Observational purity & encapsulation

David A. Naumann
Stevens Institute of Technology
Hoboken, New Jersey, USA

See also Mike Barnett, D.N., Wolfram Schulte, Qi Sun:
99.44% Pure: Functional Abstractions in Specifications

Supported by NSF CCR-0208984, CCF-0429894 & Microsoft

class $\boldsymbol{Demo}\,\{$

private $\boldsymbol{arg}$: int, $\boldsymbol{isPr}$: bool;

proc $\boldsymbol{prime}(\boldsymbol{s} : \boldsymbol{Demo}, \boldsymbol{n} : \text{int}) : \text{bool}$

$\{$ $\boldsymbol{x}$: bool := “whether $\boldsymbol{n}$ is prime”; return $\boldsymbol{x}$; $\}$

proc $\boldsymbol{memo}(\boldsymbol{s} : \boldsymbol{Demo}, \boldsymbol{n} : \text{int}) : \text{bool}$

$\{$ if $\boldsymbol{n} = 0$ then $\boldsymbol{x}$:= false; return $\boldsymbol{x}$;

elseif $\boldsymbol{s.arg} \neq \boldsymbol{n}$

then $\boldsymbol{s.arg} := \boldsymbol{n}$; $\boldsymbol{s.isPr}$:= “whether $\boldsymbol{n}$ is prime”; endif ;

$\boldsymbol{x} := \boldsymbol{s.isPr}$; return $\boldsymbol{x}$; $\}$

proc $\boldsymbol{m}(\boldsymbol{s} : \boldsymbol{Demo}) : \text{bool}$

$\{$ $\boldsymbol{s.arg} := 1$; assert $\boldsymbol{memo}(\boldsymbol{s}, 2)$; return $(\boldsymbol{s.arg} == 1)$; $\}\ldots\}$

$\mathbf{class}\ \boldsymbol{Cell}\ \{\ \mathbf{public}\ \boldsymbol{val} : \mathbf{bool}\ \}$

$\mathbf{class}\ \boldsymbol{Demo}\ \{$

$\quad \mathbf{proc}\ \boldsymbol{prime}(\boldsymbol{s} : \boldsymbol{Demo}, \boldsymbol{n} : \mathbf{int}) : \boldsymbol{Cell}$

$\quad \{\quad \boldsymbol{x} : \boldsymbol{Cell} := \mathbf{new}\ \boldsymbol{Cell};\ \boldsymbol{x.val} :=$ "whether $\boldsymbol{n}$ prime"; $\ \mathbf{return}\ \boldsymbol{x};\ \}$

$\quad \ldots$

$\}$

Pure expressions in specification

What does a precondition with side effects mean?

What good is runtime checking for such an assertion?

- Eiffel: advice to use only pure methods, not checked
- ESC/Java: specifications and annotation using Java expressions *without method calls*
- JML: strong purity; only calls of pure methods— may allocate new objects but not update fields.

But lazy initialization and memoization common in libraries.

Pure expressions in specification

What does a precondition with side effects mean?

What good is runtime checking for such an assertion?

- Eiffel: advice to use only pure methods, not checked
- ESC/Java: specifications and annotation using Java expressions *without method calls*
- JML: strong purity; only calls of pure methods— may allocate new objects but not update fields.

But lazy initialization and memoization common in libraries.

Purity is also useful for program transformations etc.

Outline of talk

- criteria for a notion of purity
- strong purity
- observational purity

  Procedure p is observationally pure outside class D if no object it updates is visible in code of any class C, $C \neq D$.
- proving observational purity by equivalence with a strongly pure procedure

Criteria

(Partial correctness for simplicity; independent from particular specification/verification system.)

- $\mathbf{assert}\ \boldsymbol{p} \approx \mathbf{skip}$, provided $\boldsymbol{p}$ is pure
- $\approx$ is congruence: If $\boldsymbol{p} \approx \boldsymbol{q}$ then $\mathcal{K}[\boldsymbol{p}] \approx \mathcal{K}[\boldsymbol{q}]$ for all program contexts $\mathcal{K}[-]$.

Correctness-preserving: take $\mathcal{K}[-]$ to be $(-; \mathbf{assert}\ \boldsymbol{Q})$.

Criteria

(Partial correctness for simplicity; independent from particular specification/verification system.)

