# Formalizing an Abstract Algebra Textbook in Isabelle/HOL [1]

Jesús María Aransay Azofra and Jose Divasón Mallagaray

Universidad de La Rioja
Departamento de Matemáticas y Computación

*XIII Encuentro de Álgebra Computacional y Aplicaciones (EACA 2012)*

Alcalá de Henares, 14 de Junio de 2012

[1] This work has been supported by project MTM2009-13842-C02-01 (MEC) and by FORMATH project, nr. 243847, of the FET program within the FP7 European Commission.

## Introduction

**Project**

1. The objective of this work is to analyze the degree of formalization that a mathematic textbook has.
2. Another objective is to formalize concepts and theorems of linear algebra, concretely of vector spaces, using Isabelle/HOL.
3. We have followed a Halmos' book: *Finite-dimensional vector spaces*.

- Why to formalize mathematics?
- Why vector spaces?
- Why using this book?

We have formalized the first 10 sections in Halmos:

**Sections**

1. Fields
2. Vector Spaces
3. Examples
4. Comments
5. Linear Dependence

**Sections**

6. Linear Combinations
7. Bases
8. Dimension
9. Isomorphism
10. Subspaces

## Main Theorems

**Theorem 1**

Every linearly independent set of a finite dimensional vector space $V$ can be extended to a basis.

**Theorem 2**

Any two bases of a finite dimensional vector space have the same cardinality.

**Theorem 3**

An n-dimensional vector space $V$ over a field $\mathbb{K}$ is isomorphic to $\mathbb{K}^n$.

## Important Concepts

### Isabelle

- **Isabelle:** The theorem proving assistant in which we have made the development.
- **HOL:** Higher-order logic.
- **HOL-Algebra:** A library of algebra implemented in Isabelle using HOL.
- **Locales:** A kind of module system in which we can fix variables and declare assumptions.

## Example of Isabelle code

```
locale vector_space = K: field K + V: abelian_group V
  for K (structure) and V (structure) +
  fixes scalar_multiplication:: "'a => 'b => 'b" (infixr "·" 70)
  assumes mult_closed: "⟦x ∈ carrier V;a ∈ carrier K⟧
  ⟹ a · x ∈ carrier V"
  and mult_assoc: "⟦x ∈ carrier V; a ∈ carrier K; b ∈ carrier K⟧
  ⟹ (a ⊗K b) · x = a · (b · x)"
  and mult_1: "⟦x ∈ carrier V⟧ ⟹ 1K · x = x"
  and add_mult_distrib1:
  "⟦x∈ carrier V; y ∈ carrier V; a ∈ carrier K⟧
  ⟹ a · (x ⊕V y)= a·x ⊕V a·y"
  and add_mult_distrib2:
  "⟦x∈ carrier V; a ∈ carrier K; b ∈ carrier K⟧
  ⟹ (a ⊕K b) · x = a·x ⊕V b·x"
```

## Indexed Sets

1. In mathematics, we usually represent a set of $n$ elements this way:

$$A = \{a_1, \dots, a_n\}$$

2. Really a set doesn't have an order by default (but we can give one for it).
3. This is not important...unless the order has influence on the proof.
4. So we have to implement the notion of indexed set as any bijection between the elements of the set and its indices.
5. We have to define operations to insert and remove one element of an indexed set. We also have to develop an induction rule in order to prove theorems and properties about this structure.

## THEOREM 1

**Auxiliary Result**

If the set of non-zero vectors $x_1, \ldots, x_n$ is linearly dependent, then there exists at least one $x_k$, $2 \leq k \leq n$, which is a linear combination of the preceding ones.

Note that the given order is very important, so the use of indexed sets is indispensable.

**Theorem 1**

Every linearly independent set of a finite dimensional vector space $V$ can be extended to a basis.

## Theorem 1

Every linearly independent set of a finite dimensional vector space $V$ can be extended to a basis.

- Let $A = \{a_1, \dots, a_n\}$ an independent set and $B = \{b_1, \dots, b_m\}$ a basis of $V$. We apply the auxiliary result to the set:
  $C = \{ \underbrace{a_1, \dots, a_n}_{Elements\ of\ A}, \underbrace{b_1, \dots, b_m}_{Elements\ of\ B} \}$
- Since the first $n$ elements are in an independent set (they are contained in $A$), hence the element which is a linear combination of the preceding ones is in $B$.
- Let $b_i$ that element, then we remove it and we obtain:
  $C' = \{a_1, \dots, a_n, b_1, \dots, b_{i-1}, b_{i+1}, \dots b_m\}$
- If $C'$ is independent we have already finished (the basis is $C'$), if not we iterate the process.

