A library for reasoning on randomized algorithms in Coq

Christine Paulin-Mohring

February 16, 2007

Contents

1 Introduction

This library forms a basis for reasoning on randomised algorithms in the proof assistant Coq [6]. The source files are available as a Coq contribution (see http://coq.inria.fr).

The theoretical basis of this work is a joint work with Philippe Audebaud and is described in [1, 2].

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. We study the example of a program implementing a Bernoulli distribution using a coin flip as a primitive. We prove the probabilistic termination of a linear random walk. We also extend this approach in order to measure probability of traces in a probabilistic transition system.

The library is composed of the following files :

Ubase An axiomatisation of the interval $[0,1]$. The primitive operations are bounded addition $(x,y) \mapsto \min(x+y,1)$, multiplication $(x,y) \mapsto x \times y$ and an inverse function $x \mapsto 1-x$ as well as a function which associates $\frac{1}{n+1}$ to each integer n. We also introduce the predicates $\leq$ and $=$ and a least-upper bound on all sequences of elements of $[0,1]$.

Uprop Derived operations and properties of operators on $[0,1]$. We define the operations max, a bounded difference $(x,y) \mapsto \max(x-y,0)$, the special operator $x\&y$ defined as $\max(x+y-1,0)$, the functions $(n,x) \mapsto x^n$, $(n,x) \mapsto nx$, with n an integer, the function $(f,n) \mapsto \Sigma_{i=0}^{i=n-1} f(i)$, the mean of two points $(x,y) \mapsto \frac{1}{2}x + \frac{1}{2}y$.

Monads Definition of the basic monad for randomized constructions, the type α is mapped to the type $(\alpha \to [0,1]) \to [0,1]$ 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 $(\alpha \to [0,1]) \to [0,1]$ that enjoys extra properties such as monotonicity, stability with respect to basic operations. We prove that functions produced by `unit` and `star` satisfy these extra properties under apropriate 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 $(\alpha \to [0,1]) \to [0,1]$ and proofs that this function enjoys the stability properties of measures. These properties are :

$$\begin{array}{rll} \mu(f_1+f_2) & =\mu(f_1)+\mu(f_2) & \text{when } f_1 \leq 1-f_2 \\ \mu(k\times f) & = k\times\mu(f) & \\ \mu(1-f) & \leq 1-\mu(f) & \\ \mu(f) & \leq \mu(g) & \text{when } f\leq g \quad (i.e.\ \forall x. f(x)\leq g(x)) \end{array}$$

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 $\frac{1}{n+1}$.

Prog Definition of randomized programs constructions. We define the conditional construction 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 $\tau\to[0,1]$, we define $p\leq\langle e\rangle(q)$ 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, $p\leq\langle e\rangle(q)$ 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 $p\leq\langle e\rangle(q)$ 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.

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`(p_1,p_2) 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).

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 `bernouilli`(p) to answer `true` is exactly p. We use this distribution in order to simulate a binomial distribution such that $\mathsf{Pr}((\texttt{binomial}\ p\ n)=k)=C_k^n p^k(1-p)^{n-k}$.

Carac A definition of characteristic functions for decidable predicates. This file contains also the proof of the principle :

$$\frac{1\leq\langle a\rangle(\mathbb{I}_P) \qquad \forall x,(P\ x)\Rightarrow k\leq\langle b\rangle(f)}{k\leq\langle \textsf{let}\ x=a\ \textsf{in}\ b\rangle(f)}$$

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

Ycart Evaluation of probability of termination for a program due to B. Ycart, parameterized by a function F of type `nat` $\to[0,1]$.

```
let rec ycart n = if uniform () <= F n then n else ycart (n+1)
```

Probability of termination of (`ycart` n) is shown to be equal to $\prod_{k=n}^{\infty}(1-Fk)$.

This file also contains an axiomatisation of a uniform measure.

Libwp A definition of partial correctness for programs. This file contains various theorems for proving probabilistic termination of programs.

Transitions A probabilistic transition system is defined by a set of states and a probabilistic transition function which associates to a state a the probability to go to a state b. In our system it corresponds to a function from states to distribution on states. We use this function in order to define the corresponding distribution on paths of length k for a given integer k. This library uses a module `Nelist` defining non empty lists.

