数理解析研究所講究録 1437 巻 2005 年 21-28

Formalization of the valuation theory

Hidetsune Kobayashi
College of Science and Technology,
Nihon University *

Hideo Suzuki
Department of Information Technology,
Polytechnic University †

Yoko Ono
Department of policy studies,
The University of Shimane ‡

Shunichi Kurino
College of Science and Technology,
Nihon University §

1 Introduction

In this report, we present formalization of the valuation theory in Isabelle/HOL which is executed on a personal computer with linux OS.

Our aim is to express abstract algebra in formal language and make mathematical logic clear and easy to understand for beginners. Because of the limit of capability of Isabelle/HOL, some concepts cannot be treated, we choose the Valuation theory as a target of our formalization. We know mathematical objects which should have unfixed number of types is not treatable in Isabelle/HOL. Up to now, we have successfully formalized some topics on the valuation theory, namely definition of a valuation, normal valuation derived from a valuation, elementary properties of the valuation ring, topology with respect to a valuation and approximation theory by using Ostrowski elements.

In our report we have nothing new added to the theory from the mathematical point of view, but our work shows an extended usage of a computer to abstract mathematics, and moreover it will be easy to see that we have got a gear to avoid making human mistakes. One more point we want to stress is, human inference is not completely free from intuitive feelings, and intuitively natural inference is easy to understand. But mechanical proof often gives a strange feeling.

As noted above, we introduced the set of augmented integers as a value group, we give here a revised version of formalization, the original version that we presented on February 21 has a cyclic group with infinity instead. New version is much easier to read than the original one.

2 Definition of valuations

In this report we treat only discrete valuations. In Isabelle/HOL, the definition of a valuation is expressed as:

*1-8 Kanda-surugadai Chiyoda, Tokyo
†2-32-1 Ogawa-nishi Kodaira, Tokyo
‡2433-2 Nobara Hamada, Shimane
§1-8 Kanda-surugadai, Tokyo

```
valuation::"'b FieldType ⇒ ('b ⇒ ant) ⇒ bool"
    "valuation K h == h ∈ extensional (carrier K) ∧
    h ∈ carrier K → Z∞ ∧
    h (0K) = ∞ ∧ (∀x ∈ ((carrier K) - 0K).  h x ≠ ∞) ∧
    (∀x ∈ (carrier K). ∀y ∈ (carrier K). h (x ·K y) = (h x) + (h y)
    ∧ (∀x ∈ (carrier K). 0 ≤ (h x) → 0 ≤ h (1K +K x))
    ∧ (∃x.  x ∈ carrier K ∧ (h x) ≠ ∞ ∧ (h x) ≠ 0 )"
```

Here, 'b FieldType means an algebraic system having field structure, a base set (called carrier) consisting of elements of type 'b, two operations addition and multiplication satisfying field conditions, and K is the field of type 'b FieldType. Type 'b has no special meaning, but if we have an element with type 'a, then Isabelle see elements having type 'a and type 'b are different, automatically. "ant" means a type of augmented integer. "extensional" is a special concept of Isabelle and it means that the function h has "arbitrary" value outside the carrier K. "carrier K" is the basic set of the field K. "Z_∞" is the set of augmented integers without minus infinity. "x $\cdot_K$ y" is the product in the field K of two elements (with type 'b). In the set of augmented integers no integer is neither equal to infinity nor minus infinity. Therefore the final line of the definition guarantees that the field having a valuation is a non-zero field.

In "Z_∞" we have $\infty + \infty = \infty$, $z + \infty = \infty$ forall integer z and $0 \cdot \infty = 0$, $z \cdot \infty = \infty$ if $0 < z$, $z \cdot \infty = -\infty$ if $z < 0$.

Elementary properties of a valuation are proved with some lines of proof. For example **lemma** "field K; valuation K v; x ∈ carrier K; x ≠ 0 ⇒ int n •a (v x) = v (x ^K n)" is proved as

```
apply (frule field_is_ring[of "K"], induct_tac n,
       simp add:ring_r_one asprod_1, simp add:value_of_one,
       frule val_nonzero_z[of "K" "v" "x"], assumption+,
       erule exE,  simp add:asprod_0_x)
apply (frule sym, thin_tac "int n *a v x = v (x^K n)", simp,
       subst val_t2p[of "K" "v" "x"], assumption+, rule npClose,
       assumption+, simp, frule val_nonzero_z[of "K" "v" "x"],
       assumption+,  erule exE, simp add:asprod_mult a_zpz,
       simp add:zadd_zmult_distrib)
done
```

The key points of this proof is to use induction and the property of a valuation appearing at the fourth line of the definition. "frule", "induct_tac", "simp add:", "assumption", "erule" and "subst" are built in functions and others e.g. "field_is_ring" etc. are names of lemmas we have already proved earlier.

