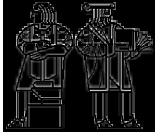


6.170 Lecture 5 Abstraction and Specification, Continued



John Guttag
MIT EECS

Using Specification Ensures Modifiability

Replace implementation of the abstraction
Need not change using programs

Might have to recompile
But never reprogram

Why change implementation of abstraction?
Fix bug (I.e., failure to correspond to specification)
Using programs should not depend upon bug!
Port to new platform
Improve efficiency

“I could really speed up foo, but the last time I changed foo, bar stopped working -- and I don’t understand bar worth a ...”

An (Abstract) Example

Specification: reference manual for Java

Implementation: compiler and runtime environment

Locality

Write programs without understanding the compiler
Write compiler without understanding programs that will use it

Modifiability

Programs port across compilers
Compiler writers free to add optimizations

Specifying Procedures

Procedure - code written in a formal executable language

Perhaps, but not necessarily, Java
Some library routines may be hand coded in assembler

Specification - written in a specification language

Usually not formal (or executable)
Most often highly stylized natural language

Each of these is tangible text

Each denotes something less tangible

A Short Mathematical Digression

Relation: A set (perhaps infinite) of ordered pairs

Can define a relation by writing its characteristic function

$R: S1, S2 \rightarrow \text{Boolean}$

$x \in S1$ is in the domain of R if

$\exists y \in S2 \text{ s.t. } \langle x, y \rangle \in R$

$y \in S2$ in the range of R if

$\exists x \in S1 \text{ s.t. } \langle x, y \rangle \in R$

A relation is total iff

$\forall x \in S1 \text{ there } \exists y \in S2 \text{ s.t. } \langle x, y \rangle \in R$

A relation that is not total is partial

A relation, F , is a function if

$\forall x \in S1, \exists \text{ at most one } y \in S2 \text{ s.t. } \langle x, y \rangle \in R$

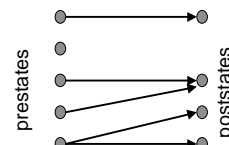
What Specifications Denote

Think of specification as denoting a transition relation

Spec: $\langle \text{State, Arguments} \rangle, \langle \text{State, Results} \rangle \rightarrow \text{Boolean}$

Why a relation?

Undesirable to constrain abstraction to a single mapping
E.g., choose an element from a set and delete it



The Transition Relation, Again

Some procedures perform mutations
Observable behavior of a procedure is a relation on states
Transition relation relates prestates to poststates.
Transition relation of a procedure is not necessarily:
total (some prestates cause failures, no poststate)
injective (multiple prestates \Rightarrow same poststate)
void remLast(List l)
functional (one prestate \Rightarrow multiple poststates)
int choose(IntSet x) // caller cannot see rep of IntSet

Think of code as also denoting a relation
Proc: <State, Arguments>, <State, Results> \blacklozenge Boolean

© John Chapin/John Guttag Spring 2000 Slide 7

Satisfaction of a Specification

Let P be an implementation and S a specification
P satisfies S iff
Relation denoted by P is a subset of relation denoted by S
(A white lie)

The statement “P is correct” is meaningless
Though often made!

If P does not satisfy S, either (or both!) could be “wrong”
“One person’s feature is another person’s bug.”
Usually better to change the program
Hate to publish amendments to library specifications

© John Chapin/John Guttag Spring 2000 Slide 8

A Trivial Example

Specification
int anyInt (int x)
requires x = 0
effects returns any int

Code
int anyInt(int x) {
return 3;
}

Does this program satisfied the specification?
Not according to the definition I just supplied
S = {<0, 0>, <0, 1>, <0, -1>, ..., <0, 3>, ...}
P = {<0, 3>, <1, 3>, <-1, 3>, ...}

© John Chapin/John Guttag Spring 2000 Slide 9

Satisfaction Revisited

Let P | D = P restricted to the domain D
Remove from P all pairs whose first member is not in D

P satisfies S iff
P | (Domain of S) is a subset of relation denoted by S
(Still a white lie)

© John Chapin/John Guttag Spring 2000 Slide 10

General Form of a Specification of a Procedure

<qualifiers> <return> <name> <formals>
<requires>
<modifies>
<effects>

public static Boolean isPrime (int i)
requires i > 0
modifies nothing
effects returns true if i is a prime number and false otherwise

requires \Rightarrow (modifies \leftrightarrow) effects
Recall that $\forall X, X \Rightarrow \text{true}, \text{false} \Rightarrow X$

© John Chapin/John Guttag Spring 2000 Slide 11

Another Example

int find(int[] a, int value)
requires value occurs in a
modifies nothing
effects returns an i such that a[i] = value

requires (the precondition)
Client must ensure prestate satisfies the precondition

modifies
No side effects

effects
Defines possible poststates using values from the prestate
Implementer must guarantee the postcondition
If precondition holds

© John Chapin/John Guttag Spring 2000 Slide 12

6.170
Spring 2000

6.170 **Another Example**


```
public static int test(Vector v, Object o, Object o1)
  requires o occurs in v
  modifies v
  effects returns an i such that ith element of v is o
    & sets the ith element of v to o1
    & makes no other changes to v
-----
public static int test(Vector v, Object o, Object o1) {
  int i = 0;
  while (true) {
    if (v.elementAt(i) == o) {
      v.setElementAt(o1, i);
      return i;}
    i++;}
}
```

© John Chapin/John Guttag Spring 2000 Slide 13

6.170 **requires \Rightarrow (modifies \Leftarrow effects)**

When is specification S1 weaker than S2?
When $\forall P, (P \text{ satisfies } S2) \Rightarrow (P \text{ satisfies } S1)$

We can weaken a specification by
Making requires harder to satisfy
Adding things to modifies clause
Making effects easier to satisfy



What is the strongest (most constraining) requires clause?
true

What is the strongest (most constraining) effects clause?
false

What is the strongest (most constraining) modifies clause
modifies nothing

© John Chapin/John Guttag Spring 2000 Slide 14