Proof of the equivalence of the order-based and the ring-based definitions of the GCD

Frédéric Boulanger

May 16, 2016

Contents

1 Introduction 1

2 Common divisors 1

3 Greatest common divisor, defined on order 3

4 Greatest common divisor, ring definition 5

5 Proof of the equivalence of the two definitions 5

1 Introduction

This Isabelle theory presents a proof of the equivalence of the *natural* definition of the Greatest Common Divisor for integers (as a common divisor that is greater than all other common divisors), and the definition on rings (as a common divisor that is divided by all other common divisors).

We finally show the equivalence between our definitions of the GCD using predicates, and the functional definition in Isabelle, which relies on Euclid's algorithm.

theory *IntGCD*
imports *Main GCD*

begin

2 Common divisors

We define a predicate for characterizing common divisors of two integers, and prove some theorems that will be needed for proving properties of the GCD.

```
definition common_div :: "int ⇒ int ⇒ int ⇒ bool"
where
  "common_div a b p ≡ p dvd a ∧ p dvd b"

lemma common_div_comm:
  "common_div a b p = common_div b a p"
using common_div_def by blast
```

Two integers have 0 as common divisor only if one of them is 0:

```
lemma cdiv_0: "common_div a b 0 ⟷ a = 0 ∧ b = 0"
using common_div_def by simp
```

Common divisors are not changed by absolute values:

```
theorem common_div_abs:
  "common_div a b d = common_div |a| |b| d"
using common_div_def by simp
```

The common divisors of a and b are the common divisors of $a - b$ and b. This theorem is the basis of the proof of the equivalence of the two definitions of the GCD.

```
lemma common_div_ab_dir:
  assumes "common_div a b p"
  shows   "common_div (a - b) b p"
proof -
  from assms and dvd_def
    obtain ka where "a = p * ka" unfolding common_div_def by blast
  moreover from assms and dvd_def
    obtain kb where "b = p * kb" unfolding common_div_def by blast
  ultimately have "a - b = (ka - kb) * p" by algebra
  hence "p dvd (a - b)" by simp
  moreover from assms have "p dvd b" using common_div_def by simp
  ultimately show ?thesis using common_div_def by simp
qed

lemma common_div_ab_rev:
  assumes "common_div (a - b) b p"
  shows   "common_div a b p"
proof -
  from assms and dvd_def
    obtain ka where "(a - b) = p * ka" unfolding common_div_def by blast
  moreover from assms and dvd_def
    obtain kb where "b = p * kb" unfolding common_div_def by blast
  ultimately have "a = (ka + kb) * p" by algebra
  hence "p dvd a" by simp
  moreover from assms have "p dvd b" using common_div_def by simp
  ultimately show ?thesis using common_div_def by simp
qed
```

```
theorem common_div_ab:"common_div a b p = common_div (a - b) b p"
using assms and common_div_ab_dir and common_div_ab_rev by blast

theorem common_div_ba:"common_div a b p = common_div a (b - a) p"
using assms and common_div_ab and common_div_comm by simp
```

3 Greatest common divisor, defined on order

Here we define the greatest common divisor using the order on integers. We define a predicate for identifying upper bounds of all common divisors:

```
definition no_greater_div :: "int ⇒ int ⇒ int ⇒ bool"
where
  "no_greater_div a b g ≡ ∀ p. common_div a b p ⟶ p ≤ g"
```

Such an upper bound is always strictly positive:

```
lemma greater_div_pos: "no_greater_div a b g ⟹ g > 0"
proof -
  assume h:"no_greater_div a b g"
  have "1 dvd a" by simp
  moreover have "1 dvd b" by simp
  ultimately have "common_div a b 1" using common_div_def by simp
  with h have "g ≥ 1" using no_greater_div_def by simp
  thus ?thesis by simp
qed

theorem greater_div_abs:
  "no_greater_div a b g = no_greater_div |a| |b| g"
proof
  assume h:"no_greater_div a b g"
  {
    fix p assume "common_div |a| |b| p"
    with common_div_abs have "common_div a b p" by simp
    with h have "p ≤ g" using no_greater_div_def by simp
  }
  thus "no_greater_div |a| |b| g" using no_greater_div_def by simp
next
  assume h:"no_greater_div |a| |b| g"
  {
    fix p assume "common_div a b p"
    with common_div_abs have "common_div |a| |b| p" by simp
    with h have "p ≤ g" using no_greater_div_def by simp
  }
  thus "no_greater_div a b g" using no_greater_div_def by simp
qed
```

