Abstraction Functions and Representation Invariants

Nate Foster
Spring 2019

Today’s music: *Never Change* by JAY-Z
Review

Previously in 3110:
• Specifying functions

Today:
• Specifying data abstractions
Back to: Audience of specification

- **Clients**
  - Spec informs what they must guarantee (preconditions)
  - Spec informs what they can assume (postconditions)

- **Implementers**
  - Spec informs what they can assume (preconditions)
  - Spec informs what they must guarantee (postconditions)

But the spec isn’t **enough** for implementers...
REPRESENTATION TYPES
Representation types

• **Q:** How to interpret the representation type as the data abstraction?
  • **A:** Abstraction function

• **Q:** How to determine which values of representation type are meaningful?
  • **A:** Representation invariant
Abstraction function

client’s view
{1,2} {7} abstract value: set

implementer’s view
[1;2] [2;1] [7] concrete value: list (no dups)

abstraction barrier

the black arrows are the abstraction function
Abstraction function maps valid concrete values to abstract values
Documenting the AF

- Above rep type in implementation you write:
  (* AF: comment *)

- Write it first before implementing operations
Representation types

• **Q:** How to interpret the representation type as the data abstraction?
  • **A:** Abstraction function

• **Q:** How to determine which values of representation type are meaningful?
  • **A:** Representation invariant
REPRESENTATION INVARIANTS
Abstraction function

client’s view

implementer’s view

{1,2} {7}

abstract value: set

[1;2] [2;1] [7]

concrete value: list (no dups)

abstraction barrier
Representation invariant

Abstract value: set

Valid concrete values: satisfy rep. invariant

Invalid concrete values: do not satisfy rep. invariant

The thick red line is the rep. invariant
Rep. invariant distinguishes valid concrete values from invalid concrete values
Documenting the RI

- Above rep type in implementation you write: 
  (* RI: comment *)

- Write it first before implementing operations
Rep. invariant implicitly part of every precondition and every postcondition in abstraction
Invariant may temporarily be violated

concrete input \[\xrightarrow{\text{operation}}\] concrete operation \[\xrightarrow{\text{output}}\] concrete output

RI holds \[\xrightarrow{\text{operation}}\] RI maybe violated \[\xrightarrow{\text{operation}}\] RI holds

Demo
Discussion

When and how would you implement a RI as part of a data abstraction?
Implementing the RI

**Idiom**: if RI fails then raise exception, otherwise return concrete value
Recap

- **Q**: How to *interpret* the representation type as the data abstraction?
  - **A**: Abstraction function

- **Q**: How to determine which values of representation type are *meaningful*?
  - **A**: Representation invariant
Upcoming events

• [Today] Foster Office Hours 1:15-2:15pm

This is invariant.

THIS IS 3110