Contents

2 Preliminaries

2.1 Definition of iterator *comp*

comp f u n x is defined as $(f\ (u\ (n-1))..(f(u\ 0)\ x))$

Fixpoint $comp\ (A{:}Type)\ (f : A \to A \to A)\ (x : A)\ (u : nat \to A)\ (n{:}nat)\ \{struct\ n\}{:}\ A :=$
$match\ n\ with\ O \Rightarrow x|\ (S\ p) \Rightarrow f\ (u\ p)\ (comp\ f\ x\ u\ p)\ end.$

Lemma $comp0 : \forall\ (A{:}Type)\ (f : A \to A \to A)\ (x : A)\ (u : nat \to A),\ comp\ f\ x\ u\ 0 = x.$

Lemma $compS : \forall\ (A{:}Type)\ (f : A \to A \to A)\ (x : A)\ (u : nat \to A)\ (n{:}nat),$
$comp\ f\ x\ u\ (S\ n) = f\ (u\ n)\ (comp\ f\ x\ u\ n).$

2.2 Monotonicity of sequences for an arbitrary relation

Definition $mon\_seq\ (A{:}Type)\ (le : A \to A \to Prop)\ (f{:}nat \to A)$
$:= \forall\ n\ m,\ (n \le m) \to (le\ (f\ n)\ (f\ m)).$

Definition $decr\_seq\ (A{:}Type)\ (le : A \to A \to Prop)\ (f{:}nat \to A)$
$:= \forall\ n\ m,\ (n \le m) \to (le\ (f\ m)\ (f\ n)).$

2.3 Reducing if constructs

Lemma $if\_then : \forall\ (P{:}Prop)\ (b{:}\{P\}+\{\neg P\})(A{:}Type)(p\ q{:}A),\ P \to (if\ b\ then\ p\ else\ q) = p.$

Lemma $if\_else : \forall\ (P{:}Prop)\ (b{:}\{P\}+\{\neg P\})(A{:}Type)(p\ q{:}A),\ \neg P \to (if\ b\ then\ p\ else\ q) = q.$

2.4 Classical reasoning

Definition $class\ (A{:}Prop) := \neg\neg A \to A.$

Lemma $class\_neg : \forall\ A{:}Prop,\ class\ (\neg A).$

Lemma $class\_false : class\ False.$
Hint $Resolve\ class\_neg\ class\_false.$

Definition $orc\ (A\ B{:}Prop) := \forall\ C{:}Prop,\ class\ C \to (A \to C) \to (B \to C) \to C.$

Lemma $orc\_left : \forall\ A\ B{:}Prop,\ A \to orc\ A\ B.$

Lemma $orc\_right : \forall\ A\ B{:}Prop,\ B \to orc\ A\ B.$

Hint $Resolve\ orc\_left\ orc\_right.$

Lemma $class\_orc : \forall\ A\ B,\ class\ (orc\ A\ B).$

Implicit $Arguments\ class\_orc\ [].$

Lemma $orc\_intro : \forall\ A\ B,\ (\neg A \to \neg B \to False) \to orc\ A\ B.$

Lemma $class\_and : \forall\ A\ B,\ class\ A \to class\ B \to class\ (A \wedge B).$

Lemma $excluded\_middle : \forall\ A,\ orc\ A\ (\neg A).$

Definition $exc\ (A : Type)(P{:}A \to Prop) :=$
$\forall\ C{:}Prop,\ class\ C \to (\forall\ x{:}A,\ P\ x \to C) \to C.$

Lemma $exc\_intro : \forall\ (A : Type)(P{:}A \to Prop)\ (x{:}A),\ P\ x \to exc\ P.$

Lemma $class\_exc : \forall\ (A : Type)(P{:}A \to Prop),\ class\ (exc\ P).$

Lemma $exc\_intro\_class : \forall\ (A{:}Type)\ (P{:}A \to Prop),\ ((\forall\ x,\ \neg P\ x) \to False) \to exc\ P.$

Lemma $not\_and\_elim\_left : \forall\ A\ B,\ \neg\ (A \wedge B) \to A \to \neg B.$

Lemma $not\_and\_elim\_right : \forall\ A\ B,\ \neg\ (A \wedge B) \to B \to \neg A.$

Hint $Resolve\ class\_orc\ class\_and\ class\_exc\ excluded\_middle.$

3 Ubase.v: Specification of U, interval $[0,1]$

Require Export *Setoid*.
Require Export *Prelude*.

