ITI8610 Software Assurance 2017

2nd part of Lecture on 29/11/2017

Leonidas Tsiopoulos and Jüri Vain

Multi-View Contracts with JML „Specification Cases“, based on Chapters 7 and 8 of book:

„Deductive Software Verification – The KeY Book“

Multi-View Contracts

- Contracts handle interface properties representing the ***assumptions*** (pre-conditions) and the ***guarantees*** (post-conditions) under these assumptions.

- A *complete* contract can be a conjunction of ***multiple viewpoints*** (aspects), each covering a specific concern (behavioral, timing, safety, etc.) of the design and specified by an individual (view) contract.

Multi-View Contracts - JML

- How multi-view contracts can be specified with JML?

- When specifying a method, it is often useful—and sometimes necessary—to describe the behavior separately for different parts of the prestate/input space.

- JML allows the formulation of structured specifications.

- The behavior of a method does not need to be formulated as a single contract, but can be split up into multiple, ***possibly nested*** individual contracts that model different parts of the behavior.

JML Specification Cases

- The structuring mechanism for that is the ***specification case***, each of which is specific for a particular pre-condition.

- Specification cases are combined by the **`also`** keyword.

Specification case example 1 in JML

- Specification of a method adding an integer element to a set:

…

```
/*@ requires size < limit && !contains(elem); -One case
@ ensures \result == true;
@ ensures contains(elem);
@ ensures (\forall int e; e != elem;
                  contains(e) <==> \old(contains(e)));
@ ensures size == \old(size) + 1;
```

Specification case example 1 in JML – Cont.

```
@ also
@
@ requires size == limit || contains(elem); - The other case
@ ensures \result == false;
@ ensures (\forall int e;
@ contains(e) <==> \old(contains(e)));
@ ensures size == \old(size);
@*/
public boolean add(int elem) {/*...*/}
```

Specification case example 2 in JML- Functional Behavior Contract

```
class Example {
/*@ public behavior
@ requires a!= null                                    // Precondition
@ && \invariant_for(this) && to >= from;  // Precondition
@ signals (Throwable e)                          // Functional Postcondition
@ (e instanceof IndexOutOfBoundsException ==>
@ from < 0 || to >= a.length)
@ && (e instanceof Throwable ==> \invariant_for(this))
@ && (e instanceof IndexOutOfBoundsException);
@ ensures a[\result] >= a[from] && \invariant_for(this); // Functional
Postcondition
@ diverges false;
@ assignable \nothing;
```

Specification case example 2 in JML – Dependency Contract

```
 @ also
  @ requires array != null      // Common Preconditions to Functional Contract
  @ && \invariant_for(this) && to >= from;
  @ accessible a[*]; // Condition for Dependency
  @*/
   /*@ helper */
   public int maxIntArray(/*@nullable*/int[] a, int from, int to) {
  // ...
  }
}
```