## PROBLEMS

INADEQUACIES OF HALMOS:

- $C = \{\overbrace{a_1, \ldots, a_n}^{Elements\ of\ A}, \overbrace{b_1, \ldots, b_m}^{Elements\ of\ B}\}$ could be a multiset. **SOLUTION:** $C = A \cup (B - A)$.
- There could be some elements of $B$ which are linear combination of the preceding ones (there is no unicity). **SOLUTION:** Take the least.
- Why will the process finish?

DIFFICULTIES WITH OUR IMPLEMENTATION:

- Reasoning about iterative processes can be hard to be implemented in Isabelle. Functions in HOL are total. **SOLUTION:** Partial functions (tail recursive).

We define two functions: *remove_ld* and *iterate_remove_ld*.

- The first one removes the least element of a dependent set which is a linear combination of the preceding ones.
- The second one iterates the previous function until achieving an independent set:

```
partial_function (tailrec) iterate_remove_ld ::
  "'c iset => 'c set"
  where "iterate_remove_ld (A,f)
  = (if linear_independent A then A
    else iterate_remove_ld (remove_ld (A, f)))"
```

There are three important properties which *iterate_remove_ld* must satisfy to demonstrate the theorem:

1. The result is a linearly independent set (about 100 lines).
2. The result is a spanning set (about 130 lines).
3. The independent set $A$ is contained in the result of the function (about 350 lines).

The total number of lines necessary to prove this theorem were 984.

```
lemma extend_independent_set_to_a_basis:
  assumes "linear_independent A"
  shows "∃S. basis S ∧ A ⊆ S"
proof -
  def C ≡"A∪(B-A)"
  have "linear_independent (iterate_remove_ld (C,h))"
  proof (rule linear_independent_iterate_remove_ld)
    ...
  qed
  have "span(iterate_remove_ld (C,h))=carrier V"
  proof (rule iterate_remove_ld_preserves_span)
    ...
  qed
  have "A ⊆ (iterate_remove_ld (C,h))"
  proof (rule A_in_iterate_remove_ld)
    ...
  qed
  ...
qed
```

## THEOREM 2

**Swap theorem**

If $A$ is a linearly independent set of $V$ and $B$ is any spanning set of $V$, then $card(A) \leq card(B)$.

**Corollary: theorem 2**

Any two bases of a finite dimensional vector space have the same cardinality.

## Swap theorem

If $A$ is a linearly independent set of $V$ and $B$ is any spanning set of $V$, then $card(A) \leq card(B)$.

SUMMARY OF THE PROOF IN HALMOS:

Let $A = \{a_1, \dots, a_n\}$ be an independent set, $B = \{b_1, \dots, b_m\}$ a spanning set and $m < n$. Then:

1. Construct the set $S = \{a_1, b_1, \dots, b_m\}$. $S$ is a linearly dependent set.
2. Apply the auxiliary result to $S$, obtaining $S' = \{a_1, b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_m\}$. $S'$ is a spanning set.
3. Iterate the process $m$ times to obtain a contradiction (i. e. add the first $m$ elements of $A$ and remove all elements of $B$).

## INADEQUACIES OF HALMOS:

1. Construct the set $S = \{a_1, b_1, \dots, b_m\}$. $S$ is a linearly dependent set.
   - Need to take into account the positions of the elements.
   - What happens if $a_1 \in \{b_1, \dots, b_m\}$? The set $S$ could be **not** a linearly dependent set!!
2. Apply the auxiliary result to $S$, obtaining $S' = \{a_1, b_1, \dots, b_{i-1}, b_{i+1}, \dots, b_m\}$. $S'$ is a spanning set.
   - There could be some elements of $B$ which are linear combination of the preceding ones (there is no unicity).
3. Iterate the process $m$ times to obtain a contradiction (i. e. add the first $m$ elements of $A$ and remove all elements of $B$).

## PROBLEMS WITH OUR IMPLEMENTATION

- We can't follow a similar reasoning than in theorem 1 to prove the result: now we need to have control in the number of iterations.
- The iterative reasoning is implemented applying $m$ times the next function:

$$\begin{aligned} \mathit{swap\_function}\ (\{a_1, \ldots, a_n\} \times \{b_1, \ldots, b_m\}) \\ = (\{a_2, \ldots, a_n\} \times \{a_1, b_1, \ldots, b_{i-1}, b_{i+1}, \ldots, b_m\}) \end{aligned}$$

- We have to make use of the power of a function...however, this is not implemented in Isabelle. We have to make it:

```
instantiation "fun" :: (type, type) power
begin
definition one_fun :: "'a => 'a"
  where one_fun_def: "one_fun = id"
definition times_fun :: "('a => 'a) => ('a => 'a) => 'a => 'a"
  where "times_fun f g = (∀x. f (g x))"
end
```

- Once we have defined the power of a function, we have to prove the properties that *swap_function* satisfies; after that, we have to generalize that properties to the case where the function is applied *m* times, by using induction. The following lemma is indispensable:

```
corollary fun_power_suc_eq:
  shows "(f^(n+1)) x = f ((f^n) x)"
  using fun_power_suc by (metis id_o o_eq_id_dest)
```

- This is a long and tedious process: the proofs of all necessary properties and lemmas to make the demonstration take up 1800 lines.

