Formalization of Shannon's Theorems in SSReflect-Coq[⋆]

Reynald Affeldt and Manabu Hagiwara

Research Institute for Secure Systems,
National Institute of Advanced Industrial Science and Technology, Japan

Abstract. The most fundamental results of information theory are Shannon's theorems. These theorems express the bounds for reliable data compression and transmission over a noisy channel. Their proofs are non-trivial but rarely detailed, even in the introductory literature. This lack of formal foundations makes it all the more unfortunate that crucial results in computer security rely solely on information theory (the so-called "unconditional security"). In this paper, we report on the formalization of a library for information theory in the SSReflect extension of the Coq proof-assistant. In particular, we produce the first formal proofs of the source coding theorem (that introduces the entropy as the bound for lossless compression), and the direct part of the more difficult channel coding theorem (that introduces the capacity as the bound for reliable communication over a noisy channel).

1 Introduction

"Information theory answers two fundamental questions in communication theory: What is the ultimate data compression (answer: the entropy H), and what is the ultimate transmission rate of communication (answer: the channel capacity C)." This is the very first sentence of the reference book on information theory by Cover and Thomas [5]. This paper is precisely about the formalization of Shannon's theorems that answer these two fundamental questions.

The proofs of Shannon's theorems are non-trivial but are rarely detailed (let alone formalized), even in the introductory literature. Shannon's original proofs [1] in 1948 are well-known to be informal; rigorous versions appeared several years later. Even today, the bounds that appear in Shannon's theorems (these theorems are asymptotic) are never made explicit and their existence is seldom proved carefully.

This lack of formal foundations makes it all the more unfortunate that several results in computer security rely crucially on information theory: this is the so-called field of "unconditional security" (one-time pad protocol, evaluation of information leakage, key distribution protocol over a noisy channel, etc.). A formalization of information theory would be a first step towards the verification

⋆ Conference version appeared in the proceedings of ITP 2012 (`http://itp2012.cs.princeton.edu`).

of cryptographic systems based on unconditional security, and, more generally, for the rigorous design of critical communication devices.

In this paper, our first contribution is to provide a library of formal definitions and lemmas for information theory. First, we formalize finite probability, up to the weak law of large numbers, and apply this formalization to the formalization of basic information-theoretic concepts such as entropy and typical sequences. This line of work has already been investigated by Hasan et al. [6,11,13] and by Coble [9], with the HOL proof-assistants. The originality of our library (besides the fact that we are working with the Coq proof-assistant [10]) lies in the formalization of advanced concepts such as channels, codes, jointly typical sequences, etc., that are necessary to state and prove Shannon's theorems.

Our second and main contribution is to provide the first (to the best of our knowledge) formal proofs of Shannon's theorems. Precisely, we formalize the source coding theorem (direct and converse parts), that introduces the entropy as the bound for lossless compression, and the direct part of the channel coding theorem, that introduces the channel capacity as the bound for reliable communication over a noisy channel.

The formalization of Shannon's theorems is not a trivial matter because, in addition to the complexity of a theorem such as the channel coding theorem, the literature does not provide proofs that are organized in a way that facilitates formalization. Most importantly, it is necessary to rework the proofs so that the (asymptotic) bounds can be formalized. Indeed, information theorists often resort to claims such as "this holds for n sufficiently large", but there are in general several parameters that are working together so that one cannot choose one without checking the others. Another kind of approximation that matters when formalizing is the type of arguments. For example, in the proof of the source coding theorem, it is mathematically important to treat the source rate as a rational and not as a real, but such details are often overlooked. In order to ease formalization, we make several design decisions to reduce the number of concepts involved. For example, we do not use conditional entropy in an explicit way, and, more generally, we avoid explicit use of conditional probabilities, except for the definition of discrete channels. In fact, we believe that this even facilitates informal understanding because proofs are more "to the point".

We carried out formalization in the SSReflect extension [12] of the Coq proof-assistant [10]. This is because information theory involves many calculations with Σ/Π-notations over various kinds of sets (tuples, functions, etc.) for which SSReflect's library (in particular, canonical big operators [7]) are well-suited. Formal definitions and lemmas that appear in this paper are taken directly from the scripts (available at [14]), modulo enhancements with colors and standard non-ASCII characters to improve reading.

Paper Outline In Sect. 2, we formalize definitions and properties about finite probability to be used in the rest of the paper. In Sect. 3, we introduce the concept of typical sequence. In Sect. 4, we state the source coding theorem and give an outline of the proof of the direct part. In Sect. 5, we formalize the concept of channel and illustrate related definitions thoroughly using the example of the binary symmetric channel. Finally, we state and prove the direct part of the channel coding theorem in Sect. 6. Section 7 is dedicated to related work.

2 The Basics: Finite Probability

We introduce basic definitions about probability (to explain the notations to be used in this paper) and formalize the weak law of large numbers. We do not claim that this formalization is a major contribution in itself because there exist more general formalizations of probability theory (in particular in the HOL proof-assistant [6,11,13]) but providing a new formalization using SSReflect will allow us to take advantage of its library to prove Shannon's theorems.

