Formal Methods for Java Lecture 14: Proving with Key

Jochen Hoenicke



Software Engineering Albert-Ludwigs-University Freiburg

December 9, 2011

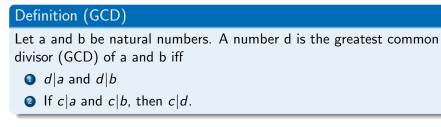
Java code to compute gcd of non-negative numbers:

```
public static int gcd(int a, int b) {
    while (a != 0 && b != 0) {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }
    return (a > b) ? a : b;
}
```

Lets prove it with KeY-System.

Specification

We first need a specification.



d|a means d divides a. $d|a :\Leftrightarrow \exists q.d * q = a$

JML Specification

So lets start proving ...

Loop-Invariant

What is the loop invariant?

The algorithm changes a and b, but the gcd of a and b should stay the same.

In fact the set of common divisors of a and b never changes. This suggests the following invariant:

 $orall d.(d| old(a) \wedge d| old(b) \leftrightarrow d|a \wedge d|b)$

```
In JML this can be specified as:
    /*@ loop_invariant a >= 0 && b >= 0 &&
    @ (\forall int d; true;
    @ (\exists int q; \old(a) == q*d)
    @ && (\exists int q; \old(b) == q*d)
    @ <==>(\exists int q; a == q*d) && (\exists int q; b == q*d)
    @ );
    @ assignable a, b;
    @ decreases a+b;
    @*/
```