The Modularity of Object Propositions

Ligia Nistor
lnistor@cs.cmu.edu
Carnegie Mellon University, USA

Motivation

- **Modular verification in the presence of aliasing**
  - Multiple clients depend on property of one object
  - One client can break that property
- **Limitations of existing techniques**
  - **Classical invariant technique**: all objects of same class need to satisfy the same invariant
  - **Separation logic**: specifications of methods need to reveal the exact structure of objects they modify
- **Object propositions**
  - Abstract predicates + fractional permissions
  - Unpack a shared object, modify it, pack it back
  - Modify objects without owning them → information hiding
  - Hide shared data between two abstractions
  - Automatable → the Oprop tool: https://github.com/ligianistor/Oprop

Object Propositions

- **Oprop language**: Featherweight Java + object propositions
- **Abstract predicates** characterize state of objects
- **Fractional permissions** specify sharing
- **obj@k Pred(x) : obj** is the object, **k** is the fractional permission, **Pred(x)** is the abstract predicate that holds

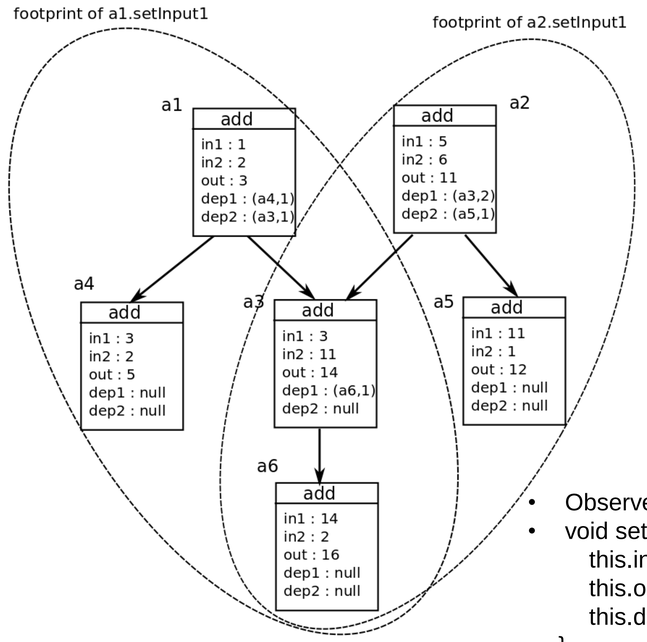
permission of 1
read/write access.
Can modify predicate.

permission of 1/2
read/write access.
Invariant holds.

Packing ensures that the initial predicate is satisfied

Unpacking gives access to the fields of an object

Spreadsheet Example



- Observer design pattern
- void setInput1(int x) {
  this.in1 = x;
  this.out = this.in1 + this.in2;
  this.dep1.setInput1(this.out);
  }

Verification Using Separation Logic

- Predicates **Basic** and **SepOK** in separation logic

$$Basic(cell) \equiv \exists x1 : int, x2 : int, o : int, d1 : Cell, d2 : Cell.(cell.in1 \rightarrow x1) \star (cell.in2 \rightarrow x2) \star (cell.out \rightarrow o) \star (cell.dep1 \rightarrow d1) \star (cell.dep2 \rightarrow d2)$$

$$SepOK(cell) \equiv \exists x1 : int, x2 : int, o : int, d1 : Cell, d2 : Cell.(cell.in1 \rightarrow x1) \star (cell.in2 \rightarrow x2) \star (cell.out \rightarrow o) \star (cell.dep1 \rightarrow d1) \star (cell.dep2 \rightarrow d2) \star (x1 + x2 == o) \star SepOK(d1) \star SepOK(d2)$$

- Verification of code using separation logic

$\{Basic(a2) \star Basic(a5) \star SepOK(a1)\}$
`a1.setInput1(10);`
$\{Basic(a2) \star Basic(a5) \star SepOK(a1)\}$
$\{********missing\ step********\}$
$\{Basic(a4) \star Basic(a1) \star SepOK(a2)\}$
`a2.setInput1(20);`

Verification Using Object Propositions

- Predicate **OK** expressed using object propositions

$$predicate\ OK() \equiv \exists k1 : real, k2 : real, x1 : int, x2 : int, o : int, d1 : Cell, d2 : Cell\ .\ x1 + x2 = o \otimes this.out \rightarrow o \otimes this.dep1 \rightarrow d1 \otimes this.dep2 \rightarrow d2 \otimes 0 < k1, k2 \leq 1 \otimes d1@k1\ OK() \otimes d2@k2\ OK()$$

- Verification of code using object propositions

$\{a1@k1\ OK() \otimes a2@k2\ OK()\}$
`a1.setInput1(10);`
$\{a1@k1\ OK() \otimes a2@k2\ OK()\}$
`a2.setInput1(20);`