2.1 Probability Distributions

A distribution over a finite type `A` (i.e., of type `finType` in SSReflect) is defined as a real-valued probability mass function `pmf` (`R` is the type of reals in Coq standard library) with positive outputs (proof `pmf0` below) that sum to 1 (proof `pmf1`; the big sum operator comes from SSReflect [7]):

```
0 Record dist := mkDist {
1   pmf :> A → R ;
2   pmf0 : ∀ a, 0 ≤ pmf a ;
3   pmf1 : Σ_(a ∈ A) pmf a = 1 }.
```

`P : dist A` is a `Record` but, thanks to the coercion line 1, we can write "`P a`" as a function application to represent the probability associated with `a`.

We will be led to define several kinds of distributions in the course of this paper. Here is a first example. Given distributions `P1` over `A` and `P2` over `B`, the *product distribution* `P1` × `P2` over `A * B` is defined as follows:

```
Definition Pprod_dist : dist [finType of A * B].
apply mkDist with (fun x ⇒ P1 x.1 * P2 x.2) ... Defined.
```

(We omit the proofs of `pmf0` and `pmf1` in this paper; the `.1` (resp. `.2`) notation is for the first (resp. second) pair projection); the notation `[finType of ...]` is just a type cast.)

Given a distribution `P` over `A`, the probability of an event (encoded as a boolean predicate of type `pred A`) is defined as follows:

```
Definition Pr (Q : pred A) := Σ_(a ∈ A | Q a) P a.
```

2.2 Random Variables

We formalize a random variable as a distribution coupled with a real-valued function: `Record rvar A := {rv_dist : dist A ; rv_fun :> A →R }.` (This definition is sufficient because `A` is a finite type.) Again, thanks to the coercion, given a random variable `X` and `a` belonging to its sample space, one can write "`X a`" as in standard mathematical writing despite the fact that `X` is actually a `Record`. Furthermore, we note p_`X` the distribution underlying the random variable `X`.

Given a random variable `X` over `A`, and writing `img X` for its image, we define for example the expected value as follows:

```
Definition E := Σ_(r ← img X) r * Pr p_X [pred i | X i =R r].
```

Let us now define the sum of random variables. Below, `n.-tuple A` is the SSReflect type of `n`-tuples over `A` (A^n in standard mathematical writing).

Given distributions `P1` over `A` and `P2` over `n.-tuple A` the *joint distribution* `P` over `n+1.-tuple A` is defined in terms of marginal distributions by the following predicate:

```
Definition joint :=
  (∀ x, P1 x = Σ_(i ∈ {:n+1.-tuple A} | thead i = x) P i) ∧
  (∀ x, P2 x = Σ_(i ∈ {:n+1.-tuple A} | behead i = x) P i).
```

Informally speaking, `joint P1 P2 P` is a relation that defines the distribution `P1` (resp. `P2`) from the distribution `P` by taking into account only the first element (resp. all the elements but the first) of the tuples from the sample space (`thead` returns the first element of a tuple; `behead` returns all the elements but the first).

The random variable `X` is the sum of `X1` and `X2` when the distribution of `X` is the joint distribution of the distributions of `X1` and `X2` and the output of `X` is the sum of the outputs of `X1` and `X2`:

```
Definition sum := joint p_X1 p_X2 p_X ∧
  X =1 [ffun x ⇒ X1 (thead x) + X2 [tuple of (behead x)]].
```

(`[ffun x ⇒...]` is a SSReflect notation to define partial functions over finite domains: `[tuple of ...]` is just a type cast.)

The random variables `X` over `A` and `Y` over `n.-tuple A` are *independent* for a distribution `P` over `n+1.-tuple A` when:

```
Definition inde_rvar := ∀ x y,
  Pr P [pred xy | (X (thead xy) =R x) ∧
                  (Y [tuple of (behead xy)] =R y)] =
  Pr p_X [pred x' | X x' =R x] * Pr p_Y [pred y' | Y y' =R y].
```

We define the sum of several random variables by generalizing `sum` to an inductive predicate (like in [6]). Let `Xs` be a tuple of `n` random variables over `A` and `X` be a random variable over `n.-tuple A`. `sum_n Xs X` holds when `X` is the sum of `Xs`. We also specialize this definition to the sum of independent random variables. Equipped with above definitions, we derive the standard properties of the expected value, such as its linearity, but also properties of the variance. See [14] for details.

2.3 The Weak Law of Large Numbers

The weak law of large numbers is the first fundamental theorem of probability. Intuitively, it says that the average of the results obtained by repeating an experiment a large number of times is close to the expected value. Formally, let `Xs` be a tuple of `n` *identically distributed* random variables, i.e., random variables with the same distribution `P`. Let us assume that these random variables are independent and let us write `X` for their sum, μ for their common expected value, and σ^2 for their common variance. The weak law of large numbers says that the outcome of the average random variable `avg_rv X` gets closer to μ:

```
Lemma wlln ε : 0 < ε →
  Pr p_X [pred x | Rabs (avg_rv X x - μ) ≥_R ε] ≤
    σ² / (n+1 * ε ^ 2).
```

See [14] for the proof of this lemma using the Chebyshev inequality.

