Reasoning with Conditional Probabilities and Joint Distributions in Coq

Reynald Affeldt[1], Jacques Garrigue[2], Takafumi Saikawa[2,3]

[1] National Institute of Advanced Industrial Science and Technology

[2] Graduate School of Mathematics, Nagoya University

[3] Peano System Inc.

Abstract Probabilities occur in many applications of computer science, such as communication theory and artificial intelligence. These are critical applications that require some form of verification to guarantee the quality of their implementations. Unfortunately, probabilities are also the typical example of a mathematical theory whose abuses of notations make pencil-and-paper proofs difficult to formalize. In this paper, we experiment a new formalization of conditional probabilities that we validate with two applications. First, we formalize the foundational definitions and theorems of information theory, extending previous work with new lemmas. Second, we formalize the notion of conditional independence and its properties, paving the road for a formalization of graphical probabilistic models.

1 Introduction

Probabilities occur in many applications of computer science. One finds them in information theory to reason about the size of compressed data, in quantitative information flow analysis, in the analysis of side-channel attacks, etc. They are also omnipresent in artificial intelligence. These are critical applications that require some form of verification to guarantee the quality of their implementations.

Unfortunately, pencil-and-paper proofs about probabilities are not easily amenable to formalization. The **first reason** is that it is a difficult topic. The "ideal" approach (in the sense of "most general") seems to be to derive the formalization of probabilities from a formalization of measure theory and Lebesgue integration. Unfortunately, not all proof assistants have such formalizations available (there is one in HOL [MHT11] and one in Isabelle/HOL [Höl12]). In fact, most proof assistants do not even have a decent formalization of real analysis, and, more generally, what is a good formalization of real analysis is still a topic of research [ACR18]. On the other hand, information theory and artificial intelligence (the computer science topics we are interested in) are mostly about discrete probabilities, so that a full-fledged formalization of measure theory and Lebesgue integration looks like a sledgehammer to crack a nut. There therefore seems to be much value in exploring a mechanization of discrete probabilities that allows for formal reasoning without this detour. Though arguably a substantial simplification, this does not provide an immediate solution because probabilities are the typical example of a mathematical theory where abuses of notations are omnipresent. This is the **second reason** why formalization of probabilities is difficult. Indeed, it is not obvious how to recover the terseness of the presentation of information theory or artificial intelligence textbooks. The latter typically feature many implicit assumptions; the main example is the omission of the underlying distributions when talking about probabilities, often ruled out as "obvious from context". But what is obvious for a human reader is generally not for a proof assistant.

In this paper, we focus on the problem of formalizing and using conditional probabilities and joint distributions. Concretely, we extend an existing formalization of information theory [AHS+18] with an "explicit" formalization of conditional probability. We say "explicit" because the original formalization [AHS+18] was already dealing with conditional probabilities but given in the form of stochastic matrices, as used to model channels in information theory (see Sect. 5 for a more precise explanation). To show that our extension is indeed useful, we validate it with two substantial applications:

- First, we formalize the foundations of information theory. This is not a new topic but we are able to improve on previous work by formalizing lemmas there were not formalized before. In fact, we are on the verge of completing the formalization of [CT06, Chapter 2], which provides the basic definitions of the standard textbook for information theory.

- Second, we formalize the notion of conditional independence. We validate this definition by proving the so-called *graphoid axioms* [PP85]. We are not aware of a previous attempt to formalize the graphoid axioms from the ground up, but they are key to reason about probabilistic graphical models used in artificial intelligence.

Paper Outline In Sect. 2, we explain how we formalize distributions, joint distributions, and conditional probabilities. In Sect. 3, we explain our formalization of information theory. In Sect. 4, we explain our formalization of conditional independence. We review related work in Sect. 5 and conclude in Sect. 6.

About Notations This paper displays Coq code verbatim. We rely on the MathComp library whose notations are explained when used for the first time. We enforced a uniform naming convention for variables. We use `A`, `B`, `C`, `D`, etc. for finite types. We use `a`, `b`, etc. for elements of resp. `A`, `B`, etc. We use `E : {set A}`, `F : {set B}`, `G : {set C}`, `H : {set D}`, etc. for events. We use `P`, `Q`, etc. to range over distributions. We use `X`, `Y`, `Z`, `W`, etc. for random variables with values in `A`, `B`, `C`, `D`, etc.

2 Formalization of Conditional Probability

2.1 Background: Formalization of Distributions with a Finite Support

We use the following formal definition of distributions with a finite support [AHS14]:

```
Record dist := mkDist {
  pmf :> A -> R+ ;
  pmf1 : \rsum_(a in A) pmf a = 1}.
```

`A` is a type with a finite number of elements: it has type `finType` [GMT08]. `A -> R+` is the type of functions with domain `A` and real, nonnegative outputs. `\rsum_(a in A)` (where `a` is a binder) is a notation for the sum over `A`. `{dist A}` is a notation that causes the type inference to correctly identify `A` has a `finType` *even when* `A` is a composite `finType` (e.g., the product of two finite types `A1 * A2`).

Given a distribution `P : {dist A}`, we can formalize an event as a finite set over `A` (i.e., a Boolean predicate over `A` represented as a list, whose type is denoted `{set A}` [MT16]) and compute its probability by summing the individual probabilities of its elements:

```
Variables (A : finType) (P : {dist A}).
Definition Pr (E : {set A}) := \rsum_(a in E) P a.
```

2.2 Joint Distributions

Using the definition of Sect. 2.1, a joint distribution can be represented by a distribution over a product type such as `{dist A * B}` or `{dist 'rV[A]_n}` (where `'rV[A]_n` represents the type of row vectors of size `n` [MT16]).

Given a joint distribution, we often need to consider marginal distributions. For that purpose, we provide a number of functions.

Joint Distributions over a Product Type `Bivar.fst` (`Bivar` for "bivariate") is the left marginal built from a joint distribution over a product type. It is defined as follows (using the distributions from Sect. 2.1):

```
(* Module Bivar *)
Variables (A B : finType) (P : {dist A * B}).
Definition ml a := \rsum_(x in {: A * B} | x.1 == a) P x.
Lemma ml0 a : 0 <= ml a. Proof. ... Qed.
Lemma ml1 : \rsum_(a in A) ml a = 1. Proof. ... Qed.
Definition fst : dist A := locked (makeDist ml0 ml1).
```

`ml` is the probability mass function. Its definition uses a general version of the `\rsum` notation that allows for filtering (here, only the `x`'s that satisfy `x.1 == a` are considered). `{: A * B}` is the product type understood as a finite type. `ml0` and `ml1` prove that `ml` forms a distribution. Finally, `fst` builds the distribution using the `makeDist` constructor. `locked X` is a MathComp construct to prevent unwanted unfolding/evaluation of `X` during formal proofs: it needs to be removed explicitly using specific tactics or lemmas (this contributes to more maintainable proofs).