3 Normal valuations

Let K be a field with valuation v. We see that there is an augmented positive integer a such that any element of the image v (carrier K) is expressed as a · x with some augmented integer x. This a is equal to the minimum of the set {x. x ∈V(carrier K) ∧ 0 < x}. Therefore for any x in carrier K, we have an equation

$$v\ x = a \cdot x'$$

Here, x' is uniquely determined by x, and we denote x' as "n_val K v x". Now we have a function from the carrier K to Z_∞. We can prove this function is a valuation, and this is called as the normal valuation of v. The definition is formalized as:

```
Lv::"['r FieldType, 'r ⇒ ant] ⇒ ant"
                (** Least nonnegative value **)
    "Lv K v == AMin {x.  x ∈ v ' carrier K ∧ 0 < x}"

n_val::"['r FieldType, 'r ⇒ ant] ⇒ ('r ⇒ ant)"
    "n_val K v == λx ∈ carrier K.
    (THE l.  (l · (Lv K v)) = v x)"
```

Here, THE l. (l · (Lv K v)) = v x is the unique augmented integer l satisfying the equation

$$l \cdot (\mathrm{Lv}\ K\ v)) = v\ x$$

We have a lemma

lemma n_val_valuation:"[| field K; valuation K v |] ⇒ valuation K (n_val K v)"

We have also a lemma:

lemma n_val:"[| field K; valuation K v; x ∈ carrier K |] ⇒
(n_val K v x) · (Lv K v) = v x"
by (frule n_valTr[of "K" "v" "x"], assumption+, simp add:n_val_def)

Since Lv K v is positive (augmented) integer, we have a proposition "v (x) is positive if and only if n_val K v x is positive".

The following lemmas are easy to see.

lemma val_surj_n_val:"[| field K; valuation K v; ∃x ∈ carrier K. v x = 1 |] ⇒
(n_val K v) = v"

```
apply (rule funcset_eq[of _ "carrier K"],
       simp add:n_val_def restrict_def extensional_def,
       simp add:valuation_def)
apply (rule ballI,
        frule val_surj_n_valTr[of "K" "v"], assumption+,
        frule_tac x = x in n_val[of "K" "v"], assumption+,
        simp add:amult_one_r)
done
```

lemma n_val_n_val:"[| field K; valuation K v |] ⇒ n_val K (n_val K v) = n_val K v"
by (frule n_val_valuation[of "K" "v"], assumption+,
frule n_val_surj[of "K" "v"], assumption+,
simp add:val_surj_n_val)

The last lemma shows that the normal valuation of a normal valuation is the normal valuation. The proofs of above lemmas are concise and easy to read.

4 The valuation ring determined by a valuation v

The valuation ring determined by a valuation v is a ring consisting of the elements having positive value of v. We introduce a formalized definition of the valuation ring:

```
Vr::"['r FieldType, 'r ⇒ ant] ⇒ 'r RingType"
    "Vr K v == Sr K ({x.  x ∈ carrier K ∧ 0 ≤ (v x)})"
```

Here Sr denotes a ring structure derived from K. The definition of Sr is:

```
Sr ::"[('a, 'more) RingType_scheme, 'a set] ⇒ 'a RingType"
    "Sr R S == (| carrier = S, pOp = λx ∈ S. λy ∈ S. x +R y,
    mOp = λx ∈ S. -R x, zero = 0R, tOp = λx ∈ S. λy ∈ S.
    x ·R y, one = 1R |)"
```

Here $\lambda x \in S.\ \lambda y \in S.\ x +_R y$ is a function with variables x and y taken from a set S mapping to $x +_R y$. From this definition, we see that the valuation ring is an integral domain:

lemma Vr_integral:"[| field K; valuation K v |] ⇒ integral_domain (Vr K v)"

```
apply (simp add:integral_domain_def,
       simp add:Vr_ring, (rule ballI)+, rule impI,
       simp add:Vr_tOp_f_tOp, simp add:Vr_0_f_0)
apply (rule contrapos_pp, simp+, erule conjE,
       frule field_is_idom[of "K"],
       frule_tac x = x in Vr_mem_f_mem[of "K" "v"], assumption+,
       frule_tac x = y in Vr_mem_f_mem[of "K" "v"], assumption+,
       frule_tac x = x and y = y in idom_tOp_nonzeros[of "K"],
       assumption+, simp)
done
```

We explain this proof line by line. At first, we give a definition of the integral domain. Since the definition of the integral domain is (1) it is a ring and (2) satisfying a condition that it has no zero-divisor. simp add:Vr_ring gives the fact that "Vr K v" is a ring, and (rule ball)+. rule impI etc. are a preparatory operations to prove that there is no nonzero zero-divisor.