3 Entropy and Typical Sequences

We formalize the central concept of a typical sequence. Intuitively, a typical sequence is an n-tuple of symbols (where n is large) that is expected to be observed. For example, a tuple produced by a binary source that emits 0's with probability 2/3 is typical when it contains approximately two thirds of 0's. The precise definition of typical sequences requires the definition of the entropy and their properties relies on a technical result known as the Asymptotic Equipartition Property. (One can find an alternative HOL version of most definitions and properties in this section in [13].)

3.1 Entropy and Asymptotic Equipartition Property

We define the entropy of a random variable with distribution `P` over `A` as follows (where log is the binary logarithm, derived from the standard library of Coq):

```
Definition H := - Σ_(i ∈ A) P i * log (P i).
```

The Asymptotic Equipartition Property (AEP) is a property about the outcome of several random variables that are independent and identically distributed (i.i.d.). Let us assume an `n`-tuple of i.i.d. random variables with distribution `P` over `A`. The probability of the outcome `x` (of type `n.-tuple A`) is:

```
Definition Ptuple x := Π_(i < n) P x_i.
```

(The big product operator comes from SSReflect, `x_i` is for accessing the `i`th element of the tuple `x`.) Informally, the AEP states that, in terms of probability, `- (1 / n) * log(Ptuple P x)` is "close to" the entropy `H P`. Here, "close to" means that, given an ε `> 0`, the probability that `- (1 / n) * log(Ptuple P x)` and `H P` differ by more than ε is less than ε, for `n` greater than the bound `aep_bound` ε defines as follows:

```
Definition aep_σ² := Σ_(x ∈ A) P x * (log (P x))^2 - (H P)^2.
Definition aep_bound ε := aep_σ² P / ε^3.
```

The probability in the AEP is taken over a *tuple distribution*. Given a distribution `P` over `A`, the tuple distribution `P^n` over `n.-tuple A` is defined as follows:

```
Definition Ptuple_dist : dist [finType of n.-tuple A].
apply mkDist with Ptuple. ... Defined.
```

Using above definitions, the AEP can now be stated formally. Its proof is an application of the weak law of large numbers (Sect. 2.3):

```
Lemma aep : aep_bound P ε ≤ n+1 →
  Pr (P^n+1) [pred x | 0 <_R P^n+1 x ∧
    Rabs (- (1 / n+1) * log (Ptuple P x) - H P) ≥_R ε ] ≤ ε.
```

3.2 Typical Sequences: Definition and Properties

Given a distribution `P` over `A` and an ε, a typical sequence is an `n`-tuple with probability “close to” 2^{-nHP}:

```
Definition typ_seq (x : n.-tuple A) ε :=
  exp (- n * (H P + ε)) <_R= Ptuple P x <_R= exp (- n * (H P - ε)).
```

Let us note $\mathcal{TS}$ the set of typical sequences. Using the AEP, we prove that the probability to observe a typical sequence for large `n` is close to 1, corresponding to the intuition that it is expected to be observed in the long run:

```
Lemma Pr_𝒯𝒮_1 : aep_bound P ε ≤ n+1 →
  Pr (P^n+1) [pred i ∈ 𝒯𝒮 P n+1 ε] ≥ 1 - ε.
```

The cardinal of $\mathcal{TS}$ is nearly 2^{nHP}. Precisely, it is upper-bounded by $2^{n(HP+\varepsilon)}$, and lower-bounded by $(1-\varepsilon)2^{n(HP-\varepsilon)}$ for n big enough:

```
Lemma 𝒯𝒮_sup : | 𝒯𝒮 P n ε | ≤ exp (n * (H P + ε)).
Lemma 𝒯𝒮_inf : aep_bound P ε ≤ n+1 →
  (1 - ε) * exp (n+1 * (H P - ε)) ≤ | 𝒯𝒮 P n+1 ε |.
```

4 The Source Coding Theorem

The source coding theorem (a.k.a. the noiseless coding theorem) is a theorem for data compression. The basic idea is to replace frequent words with alphabet sequences and other words with a special symbol. Let us illustrate this with an example. The combination of two Roman alphabet letters consists of 676 ($= 26^2$) words. Since $2^9 < 676 < 2^{10}$, 10 bits are required to represent all the words. However, by focusing on often-used English words (“as”, “in”, “of”, etc.), we can encode them with less than 9 bits. Since this method does not encode rarely-used words (such as “pz”) decoding errors can happen. Given an information source known as a discrete memoryless source (DMS) that emits all symbols with the same distribution `P`, the source coding theorem gives a theoretical lower-bound (namely, the entropy `H P`) for compression rates for compression with negligible error-rate.

4.1 Definition of a Source Code

Given a set `A` of symbols, a `k,n`-source code is a pair of an encoder and a decoder. The encoder maps a `k`-tuple of symbols to an `n`-tuple of bits and the decoder performs the corresponding decoding operation:

```
Definition encT := k.-tuple A → n.-tuple bool.
Definition decT := n.-tuple bool → k.-tuple A.
Record scode := mkScode { enc : encT ; dec : decT }.
```

The rate of a `k,n`-source code `sc` is defined as the ratio of bits per symbol:

```
Definition SrcRate (sc : scode) := n / k.
```

Given a DMS with distribution `P` over `A`, the error rate of a source code `sc` (notation: $\bar{e}_{src}$`(P , sc)`) is defined as the probability of failure for the decoding of encoded sequences:

```
Definition SrcCodeErrRate :=
  Pr (P^k) [pred x | dec sc (enc sc x) ≠ x].
```