The GCD is a common divisor which is an upper bound of the common divisors:

```
definition is_gcd :: "int ⇒ int ⇒ int ⇒ bool"
```

```
where
  "is_gcd a b g ≡ common_div a b g ∧ no_greater_div a b g"
```

We now derive properties of the GCD from properties of divisors.

```
lemma gcd_comm: "is_gcd a b g = is_gcd b a g"
using is_gcd_def and common_div_def and no_greater_div_def by auto

lemma gcd_pos: "is_gcd a b g ⟹ g > 0"
using is_gcd_def and greater_div_pos by blast

lemma gcd_neq_zero:
  assumes "is_gcd a b g"
  shows   "g ≠ 0"
using gcd_pos[OF assms] by simp

lemma gcd_a0:
  assumes "a ≠ 0"
  shows   "is_gcd a 0 |a|"
proof -
  from dvd_imp_le_int[OF assms] have "∀ p. p dvd a ∧ p dvd 0 ⟶ |p| ≤ |a|"
    by simp
 hence "no_greater_div a 0 |a|" unfolding no_greater_div_def and common_div_def
    by auto
  thus ?thesis using abs_div is_gcd_def common_div_def by simp
qed

lemma gcd_0b:
  assumes "b ≠ 0"
  shows   "is_gcd 0 b |b|"
using assms and gcd_a0 and gcd_comm by auto

lemma gcd_self:
  assumes "a ≠ 0"
  shows   "is_gcd a a |a|"
proof -
  from dvd_imp_le_int[OF assms] have "∀ p. p dvd a ∧ p dvd a ⟶ |p| ≤ |a|"
    by simp
 hence "no_greater_div a a |a|" unfolding no_greater_div_def and common_div_def
    by auto
  moreover from abs_div have "common_div a a |a|" using common_div_def by
simp
  ultimately show ?thesis using is_gcd_def by simp
qed

lemma gcd_abs:
  "is_gcd a b g = is_gcd |a| |b| g"
using is_gcd_def and common_div_abs and greater_div_abs by simp

theorem gcd_ab:"is_gcd a b g = is_gcd (a - b) b g"
```

```
using assms is_gcd_def no_greater_div_def common_div_ab by simp

theorem gcd_ba:"is_gcd a b g = is_gcd a (b - a) g"
using assms and gcd_ab and gcd_comm by simp
```

With the definition of the GCD based on the order on integers, the GCD is unique.

```
lemma gcd_unique:
  assumes "is_gcd a b g"
  and     "is_gcd a b g'"
  shows   "g = g'"
proof -
  from assms(1) have "∀ p. common_div a b p ⟶ p ≤ g"
    using is_gcd_def and no_greater_div_def by simp
  moreover from assms(2) have "common_div a b g'"
    using is_gcd_def by simp
  ultimately have 1:"g' ≤ g" by simp
  from assms(2) have "∀ p. common_div a b p ⟶ p ≤ g'"
    using is_gcd_def and no_greater_div_def by simp
  moreover from assms(1) have "common_div a b g" using is_gcd_def by simp
  ultimately have 2:"g ≤ g'" by simp
  from 1 and 2 show ?thesis by simp
qed
```

4 Greatest common divisor, ring definition

We now define the greatest common divisor as one which is divided by all other common divisors. We keep the positive one, so that this definition match the previous one.

```
definition is_gcd_div :: "int ⇒ int ⇒ int ⇒ bool"
where
  "is_gcd_div a b g ≡ (g > 0) ∧ common_div a b g
                  ∧ (∀ p. common_div a b p ⟶ p dvd g)"
```

With this definition, the GCD cannot be null. Although the GCD of 0 and 0 is 0 using the ring definition of the GCD, this makes no sense with regard to the definition based on the order on integers: any integer is a common divisor of 0 and 0, so there is no greatest one.

```
lemma gcd_div_neq_zero:"is_gcd_div a b g ⟹ g ≠ 0"
using is_gcd_div_def by simp
```

