CafeOBJ as a Tool for Software Checking

Akira Mori (JAIST, Japan)
Toshimi Sawada (SRA, Japan)
Kokichi Futatsugi (JAIST, Japa

System Description

- automatic safety model checker for (*infi*
  stract state machines (ASMs)
- ASM defined as special algebra (hidden a
  - behavioral specification
  - input and output as abstract data types
  - supported in **CafeOBJ** system
- model checking conducted using predicate
  - predicate as set of states
  - previous states by predicate transforme
  - **PigNose**: resolution engine for CafeOI
- *same logic* used for specification and veri

Brief Overview of CafeOBJ

- algebraic specification in tradition of OB
  abstract data types, order-sorted algebra/equati
  parameterized modules, module expressions, term
- new features
  - behavioral specification based on hidde
  - predicate calculus and resolution engine
  - safety model checking and behavioral
    tion

Algebraic Spec. of Dynamic System Beh

Examples: Java Bank Account Object

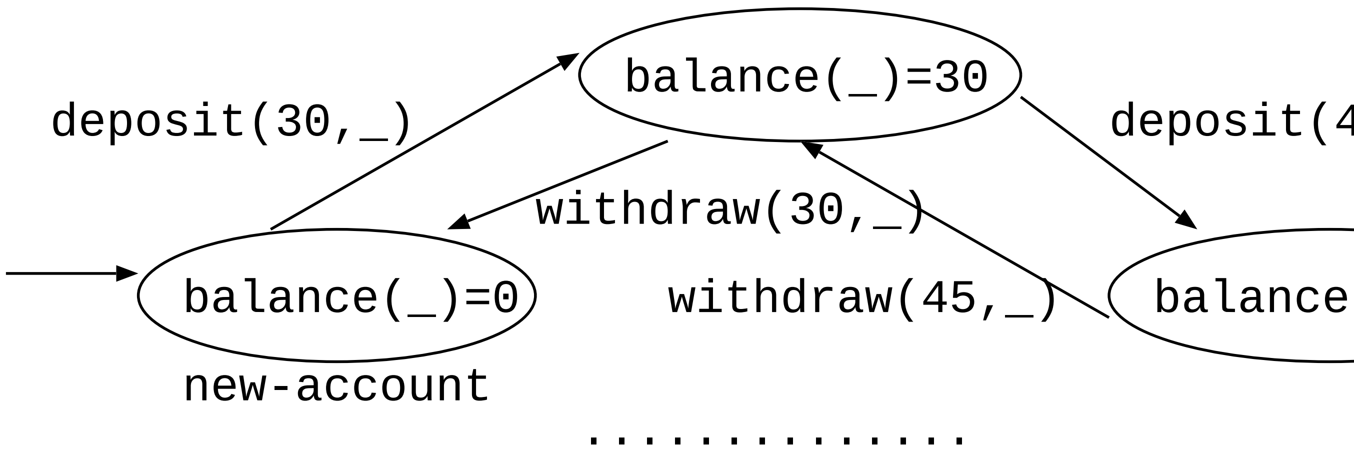
```
public class Account {
  public int balance = 0;
  public void deposit(int amount) {
    if (0 <= amount) balance += amount;
  }
  public void withdraw(int amount) {
    if (amount <= balance) balance -= amoun
  }
}
```

```
mod* ACCOUNT {
protecting(INT)                          -- d
*[ Account ]*                            -- h
op new-account : -> Account              -- n
bop balance : Account -> Int             -- a
bop deposit : Int Account -> Account     -- m
bop withdraw : Int Account -> Account    -- m
var A : Account    vars N : Int
ax balance(new-account) = 0 .
ax 0 <= N -> balance(deposit(N,A)) = balanc
ax ~(0 <= N) -> balance(deposit(N,A)) = bal
ax N <= balance(A) ->
            balance(withdraw(N,A)) = balanc
ax ~(N <= balance(A)) ->
            balance(withdraw(N,A)) = balanc
}
```

Behavioral Spec. based on Hidden Alg.

- abstract data type + abstract state mach
- hidden sort (states) vs. visible sort (data
- only one hidden sort in co-arity of behav eration (methods and attributes)
- covers well object-oriented concepts

Hidden Algebra as State Machine



Model checking?

- transitions parameterized
- state space unbounded
- must combine deduction and exploration

Invariant Checking for Bank Account

prototype of safety model checking

$\mathtt{balance(A)} \geq 0$ for any reachable state $\mathtt{A}$ of $\mathtt{A}$