- $\mathbf{assert}\ \boldsymbol{p} \approx \mathbf{skip}$, provided $\boldsymbol{p}$ is pure
- $\approx$ is congruence: If $\boldsymbol{p} \approx \boldsymbol{q}$ then $\mathcal{K}[\boldsymbol{p}] \approx \mathcal{K}[\boldsymbol{q}]$ for all program contexts $\mathcal{K}[-]$.

Correctness-preserving: take $\mathcal{K}[-]$ to be $(-; \mathbf{assert}\ \boldsymbol{Q})$.

Also want determinacy, totality—beyond our scope.

Criteria

(Partial correctness for simplicity; independent from particular specification/verification system.)

- $\mathbf{assert}\ \boldsymbol{p} \approx \mathbf{skip}$, provided $\boldsymbol{p}$ is pure
- $\approx$ is congruence: If $\boldsymbol{p} \approx \boldsymbol{q}$ then $\mathcal{K}[\boldsymbol{p}] \approx \mathcal{K}[\boldsymbol{q}]$ for all program contexts $\mathcal{K}[-]$.

Correctness-preserving: take $\mathcal{K}[-]$ to be $(-; \mathbf{assert}\ \boldsymbol{Q})$.

Also want determinacy, totality—beyond our scope.

Semantics: $\boldsymbol{h} \dashv\!\mathbf{p}\!\vdash\!\!\rightarrow \boldsymbol{k}, \boldsymbol{v}$ means procedure $\boldsymbol{p}$ takes initial heap $\boldsymbol{h}$ to final heap $\boldsymbol{k}$ and value $\boldsymbol{v}$ (ignoring arguments). Commands: $\boldsymbol{h} \dashv\!\mathbf{assert}\ \boldsymbol{p}\!\vdash\!\!\rightarrow \boldsymbol{k}$ iff $\boldsymbol{h} \dashv\!\mathbf{p}\!\vdash\!\!\rightarrow \boldsymbol{k}, \boldsymbol{v}$ and $\boldsymbol{v} = \boldsymbol{true}$.

Strong purity

Def: $\boldsymbol{p}$ is strongly pure iff the final heap, restricted to initially allocated objects, is the same as initial:
$\boldsymbol{h} \dashv \boldsymbol{p} \models \boldsymbol{k}$ implies $(\mathsf{dom}\, \boldsymbol{h} \lhd \boldsymbol{k}) = \boldsymbol{h}$ (for all $\boldsymbol{h}, \boldsymbol{k}$).

Strong purity

Def: p is strongly pure iff the final heap, restricted to initially allocated objects, is the same as initial:
$h \dashv p \mapsto k$ implies $(\mathrm{dom}\, h \lhd k) = h$ (for all h, k).

Heap equivalence: given bijection β on locations, define
$h \sim_\beta h'$ iff $h\, o \sim_\beta h'\, o'$ for all $(o, o') \in \beta$.

Strong purity

Def: p is strongly pure iff the final heap, restricted to initially allocated objects, is the same as initial:

$h \dashv p \mapsto k$ implies $(\mathrm{dom}\, h \lhd k) = h$ (for all h, k).

Heap equivalence: given bijection β on locations, define $h \sim_\beta h'$ iff $h\, o \sim_\beta h'\, o'$ for all $(o, o') \in \beta$.

Def: $p \approx p'$ iff $\begin{array}{ccc} h & \sim_\beta & h' \\ p \downarrow & & \downarrow p' \\ k & & k' \end{array}$ implies $k \sim_\gamma k'$ for $\gamma \supseteq \beta$.

Thm: If p strongly pure then $\mathbf{assert}\ p \approx \mathbf{skip}$.

For Java-like language and specifications, $\approx$ is congruence.

Observational purity

Let $\mathrm{vis}\, C \lhd h\, o$ be the fields of object $h\, o$ visible in class C.

Def: $h \sim_{\beta}^{C} h'$ iff $(\mathrm{vis}\, C \lhd h\, o) \sim_{\beta} (\mathrm{vis}\, C \lhd h'\, o')$ for all $(o, o') \in \beta$

Accordingly for $p \approx^{C} p'$.

Observational purity

Let vis $C \triangleleft h\, o$ be the fields of object $h\, o$ visible in class C.

