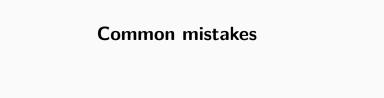
Software Engineering: Tutorial 10

David Voigt January 12th, 2023

Agenda

- 1. Common homework mistakes
- 2. Recap: Tests
- 3. Exercises



Converting a precondition into a postcondition

The general idea is to make a distinction in the postcondition: "If A is null, then ... else ..."

Example: Task 1: findIndex

Preconditions:

- searched is not null
- list is not null
- all elements of list are not null

Postconditions:

result is between -1 and the length of list - 1

Before conversion

```
def findIndex(searched: String, list: List[String]): Int = {
  require(
    searched != null
    && list != null
    && list.forall(x => { x != null })
  )
  ...
  result
} ensuring { i =>
    i >= -1 && i <= list.length - 1
}</pre>
```

Solution

```
def findIndex(searched: String, list: List[String]): Int = {
  // notice the precondition regarding searched is gone
  require(
    && list != null
    && list.forall(x => { x != null })
  result.
} ensuring { idx =>
  if searched != null && list.contains(searched) then
    -1 < idx && idx < list.length
  else
   i == -1
```

Example: Task 1: getNth

Convert preconditions regarding n to a postcondition

• $0 \le n <$ list.length

Solution

```
def getNth(n: Int, list: List[String]): String = {
  require(list != null && list.forall(_ != null))
  ...
} ensuring { elem =>
  if n < 0 || n >= list.length then
    elem == ""
  else
    elem != null
}
```

Nil is not the same as null

- Even though they have similar meanings, Nil and null are not to be confused
- Nil is the same as an empty list: List() == Nil
- Whereas null is a subtype of every other type, but cannot be accessed without an exception being thrown

Weak postconditions

Given this function

```
def reverse[T](list: List[T]): List[T] = {
  require(???)
  var result: List[T] = List(list.head)
  list.tail.foreach { elem =>
    result = elem :: result
  }
  result
} ensuring (???)
```

- What precondition is neccessary?
- Which postconditions can you think of?

Precondition

- The list must be non-empty and not null
- We do not need to check that all element are non-null since we are not directly accessing them!

Postconditions

- (obligatory list is not null)
- The list contains exactly the same elements as the input list
- Without ensuring the reverse of the resulting list is the input list, we can ensure, that the head element is now the last element and vice versa.

Solution

```
def reverse[T](list: List[T]): List[T] = {
  require(list != null)
  var result: List[T] = List(list.head)
  list.tail.foreach { elem =>
    result = elem :: result
  result
} ensuring { xs =>
  xs != null
  && xs.forall(list.contains(_))
  && list.forall(xs.contains())
  && xs.head == list.last
  && xs.last == list.head
}
```



Test coverage

- Test coverage meassures how much code your tests cover
- There is a distinction between code coverage, test coverage and control-flow coverage
 - Test coverage is measured by (but not limited to) how much the the tests the code consisting of e.g. expressions, functions, classes
 - Test coverage is also measured by the control-flow coverage, that is, how many branches in match or if-else expressions/statements have been executed by the tests

Unit tests (example based tests)

Test the behavioural correctness of individual software components (units) with varying granularity, that is, on functions, classes, modules, . . .

Steps

- 1. Setup a know state
- 2. Interact with the "unit"
- 3. Assert that the observed behaviour matches the expected
- 4. Teardown the state and check if it is in the expected end state

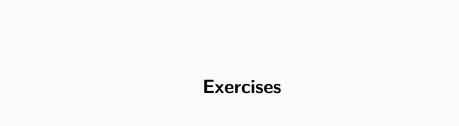
Property tests (property based tests)

Instead of testing with handpicked examples, we formulate general properties we expect the tested system to uphold for all possible inputs.

Example¹

```
// unit teste
assert(List().reverse == List())
assert(List(1,2,3).reverse == List(3,2,1))
...
// property tests
property("reversing a list twice corresponds to the identity") {
    forAll(lists) { xs =>
        assert(xs == xs.reverse.reverse)
    }
}
```

¹This is not a sufficient test but only a neccessary



Link

https://github.com/se-tuebingen-exercises/tut7-exercise10

Usefull resources

- https://github.com/se-tuebingen-exercises/se-lecturetesting/tree/main/src/test/scala
- https://scalameta.org/munit/docs/getting-started.html
- https:
 - //scalameta.org/munit/docs/integrations/scalacheck.html
- Guide for property testing on the forum