4.2 Source Coding Theorem—Direct Part

Given a source of symbols from the alphabet `A` with distribution `P`, there exist source codes of rate `r` $\in \mathcal{Q}^+$ (the positive rationals) larger than the entropy `H P` such that the error rate can be made arbitrarily small:

```
Theorem source_coding_direct : ∀ λ, 0 < λ < 1 →
  ∀ r : Q+, H P < r ≤ 1 →
    ∃ k, ∃ n, ∃ sc : scode A k n,
      r = SrcRate sc ∧ ē_src(P , sc) ≤ λ.
```

Source Coding using the Typical Set The crux of the proof is to instantiate with an adequate source code. We first define the corresponding encoder and decoder functions. For a set `S` of `k+1`-tuples, the encoder `f` encodes the ith element of `S` as the binary encoding of $i + 1$ and elements not in `S` as a string of 0's:

```
Definition f : encT A k+1 n := fun x ⇒
 if x ∈ S then
  let i := index x (enum S) in Tuple (size_nat2bin_b i+1 n)
 else
  [tuple of nseq n false].
```

(`enum S` is the lists of all the elements of `S`; `index` returns the index of an element in a list; `nat2bin_b` is a function that converts an integer $i < 2^n$ to a bitstring, `size_nat2bin_b` being the proof that this bitstring has length n.)

The definition of the decoder requires to have a default element `def ∈ S`. The decoder ϕ returns the $i-1$th element of `S` if i is smaller than the cardinal of `S`, and some default value from `S` otherwise:

```
Definition ϕ : decT A k+1 n := fun x ⇒
  let i := tuple2N x in
  if i is 0 then def else
    if i-1 < | S | then nth def (enum S) i-1 else def.
```

(`tuple2N` interprets bitstrings as Peano integers; `nth` picks up the nth element of a list.) By construction, `f` and ϕ perform lossless coding:

```
Lemma ϕ_f i : ϕ (f i) = i ↔ i ∈ S.
```

In the proof of the source coding theorem, the set `S` is actually taken to be the set $\mathcal{TS}$ of typical sequences and there exists a default element `def` $\in \mathcal{TS}$ when `k` is big enough, bound to be made more precise below.

Formalization of the Bounds Above, we explained how to construct the required source code. Technically, in the formal proof, it is also important to correctly instantiate `n` and `k` (given the source rate `r`, λ and the distribution `P`), such that `k` is "big enough" for the lemma ϕ`_f` to hold. This aspect of the proof is usually overlooked in the information theory literature, so that the (precise) formal definition of these bounds is one of our contributions.

Let us define the following quantities:

```
Definition ε := Rmin (r - H P) λ.
Definition δ := Rmax (aep_bound P (ε / 2)) (2 / ε).
```

`k` must satisfy $\delta \leq$`k` and `k * r` must be a natural. Such a `k` can be constructed using the following lemma:

```
Lemma SrcDirectBound n d m : 0 < m →
  { k | m ≤ (k+1 * d+1) ∧
        frac_part ((k+1 * d+1) * (n / d+1)) = 0}.
```

Let us assume that the rate is `r = num / den+1`. If we note `k'` the natural constructed via the above lemma by taking `n` to be the numerator `num`, `d` to be `den`, and `m` to be δ, then it is sufficient to take `n` equal to `k'+1 * num` and `k` equal to `k'+1 * den+1`.

At this point, we have thoroughly explained how to instantiate the source code required by the source coding theorem. The proof is completed by appealing to the properties of typical sequences (in particular, lemmas `Pr_`$\mathcal{TS}$`_1` and $\mathcal{TS}$`_sup` from Sect. 3.2). The successive steps of the proof can be found in [14].

4.3 Source Coding Theorem—Converse Part

The converse of the Shannon's source coding theorem shows that any source code whose rate is smaller than the entropy of a source with distribution `P` over `A` has non-negligible error-rate:

```
Theorem source_coding_converse : ∀ λ, 0 < λ < 1 →
  ∀ r : Q⁺, 0 < r < H P →
    ∀ n k (sc : scode A k+1 n),
      r = SrcRate sc →
```

```
 SrcConverseBound P (num r) (den r) n λ ≤ k+1 →
$\bar{e}_{src}$(sc , P) ≥ λ.
```

where the bound `SrcConverseBound` gives a precise meaning to the claim that would otherwise be informally summarized as "for `k` big enough":

```
Definition ε := Rmin ((1 - λ) / 2) ((H P - r) / 2).
Definition δ := Rmin ((H P - r) / 2) (ε / 2).
Definition SrcConverseBound := Rmax (Rmax
  (aep_bound P δ) (- ((log δ) / (H P - r - δ)))) (n / r).
```

The proof of the converse part of the source coding theorem is a bit simpler than the direct part because no source code needs to be constructed. See [14] for the detail of the proof steps.