5 Proof of the equivalence of the two definitions

We can now show that both definitions of the GCD are equivalent. Showing that being the GCD with the ring definition implies being the GCD with the order definition is straightforward:

```
lemma gcd_div_inf:
  assumes "is_gcd_div a b g"
  shows   "is_gcd a b g"
proof -
  from assms have 1:"common_div a b g" using is_gcd_div_def by simp
  from assms have 2:"∀ p. common_div a b p ⟶ p dvd g"
    using is_gcd_div_def by simp
  from assms have 3:"g > 0" using is_gcd_div_def by simp
  have "∀ p. common_div a b p ⟶ p ≤ g"
  proof -
    {
      fix p assume h:"common_div a b p"
      with 2 have dp:"p dvd g" by simp
      from 3 have "|g| = g" and "g ≠ 0" by simp+
      with zdvd_imp_le[OF dp] have "p ≤ g" by simp
    }
    thus ?thesis by auto
  qed
  thus ?thesis using 1 and is_gcd_def and no_greater_div_def by simp
qed
```

The other way is more difficult. We use induction on natural numbers with an upper bound on the sum of the absolute values, and use the fact that *is_gcd (a - b) b g = is_gcd a b g*

```
lemma cdiv_div_gcd:
  "(|a| + |b| > 0) ∧ (nat (|a| + |b|) ≤ Suc n) ∧ is_gcd |a| |b| g
  ⟹ (∀ p. common_div |a| |b| p ⟶ p dvd g)"
proof (induction n arbitrary: a b)
  case 0
    hence pos:"|a| + |b| > 0"
      and leq1:"|a| + |b| ≤ 1"
      and gcd:"is_gcd |a| |b| g" by auto
    show ?case
    proof (cases "a = 0")
      case True
        with leq1 and pos have "|b| = 1" by simp
        moreover with this have "g = 1"
          using gcd_0b[of "|b|"] and ‘a = 0‘ and gcd and gcd_unique by simp
        ultimately show ?thesis using common_div_def by simp
    next
      case False
        with leq1 and assms have "|a| = 1" and "b = 0" by auto
        moreover with this have "g = 1"
          using gcd_a0[of "|a|"] and ‘¬a = 0‘ and gcd and gcd_unique by simp
        ultimately show ?thesis using common_div_def by simp
    qed
next
    case (Suc k)
      from Suc.prems have
```

```
  1:"nat (|a| + |b|) ≤ Suc (Suc k)" and
  2:"is_gcd |a| |b| g" and 3:"|a| + |b| > 0" by auto
show ?case
proof (cases "nat (|a| + |b|) ≤ Suc k")
  case True
    thus ?thesis using Suc.IH and 2 and 3 by simp
next
  case False
    with 1 have ab:"nat (|a| + |b|) = Suc (Suc k)" by simp
    show ?thesis
    proof (cases "|a| ≥ |b|")
      case True
        show ?thesis
        proof (cases "|b| = 0")
          case True
            with ab have "|a| ≠ 0" by simp
            with `|b| = 0` and 2 and gcd_a0[of "|a|"] and gcd_unique
              have "g = |a|" by simp
            thus ?thesis using common_div_def by simp
        next
          case False
            with ab have "nat (|a| - |b| + |b|) ≤ Suc k"
              using `|a| ≥ |b|` by simp
            moreover from 2 and gcd_ab have "is_gcd (|a| - |b|) |b| g"
              by simp
            moreover from `|b| ≠ 0` and `|a| ≥ |b|` have "|a| + |b| - |b| > 0"
              by simp
            ultimately show ?thesis
             using Suc.IH[of "|a| - |b|" "|b|"] and `|a| ≥ |b|` and common_div_ab
              by simp
        qed
    next
      case False
        show ?thesis
        proof (cases "|a| = 0")
          case True
            with ab have "|b| ≠ 0" by simp
            with True and 2 and gcd_0b[of "|b|"] and gcd_unique
              have "g = |b|" by simp
            thus ?thesis using common_div_def by simp
        next
          case False
            with ab have "nat (|a| + |b| - |a|) ≤ Suc k"
              using `¬ |a| ≥ |b|` by simp
            moreover from 2 and `¬ |a| ≥ |b|` and gcd_ba
              have "is_gcd |a| (|b| - |a|) g" by simp
            moreover from `|a| ≠ 0` and `¬ |a| ≥ |b|`
              have "|a| + |b| - |a| > 0" by simp
            ultimately show ?thesis
```

```
          using Suc.IH[of "|a|" "|b| - |a|"] and ‘¬ |a| ≥ |b|‘
           and common_div_ba by simp
      qed
    qed
  qed
qed
```