Regarding joint distributions over a product type, one might also want to build the right marginal `Bivar.snd` (which is of course defined similarly) and the distribution `Swap.d` that swaps the left and the right elements.

The following table summarizes the distributions that one might want to build from of a joint distribution over a product type:

Original distribution	*Built distribution*	*and its type*
`P : {dist A * B}`	`Swap.d P`	`{dist B * A}`
	`Bivar.fst P`	`{dist A}`
	`Bivar.snd P`	`{dist B}`

A Zoo of Distributions Similarly to joint distributions over a product type, one might want to build distributions from joint distributions over a triple of distributions. For example, in the table below, `TripC12.d` permutes the first and the second element, while `Proj13.d` build the marginal w.r.t. the second element:

Original distribution	*Built distribution*	*and its type*
`P : {dist A * B * C}`	`TripA.d P`	`{dist A * (B * C)}`
	`TripC12.d P`	`{dist (B * A) * C}`
	`TripC23.d P`	`{dist (A * C) * B}`
	`TripC13.d P`	`{dist (C * B) * A}`
	`Proj13.d P`	`{dist A * C}`
	`Proj23.d P`	`{dist B * C}`

In Sect. 4, we also deal with distributions made of quadruples of distributions. They are prefixed with `Quad` and follow a similar naming convention. See Sect. 4.3.2 for example.

In the case of multivariate distributions (i.e., distributions of type `{dist 'rV[A]_n}`), one might want to build the marginals made of the head or the tail, like `Multivar.head_of` or `Multivar.tail_of` in the table below:

Original distribution	*Built distribution*	*and its type*
`P : {dist 'rV[A]_n.+1}`	`Multivar.to_bivar P`	`{dist A * 'rV[A]_n}`
	`Multivar.head_of P`	`{dist A}`
	`Multivar.tail_of P`	`{dist 'rV[A]_n}`
	`Multivar.belast_last P`	`{dist 'rV[A]_n * A}`
`P : {dist A * 'rV[A]_n}`	`Multivar.from_bivar P`	`{dist 'rV[A]_n.+1}`
`P : {dist 'rV[A]_n}`	`Take.d P i`	`{dist 'rV[A]_i}`
	`Nth.d P i`	`{dist A}`
`P : {dist 'rV[A]_n.+1 * B}`	`PairTake.d P`	`{dist ('rV[A]_i * A) * B}`

See 3.2.3 for an explanation of `PairTake.d`, other distributions should be self-explanatory.

In the following, we will need to make explicit mention of the distributions above, whereas they are often implicit in pencil-and-paper proofs.

2.3 Conditional Probability

We use joint distributions to define conditional probabilities.

Given a joint distribution `P : {dist A * B}` and two events `E : {set A}` and `F : {set B}`, the probability of `E` knowing `F` (i.e., $\Pr[E|F]$) is formalized as follows:

```
Variables (A B : finType) (P : {dist A * B}).
Definition cPr E F := Pr P (setX E F) / Pr (Bivar.snd P) F.
```

`setX E F` is the Cartesian product of the sets `E` and `F`. Using a pencil-and-paper prose that mentions explicitly the underlying distributions (what many textbooks do not do), the definition of `cPr P E F` would read as $\Pr_P[E|F] \stackrel{\text{def}}{=} \frac{\Pr_P(E,F)}{\Pr_{P_2}(F)}$, where P_2 represents the right marginal of P. This is why we use the notation `\Pr_P[E | F]` for `cPr P E F` in the Coq scripts.

We formalized a number of lemmas about conditional probabilities, such as Bayes' theorem, its general form, etc. We do not provide a complete presentation because their proofs follow known formal proofs [HT11]. Let us just provide two examples that are used to prove the graphoid axioms in Appendix A. The proofs of the weak union, contraction, and intersection axioms use the product rule (i.e., $\Pr[E, F|G] = \Pr[E|F, G]\Pr[F|G]$), which we state formally as follows:

```
Lemma product_rule E F G :
  \Pr_P [setX E F | G] = \Pr_(TripA.d P) [E | setX F G] * \Pr_(Proj23.d P) [F | G].
```

The proof of intersection uses the conditional property of the universal set, i.e., the fact that $\Pr[U|A] = 1$ where U is the universal set and A is an event with non-zero probability:

```
Variables (A B : finType) (P : {dist B * A}).
Lemma cPr_1 a : Bivar.snd P a != 0 -> \rsum_(b in B) \Pr_P[[set b] | [set a]] = 1.
```

In the formal definition, `[set x]` is a MathComp notation for the singleton set $\{x\}$.

3 Application 1: Formalization of Information Theory

As a first application of our formalization of conditional probability and joint distributions, we formalize the basic elements of information theory. This includes lemmas that (to the best of our knowledge) were not formalized before (e.g., the chain rule for information).

3.1 Entropy and Conditional Entropy

3.1.1 Background: Entropy

Entropy is a measure of the uncertainty of a random variable. In [CT06, Sect. 2.1], it is defined as follows (for a random variable X with alphabet $\mathcal{X}$ and probability mass function $p(x) \stackrel{\text{def}}{=} \Pr(X = x)$):

$$\mathcal{H}(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x)$$

Since the body of the definition uses only distributions, we formalized it as follows in [AHS14] (Coq notation: `H):

```
Variables (A : finType) (P : {dist A}).
Definition entropy := - \rsum_(a in A) P a * log (P a).
```

One may wonder what this definition means if $p(a) = 0$ for some a. Here we rely on the fact that Coq assumes all functions (including `log`) to be total. As a result, the product of 0 with `log` 0 is equal to 0, which happens to be what we want here. While Coq uses completed versions of logarithm and division for instance, assuming both of them to be 0 outside of their usual definition domains, theorems on specific functions are restricted to their usual definition domains. For instance $x/x = 1$ is true only if $x \neq 0$, *cf.* lemma `cPr_1` (Sect. 2.3).