5 Formalization of Channels

5.1 Discrete Memoryless Channel

A *discrete channel* with input alphabet `X` and output alphabet `Y` is a (probability transition) matrix (`ptm`) that expresses the probability of observing an output symbol given some input symbol; it associates to each input a distribution of the corresponding outputs (`channel` is noted `W` hereafter):

```
Definition channel := X → dist Y.
```

The *nth extension of a discrete channel* is the generalization of a discrete channel to the communication of several symbols (`channeln` is noted `Wn` hereafter):

```
Definition channeln n :=
  n.-tuple X → dist ([finType of n.-tuple Y]).
```

A *discrete memoryless channel* (DMC) models channels whose inputs do not depend on past outputs. It is the special case of the nth extension of a discrete channel defined as follows (again, we omit the proofs for `pmf0` and `pmf1`):

```
Definition DMC (w : W) n : Wn n. move⇒ x.
apply mkDist with (fun y ⇒ Π_(i < n) w x_i y_i ). ... Defined.
```

5.2 Mutual Information and Channel Capacity

Given a discrete channel `w` with input alphabet `X` and output alphabet `Y`, and an input distribution `P`, there are two important distributions: the output distribution and the mutual distribution. The *output distribution* (notation: `d(P , w)`) is the distribution of the outputs:

```
Definition out_dist (P : dist X) (w : W X Y) : dist Y.
apply mkDist with (fun y ⇒ Σ_(x ∈ X) w x y * P x). ... Defined.
```

The *mutual distribution* (notation: `d(P ; w)`) is the joint distribution of the inputs and the outputs:

```
Definition mut_dist (P : dist X) (w : W X Y) :
  dist ([finType of X * Y]).
apply mkDist with (fun xy ⇒ w xy.1 xy.2 * P xy.1).
  ... Defined.
```

The *output entropy* (resp. *mutual entropy*) is the entropy of the output distribution (resp. mutual distribution), hereafter noted `H(P , w)` (resp. `H(P ; w)`).

The *mutual information* (notation: `I(P ; w)`) is a measure of the amount of information that the output distribution contains about the input distribution:

```
Definition mut_info_W (P : dist X) (w : W X Y) :=
  H P + H(P , w) - H(P ; w).
```

Finally, the *information channel capacity* is defined as the least upper bound of the mutual information taken over all possible input distributions:

```
Definition upper_bound {A} (f : A → R) b := ∀ a, f a ≤ b.
Definition lub {A} (f : A → R) b :=
  upper_bound f b ∧ ∀ b', upper_bound f b' → b ≤ b'.
Definition capacity (w : W X Y) c := lub (fun P ⇒ I(P ; w)) c.
```

It may not be immediate why the maximum mutual information is called capacity. The goal of the channel coding theorem is to ensure that we can distinguish between two outputs (actually sets of outputs because of potential noise), so as to be able to deduce the corresponding inputs without ambiguity. For each input (of n symbols), there are approximately $2^{n(H(P;w)-HP)}$ typical outputs because $H(P;w) - HP$ is the entropy of the output knowing the input. On the other hand, the total number of typical outputs is approximately $2^{nH(P,w)}$. Since this set has to be divided into sets of size $2^{n(H(P;w)-HP)}$, the total number of disjoint sets is less than or equal to $2^{n(H(P,w)-(H(P;w)-HP))} = 2^{nI(P;w)}$.

5.3 Example: The Binary Symmetric Channel

We illustrate above definitions with the simplest model of channel with errors: the `p`-binary symmetric channel (`BSC` below). In such a channel, the input and output symbols are taken from the same alphabet `X` with only two symbols (hypothesis noted `HX` below). Upon transmission, the input is flipped with probability `p` (with hypothesis `Hp : 0 < p < 1`):

```
Definition BSC : W X X.
move⇒ x. apply mkDist with
  (fun y ⇒ if x = y then 1 - p else p). ... Defined.
```

For convenience, we introduce the *binary entropy function*:

```
Definition H₂ p := - p * log p - (1 - p) * log (1 - p).
```

For any input distribution `P`, we prove that the mutual information can actually be expressed by only the entropy of the output distribution and the binary entropy function:

```
Lemma IPW : I(P ; BSC HX Hp) = H(P , BSC HX Hp) - H₂ p.
```

The maximum of the binary entropy function on the interval $(0, 1)$ is 1, fact that we proved formally in Coq by appealing to the standard library for reals[1]:

```
Lemma H₂_max : ∀ q, 0 < q < 1 → H₂ q ≤ 1.
```

This fact gives an upper-bound for the entropy of the output distribution:

```
Lemma H_out_dist_max : H(P , BSC HX Hp) ≤ 1.
```

The latter bound is actually reached for the uniform input distribution:

```
Definition binary_uniform : dist X.
apply mkDist with (fun x ⇒ 1 / 2). ... Defined.
Lemma H_binary_uniform : H(binary_uniform , BSC HX Hp) = 1.
```

Above facts imply that the capacity of the `p`-binary symmetric channel can be expressed by a simple closed formula:

```
Theorem BSC_capacity : capacity (BSC HX Hp) (1 - H₂ p).
```

5.4 Jointly Typical Sequences

Let us consider a channel `w` with input alphabet `X`, output alphabet `Y`, and input distribution `P`. A *jointly typical sequence* is a pair of two sequences such that: (1) the first sequence is typical for `P`, (2) the second sequence is typical for the output distribution `d(P , w)`, and (3) the pair is typical for the mutual distribution `d(P ; w)`[2]:

```
Definition jtyp_seq n (xy : n.-tuple (X * Y)) ε :=
  typ_seq P ε (uzip1 xy) ∧
  typ_seq (d(P , w)) ε (uzip2 xy) ∧
  typ_seq (d(P ; w)) ε xy.
```

