# Formalizing a coding theory

Saikawa, Takefumi
名古屋大学



出版情報：MI lecture note series. 61, pp.115-118, 2015-03-06. 九州大学マス・フォア・インダストリ研究所
バージョン：
権利関係：

**TPP2014**

Theorem Proving and Provers for Reliable Theory and Implementations

**December 3-5, 2014, Nishijin Plaza, Kyushu University, Fukuoka, Japan**

# Formalizing a coding theory

**Takefumi SAIKAWA**

Nagoya University

We are working on formalizing the core properties of Reed-Solomon codes. The presentation of the theory tightly follows the one given in Hagiwara's textbook[3]. As this is an unfinished work with almost no result yet, we can only present our objecives. First, we are trying to completely understand this basic theory. We shall be able to find every detail and necessary technicalities by formalizing it. After completing this first step, we expect that we can extend our formalization to more advanced coding theories such as BCH codes and Gabidulin codes. Second, we are benchmarking the applicability of the ssreflect/mathcomp library[1] which is provided for the proof assitant coq[2]. The mathcomp library contains many results of algebra, yet it was originally developed to deal with a specific result in group theory. We want to find mismatches between the library and our use, if there is any. And last, we are examining the effectiveness of a method to debug a book by formalizing it. So far our development have successfully pointed out in the textbook[3] the need of a few corrections which are more than typos, i.e., which actually affect the theory. We expect that this method can correct errors more consistently than usual reviewing process. Aside from these objectives, we propose some possible improvements in the graphical interface of coq. We plan to implement them in parallel to the development of our formalization.

## References

[1] `http://ssr.msr-inria.inria.fr/`
[2] `http://coq.inria.fr/`
[3] 萩原学, 「符号理論」, 2012, 日本評論社.

# Formalizing a coding theory

Takafumi Saikawa

December 18, 2014

## about the project

A part of infotheo project (Reynald's talk, yesterday).
Formalizing the final chapter of Hagiwara's book.

## motivation

General motivations for formalizing a theory: we want to

- rigorously understand the theory,
  we need details to really understand a theory
- build a good library, generalizing notations and patterns used in math,
  notations are not trivial
- and debug the textbook.

## motivation

I want to

- get used to ssreflect & mathcomp,
- understand a coding theory from scratch.

## theory

Reed-Solomon code

- is an error-correction code,
- takes code words in a polynomial ring $\mathbb{F}[X]$,
- is powerful and used everywhere.

## presentation in the textbook

Reed-Solomon code in Hagiwara's book:

- Presented mostly in terms of polynomials. Very little linear algebra.
- Employs the Euclidean decoding algorithm.
- Main theorem : the decoder corrects up to $(d-1)/2$ errors, where $d$ is the distance that appears in the definition of RS-code :

$$C(n,a,d) := \{f(X) | \mathrm{deg} f(X) < n \wedge \forall (1 \leq n_0 \leq d-1), f(a^{n_0}) = 0\}$$

## presentation in the textbook

The technical ingredients in the Euclidean decoding algorithm are:

- Euclid's gcd
- derivation

They are already given in `poly.v` and `polydiv.v` of mathcomp library.
It seems to be fairly easy to formalize the theory now...

Not finished yet (by this morning).

## interface

"interface matters!"

We want (I want) an improved ProofGeneral:

- Independent "`Search`" buffer, independent from `proof-response-buffer`.
- Graphical subterm selection by a pointing device.
- (Following the selection of a subterm,) automatic listing of immediately "`rewrite`"-able lemmas.
- Let a mouseover trigger a `Search`, `Locate`, ...
- Intermediate results
  - in a sequence of tactics at each semicolon,
  - and in a sequence of `rewrite`s at each lemma,

  i.e. finer step-execution.

Maybe we need several concurrent coqtop processes; possible with much RAM and many CPUs.

## conclusion

work-in-progress; should be commited (hopefully) in some weeks.