3.1 Basic operators of U

- Constants : 0 and 1
- Constructor : *Unth* $n(\equiv \frac{1}{n+1})$
- Operations : $x + y$ $(\equiv min(x+y,1))$, $x * y$, *inv* x $(\equiv 1-x)$
- Relations : $x \leq y$, $x == y$

Module *Type Universe*.
Parameter U : *Type*.
Delimit Scope U_scope with U.

Parameters *Ueq Ule* : $U \to U \to$ *Prop*.
Parameters *U0 U1* : U.
Parameters *Uplus Umult* : $U \to U \to U$.
Parameter *Uinv* : $U \to U$.
Parameter *Unth* : *nat* $\to U$.

Infix "$+$" := *Uplus* : *U_scope*.
Infix "$\times$" := *Umult* : *U_scope*.
Infix "$==$" := *Ueq* (*at level* 70) : *U_scope*.
Infix "$\leq$" := *Ule* : *U_scope*.
Notation "[1-] x" := (*Uinv* x) (*at level* 35, *right associativity*) : *U_scope*.
Notation "0" := *U0* : *U_scope*.
Notation "1" := *U1* : *U_scope*.
Notation "[1/]1+ n" := (*Unth* n) (*at level* 35, *right associativity*) : *U_scope*.
Open Local *Scope U_scope*.

3.2 Basic Properties

Hypothesis *Ueq_refl* : $\forall\, x{:}U,\ x == x$.
Hypothesis *Ueq_sym* : $\forall\, x\ y{:}U,\ x == y \to y == x$.

Hypothesis *Udiff_0_1* : $\neg 0 == 1$.
Hypothesis *Upos* : $\forall\, x{:}U,\ 0 \leq x$.
Hypothesis *Unit* : $\forall\, x{:}U,\ x \leq 1$.

Hypothesis *Uplus_sym* : $\forall\, x\ y{:}U,\ x + y == y + x$.
Hypothesis *Uplus_assoc* : $\forall\, x\ y\ z{:}U,\ x + (y + z) == x + y + z$.
Hypothesis *Uplus_zero_left* : $\forall\, x{:}U,\ 0 + x == x$.

Hypothesis *Umult_sym* : $\forall\, x\ y{:}U,\ x \times y == y \times x$.
Hypothesis *Umult_assoc* : $\forall\, x\ y\ z{:}U,\ x \times (y \times z) == x \times y \times z$.
Hypothesis *Umult_one_left* : $\forall\, x{:}U,\ 1 \times x == x$.

Hypothesis *Uinv_one* : [1-] $1 == 0$.
Hypothesis *Uinv_opp_left* : $\forall\, x$, [1-] $x + x == 1$.

Property : $1-(x+y)+x = 1-y$ holds when $x+y$ does not overflow
Hypothesis *Uinv_plus_left* : $\forall\, x\ y,\ y \leq$ [1-] $x \to$ [1-] $(x + y) + x ==$ [1-] y.

Property : $(x+y)\times z = x \times z + y \times z$ holds when $x+y$ does not overflow
Hypothesis *Udistr_plus_right* : $\forall\, x\ y\ z,\ x \leq$ [1-] $y \to (x + y) \times z == x \times z + y \times z$.

Property : $1-(x \times y) = (1-x)\times y + (1-y)$
Hypothesis *Udistr_inv_right* : $\forall\, x\ y{:}U$, [1-] $(x \times y) ==$ ([1-] $x) \times y +$ [1-] y.

The relation $x \leq y$ is reflexive, transitive and anti-symmetric
Hypothesis *Ueq_le* : ∀ *x y*:*U*, *x* == *y* → *x* ≤ *y*.

Hypothesis *Ule_trans* : ∀ *x y z*:*U*, *x* ≤ *y* → *y* ≤ *z* → *x* ≤ *z*.

Hypothesis *Ule_antisym* : ∀ *x y*:*U*, *x* ≤ *y* → *y* ≤ *x* → *x* == *y*.

Totality of the order
Hypothesis *Ule_class* : ∀ *x y* : *U*, *class* (*x* ≤ *y*).

Hypothesis *Ule_total* : ∀ *x y* : *U*, *orc* (*x*≤*y*) (*y*≤*x*).
Implicit *Arguments Ule_total* [].

The relation $x \leq y$ is compatible with operators
Hypothesis *Uplus_le_compat_left* : ∀ *x y z*:*U*, *x* ≤ *y* → *x* + *z* ≤ *y* + *z*.

Hypothesis *Umult_le_compat_left* : ∀ *x y z*:*U*, *x* ≤ *y* → *x* × *z* ≤ *y* × *z*.