3.1.2 Conditional Entropy

When it comes to conditional entropy, [CT06] gives a choice of several definitions (of course all equivalent) using slightly different notions: a more primitive notion of conditional entropy (which is defined simultaneously), using conditional probabilities, using joint distributions, or using the expectation of a random variable [CT06, Equations 2.10–2.13]. For example, using conditional probabilities, the definition of conditional entropy reads as follows:

$$\mathcal{H}(Y \mid X) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x) \Pr[y|x] \log \Pr[y|x]$$

Let us use the definition of conditional probability from Sect. 2.3 to write this definition in Coq. Like for entropy (Sect. 3.1.1), we only care about distributions. We are given a joint distribution `QP : {dist B * A}` (to be able to write down the conditional probability $\Pr[y|x]$) from which we compute the marginal `P := Bivar.snd QP` (so as to be able to write $p(x)$):

```
(* Module CondEntropy *)
Variables (A B : finType) (QP : {dist B * A}).
Definition h1 a := - \rsum_(b in B)
  \Pr_QP[[set b] | [set a]] * log (\Pr_QP[[set b] | [set a]]).
Let P := Bivar.snd QP.
Definition h := \rsum_(a in A) P a * h1 a.
```

Let us check the validity of our formal definition with an example of computation and a (non-trivial) property of conditional entropy.

Computation of Conditional Entropy Let us consider the joint distribution with the following probability mass function [CT06, Example 2.2.1]:

```
Definition sample_f : 'I_4 * 'I_4 -> R := [eta (fun=>0) with
(zero,zero) |-> 1/8, (zero,one) |-> 1/16, (zero,two) |-> 1/16, (zero,three) |-> 1/4,
(one,zero) |-> 1/16, (one,one) |-> 1/8, (one,two) |-> 1/16, (one,three) |-> 0,
(two,zero) |-> 1/32, (two,one) |-> 1/32, (two,two) |-> 1/16, (two,three) |-> 0,
(three,zero) |-> 1/32, (three,one) |-> 1/32, (three,two) |-> 1/16, (three,three) |-> 0].
```

Since it is nonnegative and sums to 1, it is a distribution:

```
Lemma sample_f0 : forall x, 0 <= sample_f x. Proof. ... Qed.
Lemma sample_f1 : \rsum_(x in {: 'I_4 * 'I_4}) sample_f x = 1. Proof. ... Qed.
Definition sample_d : {dist 'I_4 * 'I_4} := locked (makeDist sample_f0 sample_f1).
```

We compute $H(X|Y)$ where X and Y are random variables whose joint distribution is `sample_d`. Using the symbolic computations capabilities of Coq as implemented by tactics such as `lra`, we recover the expected the result:

```
Lemma sample_conditional_entropyE : CondEntropy.h sample_d = 11/8.
```

"Information Can't Hurt" One important property of information theory is that conditioning reduces entropy, i.e., $H(X|Y) \leq H(X)$ [CT06, Thm 2.6.5]. Using the formal definitions explained so far, this can be stated in Coq as follows:

```
Variables (A B : finType) (PQ : {dist A * B}).
Let P := Bivar.fst PQ.
Lemma information_cant_hurt : CondEntropy.h PQ <= `H P.
```

We provide a succinct Coq proof in Sect. 3.2.1, using the properties of mutual information.

3.1.3 The Chain Rule for Entropy

The chain rule for entropy is stated for n random variables X_1, X_2, ..., X_n drawn according to $p(x_1, x_2, \ldots, x_n)$ [CT06, Thm 2.5.1]:

$$\mathcal{H}(X_1, \ldots, X_n) = \sum_{i=1}^{n} \mathcal{H}(X_i \,|\, X_{i-1}, \ldots, X_1)$$

First, we observe that $0 < n$. We reflect this in the formal statement below by considering `n.+1` instead of `n`. Second, we observe that the pencil-and-paper notation is such that $\mathcal{H}(X_i \,|\, X_{i-1}, \ldots, X_1) = \mathcal{H}(X_1)$ when $i = 1$. This is why, in the formal statement below, we distinguish the case where $i = 1$ from the case where $1 < i$ (which become `i = 0` and `0 < i` in Coq where we count from `0`):

```
Lemma chain_rule_rV A n (P : {dist 'rV[A]_n.+1}) :
  `H P = \rsum_(i < n.+1)
          if i == 0 then
            `H (Multivar.head_of P)
          else
            CondEntropy.h (Swap.d (Multivar.belast_last (Take.d P (lift ord0 i)))).
```

In the case where `i = 0`, the chain rule returns `` `H (Multivar.head_of P) ``, i.e., `` `H P ``, since `P` is a vector with only one element. The case with `0 < i` is a little bit involved. `lift ord0 i` means $i + 1$; it is written with `lift` so that it has the appropriate type `'I_n.+2`, the type of natural number strictly less than `n.+2` [MT16]. The calls to `Take.d`, `Multivar.belast_last`, and `Swap.d` (see Sect. 2.1) build the product distribution corresponding to $(X_{i+1}, (X_i, \ldots, X_1))$. This may look a little bit more complicated than what one could expect with lists (instead of (finite-size) vectors), but since distributions have finite support we cannot easily recast the problem in terms of "distributions of lists".

The proof is by induction on n and uses as an intermediate lemma the "chain rule" [CT06, Thm 2.2.1], which is actually a version of the chain rule for entropy but restricted to three random variables. One can find this restricted version in related work (e.g., [Mor12]), but we

are not aware of an existing generalization to n distributions. The formal statement above show that the need to build a variety of joint distributions makes the generalization not immediate. It is nevertheless useful, for example to prove the chain rule for information (see Sect. 3.2.3) or Han's inequality (see [AHS+18]).

3.2 Mutual Information and Conditional Mutual Information

In this section, we give an overview of our formalization of mutual information, conditional mutual information, and the chain rule for information. It is technically a bit more involved than the formalization of conditional entropy in Sect. 3.1, but it uses similar ideas, hence the shorter presentation.