Def: $h \sim_{\beta}^{C} h'$ iff $(\text{vis}\, C \triangleleft h\, o) \sim_{\beta} (\text{vis}\, C \triangleleft h'\, o')$ for all $(o, o') \in \beta$

Accordingly for $p \approx^{C} p'$.

Lemma: p strongly pure iff
$h \dashv p \vdash\!\!\rightarrow k \Rightarrow k \sim_{\delta} h$, for $\delta = id_h$

Observational purity

Let vis $C \triangleleft h\, o$ be the fields of object $h\, o$ visible in class C.

Def: $h \sim_{\beta}^{C} h'$ iff $(\mathrm{vis}\, C \triangleleft h\, o) \sim_{\beta} (\mathrm{vis}\, C \triangleleft h'\, o')$ for all $(o, o') \in \beta$

Accordingly for $p \approx^{C} p'$.

Lemma: p strongly pure iff
$h \dashv\!p\!\mapsto k \Rightarrow k \sim_{\delta} h$, for $\delta = id_h$

Def: p is observationally pure outside D iff
$h \dashv\!p\!\mapsto k \Rightarrow k \sim_{\delta}^{C} h$, for $\delta = id_h$ and all $C \neq D$.

Observational purity

Let vis $C \triangleleft h\,o$ be the fields of object $h\,o$ visible in class C.

Def: $h \sim_{\beta}^{C} h'$ iff $(\text{vis}\,C \triangleleft h\,o) \sim_{\beta} (\text{vis}\,C \triangleleft h'\,o')$ for all $(o, o') \in \beta$

Accordingly for $p \approx^{C} p'$.

Lemma: p strongly pure iff
$h \dashv\!p\!\mapsto k \Rightarrow k \sim_{\delta} h$, for $\delta = id_h$

Def: p is observationally pure outside D iff
$h \dashv\!p\!\mapsto k \Rightarrow k \sim_{\delta}^{C} h$, for $\delta = id_h$ and all $C \neq D$.

Example: $memo$ is observationally pure outside class $Demo$, because arg and $isPr$ are not visible.

First steps

Thm: If p observationally pure outside D then $\mathbf{assert}\ p \approx^C \mathbf{skip}$.

Hazards:

- postconditions sensitive to garbage, e.g., “no $Cell$ exists”—break strong purity too, i.e., congruence for $\approx$
- violation of encapsulation breaks congruence:

  $\mathbf{proc}\ leak(s : Demo) : \mathbf{int}\ \{\ \mathbf{return}\ s.arg;\}$

  $\mathbf{assert}\ memo(s, x); y := leak(s) \ \not\approx^C \ \mathbf{skip}; y := leak(s)$
- encapsulation is difficult with mutable objects

Problem

Unfortunately, $\approx^C$ is not a congruence even without leaks:

$memo \not\approx^C memo$ —because $h \sim_{\beta}^{C} h'$ allows

$o.arg = 3, o.isPr =$ *false* in h and

$o.arg = 3, o.isPr =$ *true* in h'

Solution

Relation $\asymp$ is a D-simulation iff initialized and

- $h \asymp_{\alpha} g$ and $g \sim_{\beta} k$ implies $h \asymp_{\alpha \cdot \beta} k$
- $h \asymp_{\beta} k$ implies $h \sim_{\beta}^{C} k$ for all $C \neq D$
- $p \asymp p$ for every procedure p in class D

Solution

Relation $\asymp$ is a D-simulation iff initialized and

- $h \asymp_{\alpha} g$ and $g \sim_{\beta} k$ implies $h \asymp_{\alpha \cdot \beta} k$
- $h \asymp_{\beta} k$ implies $h \sim^{C}_{\beta} k$ for all $C \neq D$
- $p \asymp p$ for every procedure p in class D

Assumption (parametricity) [Banerjee,Naumann POPL02]:
$p \asymp p' \Rightarrow \mathcal{K}[p] \asymp \mathcal{K}[p']$ and moreover $p \asymp p$

Solution

Relation $\asymp$ is a D-simulation iff initialized and

- $h \asymp_\alpha g$ and $g \sim_\beta k$ implies $h \asymp_{\alpha\cdot\beta} k$
- $h \asymp_\beta k$ implies $h \sim_\beta^C k$ for all $C \neq D$
- $p \asymp p$ for every procedure p in class D

Assumption (parametricity) [Banerjee,Naumann POPL02]:
$p \asymp p' \Rightarrow \mathcal{K}[p] \asymp \mathcal{K}[p']$ and moreover $p \asymp p$

Def: p is observationally pure for $\asymp$ iff $h \dashv p \mapsto k \Rightarrow k \asymp_\delta h$.