Hypothesis *Uinv_le_compat* : ∀ *x y*:*U*, *x* ≤ *y* → [1-] *y* ≤ [1-] *x*.

Properties of simplification in case there is no overflow
Hypothesis *Uplus_le_simpl_right* : ∀ *x y z*, *z* ≤ [1-] *x* → *x* + *z* ≤ *y* + *z* → *x* ≤ *y*.

Hypothesis *Umult_le_simpl_left* : ∀ *x y z*: *U*, ¬0 == *z* → *z* × *x* ≤ *z* × *y* → *x* ≤ *y* .

Property *Unth* $\frac{1}{n+1} == 1 - n \times \frac{1}{n+1}$
Hypothesis *Unth_prop* : ∀ *n*, [1/]1+*n* == [1-](*comp Uplus* 0 (*fun k* ⇒ [1/]1+*n*) *n*).

Archimedian property
Hypothesis *archimedian* : ∀ *x*, ¬0 == *x* → *exc* (*fun n* ⇒ [1/]1+*n* ≤ *x*).

3.3 Least upper bound, corresponds to limit for increasing sequences

Variable *lub* : (*nat* → *U*) → *U*.

Hypothesis *le_lub* : ∀ (*f* : *nat* → *U*) (*n*:*nat*), *f n* ≤ *lub f*.
Hypothesis *lub_le* : ∀ (*f* : *nat* → *U*) (*x*:*U*), (∀ *n*, *f n* ≤ *x*) → *lub f* ≤ *x*.

Stability properties of lubs with respect to + and ×
Hypothesis *lub_eq_plus_cte_right* : ∀ (*f*:*nat*→*U*) (*k*:*U*), *lub* (*fun n* ⇒ (*f n*) + *k*) == (*lub f*) + *k*.

Hypothesis *lub_eq_mult* : ∀ (*k*:*U*) (*f*:*nat*→*U*), *lub* (*fun n* ⇒ *k* × (*f n*)) == *k* × *lub f*.

End *Universe*.

4 Uprop.v : Properties of operators on [0,1]

Require Export *Ubase*.
Require Export *Arith*.
Require Export *Omega*.
Module *Univ_prop* (*Univ*:*Universe*).
Import *Univ*.

Hint *Resolve Ueq_refl*.
Hint *Resolve Upos Unit Udiff_0_1 Unth_prop Ueq_le*.
Hint *Resolve Uplus_sym Uplus_assoc Umult_sym Umult_assoc*.
Hint *Resolve Uinv_one Uinv_opp_left Uinv_plus_left*.
Hint *Resolve Uplus_zero_left Umult_one_left Udistr_plus_right Udistr_inv_right*.
Hint *Resolve Uplus_le_compat_left Umult_le_compat_left Uinv_le_compat*.
Hint *Resolve lub_le le_lub lub_eq_mult lub_eq_plus_cte_right*.
Hint *Resolve Ule_total Ule_class*.
Hint Immediate *Ueq_sym Ule_antisym*.
Open Scope nat_scope.
Open Scope U_scope.

4.1 Direct consequences of axioms

Lemma *Ueq_class* : $\forall\ x\ y$, *class* $(x{=}{=}y)$.

Lemma *Ueq_double_neg* : $\forall\ x\ y : U, \neg \neg x == y \rightarrow x == y$.
Hint *Resolve Ueq_class*.
Hint Immediate *Ueq_double_neg*.

Lemma *Ule_orc* : $\forall\ x\ y$, *orc* $(x \leq y)$ (~ $x \leq y$).
Implicit *Arguments Ule_orc* [].

Lemma *Ueq_orc* : $\forall\ x\ y$, *orc* $(x{=}{=}y)$ (~ $x{=}{=}y$).
Implicit *Arguments Ueq_orc* [].

Lemma *Ule_0_1* : $0 \leq 1$.

Lemma *Ule_refl* : $\forall\ x{:}U, x \leq x$.
Hint *Resolve Ule_refl*.

Add Relation U Ule reflexivity proved by Ule_refl transitivity proved by Ule_trans as Ule_Relation.

4.2 Properties of == derived from properties of ≤

Lemma *Ueq_trans* : $\forall\ x\ y\ z{:}U,\ x == y \rightarrow y == z \rightarrow x == z$.
Hint *Resolve Ueq_trans*.