3.2.1 Mutual Information

Mutual information quantifies the information shared by two random variables. Given two random variables X and Y with joint probability mass function $p(x, y)$ and marginals $p(x)$ and $p(y)$, mutual information is defined as $\mathcal{I}(X\,;\,Y) = \mathcal{D}(p(x,y)\,||\,p(x)p(y))$ where $\mathcal{D}(\cdot\,||\,\cdot)$ is the Kullback-Leibler distance [CT06, Sect. 2.3] (also called *relative entropy*). The formal definition is immediate given that we have already defined in previous work [AHS14] the relative entropy `D(... || ...)` and the product distribution `` `x ``:

```
Variables (A B : finType) (PQ : {dist A * B}).
Let P := Bivar.fst PQ.
Let Q := Bivar.snd PQ.
Definition mi := D(PQ || P `x Q).
```

The notation `P `x Q` is for the distribution with probability mass function $(x, y) \mapsto P(x) \times Q(x)$. It is defined using a more generic construct explained in Sect. 3.3.

This formal definition is equipped with the mandatory lemmas that, among others, provide a short proof for the "information can't hurt" property of Sect. 3.1.2:

```
Lemma information_cant_hurt : CondEntropy.h PQ <= `H P.
Proof. rewrite -subR_ge0 -MutualInfo.miE2; exact: MutualInfo.mi_ge0. Qed.
```

`MutualInfo.miE2` provides an alternative definition of mutual information in terms of entropy (i.e., $\mathcal{I}(X\,;\,Y) = \mathcal{H}(X) - \mathcal{H}(X\,|\,Y)$ [CT06, Thm 2.4.1]) and `MutualInfo.mi_ge0` establishes that mutual information is nonnegative [CT06, p. 28].

3.2.2 Conditional Mutual Information

Conditional mutual information quantifies the information shared by two random variables *under the assumption that the result of a third random variable is already known*: $\mathcal{I}(X;Y|Z) = \mathcal{H}(X\,|\,Z) - \mathcal{H}(X\,|\,Y,Z)$ [CT06, Sect. 2.5]. The formalization is immediate given the definition of conditional entropy from Sect. 3.1.2:

```
Variables (A B C : finType) (PQR : {dist A * B * C}).
Definition cmi := CondEntropy.h (Proj13.d PQR) - CondEntropy.h (PairA.d PQR).
```

3.2.3 The Chain Rule for Information

Mutual information and conditional mutual information satisfy a chain rule [CT06, Thm 2.5.2]:

$$\mathcal{I}(X_1, X_2, \ldots, X_n\,;\,Y) = \sum_{i=1}^{n} \mathcal{I}(X_i; Y | X_{i-1}, \ldots, X_1)$$

It is similar to the chain rule for entropy and conditional entropy of Sect. 3.1.3.

For the formal statement, we first prepare the joint distribution $X_1, X_2, \ldots, X_n, Y$ as the distribution `PY` of type `{dist 'rV[A]_n.+1 * B}`:

```
Variables (A : finType). Let B := A.
Variables (n : nat) (PY : {dist 'rV[A]_n.+1 * B}).
```

For the case where $i = 1$, we take the mutual information of the distribution X_1, Y, i.e., `(PairNth.d PY ord0)` (`ord0` is the natural number 0 but with the type `'I_n.+1`). For the case where $1 < i$, we take the conditional mutual information of the distribution $X_i, Y, (X_{i-1}, \ldots, X_1)$, i.e., `TripC23.d (TripC12.d (PairTake.d PY i))`:

```
Let f (i : 'I_n.+1) : {dist A * 'rV[A]_i * B} := TripC12.d (PairTake.d PY i).
Let f23 (i : 'I_n.+1) : {dist A * B * 'rV[A]_i} := TripC23.d (f i).

Lemma chain_rule_information :
  MutualInfo.mi PY = \rsum_(i < n.+1)
    if i == 0 then
      MutualInfo.mi (PairNth.d PY ord0)
    else
      cmi (f23 i).
```

The proof uses the chain rule for entropy of Sect. 3.1.3.

3.3 Convexity of Entropy and Mutual Information

The goal of this section are formal proofs of convexity for entropy and mutual information. We start by defining convexity.

Let `A` and `B` be convex spaces. A function `f` of type `A -> B` is convex when it satisfies the predicate `convex_function`:

```
Definition convex_function_at a b (t : prob) := f (a <| t |> b) <= f a <| t |> f b.
Definition convex_function := forall a b (t : prob), convex_function_at f a b t.
```

The type `prob` is for reals between 0 and 1; `a <| t |> b` is a notation for $t\,a + (1-t)\,b$. Similarly, the predicate `concave_function` is for functions f such that $-f$ is convex. The type of distributions `{dist A}` (where `A` is a finite type) and the type of reals `R` happen to be convex spaces. See [AHS+18, file `convex.v`] for details.

Given the above definitions, we can state the convexity of entropy as follows [CT06, Thm 2.7.3]:

```
Variable (A : finType).
Hypothesis A_not_empty : 0 < #|A|.
Lemma entropy_concave : concave_function (fun P : {dist A} => `H P).
```

The proof relies on a convexity property of relative entropy.

The convexity of mutual information requires an additional construct. Let us first recall the pencil-and-paper statement. Let (X, Y) be two random variables drawn according to $p(x, y) = p(x) \Pr[y|x]$. For fixed $\Pr[y|x]$, $\mathcal{I}(X\,;\,Y)$ is a concave function of $p(x)$ [CT06, Thm 2.7.4, first part]. In the formal statement, we represent $\Pr[y|x]$ as `W : A -> {dist B}` (i.e., a stochastic matrix), and $p(x)$ is given as the parameter `P : {dist A}`. We build the joint distribution $p(x, y)$ using the function `make_joint` defined as follows:

```
(* Module CDist *)
Variables (A B : finType).
Record t := mkt {P : dist A ; W :> A -> dist B}.
Definition joint_of (x : t) : {dist A * B} := ProdDist.d (P x) (W x).
Definition make_joint (P : dist A) (W : A -> dist B) : {dist A * B} :=
  joint_of (mkt P W).
```

Given a distribution `P` and a stochastic matrix `W`, `ProdDist.d` builds the "product distribution" with probability mass function $(x, y) \mapsto P(x) \times Q(x, y)$. (This explains the product distribution of Sect. 3.2.1: `P `x Q` is a notation for `ProdDist.d P (fun=> Q)`.) By construction, the left marginal of the joint distribution is `P`, which is one part of the original condition $p(x, y) = p(x) \Pr[y|x]$ and lets us prove the convexity of mutual information:

```
Variables (A B : finType) (Q : A -> dist B).
Hypothesis B_not_empty : 0 < #|B|.
Lemma mutual_information_concave :
  concave_function (fun P => MutualInfo.mi (CDist.make_joint P Q)).
```

The proof uses the convexity of entropy as an intermediate step.

