# Formal Methods for Java Lecture 25: Proving a JML-Program with KeY

#### Jochen Hoenicke



Software Engineering Albert-Ludwigs-University Freiburg

January 29, 2013

Jochen Hoenicke (Software Engineering)

Formal Methods for Java

January 29, 2013 1 / 7



- Theorem Prover
- Developed at University of Karlsruhe
- http://www.key-project.org/.
- Theory specialized for Java(Card).
- Can generate proof-obligations from JML specification.
- Underlying theory: Sequent Calculus + Dynamic Logic

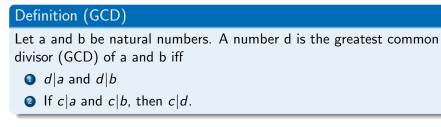
Java code to compute gcd of non-negative numbers:

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

```
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) &&
    @
    @ (\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)
```

#### The rule while\_invariant\_with\_variant\_dec

The rule while\_invariant\_with\_variant\_dec takes an invariant *inv*, a modifies set  $\{m_1, \ldots, m_k\}$  and a variant v. The following cases must be proven.

- Initially Valid:  $\implies inv \land v \ge 0$
- Body Preserves Invariant:

$$\implies \{m_1 := x_1 \| \dots \| m_k := x_k\} (inv \land [\{b = COND; \}]b = true \\ \rightarrow \langle BODY \rangle inv$$

Use Case:

$$\implies \{m_1 := x_1 \| \dots \| m_k := x_k\} (inv \land [\{b = COND; \}]b = \mathsf{false} \\ \rightarrow \langle \dots \rangle \phi$$

• Termination:

$$\implies \{m_1 := x_1 \| \dots \| m_k := x_k\} (inv \land v \ge 0 \land [\{b = COND; \}] b = true \\ \rightarrow \{old := v\} \langle BODY \rangle v \le old \land v \ge 0$$

Jochen Hoenicke (Software Engineering)

Formal Methods for Java

### 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;
    @*/
```