We note $\mathcal{JTS}$ the set of jointly typical sequences. The number of jointly typical sequence is upper-bounded by $2^{n(H(P;w)+\varepsilon)}$:

```
Lemma 𝒥𝒯𝒮_sup ε : | 𝒥𝒯𝒮 P w n ε| ≤ exp (n * (H(P ; w) + ε)).
```

Now follow two lemmas that will be key to prove the channel coding theorem. With high probability (probability taken over the tuple distribution of the mutual distribution), the sent input and the received output are jointly typical:

```
Lemma 𝒥𝒯𝒮_1 : 𝒥𝒯𝒮_1_bound ≤ n →
  Pr ((d( P ; w)^n)) [pred x ∈ 𝒥𝒯𝒮 P w n ε] ≥ 1 - ε.
```

The bound $\mathcal{JTS}$`_1_bound` is defined as follows:

[1] Modulo a slight extension of the corollary of the mean value theorem to handle derivability of partial functions.

[2] Informal definitions about jointly typical sequences seeminglessly switch between $(X \times Y)^n$ and $X^n \times Y^n$; this translates formally to projections and casts that we do not represent explicitly in this paper.

```
Definition 𝒥𝒯𝒮_1_bound :=
  maxn (up (aep_bound P (ε/3)))
 (maxn (up (aep_bound (d(P , w)) (ε/3)))
       (up (aep_bound (d(P ; w)) (ε/3)))).
```

(`up r` is the ceiling of `r`, this is a function from the Coq standard library.) This bound will later appear again in the proof of the channel coding theorem (Sect. 6.3).

In contrast, the probability of the same event (joint typicality) taken over the product distribution of the inputs and the outputs considered independently tends to 0 as `n` gets large:

```
Lemma non_typical_sequences : Pr ((P^n) × ((d(P , w))^n))
  [pred x ∈ 𝒥𝒯𝒮 P w n ε] ≤ exp (- n * (I( P ; w) - 3 * ε)).
```

6 The Channel Coding Theorem

6.1 Formalization of a Channel Code

The purpose of a code is to transform the input of a channel (typically, by adding some form of redundancy) so that the transmitted information can be recovered correctly from the output despite of potential noise. Concretely, given input alphabet `X` and output alphabet `Y`, a (channel) code is (1) a set `M` of codewords, (2) an encoding function that turns a codeword into `n` input symbols, and (3) a decoding function that turns `n` output symbols back into the original codeword (or possibly fails):

```
Definition encT := {ffun M → n.-tuple X}.
Definition decT := {ffun n.-tuple Y → option M}.
Record code := mkCode { enc : encT ; dec : decT }.
```

The *rate* of a code is defined as follows:

```
Definition CodeRate (c : code) := log (| M |) / n.
```

For convenience, we introduce the following predicate to characterize (channel) code rates:

```
Definition CodeRateType r := ∃ n, ∃ d,
  0 < n ∧ 0 < d ∧ r = log n / d.
```

We now define the error-rate. Given a channel `w` and a tuple of inputs `x`, we note “`w (| x )`” the distribution of outputs knowing that `x` was sent. Using this distribution, we first define the probability of decoding error knowing that the codeword `m` from the code `c` was sent (notation: `e(w , c) m`):

```
Definition e (w : W X Y) c m :=
  Pr (w (| enc c m) ) [pred y | dec c y ≠ Some m].
```

Finally, we define the error rate as the average probability of error for a code `c` over channel `w` (notation: $\bar{e}_{cha}$`(w , c)`):

```
Definition ChanCodeErrRate := 1 / | M | * Σ_(m ∈ M) e(w, c) m.
```

6.2 Channel Coding Theorem—Statement of the Direct Part

The (noisy-)channel coding theorem (a.k.a. Shannon's theorem) is a theorem for reliable information transmission over a noisy channel. The basic idea is to represent the original message by a longer message. Let us illustrate this with an example. Assume the original message is either 0 or 1 and is sent over a p-binary symmetric channel (see Sect. 5.3). The receiver obtains the wrong message with probability p. Let us now consider that the original message is 0 and encode 0 into 000 before transmission (in other words, we use a repetition encoding with code rate 1/3). The receiver obtains a message from $\{000, 001, 010, 100\}$ with probability $(1-p)^3 + 3p(1-p)^2$ and it guesses the original message 0 by majority vote. The error probability $1 - ((1-p)^3 + 3p(1-p)^2)$ is smaller than p.

One may guess that the smaller the code rate is, the smaller the error probability becomes. Given a discrete channel `w` (with input alphabet `X` and output alphabet `Y`), the channel coding theorem guarantees the existence of an encoding function and a decoding function such that the code rate is not small (but smaller than the capacity `cap`—hypothesis `capacity w cap`) but is with negligible error-rate:

```
Theorem channel_coding r : CodeRateType r → r < cap →
  ∀ ε, 0 < ε →
    ∃ n, ∃ M, ∃ c : code X Y M n,
      r = CodeRate c ∧ ē_cha(w, c) < ε.
```

6.3 Channel Coding Theorem—Proof of the Direct Part

We formalize a proof by "random coding". In a nutshell: we first fix the decoding function and then select an appropriate encoding function by checking all the possible ones. Selection operates using a criterion about the average error-rate of all the possible encoding functions, weighted according to a well-chosen distribution.