The convexity of mutual information (as a function of $\Pr[y|x]$ for fixed $p(x)$) [CT06, Thm 2.7.4, second part] follows along similar lines (see [AHS+18]).

3.4 More Information Theory

We were actually able to formalize more information theory than what we have explained so far. At the time of this writing, we have almost completed the formalization of [CT06, Chapter 2]: only the sections about sufficient statistics and Fano's inequality are left unexplored.

Among theorems that we have not explained are also the chain rule for relative entropy [CT06, Thm 2.5.3], the independence bound on entropy [CT06, Thm 2.6.6], the definition of Markov chains, and the data-processing inequality [CT06, Thm 2.8.1].

This work uses and complements our previous work that already provides in Coq several theorems from [CT06, Chapter 2]: Jensen's inequality [CT06, Thm 2.6.2] (see [AGS18]), the log sum inequality [CT06, Thm 2.7.1] (see [AHS14]), and the proof that the nonnegativity of the second derivative implies convexity [CT06, Thm 2.6.1].

4 Application 2: Formalization of Conditional Independence

As a second application of our formalization of conditional probability and joint distributions, we formalize conditional independence and we validate this formal definition by establishing the graphoid axioms.

4.1 Random Variables

Given a distribution `P : {dist U}`, a random variable over `A` is defined as a function of type `U -> A`, where `A` is some finite type for the values that the random variable can take:

```
Definition RV {U : finType} (P : {dist U}) (A : finType) := U -> A.
```

This is the same definition as [Mor12]. In the following, we `{RV P -> A}` as a notation for `RV P A`.

A random variable `X : {RV P -> A}` induces a distribution over its codomain. This is the distribution whose probability mass function associates to each element `a` of `A` the probability of the event corresponding to the pre-image of `a` via `X`:

```
Definition f a := Pr P [set i | X i == a].
```

`[set i | X i == a]` is a MathComp notation for the set $X^{-1}(a)$. `RVar.d X` is the formalization of this probability distribution (it just packs the above probability mass function with the proofs that it is nonnegative and that the sum over its codomain is 1, like we did in Sect. 2.2; see [AHS+18] for the the complete Coq definitions and proofs).

In this setting, we observe in particular the following:

- The pencil-and-paper definition $P(X = a)$ becomes `Pr (RVar.d X) [set a]`.

- We can create random variables by pairing existing random variables:

```
Variables (U : finType) (P : {dist U}).
Variables (A : finType) (X : {RV P -> A}) (B : finType) (Y : {RV P -> B}).
Definition RV2 : {RV P -> A * B} := fun x => (X x, Y x).
```

  In the following, `[% X, Y, .., Z]` denotes the iterated pairing of the random variables `X`, `Y`, ..., `Z`.

4.2 Conditional Independence

We are concerned with the formalization of the predicate $X \perp Y \mid Z$, which intuitively means that X is independent of Y given Z. We first recall the textbook definition [KF09, Definition 2.4].

Let X, Y, and Z be random variables and P be a distribution. X is conditionally independent of Y given Z in the distribution P if for all values a, b, and c (belonging resp. to the codomains of X, Y, and Z), we have:

$$\Pr(X = a, Y = b | Z = c) = \Pr(X = a | Z = c) \Pr(Y = b | Z = c).$$

We now move on to the formal definition of conditional independence. First, we give ourselves a sample space `U` with a distribution `P : {dist U}`:

```
Variables (U : finType) (P : {dist U}).
```

Second, we declare three random variables `X`, `Y`, and `Z`. Each of these random variables induces a distribution; let us note `Q` the resulting joint distribution:

```
Variables (A B C : finType).
Variables (X : {RV P -> A}) (Y : {RV P -> B}) (Z : {RV P -> C}).
Let Q := RvarDist.d [% X, Y, Z].
```

Finally, we state the property of conditional independence (Coq notation: `X _|_ Y | Z`):

```
Definition cinde_drv := forall a b c,
  \Pr_Q[[set (a, b)] | [set c]] =
  \Pr_(Proj13.d Q) [[set a] | [set c]] * \Pr_(Proj23.d Q) [[set b] | [set c]].
```

Let us compare with the pencil-and-paper definition. In Coq, the random variables `X`, `Y`, and `Z` have not disappeared: they are inside the `Q` notation. Since we use the definition of conditional probability from Sect. 2.3, both sides of the equality use different joint distributions in Coq. It is `Q` in the left hand-side, whereas the distributions in the right hand-side are marginals: `Proj13.d Q` is the marginal of `Q` w.r.t. `Y` and `Proj23.d Q` the marginal of `Q` w.r.t. `X`.

4.3 Conditional Independence is a Graphoid

Conditional independence satisfies the graphoid axioms [PP85]:

- Symmetry: $X \perp Y \mid Z$ implies $Y \perp X \mid Z$.
- Decomposition: $X \perp Y, W \mid Z$ implies $X \perp Y \mid Z$.
- Weak union: $X \perp Y, W \mid Z$ implies $X \perp Y \mid Z, W$.
- Contraction: $X \perp W \mid Z, Y$ and $X \perp Y \mid Z$ imply $X \perp Y, W \mid Z$.
- Intersection: $X \perp Y \mid Z, W$ and $X \perp W \mid Z, Y$ imply $X \perp Y, W \mid Z$ for positive distributions.

These axioms have an intuitive interpretation. For example, symmetry means that in a state Z where X says nothing new about Y, then Y says nothing new about X; decomposition means that if Y, W say nothing new about X, then Y alone does not say anything new about X; etc.

We have proved the graphoid axioms for conditional independence in Coq. We detail the proofs of symmetry and decomposition in Sections 4.3.1 and 4.3.2; see Appendix A for information about the proofs of weak union, contraction, and intersection.

