two kinds of formal models in CafeOBJ

- **Abstract data types** with tight semantics
  - Modeling data
  - Initial algebra semantics
  - Induction based reasoning
- **Abstract machines (abstract process types)** with loose semantics
  - Modeling processes or behaviors
  - Coherent hidden algebra semantics
  - Co-induction based reasoning
  - CafeOBJ is the first algebraic formal specification language which supports the abstract machines

These two models can provide unified specification style both for static and dynamic systems

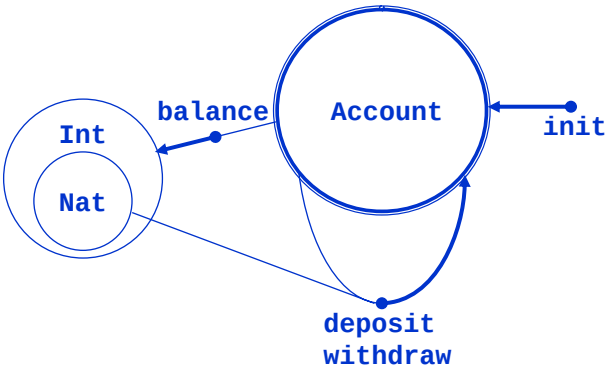
CafeOBJ

Modeling Behaviors by sequences of actions



CafeOBJ

Simple Bank Account -- signature --



CafeOBJ

Simple Bank Account -- signature --

```
-- state space of an ACCOUNT system
*[ Account ]*
-- initial state of an ACCOUNT system
op init : -> Account
-- an observer of Account
bop balance : Account -> Int
-- two actions for Account
bop deposit : Account Nat -> Account
bop withdraw : Account Nat -> Account
```

CafeOBJ

Reachable state space of ACCOUNT

```
ReachableStateSpaceOfAccount
= {init}
∪
{deposit(a,n)|
   a∈ReachableStateSpaceOfAccount,n∈Nat}
∪
{withdraw(a,n)|
   a∈ReachableStateSpaceOfAccount,n∈Nat}
```

CafeOBJ

CafeOBJ specification of ACCOUNT

```
mod* ACCOUNT { protecting(INT>=)
  -- signature should comes here
     ...
  -- condition for init; initial condition
  eq balance(init) = 0 .

  -- for the action "deposit(A,N)"
  eq balance(deposit(A:Account,N:Nat)) = balance(A) + N .

  -- for the action "withdraw(A,N)"
  cq balance(withdraw(A:Account,N:Nat))
      = balance(A) - N if ((balance(A) - N) >= 0) .
  cq balance(withdraw(A:Account,N:Nat))
      = balance(A) if not((balance(A) - N) >= 0) . }
```

CafeOBJ

Current status and future issues

Current Achievements of OTS/CafeOBJ proof score approach

OTS/CafeOBJ approach has been applied to the following kinds of problems and found usable:

- Some classical mutual exclusion algorithms
- Some real time algorithms
  e.g. Fischer's mutual exclusion protocol
- Railway signaling systems
- Authentication protocol
  e.g. NSLPK, Otway-Rees, STS protocols
- Practical sized e-commerce protocol of SET
  (some of proof score exceeds 60,000 lines;
  specification is about 2,000 lines,
  20-30 minutes for reduction of the proof score)
- UML semantics (class diagram + OCL-assertions)
- Formal Fault Tree Analysis
- Secure workflow models

CafeOBJ

Three levels of CafeOBJ applications

1. Construct formal models; Develop formal specifications
2. Do rapid prototyping and check the properties of specifications; execute specifications for validations/verifications
3. Write proof scores to verify properties of specifications; verifications with reductions/rewritings

Choose an appropriate level depending on problems and situations

CafeOBJ

Prerequisites for proof score writing in CafeOBJ (1)

- **Algebraic modeling:**
  development of algebraic specifications
  - ♦ defining signature for a real problem
  - ♦ expressing the semantics of a problem in equations
    - ■ more exactly, expressing the problem in reduction rules

CafeOBJ

Prerequisites for proof score writing in CafeOBJ (2)

- Equational logic, rewriting, and propositional calculus
  - equationl reasoning
    - equivalence relation, equational calculus, ...
  - propositional calculus with "xor" normal forms which has the complete rewriting calculus
  - reduction/rewriting
    - termination, confluence, sufficiently completeness

CafeOBJ

Prerequisites for proof score writing in CafeOBJ (3)

- Proof by induction and case analysis
  - case splitting using key predicates in specifications
  - discovery of lemmas
  - decomposition of a goal predicate into an appropriate conjunctive form

These are the most difficult parts of proof score writing

But this is common to any kind of interactive verifiers!

CafeOBJ

Traceability in proof score approach with CafeOBJ

- All reductions are done exactly using equations in specifications
  - this make it easy to detect necessary changes in specs for letting something happen (or not happen)
- Usually reductions are sufficiently fast, and encourage prompt interactions between user and system

This is a quit unique feature of the proof score approach with CafeOBJ comparing to other verification method which often involves several formalisms/logics and translations between them

CafeOBJ

Equational proofs by reduction/rewriting

Why do we care about

equational reasoning by reduction ?

- It is simple and powerful and a promising light weighted formal reasoning method
  - easy to understand and can be more acceptable for software engineers
- It supports transparent relation between specs and reasoning by reduction (good traceability)

CafeOBJ

Future Issues (1)

- Development of theory and methodology for proof score writings
  - Automate case analysis and lemma discovery
    - Automation of inductive proof (Crème)
      - NSLPK verification is almost done automatically
  - Incorporation of model checking technologies (i.e. searching of state space) into proof score writings
    - Search command of the form:

      `reduce S1 =(m,n)=>* S2 suchThat P(S2) .`

      has been introduced into CafeOBJ recently
    - The command already has made the following possible:
      - Automatic counter example findings
      - Automation of some parts of proof scores

CafeOBJ

Future Issues (2)

- Development proof score writing environment
  - Standard platforms for programming environment can be naturally used: Emacs, Eclips
  - Write specs and proofs as writing programs and test cases, but in logically more sound and complete way.
- Application to important domains and development of formal specifications (precise descriptions of models) in the domain; not only technical domains but also business/social domains
  - Mobile protocols and systems
  - Digital Right Management (DRM) systems
  - E-Government and public administration
  - Internal control/governance (security/safety rules/policies in an organization)

CafeOBJ

Future Issues (3)

- Development of proof scores for verifying important properties of application domains
  - (Formal specification + Proof Score) hope to be an important asset for the organization
  - The proof score method hope to show new kind of facts which are difficult to be shown by other method
  - Systems verification has already become an important topic in systems development and maintenance.

CafeOBJ

CafeOBJ Official Home Page

http://www.ldl.jaist.ac.jp/cafeobj/

CafeOBJ