Formal Methods for Java

Lecture 14: Proving with Key

Jochen Hoenicke

Software Engineering
Albert-Ludwigs-University Freiburg

December 9, 2011

Case Study: Euklid's Algorithm

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.

Definition (GCD)

Let a and b be natural numbers. A number d is the greatest common divisor (GCD) of a and b iff

1. $d|a$ and $d|b$
2. If $c|a$ and $c|b$, then $c|d$.

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

The specifation can be converted to JML:

```
/*@
  @ requires a >= 0 && b >= 0;
  @ ensures \result >= 0;
  @ ensures (\exists int q; \result*q == a) &&
  @        (\exists int q; \result*q == b) &&
  @  (\forall int c;
  @       (\exists int q; c*q == a) && (\exists int q; c*q == b);
  @       (\exists int q; c*q == \result));
  @*/
public static int gcd(int a, int b)
```

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:

$$\forall d.(d| \backslash\texttt{old}(a) \wedge d| \backslash\texttt{old}(b) \leftrightarrow d|a \wedge d|b)$$

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;
  @*/
```