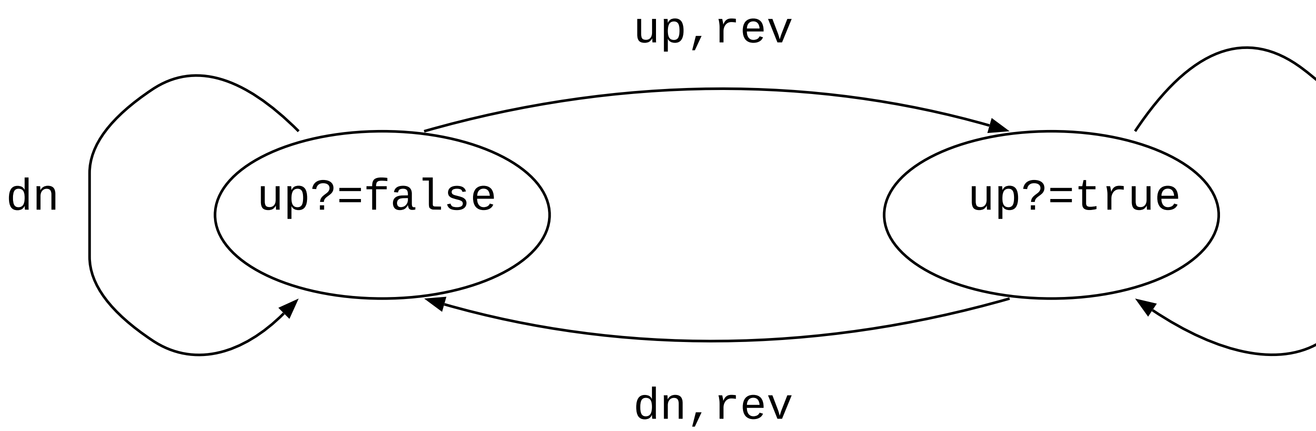
- $\mathtt{balance(new\text{-}account)} \geq 0$
- $\forall[\mathtt{A} : \mathtt{Account}, \mathtt{N} : \mathtt{Int}]$

  $\mathtt{balance(A)} \geq 0 \Rightarrow \mathtt{balance(deposit(N, A))} \geq$

  $\mathtt{balance(A)} \geq 0 \Rightarrow \mathtt{balance(withdraw(N, A))} \geq$

State Identification in Hidden Algebra

```
mod* FLAG {
  *[ Flag ]*
  bops (up_) (dn_) (rev_) : Flag -> Flag
  bop up?_ : Flag -> Bool
  var F : Flag
  eq up? up F = true .
  eq up? dn F = false .
  eq up? rev F = not up? F .
 }
```



$$\forall[\texttt{F} : \texttt{Flag}]\ \texttt{rev rev F} = \texttt{F}?$$

Behavioral Abstraction of States

Hidden elements equivalent iff behaviors (r

observation via methods and attributes) are t

Coinduction as Relational Invariant

$\forall[\mathtt{F} : \mathtt{Flag}]\ \mathtt{rev}(\mathtt{rev}(\mathtt{F})) = \mathtt{F}$

- $\forall[\mathtt{F} : \mathtt{Flag}]\ \mathtt{up?}(\mathtt{rev}(\mathtt{rev}(\mathtt{F}))) = \mathtt{up?}(\mathtt{F})$
- $\forall[\mathtt{F}, \mathtt{F}' : \mathtt{Flag}]$

  $\mathtt{up?}(\mathtt{F}) = \mathtt{up?}(\mathtt{F}') \Rightarrow \mathtt{up?}(\mathtt{up}(\mathtt{F})) = \mathtt{up?}(\mathtt{up}(\mathtt{F}'$

  $\mathtt{up?}(\mathtt{F}) = \mathtt{up?}(\mathtt{F}') \Rightarrow \mathtt{up?}(\mathtt{dn}(\mathtt{F})) = \mathtt{up?}(\mathtt{dn}(\mathtt{F}'$

  $\mathtt{up?}(\mathtt{F}) = \mathtt{up?}(\mathtt{F}') \Rightarrow \mathtt{up?}(\mathtt{rev}(\mathtt{F})) = \mathtt{up?}(\mathtt{rev}($
- relation $\mathtt{up?}(\_) = \mathtt{up?}(\_)$ is invariant start

  $(\mathtt{rev}(\mathtt{rev}(\mathtt{F})), \mathtt{F})$

Symbolic Manipulation of ASM in Hidden

- Predicate as set of states

  $\mathtt{P}(\mathtt{X} : \mathtt{Protocol}) \stackrel{\triangle}{=}$

  $\forall[\mathtt{I}, \mathtt{J} : \mathtt{Nat}]\ \mathtt{flag}(\mathtt{I}, \mathtt{X}) = \mathtt{flag}(\mathtt{J}, \mathtt{X}) = \mathtt{sharec}$

  $\qquad\qquad \mathtt{cdata}(\mathtt{I}, \mathtt{X}) = \mathtt{cdata}(\mathtt{J}, \mathtt{X}).$

  (must be defined in terms of attributes!)

- Predicate transformer as previous state fu

  $$\mathtt{pre}(\mathtt{P}(\mathtt{X} : \mathtt{h})_{\mathtt{h}})_{\mathtt{h}'} \stackrel{\triangle}{=} \sum_{\sigma : \mathtt{wh}' \to \mathtt{h}} \exists[\mathtt{V} : \mathtt{w}]\ \mathtt{P}(\sigma(\mathtt{V}, \mathtt{Y}$$

- $\mathtt{Q}(\mathtt{X}) \stackrel{\triangle}{=} \exists[\mathtt{I} : \mathtt{Index}, \mathtt{M} : \mathtt{Data}]\ \neg\mathtt{P}(\mathtt{write}(\mathtt{I}, \mathtt{M}, \mathtt{X}$

  set of states whose next states via `write` c

  may not satisfy cache coherence

Backward Safety Model Checking

To show that $I \not\subseteq \mathtt{pre}^*(\neg P)$

(I: initial states, P: safety predicate)

$\mathtt{pre}^*(P) \triangleq \mu Z.P \vee \mathtt{pre}(Z)$

(least fixpoint of $\mathtt{pre}$ including P)

set of all states from which a state satisfyin

be reached

Counterexample when $I \subseteq \mathtt{pre}^*(\neg P)$

Fixpoints can be calculated by predicated ca