It is known that Vr K v is a local ring. And the subset vp K v defined as

```
vp::"['r FieldType, 'r ⇒ ant] ⇒ 'r set"
    "vp K v == {x.  x ∈ carrier (Vr K v) ∧ 0 < (v x)}"
```

is the maximal ideal.

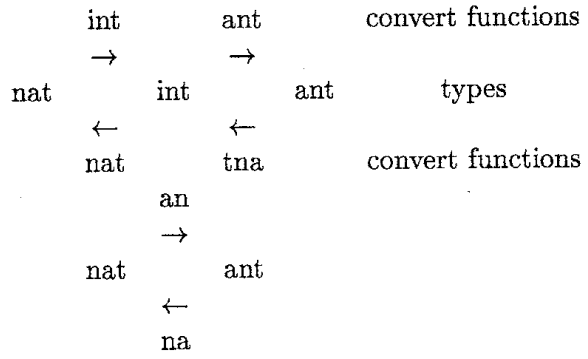
The valuation ring is a principal ideal ring and any ideal is expressed as a "power" of vp K v. In the case where we take exponent of power from a natural number, zero ideal is not expressed as a power of vp K v. But if we define the power with exponent chosen from the augmented integers, we can treat the zero ideal as a power of vp K v:

```
r_apow::"[('r, 'm) RingType_scheme, 'r set, ant] ⇒ 'r set"
    "r_apow R I a == if a = ∞ then {0R}
    else (if a = 0 then carrier R else I ◇R (na a))"
```

Here, a is an augmented integer and na a is a natural number defined as

$$\text{na a} = 0 \text{ if a} < 0 \text{ else na} = \text{int a if } 0 \leq \text{a} \wedge \text{a} \neq \infty$$

Because any element has type, and we cannot add two elements having different types, therefore Isabelle has functions converting from one type to the other. We defined "ant" as a type of augmented integers, and relation between type are as follows:



As stated above, vp K v is a maximal ideal. This fact is formalized as
lemma vp_maximal:"[| field K; valuation K v |] ⇒ maximal_ideal (Vr K v) (vp K v)"

The above statement might be a little funny, since in a text book, we have only to write "Let K be a field and let v be a valuation. Then vp K v is a maximal ideal of the ring Vr K v". In a textbook, we do not write as "valuation K v" but we write simply as valuation v. In Isabelle/HOL, we have a method to avoid putting items required to define such as K and v for Vr K v. Expressing the valuation ring as Vr is much easier to read than to express Vr K v, but we have chosen a way putting items all together, because formalization of definition is simple and clear.

To formalize Vr K v is a principal ideal domain, we formalized the following two items and then we have the lemma Ig_generate_I.

```
LI ::  "['r FieldType, 'r ⇒ ant, 'r set] ⇒ ant"
    "LI K v I == AMin (v ' I)"
```

The minimum of the set v ' I, the image of I by v.

```
Ig ::  "['r FieldType, 'r ⇒ ant, 'r set] ⇒ 'r"
    "Ig K v I == SOME x.  x ∈ I ∧ v x = LI K v I"
```

Then
lemma Ig_generate_I:"[| field K; valuation K v; ideal (Vr K v) I |] ⇒
(Vr K v) ⋄ (Ig K v I) = I"

Here, (Vr K v) ⋄ (Ig K v I) is a principal ideal of Vr K v generated by Ig K v I.
As a property of the discrete valuation ring Vr K v, we have
lemma ideal_apow_vp:"[| field K; valuation K v; ideal (Vr K v) I |] ⇒
$I = (\text{vp K v})^{(\text{Vr K v})\ (\text{n\_val K v (Ig K v I)})}$"

This lemma states that any ideal of Vr K v is expressed as a power of the maximal ideal vp K v. The following lemma gives another view point of the normal value of an element x of the valuation ring Vr K v.

lemma ideal_apow_n_val:"[| field K; valuation K v; x ∈ carrier (Vr K v) |] ⇒
(Vr K v) ◇ x = (vp K v)$^{(Vr\ K\ v)(n\_val\ K\ v\ x)}$"

From this lemma we see the principal ideal (x) is equal to the power of vp K v with exponent "n_val K v x" i.e. the normal value of x.