Solution

Relation $\preceq$ is a D-simulation iff initialized and

- $h \preceq_\alpha g$ and $g \sim_\beta k$ implies $h \preceq_{\alpha\cdot\beta} k$
- $h \preceq_\beta k$ implies $h \sim_\beta^C k$ for all $C \neq D$
- $p \preceq p$ for every procedure p in class D

Assumption (parametricity) [Banerjee,Naumann POPL02]:
$p \preceq p' \Rightarrow \mathcal{K}[p] \preceq \mathcal{K}[p']$ and moreover $p \preceq p$

Def: p is observationally pure for $\preceq$ iff $h \dashv p \mapsto k \Rightarrow k \preceq_\delta h$.

This implies p observationally pure outside D.
And $\mathbf{assert}\ p \preceq \mathbf{skip}$, whence $\mathcal{K}[\mathrm{assert}\ p] \doteq{\approx}^C \mathcal{K}[\mathrm{skip}]$.

Proving observational purity I

Avoiding observational purity property per se:

Thm: If $p \asymp q$ for D-simulation $\asymp$, and q is *strongly pure*, then $\mathcal{K}[\mathbf{assert}\ p] \stackrel{.}{\approx}^{C} \mathcal{K}[\mathbf{skip}]$ for any $C \neq D$.

Use simulation in usual way to prove equivalence of implementations.

Proving observational purity I

Avoiding observational purity property per se:

Thm: If $p \asymp q$ for D-simulation $\asymp$, and q is *strongly pure*, then $\mathcal{K}[\textbf{assert}\ p] \dot{\approx}^{C} \mathcal{K}[\textbf{skip}]$ for any $C \neq D$.

Use simulation in usual way to prove equivalence of implementations.

Example: $prime \asymp memo$ where $h \asymp h'$ iff $h \sim^{C} h'$ for all $C \neq D$ and for every $o : Demo$, $o.arg \neq 0 \Rightarrow o.isPr =$ "whether $o.arg$ is prime".

Proving observational purity II

Typically $h \asymp h'$ iff $I(h)$ and $I(h')$ and $h \sim^{C} h'$ (all $C \neq D$).

- show I is invariant
- show $\sim^{C}$ preserved using info flow analysis
  - label cache $(arg, isPr)$ as secret, all else public; check secure flow
  - for pure procedures—"write confinement": $k \sim^{C}_{\delta} h$ with $\delta = id_h$

Proving observational purity II

Typically $h \asymp h'$ iff $I(h)$ and $I(h')$ and $h \sim^C h'$ (all $C \neq D$).

- show I is invariant
- show $\sim^C$ preserved using info flow analysis
  - label cache $(arg, isPr)$ as secret, all else public; check secure flow
  - for pure procedures—"write confinement": $k \sim^C_\delta h$ with $\delta = id_h$

But secret cache used for public output. Add flow rule:

$\text{assert } secret = open; \text{ return } secret$

Proving observational purity II

Typically $h \asymp h'$ iff $I(h)$ and $I(h')$ and $h \sim^C h'$ (all $C \neq D$).

- show I is invariant
- show $\sim^C$ preserved using info flow analysis
  - label cache $(arg, isPr)$ as secret, all else public; check secure flow
  - for pure procedures—"write confinement": $k \sim^C_\delta h$ with $\delta = id_h$

But secret cache used for public output. Add flow rule:

$\text{assert } secret = open;\ \text{return } secret$

(prove the assertion using I, e.g., $memo$ returns $prime(n)$)

Conclusion

- Strong purity: beware garbage-sensitive assertions [Calcagno et al, TCS]
- Observational purity: context of use matters
- Prove equal to something pure or something public
- Sălcianu and Rinard: A combined pointer and purity analysis for Java programs [MIT TR]
- Spec#: implementation and experience
- JML: full account of strong encapsulation, w/inheritance, exceptions, file I/O . . .