

ALEA: a library for reasoning on randomized algorithms in COQ

Version 5

Christine Paulin-Mohring
with contributions by Pierre Courtieu
PROVAL Team

LRI, UMR 8623 Univ. Paris-Sud 11, CNRS & INRIA Saclay - Île-de-France

November 24, 2010

1 Introduction

This library forms a basis for reasoning on randomised algorithms in the proof assistant COQ [6]. The source files are available from the (author homepage). They can be compiled using Coq version V8.3.

The theoretical basis of this work is a joint work with Philippe Audebaud and is described in [1, 2]. It is developed in the framework of the SCALP project (ANR-07-SESU-010) on Security of Cryptographic ALgorithms with Probabilities.

As proposed by Kozen [3, 4], we interpret probabilistic programs as measures; the originality of our approach is to view this interpretation as a monadic transformation on functional programs. Using this semantics, we derive general rules for estimating the probability for a randomised algorithm to satisfy a given property. We apply this approach to the formal proof in COQ of properties of randomised algorithms. The repository contains simple examples: a program implementing a Bernoulli distribution using a coin flip as a primitive, the construction of the binomial distribution. We prove the probabilistic termination of a linear random walk. We also prove a general scheme for combining 2 different randomized expressions in order to get a better result. We also study an example of partial termination. The library ALEA has been used in larger developments, mainly the CertiCrypt environment for formal proofs for computational cryptography based on a game representation.

With respect to previous versions of this library, we use the mechanism of type classes introduced by N. Oury and M. Sozeau [5] in order to represent the order structure and the cpo structure on a given type. In this version, P. Courtieu started to design simple automatic tactics for reasoning on measures, making an essential use of the possibility to rewrite under abstraction.

2 Contents of the library

The library is composed of the following files :

Misc A few preliminary definitions, in particular primitives for reasoning classically in Coq are defined.

Cpo The definition of structures for ordered sets and ω -complete partial orders (a monotonic sequence has a least-upper bound). We define the type $O_1 \xrightarrow{m} O_2$ of monotonic functions and define the fixpoint-construction for monotonic functionals. Continuity is also defined.

Utheory An axiomatisation of the interval $[0, 1]$. The type $[0, 1]$ is given a cpo structure: we have the predicates \leq and $==$, a least element 0 and a least-upper bound on all monotonic sequences of elements of $[0, 1]$.

The primitive operations are:

Operation	Meaning	CoqNotation
bounded addition :	$(x, y) \mapsto \min(x + y, 1)$	$x + y$
multiplication :	$(x, y) \mapsto x \times y$	$x * y$
division:	$(x, y) \mapsto \min(x/y, 1)$	x / y
inverse function:	$x \mapsto 1 - x$	$[1-]x$
constants:	$(n \in \mathbb{N}) \mapsto \frac{1}{1+n}$	$[1/1+]n$

Uprop Derived operations and properties of operators on $[0, 1]$. The main defined operations are:

Operation	Meaning	CoqNotation	ALEAdefinition
bounded subtraction:	$(x, y) \mapsto \max(x - y, 0)$	$x - y$	$[1-]([1-]x + y)$
maximum:	$(x, y) \mapsto \max(x, y)$	$\mathbf{max} \ x \ y$	$(x - y) + y$
minimum:	$(x, y) \mapsto \min(x, y)$	$\mathbf{min} \ x \ y$	$[1-]((y - x) + [1-]y)$
addition dual:	$(x, y) \mapsto \max(x + y - 1, 0)$	$x \& y$	$[1-]([1-]x + [1-]y)$
difference:	$(x, y) \mapsto x - y $	$\mathbf{diff} \ x \ y$	$(x - y) + (y - x)$
mean:	$(x, y) \mapsto \frac{1}{2}x + \frac{1}{2}y$	$\mathbf{mean} \ x \ y$	$[1/2]x + [1/2]y$
multiple:	$(n \in \mathbb{N}, x) \mapsto nx$	$n * / x$	induction on n
power:	$(n \in \mathbb{N}, x) \mapsto x^n$	$x \hat{=} n$	induction on n
generalized sum:	$(f \in \mathbb{N} \rightarrow [0, 1], n \in \mathbb{N}) \mapsto \sum_{i=0}^{i=n-1} f(i)$	$\mathbf{sigma} \ f \ n$	induction on n

This file defines a tactic `Usimpl` designed to help simplify expressions using operations on $[0, 1]$.

Monads Definition of the basic monad for randomized constructions, the type α is mapped to the type $(\alpha \rightarrow [0, 1]) \xrightarrow{m} [0, 1]$ of measure functions. We define the `unit` and `star` constructions and prove that they satisfy the basic monadic properties. A measure will be a function of type $(\alpha \rightarrow [0, 1]) \xrightarrow{m} [0, 1]$ which enjoys extra properties such as stability with respect to basic operations, continuity. We prove that functions produced by `unit` and `star` satisfy these extra properties under appropriate assumptions.