4.3.1 The Proof of Symmetry in Coq

Let us recall the pencil-and-paper proof:

Lemma 1 (Symmetry)**.** *$X \perp Y \mid Z$ implies $Y \perp X \mid Z$.*

Proof.

$$\begin{aligned} P(Y,X|Z) &= P(X,Y|Z) && \text{(by set-theoretic reasoning)} \\ &= P(X|Z)P(Y|Z) && \text{(because } X \perp Y \mid Z\text{)} \\ &= P(Y|Z)P(X|Z) && \text{(by commutativity of multiplication)} \end{aligned}$$

□

In Coq, we write the statement "$X \perp Y \mid Z$ implies $Y \perp X \mid Z$" as follows:

```
Variable (U : finType) (P : {dist U}).
Variables (A B C : finType) (X : {RV P -> A}) (Y : {RV P -> B}) (Z : {RV P -> C}).
Lemma symmetry : X _|_ Y | Z -> Y _|_ X | Z.
```

The proof script follows the pencil-and-paper proof:

```
1 Proof.
2 rewrite /cinde_drv => H b a c.
3 rewrite -setX1 -cPr_TripC12 setX1 TripC12_RV3.
4 rewrite H.
5 rewrite mulRC.
6 by rewrite 2!Proj23_RV3 2!Proj13_RV3.
7 Qed.
```

At the beginning of the proof, the distributions in the goal and in the hypothesis do not syntactically match: the hypothesis is about the distribution `RVar.d [% X, Y, Z]` whereas the goal is about `RVar.d [% Y, X, Z]`). The purpose of line 3 is to swap the two leading random variables so as to be able to rewrite with the hypothesis at line 4. We apply the commutativity of multiplication at line 5. At this point, the proof is almost completed: line 6 just identifies semantically-equal distributions (e.g., `Proj23.d (RVar.d [% X, Y, Z])` and `Proj13.d (RVar.d [% Y, X, Z])`).

4.3.2 The Proof of Decomposition in Coq

Lemma 2 (Decomposition)**.** *$X \perp Y, W \mid Z$ implies $X \perp Y \mid Z$.*

Proof. [KF09, Sect. 2.1.4.3]

$$\begin{aligned} P(X,Y|Z) &= \sum_w P(X,Y,w|Z) && \text{(reasoning by cases)} \\ &= \sum_w P(X|Z)P(Y,w|Z) && \text{(by } X \perp Y,W \mid Z\text{)} \\ &= P(X|Z)\sum_w P(Y,w|Z) && \text{(by distributivity)} \\ &= P(X|Z)P(Y|Z) && \text{(reasoning by cases)} \end{aligned}$$

□

Here follows the proof statement in Coq:

```
Variables (U : finType) (P : {dist U}) (A B C D : finType).
Variables (X : {RV P -> A}) (Y : {RV P -> B}) (Z : {RV P -> C}) (W : {RV P -> D}).
Lemma decomposition : X _|_ [% Y, W] | Z -> X _|_ Y | Z.
```

For readability, we insert into the proof script intermediate goals that match the pencil-and-paper proof:

```
1  Proof.
2  move=> H; rewrite /cinde_drv => a b c.
3  rewrite Proj13_RV3 Proj23_RV3.
4  transitivity (\rsum_(d <- fin_img W) \Pr_(QuadA23.d Q)[[set (a, (b, d))] | [set c]]).
5    rewrite (reasoning_by_cases_RV2 W); apply eq_bigr => /= d _.
6    by rewrite -2![in RHS]setX1 cPr_QuadA23 -setX1.
7  transitivity (\rsum_(d <- fin_img W)
8    \Pr_(Proj14d Q)[[set a] | [set c]] * \Pr_(Proj234.d Q)[[set (b, d)] | [set c]]).
9    apply eq_bigr => d _.
10   rewrite QuadA23_RV4.
11   rewrite H.
12   by rewrite Proj13_RV3 Proj23_RV3 Proj14_RV4 Proj234_RV4 Proj23_RV3.
13 rewrite -big_distrr /=; congr (_ * _).
14   by rewrite /Proj14d Proj124_RV4 Proj13_RV3.
15 rewrite (reasoning_by_cases_RV2 W); apply eq_bigr => d _.
16 by rewrite Proj234_RV4 Proj23_RV3 setX1.
17 Qed.
```

`Q` is a shortcut for the joint distribution corresponding to the quadruple of random variables `[% X, Y, W, Z]` (`Let Q := RVar.d [% X, Y, W, Z].`). The first subgoal appears explicitly at line 4. `fin_img` is the (finite) image of a random variable. `QuadA23.d Q` is the distribution corresponding to the pairs of random variables `[% X, [% Y, W], Z]`. Solving the first subgoal is essentially a matter of reasoning by cases (see line 5). The second subgoal appears explicitly at line 7. It is essentially solved by using the hypothesis of conditional independence at line 11. Line 13 is where we use the distributivity of multiplication w.r.t. addition. Line 15 is the conclusive reasoning by cases. The lines that we have not explained in details are just low-level manipulations of distributions.

The proofs of weak union, contraction, and intersection are a bit more involved. For example, weak union uses decomposition and intersection uses contraction. See Appendix A for more details.

4.4 Application of the Graphoid Axioms

Now that we have proved the graphoid axioms, we can derive most rules to reason formally about probabilistic models. For example, the *chaining rule* uses all the *semi-graphoid axioms* (i.e., the graphoid axioms except intersection):

```
Lemma chaining_rule : X _|_ Z | Y /\ [% X, Y] _|_ W | Z -> X _|_ W | Y.
Proof.
case=> ? ?.
suff : X _|_ [% W, Z] | Y by move/decomposition.
apply/cinde_drv_2C/contraction => //.
exact/cinde_drv_3C/symmetry/weak_union/symmetry.
Qed.
```

See also the *mixing rule* for another example of derived rule [AHS+18].

5 Related Work

The theory of conditional probabilities (Bayes' theorems, etc.) is developed in HOL and applied to the analysis of the binary asymmetric channel [HT11], but there does not seem to be any theory of joint distributions in this work. We were able to reproduce the same theory of conditional probabilities as [HT11] in the setting of Sect. 2.3, except for lemmas involving countable unions which are not handled by the theory of finite sets of MathComp. Information-theoretic notions such as entropy, relative entropy, and mutual information are defined in HOL using a formalization of probability based on measure theory and Lebesgue integration [MHT11], but there does not seem to be much lemmas about them. There are more information-theoretic notions (e.g., conditional mutual information information) defined Isabelle/HOL using Lebesgue integration [Höl12]. Compared to the work above, our work features much more lemmas (all the chain rules, lemmas about conditional independence, etc.).