Lemma *Uplus_eq_compat_left* : $\forall\ x\ y\ z{:}U,\ x == y \rightarrow (x + z) == (y + z)$.

Hint *Resolve Uplus_eq_compat_left*.

Lemma *Uplus_eq_compat_right* : $\forall\ x\ y\ z{:}U,\ x == y \rightarrow (z + x) == (z + y)$.

Lemma *Umult_eq_compat_left* : $\forall\ x\ y\ z{:}U,\ x == y \rightarrow (x \times z) == (y \times z)$.
Hint *Resolve Umult_eq_compat_left*.

Lemma *Umult_eq_compat_right* : $\forall\ x\ y\ z{:}U,\ x == y \rightarrow (z \times x) == (z \times y)$.

Hint *Resolve Uplus_eq_compat_right Umult_eq_compat_right*.

Lemma *Uinv_opp_right* : $\forall\ x,\ x +$ [1-] $x == 1$.
Hint *Resolve Uinv_opp_right*.

4.3 *U* is a setoid

Lemma *Usetoid* : *Setoid_Theory U Ueq*.

Add Setoid U Ueq Usetoid as U_setoid.

Add Morphism Uplus with signature Ueq ==> Ueq ==> Ueq as Uplus_eq_compat.

Add Morphism Umult with signature Ueq ==> Ueq ==> Ueq as Umult_eq_compat.

Hint Immediate *Umult_eq_compat Uplus_eq_compat*.

Add Morphism Uinv with signature Ueq ==> Ueq as Uinv_eq_compat.

Add Morphism Ule with signature Ueq ==> Ueq ==> iff as Ule_eq_compat_iff.

Lemma *Ule_eq_compat* :
$\forall\ x1\ x2 : U,\ x1 == x2 \rightarrow \forall\ x3\ x4 : U,\ x3 == x4 \rightarrow x1 \leq x3 \rightarrow x2 \leq x4$.

4.4 Definition and properties of $x < y$

Definition *Ult* $(r1\ r2{:}U) : Prop := \neg\ (r2 \leq r1)$.

Infix "<" := *Ult* : *U_scope*.

Hint *Unfold Ult*.

Add Morphism Ult with signature Ueq ==> Ueq ==> iff as Ult_eq_compat_iff.

Lemma *Ult_eq_compat* :
$\forall\ x1\ x2 :\ U,\ x1 == x2 \rightarrow \forall\ x3\ x4 :\ U,\ x3 == x4 \rightarrow x1 < x3 \rightarrow x2 < x4$.

Lemma *Ult_class* : $\forall\ x\ y,\ class\ (x<y)$.
Hint *Resolve Ult_class*.

4.4.1 Properties of $x \leq y$

Lemma *Ule_zero_eq* : $\forall\ x,\ x \leq 0 \rightarrow x == 0$.

Lemma *Uge_one_eq* : $\forall\ x,\ 1 \leq x \rightarrow x == 1$.

Hint Immediate *Ule_zero_eq Uge_one_eq*.

4.4.2 Properties of $x < y$

Lemma *Ult_neq* : $\forall\ x\ y{:}U,\ x < y \rightarrow \neg x == y$.

Lemma *Ult_neq_rev* : $\forall\ x\ y{:}U,\ x < y \rightarrow \neg y == x$.

Lemma *Ult_trans* : $\forall\ x\ y\ z,\ x<y \rightarrow y<z \rightarrow x <z$.

Lemma *Ult_le* : $\forall\ x\ y{:}U,\ x < y \rightarrow x \leq y$.

Lemma *Ule_diff_lt* : $\forall\ x\ y :\ U,\ x \leq y \rightarrow \neg x{==}y \rightarrow x < y$.

Hint Immediate *Ult_neq Ult_neq_rev Ult_le*.
Hint *Resolve Ule_diff_lt*.

Lemma *Ult_neq_zero* : $\forall\ x,\ \neg 0 == x \rightarrow 0 < x$.

Hint *Resolve Ule_total Ult_neq_zero*.

4.5 Properties of + and $\times$

Lemma *Udistr_plus_left* : $\forall\ x\ y\ z,\ y \leq [1\text{-}]\ z \rightarrow (x \times (y + z)) == (x \times y + x \times z)$.

Lemma *Udistr_inv_left* : $\forall\ x\ y,\ [1\text{-}](x \times y) == (x \times ([1\text{-}]\ y)) + [1\text{-}]\ x$.