We can now remove the condition used to make the induction on n:

```
lemma common_div_gcd:
  assumes "a ≠ 0 ∨ b ≠ 0"
  and     "is_gcd |a| |b| g"
  shows   "(∀ p. common_div |a| |b| p ⟶ p dvd g)"
proof -
  from assms(1) have 1:"|a| + |b| > 0" by auto
  have "nat (|a| + |b|) ≤ Suc (nat (|a| + |b|))" by simp
  with assms and 1 have
    "(|a| + |b| > 0) ∧ (nat (|a| + |b|) ≤ Suc (nat (|a| + |b|))) ∧ is_gcd |a| |b| g"
    by blast
  from cdiv_div_gcd[OF this] show ?thesis .
qed
```

Therefore, we have the equivalence of the two definitions when a and b are not both null:

```
lemma gcd_inf_div:
  assumes "is_gcd a b g"
  and     "a ≠ 0 ∨ b ≠ 0"
  shows   "is_gcd_div a b g"
proof -
  from assms(1) have "is_gcd |a| |b| g" using gcd_abs by simp
  with assms(2) and common_div_gcd
    have "(∀ p. common_div |a| |b| p ⟶ p dvd g)" by simp
  hence "(∀ p. common_div a b p ⟶ p dvd g)" using common_div_abs by simp
  moreover from assms(1) have "common_div a b g" using is_gcd_def by simp
  moreover from assms(1) have "g > 0" using gcd_pos by simp
  ultimately show ?thesis using is_gcd_div_def by simp
qed

theorem gcd_inf_div_eq:
  assumes "a ≠ 0 ∨ b ≠ 0"
  shows   "is_gcd a b g = is_gcd_div a b g"
using assms and gcd_div_inf and gcd_inf_div by blast
```

The condition on the simultaneous nullity of a and b comes from the fact that there is no GCD of 0 and 0 with the definiton based on the order on integers:

```
lemma any_common_div_0:"common_div 0 0 d"
proof -
  have "d dvd 0" by simp
```

```
  thus ?thesis using common_div_def by simp
qed

theorem "¬(∃g. no_greater_div 0 0 g)"
proof
  assume "∃g. no_greater_div 0 0 g"
  from this obtain g where ngd:"no_greater_div 0 0 g" by blast
  from any_common_div_0[of "g+1"] have "common_div 0 0 (g+1)" .
  with ngd have "g+1 ≤ g" using no_greater_div_def by blast
  thus False by simp
qed
```

Finally, we prove that our definition of the GCD matches the definition of the *gcd* function in Isabelle.

```
lemma gcdfunc_imp_gcd_div:
  assumes "a ≠ 0 ∨ b ≠ 0"
  and     "g = gcd a b"
  shows   "is_gcd_div a b g"
using assms common_div_def is_gcd_div_def by auto

theorem gcd_func_is_gcd_div:
  assumes "a ≠ 0 ∨ b ≠ 0"
  shows   "(g = gcd a b) = is_gcd_div a b g"
proof
  assume "is_gcd_div a b g"
  hence h:"is_gcd a b g" using gcd_inf_div_eq[OF assms] by simp
  let ?g' = "gcd a b"
  from assms and gcdfunc_imp_gcd_div have "is_gcd_div a b ?g'" by simp
  hence "is_gcd a b ?g'" using gcd_inf_div_eq[OF assms] by simp
  from gcd_unique[OF this h] show "g = ?g'" ..
qed (simp add: gcdfunc_imp_gcd_div[OF assms])

end
```