Two valuations v and v′ are called if and only if n_val K v and n_val K v′ are equal. If v and v′ are equivalent, we see Vr K v and Vr K v′ are equal and two ideals vp K v and vp K v′ are the same. An equivalence class of a valuation v is called a prime divisor, and we see the valuations in the same prime divisor determine the same ideal. That is, if two valuations v and v′ are equivalent and if x is an element of Vr K v, then (Vr K v) ◇ x = (Vr K v′) ◇ x.

We have formalized an interesting lemma concerning the equivalence of the valuations.

lemma valuations_equiv:"[| field K; valuation K v; valuation K v′; ∀x ∈ carrier K.
0 ≤ (v x) → 0 ≤ (v′ x) |] ⇒ v_equiv K v v′"

Formalization of a proof to this lemma is not very short. The proof is completed with 9 steps 51 lines.

We present one more lemma implying the dimension of the valuation ring is 1:

lemma Vring_prime_maximal:"[| field K; valuation K v; prime_ideal (Vr K v) I;
I ≠ {0$_{(Vr\ K\ v)}$} |] ⇒ maximal_ideal (Vr K v) I"

5 Topology determined by a valuation

We noted that the valuation ring Vr K v is a local ring with the maximal ideal vp K v. We can consider (vp K v) -adic topology in Vr K v. This topology can be formalized in some ways, but in this report, we formalized the topology by using the valuation. And we formalized the completion in a naive way by using Cauchy sequence. This is because the type becomes complicated if we discuss the completion as an inverse limit.

The essential lemma are the following lemmas, because to discuss the topology, we have only to discuss the convergence of an infinite sequence.

lemma n_value_x_1:"[| field K; valuation K v; 0 ≤ n; x ∈ (vp K v)$^{(Vr\ K\ v)\ n}$ |] ⇒
n ≤ (n_val K v x)"

lemma n_value_x_2:"[| field K; valuation K v; x ∈ carrier (Vr K v); n ≤ (n_val K v x);
0 ≤ n |] ⇒ x ∈ (vp K v)$^{(Vr\ K\ v)\ n}$"

(vp K v) - adic topology is the topology determined by a system of neighborhoods {(vp K v)$^{◇(Vr\ K\ v)\ n}$ | natural number n}. Above lemma shows an element x belongs to (vp K v)$^{(Vr\ K\ v)\ n}$ if and only if n ≤ n_val K v x. Therefore we can discuss the (vp K v)-adic topology by using the normal valuation n_val K v.

A Cauchy sequence is formalized as:

```
Cauchy_seq::"['b FieldType, 'b ⇒ ant, nat ⇒ 'b] ⇒ bool"
            ("(3Cauchy_ _ _)" [90,90,91]90)
  "Cauchy K v f == (∀n.  (f n) ∈ carrier K) ∧
  (∀N. ∃M. (∀n m.  M < n ∧ M < m ((f n) +K (-K (f m)))
  ∈ (vp K v)^(Vr K G a i v)(an N)))"
```

Note that all numbers N M n m are of type nat. Although we do not specify the type of N M, since types of n m are required by the type conditions given as above, f is of type nat ⇒ 'b, the type of n and the type of m should be nat. And inequality M < n should have elements of the same type, M should have the type nat. The type inference is done automatically, we needn't specify types explicitly.

A convergent sequence is a Cauchy sequence, and this fact is formalized as:

lemma has_limit_Cauchy:"[| field K; valuation K v; ∀j. f j ∈ carrier K; b ∈ carrier K; lim K v f b |] ⇒ Cauchy $_{K\ v}$ f"

As noted above, we formalized the completion of a given field with respect to a valuation by using Cauchy sequence. You see the type of the original field is the same as the type of the completion. In the definition, Complete $_{v'}$ K' means K' is complete with respect to the valuation v' (or more precisely, Complete $_{v'}$ K' is a function returning true if K' is complete and returning false otherwise).

```
v_completion::"['b ⇒ant, 'b ⇒ ant, 'b FieldType,
    'b FieldType] ⇒ bool"
            ("(4Completion_ _ _ _ _ _)" [90,90,90,91]90)
    "Completion v v' K K' == subfield K K' ∧ Complete v' K'
    ∧ (∀x ∈ carrier K. v x = v' x) ∧
    (∀x ∈ carrier K'. (∃f.  Cauchy K v f ∧ lim K' v' f x))"
```

I am afraid the formalization of the completion above is frustrating for some mathematicians, because it is not constructive. But, if we treat an equivalence class of Cauchy sequence as an element of the completion, the type of the element is (nat ⇒'b) set and the completion field is of type "(nat ⇒'b) FieldType", therefore even an inclusion map should have complicated type. This is why we adopted formalization above.