Our work uses a Coq library [AHS+18] that comes with a formalization of distributions and probabilities (including basic lemmas such as the weak law of large numbers) and applications to information theory. This work already features definitions of conditional entropy and mutual information but specialized for a setting with one input distribution P and one stochastic matrix W, which models a channel of communication. In this setting, one starts by defining a joint distribution $J(P, W)$ and then uses this distribution to define conditional entropy $H(W|P)$ and mutual information $I(P, W)$ (these notations departs from [CT06] but are not uncommon, see, e.g., [CK11]). We can prove that each entry of W corresponds to a conditional probability using $J(P, W)$:

```
Variables (A B : finType) (W : `Ch_1(A, B)) (P : {dist A}).
Let QP := Swap.d (`J(P, W)).
Lemma WcPr : forall a b, P a != 0 -> W a b = \Pr_QP[[set b] | [set a]].
```

It follows that $H(W|P)$ is equal to `CondEntropy.h QP` and $I(P, W)$ is equal to `MutualInfo.mi QP`. Similarly, in [AG15], we defined *aposteriori probabilities* using P and W:

$$P^W(x|y) \stackrel{\text{def}}{=} \frac{P(x)W^n(y|x)}{\sum_{x' \in A^n} P(x')W^n(y|x')}.$$

We can now show that $P^W(x|y)$ is equal to `\Pr_(`J(P, W ``^ n))[[set x]|[set y]]` where `` `J(P, W ``^ n) `` is the joint distribution of the n^{th} extension of the channel W. See [AHS+18, file `chap2.v`] for details. In other words, the work we present in this paper simplifies and generalizes our previous work.

There is another formalization of information theory that uses a similar setting [Mor12]. Distributions are defined similarly, which is not surprising because it is natural to use the big operators of MathComp in this way. The definition of random variable in Sect. 4.1 is the same definition as [Mor12]. Yet, that work stops at an early stage with the chain rule for entropy restricted to three variables and the definition of mutual information. Another limitation of this formalization is that it axiomatizes the logarithm, whereas InfoTheo is compatible with the logarithm defined in the standard library of Coq.

Our work connects concretely to the semantics of probabilistic programming languages. We are currently using the InfoTheo library to provide a formal model for a monad combining probability and nondeterminism [AGNS18]. It happens that such a model requires the notion of convex spaces that we have developed in Sect. 3.3.

Regarding conditional independence, one can find a formal definition of the graphoid axioms in Coq/SSReflect [YKS+16] but this work does not seem to provide the underlying formalization of probabilities; instead, it focuses on a high-level algebraic presentation and its application.

File	Contents	Sections in this paper
InfoTheo *files that required extensions for this paper*		
`proba.v`	Distributions and probabilities	Sect. 2.1
	Joint distributions	Sect. 2.2
New files that come with this paper		
`cproba.v`	Conditional probability	Sect. 2.3
`chap2.v`	Information theory	Sections 3.1, 3.2, and 3.4
`convex.v` `convex_dist.v`	Convexity properties	Sect. 3.3
`cinde.v`	Conditional independence	Sect. 4 and Appendix A

Table 1. Overview of the formalization in this paper

The formalization of conditional independence can be tackled in a different way using *string diagrams* [CJ18, Proposition 6.10].

6 Conclusion

In this paper, we proposed a formalization of conditional probabilities and joint distributions validated by two original applications: a formalization of the foundations of information theory and a formalization of conditional independence. Our formalization of information theory extends previous work with lemmas that were not formalized before (e.g., the chain rule for information, convexity of entropy). Our second application is (to the best of our knowledge) the first formal account *from the ground up* of the graphoid axioms of conditional independence. The complete formalization is available online as a conservative extension of the InfoTheo library [AHS+18] (Table 1 summarizes the relevant files).

Using our work, it is now possible to address new challenges in formalization of information theory and artificial intelligence. Next, we plan to tackle the formalization of Fano's inequality, the formalization of Bayesian networks, and more generally formal reasoning about probabilistic graphical models.

Acknowledgments The authors are thankful to Kenta Cho, Shin-ya Katsumata, and Keisuke Nakano for fruitful discussions about the work presented in this paper, to Paul Fournier for his proofreading, and to the anonymous reviewers for their helpful comments and for suggesting the application to Han's inequality. The authors acknowledge partial support from a Grant-in-Aid for Scientific Research (B) (project number 18H03204).

References

[ACR18] Reynald Affeldt, Cyril Cohen, and Damien Rouhling. Formalization techniques for asymptotic reasoning in classical analysis. *Journal of Formalized Reasoning*, 11(1):43–76, 2018.

[AG15] Reynald Affeldt and Jacques Garrigue. Formalization of error-correcting codes: from Hamming to modern coding theory. In *6th Conference on Interactive Theorem Proving (ITP 2015), Nanjing, China, August 24–27, 2015*, volume 9236 of *Lecture Notes in Computer Science*, pages 17–33. Springer, Aug 2015.

[AGNS18] Reynald Affeldt, Jacques Garrigue, David Nowak, and Takafumi Saikawa. Monadic equational reasoning in Coq. `https://github.com/affeldt-aist/monae`, 2018. Draft paper: [AN18].

[AGS18] Reynald Affeldt, Jacques Garrigue, and Takafumi Saikawa. Examples of formal proofs about data compression. In *International Symposium on Information Theory and Its Applications*

(ISITA 2018), Singapore, October 28–31, 2018, pages 665–669. IEICE. IEEE Xplore, Oct 2018.

[AHS14] Reynald Affeldt, Manabu Hagiwara, and Jonas Sénizergues. Formalization of Shannon's theorems. *Journal of Automated Reasoning*, 53(1):63–103, 2014.

[AHS+18] Reynald Affeldt, Manabu Hagiwara, Jonas Sénizergues, Jacques Garrigue, Kazuhiko Sakaguchi, Taku Asai, Takafumi Saikawa, and Naruomi Obata. A Coq formalization of information theory and linear error-correcting codes. `https://github.com/affeldt-aist/infotheo`, 2018.