## THEOREM 3

What is $\mathbb{K}^n$?

**Definition of $\mathbb{K}^n$**

$$\mathbb{K}^n = \underbrace{\mathbb{K} \times \mathbb{K} \times \cdots \times \mathbb{K}}_{n} = \{(x_1, \ldots, x_n) | x_i \in \mathbb{K} \; \forall i, 1 \leq i \leq n\}$$

And in Isabelle?

- First we define the type vector, a pair of a function and a natural:
  **types** *'a vector = "(nat => 'a) * nat"*
  - The function maps naturals to elements of a set.
  - The natural is the length of the vector minus one.
  - **Example:** To represent $(a_1, a_2, a_3, a_4)$ we have a vector $(f, 3)$ where $f(0) = a_1$, $f(1) = a_2$, $f(2) = a_3$ and $f(3) = a_4$.
  - **Problem:** we don't have unicity of representation (a problem of partiality).

- **definition** *K_n_carrier :: "'a set => nat => ('a vector) set"*
  **where** *"K_n_carrier A n = {v. ((∀ i<n. ith v i ∈ A))*
  *∧ (∀ i≥n. ith v i =* **0***) ∧ (vlen v = (n - 1))}"*
- **definition**
  *K_n_add :: "nat => 'a vector => 'a vector => 'a vector"*
  (**infixr** *"⊕ı" 65)*
  **where** *"K_n_add n = (λv w. ((λi. ith v i $\oplus_R$ ith w i), n - 1))"*
- **definition** *K_n_zero :: "nat => 'a vector"*
  **where** *"K_n_zero n = ((λi.* $\mathbf{0}_R$*), n - 1)"*
- **definition** *K_n_mult :: "nat => 'a vector => 'a vector => 'a vector"*
  **where** *"K_n_mult n = (λv w. ((λi. ith v i $\otimes_R$ ith w i), n - 1))"*
- **definition** *K_n_one :: "nat => 'a vector"*
  **where** *"K_n_one n = ((λi.* $\mathbf{1}_R$*), n - 1)"*

## Definition of $\mathbb{K}^n$ in Isabelle

Finally using the definition of carrier, add, zero, mult and one we can define the concept of $\mathbb{K}^n$:

**definition** *K_n :: "nat => 'a vector ring"*
  **where**
```
  "K_n n = (| carrier = K_n_carrier (carrier R) n,
              mult = (λv w. K_n_mult n v w),
              one = K_n_one n,
              zero = K_n_zero n,
              add = (λv w. K_n_add n v w)|)"
```

We need to check that $\mathbb{K}^n$ is a vector space over $\mathbb{K}$, so we need to define its scalar multiplication: $a \odot (b_1, \ldots, b_n) = (a \cdot b_1, \ldots, a \cdot b_n)$

**definition** *K_n_scalar_multiplication :: "'a => 'a vector => 'a vector"*
(**infixr** *"⊙" 65)* **where** *"a ⊙ b = (λn::nat. a $\otimes_R$ ith b n, vlen b)"*

## Definition of isomorphism between vector spaces

Two vector spaces $V$ and $W$ over the same field $\mathbb{K}$ are isomorphic if there exists a linear map $f\colon V \to W$ such that is a bijection.

## Theorem 3

An n-dimensional vector space $V$ over a field $\mathbb{K}$ is isomorphic to $\mathbb{K}^n$.

Let $X = \{x_1, \ldots, x_n\}$ be a basis of $V$. The isomorphism between $V$ and $\mathbb{K}^n$ is easy to understand:

$$\begin{tikzcd} a = \alpha_1 x_1 \oplus_V \cdots \oplus_V \alpha_n x_n \in V \arrow[r, bend left, "f"] & (\alpha_1, \ldots, \alpha_n) \in \mathbb{K}^n \arrow[l, bend left, "f^{-1}"] \end{tikzcd}$$

## CONCLUSIONS AND RELATED WORK

### CONCLUSIONS

- Proofs in a book are not fully formal.
- Comparison between the length in the book and the formalized proof.
- Chapters and concepts in the book can be, sometimes out of order.
- We are able to generate code of the constructive proofs and execute it over instances.
- To formalize a book is factible.

### RELATED WORK

- Mizar
- HOL-Light
- Coq

ANY QUESTIONS?