Decoding by Joint Typicality We first fix the decoding function `jtdec`. Given the channel output `y`, `jtdec` looks for a codeword `m` such that the channel input `f m` is jointly typical with `y`. If a unique such codeword is found, it is declared to be the sent codeword (`[pick m | P m]` is a SSReflect construct that picks up an element `m` satisfying the predicate `P`):

```
Definition jtdec P w ε (f : encT X M n) : decT Y M n :=
  [ffun y ⇒ [pick m |
    ((f m, y) ∈ 𝒥𝒯𝒮 P w n ε) ∧
    (∀ m', (m' ≠ m) ⇒ ((f m', y) ∉ 𝒥𝒯𝒮 P w n ε))]].
```

Criterion for Encoder Selection We are looking for a code such that the error-rate can be made arbitrarily small. The following lemma provides a sufficient condition for the existence of such a code:

```
Lemma good_code_sufficient_condition (P : dist X) w ε
    (ϕ : encT X M n → decT Y M n) :
  Σ_(f : encT X M n) (wght P f * ē_cha(w , mkCode f (ϕ f))) < ε →
  ∃ f, ē_cha(w , mkCode f (ϕ f)) < ε.
```

where `wght` is the distribution of encoding functions defined as follows:

```
Definition wght (P : dist X) : dist [finType of (encT X M n)].
apply mkDist with
  (fun f : encT X M n ⇒ Π_(m ∈ M) Ptuple P (f m)). ... Defined.
```

The Main Lemma Our theorem can be derived from the following technical lemma by just proving the existence of appropriate ε_0 and `n`. This lemma establishes that there exists a set of codewords `M` such that decoding by joint typicality meets the above criterion:

```
0 Lemma random_coding_good_code : ∀ ε, 0 ≤ ε →
1   ∀ r, CodeRateType r →
2     ∀ ε0, ε0_condition r ε ε0 →
3       ∀ n, n_condition r ε0 n →
4   ∃ M : finType, 0 < |M| ∧ |M| = Int_part (exp (n * r)) ∧
5   let Jtdec := jtdec P w ε0 in
6   Σ_(f : encT X M n) (wght P f * ē_cha(w , mkCode f (Jtdec f))) < ε.
```

In this lemma, the fact that the rate `r` is bounded by the mutual information appears in the condition ε_0`_condition`:

```
Definition ε0_condition r e e0 :=
  0 < e0 ∧ e0 < e / 2 ∧ e0 < (I(P ; w) - r) / 4.
```

The condition `n_condition` corresponds to the formalization of the restriction "for `n` big enough" (we saw the bound $\mathcal{JTS}$`_1_bound` in Sect. 5.4):

```
Definition n_condition r e0 n := 0 < n ∧ - log e0 / e0 < n ∧
  frac_part (exp (n * r)) = 0 ∧ JTS_1_bound P w e0 ≤ n.
```

Proof of the Main Lemma The first thing to observe is that by construction the error-rate averaged over all possible encoders does not depend on which codeword `m` was sent:

```
Lemma error_rate_symmetry (P : dist X) (w : W Y X) ε :
 0 ≤ ε → let Jtdec := jtdec P w ε in
  ∀ m m',
   Σ_(f : encT X M n) (wght P f * e(w, mkCode f (Jtdec f)) m) =
   Σ_(f : encT X M n) (wght P f * e(w, mkCode f (Jtdec f)) m').
```

Therefore, the left-handside of the conclusion of the main lemma (line 6 above) can be rewritten by assuming that the codeword `0` was sent:

```
Σ_(f : encT X M n)
  wght P f * Pr (w (|f 0)) [pred y ∈ not_preimg (Jtdec f) 0]
```

where `not_preimg (Jtdec f) 0` is the set of outputs that do not decode to `0`.

Let us write $\mathcal{E}$ `f m` for the set of outputs `y` such that `(f m, y)` $\in \mathcal{JTS}$ `P w n` ε. Assuming that `0` was sent, a decoding error occurs when (1) the input and the output are not jointly typical, or (2) when a wrong input is jointly typical with the output (`~:` is a notation for set complementation):

```
[set x ∈ not_preimg (JTdec f) 0] =_i
  (~: ℰ f 0) ∪ ⋃_(i : M | i ≠ 0) ℰ f i.
```

Using the fact that the probability of a union is smaller that the sum of the probabilities, the left-handside of the conclusion of the main lemma can be bounded by the following expression:

```
Σ_(f : encT X M n)
    wght P f * Pr (w (|f 0)) [pred y ∈ ~: ℰ f 0] +   (* (1) *)
Σ_(i|i ≠ 0)Σ_(f : encT X M n)
    wght P f * Pr (w (|f 0)) [pred y ∈ ℰ f i]        (* (2) *)
```

The first summand (1) can be rewritten into

```
Pr (d(P ; w)^n) [pred y ∈ ~: 𝒥𝒯𝒮 P w n ε₀]
```

which can be bounded using the lemma $\mathcal{JTS}$`_1` (Sect. 5.4). The second summand (2) can be rewritten into

```
k * Pr (P^n × (d(P, w))^n) [pred x ∈ 𝒥𝒯𝒮 P w n ε₀]
```

which can be bounded using the lemma `non_typical_sequences` (Sect. 5.4). The bounds ε_0 and `n` have been carefully chosen so that the proof can be concluded with symbolic manipulations. See [14] for details.