Probas Definition of a dependent type for distributions on a type α . A distribution on a type α is a record containing a function μ of type $(\alpha \rightarrow [0, 1]) \xrightarrow{m} [0, 1]$ and proofs that this function enjoys the stability properties of measures. These properties are :

$$\begin{aligned}
\mu(f_1 + f_2) &= \mu(f_1) + \mu(f_2) && \text{when } f_1 \leq 1 - f_2 \\
\mu(k \times f) &= k \times \mu(f) \\
\mu(1 - f) &\leq 1 - \mu(f) \\
\mu(\text{lub}_n(f \ n)) &\leq \text{lub}_n(\mu(f \ n)) && \text{when } f \text{ is a monotonic sequence of functions}
\end{aligned}$$

We define the interpretation of specific random primitives : the distribution corresponding to a coin flip and the distribution corresponding to the random function which applied to n gives a number between 0 and n with probability $\frac{1}{n+1}$.

SProbas a relaxed notion of distributions called sub-distributions suitable to model programs using both non-deterministic and random choice. A sub-distribution is not additive but instead satisfies the following weaker properties:

$$\begin{aligned}
\mu(f_1) + \mu(f_2) &\leq \mu(f_1 + f_2) && \text{when } f_1 \leq 1 - f_2 \\
\mu(f) \& \mu(g) &\leq \mu(f \& g) \\
\mu(f + c) &\leq \mu(f) + c
\end{aligned}$$

Prog Definition of randomized programs constructions. We define the probabilistic choice and conditional constructions and a fixpoint operator obtained by iterating a monotonic functional. We introduce an

axiomatic semantics for these randomized programs : let e be a randomized expression of type τ , p be an element of $[0, 1]$ and q be a function of type $\tau \rightarrow [0, 1]$, we define $p \leq [e](q)$ to be the property : the measure of q by the distribution associated to the expression e is not less than p . In the case q is the characteristic function of a predicate Q , $p \leq [e](q)$ can be interpreted as “the probability for the result of the evaluation of e to satisfy Q is not less than p ”. In the particular case where q is the constant function equal to 1, the relation $p \leq [e](q)$ can be interpreted as “the probability for the evaluation of e to terminate is not less than p ”.

We derive inference rules for this relation.

Cover A definition of what it means for a function f to be the characteristic function of a predicate P . Defines also characteristic functions for decidable predicates. This file contains a proof of the principle :

$$\frac{1 \leq [a](\mathbb{I}_P) \quad \forall x, (P x) \Rightarrow k \leq [b](f)}{k \leq [\mathbf{let } x = a \mathbf{ in } b](f)}$$

This file uses the library `Sets.v` which defines sets as predicates and finite sets with an inductive definition.

Bernoulli Construction of a bernoulli distribution from the flip distribution. We consider the program

```
let rec bernoulli p = if flip () then
  if x < 1/2 then false else bernoulli (2 p-1)
  else if x < 1/2 then bernoulli (2 p) else true
```

We prove that the probability of `bernoulli(p)` to answer `true` is exactly p . We use this distribution in order to simulate a binomial distribution such that $\Pr(\mathbf{binomial } p \ n = k) = C_k^n p^k (1-p)^{n-k}$.

DistrTactic This file introduces a tactic `rsimplmu` which attempts to simplify expressions involving randomized programs.

3 Examples

IterFlip A proof of probabilistic termination for a random walk. We consider the program

```
let rec iter x = if flip() then iter (x+1) else x
```

We prove that the probability that this program terminates is 1.

Choice A proof of composition of two runs of a probabilistic program, when a choice can improve the quality of the result. Given two randomized expressions p_1 and p_2 of type τ and a function Q to be estimated, we consider a `choice` function such that the value of Q for `choice(x,y)` is not less than $Q(x) + Q(y)$. We prove that if p_i evaluates Q not less than k_i and terminates with probability 1 then the expression `choice(p1,p2)` evaluates Q not less than $k_1(1 - k_2) + k_2$ (which is greater than both k_1 and k_2 when k_1 and k_2 are not equal to 0).

Ycart Evaluation of probability of termination for a program adapted from an example due to B. Ycart, parameterized by a function K of type $\mathbf{nat} \rightarrow \mathbf{nat}$ and a natural number N .

```
let rec ycart n = if random N < K n then n else ycart (n+1)
```

Probability of termination of `(ycart n)` is shown to be equal to $\prod_{k=n}^{\infty} (1 - Fk)$, with $F n = K n / (1 + N)$.

References

- [1] Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. In Tarmo Uustalu, editor, *Mathematics of Program Construction, MPC'2006*, volume 4014 of *LNCS*, Kuressaare, Estonia, 2006. Springer-Verlag.
- [2] Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. *Science of Computer Programming*, 74(8):568–589, 2009.
- [3] Dexter Kozen. Semantics of probabilistic programs. *J. of Computer and System Science*, 1981.
- [4] Dexter Kozen. A probabilistic PDL. In *15th ACM Symposium on Theory of Computing*, 1983.
- [5] Matthieu Sozeau and Nicolas Oury. First-Class Type Classes. In Sofiène Tahar, Otmane Ait-Mohamed, and César Muñoz, editors, *21th International Conference on Theorem Proving in Higher Order Logics*, LNCS. Springer-Verlag, August 2008.
- [6] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V8.2*, February 2009. .