6 Approximation

In this section, we show formalization of approximation. Some lemmas on approximation require calculation of algebraic formula, hence it is taken that this section presents examples how Isabelle/HOL executes formula manipulation. Compared to existing formula manipulation language such as Maple, Mathematica and Reduce, in Isabelle calculation of formulas is executed step by step and each step is executed with a given instruction. However, we see that we can obtain nice result fit to a given problem.

Approximation lemma is "given nonequivalent valuations v_1, v_2, ... , v_n and elements y_1, y_2, ... , y_n of K, then for any natural number m there exists an element y of K such that m < v_j (y_j - y) for each j = 1, 2, ... , n".

To present approximation, we need nonequivalent valuations formalized:

```
valuations::"['r FieldType, nat, nat ⇒ ('r ⇒ ant)] ⇒bool"
    "valuations K n vv == ∀j ∈ Nset n.  valuation K (vv j)"

vals_nonequiv::"['r FieldType, nat, nat ⇒ ('r ⇒ ant)]
    ⇒ bool"
    "vals_nonequiv K n vv == valuations K n vv ∧ (∀j∈Nset n.
    ∀l∈Nset n.  j ≠ l → ¬ (v_equiv K (vv j) (vv l)))"
```

We use Ostrowski element to show there exists an approximation element.

```
Ostrowski_elem::"['b FieldType, nat, nat ⇒ ('b ⇒ ant), 'b]
    ⇒ bool"
"Ostrowski_elem K n vv x == (0 < (vv 0 (1_K +_K (-_K x)))) ∧
(∀j ∈ nset (Suc 0) n.  0 < (vv j x))"
```

In this formalization, we give nonequivalent valuations vv 0, vv 1, ..., vv n, because we use mathematical induction "induct_tac" which is already prepared in Isabelle/HOL.

The following lemma is an example how Isabelle manipulates a formula:

lemma tail_of_expansion1:"[| ring R; x ∈ carrier R |] ⇒ $(1_R +_R x)^{\hat{}R\ (Suc\ n)} =$
$x \cdot_R$ (nsum0 R (λ i. $({}_{(Suc\ n)}C_{(Suc\ i)} \times_R x^{\hat{}R\ i})$) n) $+_R 1_R$"

Here, "nsum0 (λ i. $({}_{(Suc\ n)}C_{(Suc\ i)} \times_R x^{\hat{}R\ i})$) n" is a sum
$x^{\hat{}R\ n} +_R {}_{(Suc\ n)}C_n \times_R x^{\hat{}R\ (n-1)} +_R \cdots +_R 1_R$

Approximation theorem is formalized as follows:

lemma ApproximationTr4:"[| field K; vals_nonequiv K (Suc n) vv; ∀j ∈ Nset (Suc n).
x j ∈ carrier K |] ⇒ ∃l. ∀N. l < N → (∀j ∈ Nset (Suc n).
(an m) ≤ (vv j (e ∑ K (λj ∈ Nset (Suc n).
$(x\ j) \cdot_K (1_K +_K -_K (1_K +_K -_K ((\Omega\ K_{\ vv\ (Suc\ n)})\ j)^{\hat{}K\ N})^{\hat{}K\ N})$))
(Suc n) $+_K -_K$ (x j))))"

theorem Approximation_thm:"[| field K; vals_nonequiv K (Suc n) vv; ∀j ∈ Nset (Suc n).
(x j) ∈ carrier K |] ⇒ ∃y ∈ carrier K. ∀j ∈ Nset (Suc n). (an m) ≤ (vv j (y $+_K -_K$ (x j)))"

In ApproximationTr4 we presented a constructive proof, and in Approximation_thm we formalized ordinary proposition given in a textbook.

7 Note on formalization

Although there are some exceptions, formalization is possible in many fields of pure mathematics. And we can make their logical structure clear and check their logic is correct or not. Therefore it will be useful to formalize mathematical concepts in some formalization language, and check proofs whether it is correct or not. At present, we have no advanced automated prover and we have to give instruction at each step of proof. But after accumulating mathematical knowledge in a huge database, it will be possible to choose a correct tactic to prove a problem automatically.

References

1. Kenkichi, Iwasawa: Algebraic function theory, Iwanami shoten, 1951 (in Japanese)

2. M.F.Atiyah and I.G. Macdonald: Introduction to commutative algebra, Addison-Wesley 1969.