7 Related Work

The formalization of Shannon's theorems in this paper as well as the formalization of advanced information-theoretic concepts (channels, jointly typical sequences, etc.) are new. Yet, one can find formalization of more basic concepts of information theory in the literature. [9] formalizes (conditional) entropy and (conditional) mutual information (based on the seminal work by Hurd [4]), defines a notion of information leakage, and applies it to the verification of privacy properties of a protocol. [13] provides a formalization of the AEP and presents the source coding theorem as a potential application; in other words, our paper can be seen as the direct continuation of [13], though in a different proof-assistant.

For the purpose of this paper, we formalized finite probability using SSReflect. As we have hinted at several times in this paper, this formalization was important to take advantage of SSReflect's library (in particular, canonical big operators [7]). We limit ourselves to finite probability because it is enough for our purpose (as for the information theory formalized in [9]). [8] provides an alternative formalization of probabilities in Coq but that is biased towards verification of randomized algorithms. Hasan et al. formalize probability theory on more general grounds in the HOL proof-assistant: [6] formalizes the expectation properties (this is also based on the work by Hurd [4]), [11] provides a formalization of the Chebyshev inequality and of the Weak Law of Large Numbers.

Our formalization of the source coding theorem follows [3, Chapter 1] with nevertheless much clarification (in particular, formalization of bounds).

8 Conclusion and Future Work

We presented a formalization of information-theoretic definitions and lemmas in the SSReflect extension of the Coq proof-assistant. Besides basic material such as finite probability and typical sequences, this formalization includes a formalization of channels (duly illustrated with the example of the binary symmetric channel), codes (for source and channel coding), and jointly typical sequences. We use this formalization to produce the first formal proofs of the source coding theorem (direct and converse parts), that establishes the limit to possible data compression, and the direct part of the channel coding theorem, that establishes the limit to reliable data transmission over a noisy channel. Compared to pencil-and-paper proofs, our formalization has the added value to make precise the construction of asymptotic bounds.

We believe that the library that we have formalized can be used to formalize further results about information theory (primarily, the converse of the channel coding theorem) and also results of unconditional security (e.g., the proof of the perfect secrecy of the one-time pad [2]).

The channel coding theorem proves the existence of codes for reliable data transmission. Such codes play a critical role in IT products (e.g., LDPC codes in storage devices). As a first step towards the verification of the implementation of codes, we have been working on formalizing their basic properties ([14] already provides several standard proofs about coding theory).

References

1. Shannon, C.E.: A Mathematical Theory of Communication. Bell System Technical Journal, vol. 27, pp. 379–423 and 623–656. 1948.
2. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell System Technical Journal, Vol. 28, pp. 656–715. 1949.
3. Uyematsu, T.: Modern Shannon Theory, Information theory with types. In Japanese. Baifukan (1998).
4. Hurd, J.: Formal Verification of Probabilistic Algorithms. PhD Thesis, Trinity College, University of Cambridge, UK (2001).
5. Cover, T.M., Thomas, J.A.: Elements of Information Theory, 2nd edition. Wiley-Interscience (2006).
6. Hasan, O., Tahar, S.: Verification of Expectation Using Theorem Proving to Verify Expectation and Variance for Discrete Random Variables. J. Autom. Reasoning 41:295–323 (2008).
7. Bertot, Y., Gonthier, G., Ould Biha, S., Pasca, I.: Canonical Big Operators. In: TPHOLs 2008. LNCS, vol. 5170, pp. 86–101. Springer, Heidelberg (2008).

8. Audebaud, P., Paulin-Mohring, C.: Proofs of randomized algorithms in Coq. Sci. Comput. Program. 74(8):568–589 (2009).
9. Coble, A.R.: Anonymity, Information, and Machine-Assisted Proof. PhD Thesis, King's College, University of Cambridge, UK (2010).
10. The Coq Development Team. Reference Manual. Version 8.3. Available at `http://coq.inria.fr`. INRIA (2004-2010).
11. Mhamdi, T., Hasan, O., Tahar, S.: On the Formalization of the Lebesgue Integration Theory in HOL. In: Proc. of ITP 2010. LNCS, vol. 6172, pp. 387–402. Springer, Heidelberg (2010).
12. Gonthier, G., Mahboubi, A., Tassi, E.: A Small Scale Reflection Extension for the Coq system. Version 10. Technical report RR-6455. INRIA (2011).
13. Mhamdi, T., Hasan, O., Tahar, S.: Formalization of Entropy Measures in HOL. In: Proc. of ITP 2011. LNCS, vol. 6898, pp. 233–248. Springer, Heidelberg (2011).
14. Affeldt, R., Hagiwara, M.: Formalization of Shannon's Theorems in SSReflect-Coq. Coq scripts. `http://staff.aist.go.jp/reynald.affeldt/shannon`.