When do I need an invariant?

CS 5010 Program Design Paradigms "Bootcamp"

Lesson 7.3

Learning Objectives

- At the end of this lesson, the student should be able to
  - decide whether a purpose statement needs an invariant or not.

When do I need an invariant?

- It all depends on the purpose statement.
- If your code fulfills the purpose statement for any arguments of the types listed in the contract, you don't need an invariant.
- If the function fulfills its purpose statement only for certain values or combinations of values of the arguments, then you must document that restriction with a WHERE-clause.

What kind of things belong in an invariant?

- If the function needs additional information that is not in the arguments, then you need an invariant to document the needed information
- What kind of information might you want?
  - context information (e.g. we are position **n** in the list)
  - other knowledge that isn't expressed in the contract (e.g. we've figured out the ball isn't going to bounce).

Whose responsibility is it?

- The invariant, along with the contract, sets down the assumptions that each function makes about the arguments that it processes
- It is up to each caller of the function to make sure that the invariant is true at every call.
- The function gets to assume that the invariant is true.

Example:

```
;; ball-normal-motion : Ball -> Ball
;; GIVEN: a Ball



;; RETURNS: the state of the ball after a
;; tick.
(define (ball-normal-motion b)
  (make-ball
    (+ (ball-x-pos b) BALLSPEED)))
```

Doesn't work for every Ball!..
Needs more information

Invariant provides the
necessary information

Once more: When do I need an invariant?

- If your code fulfills the purpose statement for any arguments of the types listed in the contract, you don't need an invariant.
- If the function only works for certain values or combinations of values of the arguments, then you must document the assumptions that it needs with a WHERE-clause (i.e. an invariant).

What needs to be in my purpose statement?

- The purpose statement must account for all the parameters.
  - if it doesn't then either you are passing more parameters than you need, or there's something going on that you haven't described.
- The RETURNS clause must describe the value returned by the function for all possible values of the parameters.
- If the RETURNS clause describes the value returned by the function only for some values of the arguments or some combination of arguments, then that restriction must be stated in a WHERE clause.
- It becomes the responsibility of the caller to guarantee that the restriction is satisfied.

Summary: When do I need an invariant?

- It all depends on your purpose statement!
- If the function needs additional information to fulfill its stated purpose, and that information is not in the arguments, then you need an invariant to document the needed information.
- It is up to each caller of the function to make sure that the invariant is true at every call.

Summary

- The student should now be able to
  - decide whether a purpose statement needs an invariant or not.

Next Steps

- If you have questions about this lesson, ask them on the Discussion Board
- Go on to the next lesson