[AN18] Reynald Affeldt and David Nowak. Experimenting with monadic equational reasoning in Coq. In *the proceedings of the 35th Meeting of the Japan Society for Software Science and Technology (JSSST 2018), Osaka-fu, Suita-shi, August 29–31, 2018*, Aug 2018. 14 pages. Updated version: `https://staff.aist.go.jp/reynald.affeldt/documents/monae.pdf`.

[CJ18] Kenta Cho and Bart Jacobs. Disintegration and bayesian inversion via string diagrams. Version 2. Available at `https://arxiv.org/abs/1709.00322`, 2018.

[CK11] Imre Csiszár and János Körner. *Information Theory: Coding Theorems for Discrete Memoryless Systems*. Cambridge University Press, 2011. Second Edition.

[CT06] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley, 2006. Second edition.

[GMT08] Georges Gonthier, Assia Mahboubi, and Enrico Tassi. A small scale reflection extension for the Coq system. Technical report, INRIA, 2008. Version 17 (Nov 2016).

[Höl12] Johannes Hölzl. *Construction and Stochastic Applications of Measure Spaces in Higher-Order Logic*. PhD thesis, Technische Universität München Institut für Informatik, 2012.

[HT11] Osman Hasan and Sofiène Tahar. Reasoning about conditional probabilities in a higher-order-logic theorem prover. *J. Applied Logic*, 9(1):23–40, 2011.

[KF09] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[MHT11] Tarek Mhamdi, Osman Hasan, and Sofiène Tahar. Formalization of entropy measures in HOL. In *Interactive Theorem Proving - Second International Conference, ITP 2011, Berg en Dal, The Netherlands, August 22-25, 2011. Proceedings*, volume 6898 of *Lecture Notes in Computer Science*, pages 233–248. Springer, 2011.

[Mor12] Diogo Arajo Carvalho Vilaa Moreira. Finite probability distributions in Coq. Master's thesis, Universidade do Minho, Escola de Engenharia, 01 2012. `http://mei.di.uminho.pt/sites/default/files/dissertacoes//eeum_di_dissertacao_pg16019.pdf`.

[MT16] Assia Mahboubi and Enrico Tassi. *Mathematical Components*. Available at: `https://math-comp.github.io/mcb/` (last access: 2018/08/20), 2016. With contributions by Yves Bertot and Georges Gonthier. Version of 2018/08/11.

[PP85] Judea Pearl and Azaria Paz. GRAPHOIDS: A graph-based logic for reasoning about relevance relations. Technical Report R-53 850038, UCLA Computer Science Department, Dec 1985.

[YKS+16] Rutaro Yamaguchi, Ken Kin, Shugo Shimoyama, Manabu Hagiwara, Mitsuharu Yamamoto, and Jinfang Wang. Formalization of the conditional independence using Coq/SSReflect. Technical Report Technical Reports of Mathematical Sciences, Chiba University, 29:1–40, Department of Mathematics and Informatics Graduate School of Science, Chiba University, Japan, 2016. In Japanese.

A Weak Union, Contraction, and Intersection

This appendix provides pencil-and-paper proofs for the weak union, contraction, and intersection properties of conditional independence that follow the formal proofs in `cinde.v` [AHS+18]. Note that what we present here is only the tip of the iceberg. Indeed, our formalization starts from the definition of finite discrete probability theory on top of finite types and real numbers, rather than a direct axiomatization of conditional independence. This requires proving many technical lemmas, used internally in these proofs. Moreover, the pencil-and-paper version of the proof hushes many administrative details of the transformations, which need to be spelled fully in the formal version.

Lemma 3 (Weak Union)**.** *$X \perp Y, W \mid Z$ implies $X \perp Y \mid Z, W$.*

Proof.

$$\begin{aligned} P(X,Y|Z,W) &= P(X|Y,Z,W)P(Y|Z,W) && \text{(by the product rule)} \\ &= P(X|Z)P(Y|Z,W) && \text{(because } X \perp Y,W \mid Z\text{)} \\ &= P(X|Z,W)P(Y|Z,W) \end{aligned}$$

(because $X \perp W \mid Z$ by decomposition (Lemma 2) from $X \perp Y, W \mid Z$)

□

Lemma 4 (Contraction)**.** *$X \perp W \mid Z, Y$ and $X \perp Y \mid Z$ imply $X \perp Y, W \mid Z$.*

Proof.

$$\begin{aligned} P(X,Y,W|Z) &= P(X|Y,W,Z)P(Y,W|Z) && \text{(by the product rule)} \\ &= P(X|Y,Z)P(Y,W|Z) && \text{(because } X \perp W \mid Z,Y\text{)} \\ &= P(X|Z)P(Y,W|Z) && \text{(because } X \perp Y \mid Z\text{)} \end{aligned}$$

□

Lemma 5 (Intersection)**.** *$X \perp Y \mid Z, W$ (1) and $X \perp W \mid Z, Y$ (2) imply $X \perp Y, W \mid Z$ provided that the distribution Y, Z, W is positive (3) and the type of the codomain of W is not empty (4).*

Proof. It suffices to show $X \perp Y \mid Z$ by contraction (Lemma 4) because we already have (2).

The goal is therefore $P(X,Y|Z) = P(X|Z)P(Y|Z)$ which is proved as follows:

$$\begin{aligned} P(X|Z)P(Y|Z) &= \sum_w P(X,w,Z)P(Y|Z) && \text{(reasoning by cases)} \\ &= \sum_w P(X,Y|Z)P(w|Z) && \text{(see (*) below)} \\ &= P(X,Y|Z)\sum_w P(w|Z) && \text{(by distributivity)} \\ &= P(X,Y|Z) && \text{(conditional probability of the universal set and (4))} \end{aligned}$$

Proof of step (*): By (2), (3), and the product rule, we have $P(X|Y,Z) = P(X|W,Z,Y)$.

By (1), (3), and the product rule, we have $P(X|W,Z) = P(X|Y,W,Z)$.

Therefore we have $P(X|Y,Z) = P(X|W,Z)$.

By (3) and the product rule, we derive $\frac{P(X,Y|Z)}{P(Y|Z)} = \frac{P(X,W|Z)}{P(W|Z)}$, which implies $P(X,Y|Z)P(W|Z) = P(X,W|Z)P(Y|Z)$ and $\sum_w P(X,Y|Z)P(w|Z) = \sum_w P(X,w|Z)P(Y|Z)$.

□