Approximate Methods in Geometry

Practical Part

Jens K. Mueller

jkm@informatik.uni-jena.de

Department of Mathematics and Computer Science
Friedrich-Schiller-University Jena

Wednesday 17$^{\text{th}}$ November, 2010

Today's lecture: Contract Programming

Documentation

Purpose

- Usage
  (Reference) Manual, etc.
- Maintaining
  Design Decisions, Overall Picture
- Understanding
  Commenting difficult/optimized code

Problems

- Keeping code and documentation in sync
- Too verbose

Contract Programming

Concepts in Contract Programming

Assertions, Preconditions, Postconditions, and Invariants

- Assertion
  Runtime check against a condition
- Precondition
  Condition that need to be fulfilled by the caller (parameters, resources)
- Postcondition
  Condition that are guaranteed by the callee (assuming normal return)
- Invariant
  Condition that holds during a computation

Function Signature and Contracts

- Type and number of arguments and return values

  ```
  double sqrt(const(double) x)
  ```

- Purity and Throwing

  ```
  pure nothrow double sqrt(double x)
  ```

You can use function signatures to impose contracts.

Writing Contracts

- Contracts in non-machine readable form are of less use (e.g. in form of documentation), if they cannot be checked automatically.
- Contracts needs to be written such that you can safely remove them without changing the logic of the program. No assert($++x > y$)!

Contracts in D

```
pure nothrow double sqrt(double x)
in {
    assert(x >= 0, "x has to be greater
                    equal than 0");
}
out {
    assert(x == result * result,
           text("Square of ", result, " does
                 not equal ", x));
}
body {
    // ...
}
```

Contracts in D (cont.)

```
// invariant in a class/struct
invariant() {
    // ...
}
```