Hint *Resolve Uinv_eq_compat Udistr_plus_left Udistr_inv_left*.

Lemma *Uplus_perm2* : $\forall\ x\ y\ z{:}U,\ x + (y + z) == y + (x + z)$.

Lemma *Umult_perm2* : $\forall\ x\ y\ z{:}U,\ x \times (y \times z) == y \times (x \times z)$.

Lemma *Uplus_perm3* : $\forall\ x\ y\ z :\ U,\ (x + (y + z)) == z + (x + y)$.

Lemma *Umult_perm3* : $\forall\ x\ y\ z :\ U,\ (x \times (y \times z)) == z \times (x \times y)$.

Hint *Resolve Uplus_perm2 Umult_perm2 Uplus_perm3 Umult_perm3*.

Lemma *Uplus_le_compat_right* : $\forall\ x\ y\ z{:}U,\ (x \leq y) \rightarrow (z + x \leq z + y)$.

Hint *Resolve Uplus_le_compat_right*.

Lemma *Uplus_le_compat* : $\forall\ x\ y\ z\ t{:}U,\ x \leq y \rightarrow z \leq t \rightarrow (x + z \leq y + t)$.
Hint Immediate *Uplus_le_compat*.

Lemma *Uplus_zero_right* : $\forall\ x{:}U,\ x + 0 == x$.
Hint *Resolve Uplus_zero_right*.

Lemma *Uinv_zero* : [1-] 0 == 1.
Hint *Resolve Uinv_zero.*

Lemma *Uinv_inv* : $\forall$ x : U, [1-] [1-] x == x.
Hint *Resolve Uinv_inv.*

Lemma *Uinv_simpl* : $\forall$ x y : U, [1-] x == [1-] y $\rightarrow$ x == y.

Hint Immediate *Uinv_simpl.*

4.6 More properties on + and × and *Uinv*

Lemma *Umult_le_compat_right* : $\forall$ x y z: U, $x \leq y \rightarrow (z \times x) \leq (z \times y)$.

Hint *Resolve Umult_le_compat_right.*

Add Morphism Umult with signature Ule ++> Ule ++> Ule as Umult_le_compat.
Hint Immediate *Umult_le_compat.*

Lemma *Umult_one_right* : $\forall$ x:U, $(x \times 1)$ == x.
Hint *Resolve Umult_one_right.*

Lemma *Udistr_plus_left_le* : $\forall$ x y z : U, $x \times (y + z) \leq x \times y + x \times z$.

Lemma *Uplus_eq_simpl_right* :
$\forall$ x y z:U, $z \leq$ [1-] $x \rightarrow z \leq$ [1-] $y \rightarrow (x + z)$ == $(y + z) \rightarrow x$ == y.

Lemma *Ule_plus_right* : $\forall$ x y, $x \leq x + y$.

Lemma *Ule_plus_left* : $\forall$ x y, $y \leq x + y$.
Hint *Resolve Ule_plus_right Ule_plus_left.*

Lemma *Ule_mult_right* : $\forall$ x y, $x \times y \leq x$.

Lemma *Ule_mult_left* : $\forall$ x y, $x \times y \leq y$.
Hint *Resolve Ule_mult_right Ule_mult_left.*

Lemma *Uinv_le_perm_right* : $\forall$ x y:U, $x \leq$ [1-] $y \rightarrow y \leq$ [1-] x.
Hint *Resolve Uinv_le_perm_right.*

Lemma *Uinv_le_perm_left* : $\forall$ x y:U, [1-] $x \leq y \rightarrow$ [1-] $y \leq x$.
Hint *Resolve Uinv_le_perm_left.*

Lemma *Uinv_eq_perm_left* : $\forall$ x y:U, x == [1-] $y \rightarrow$ [1-] x == y.
Hint Immediate *Uinv_eq_perm_left.*

Lemma *Uinv_eq_perm_right* : $\forall$ x y:U, [1-] x == $y \rightarrow x$ == [1-] y.

Hint Immediate *Uinv_eq_perm_right.*

Lemma *Uinv_plus_right* : $\forall$ x y, $y \leq$ [1-] $x \rightarrow$ [1-] $(x + y) + y$ == [1-] x.
Hint *Resolve Uinv_plus_right.*

Lemma *Uplus_eq_simpl_left* :
$\forall$ x y z:U, $x \leq$ [1-] $y \rightarrow x \leq$ [1-] $z \rightarrow (x + y)$ == $(x + z) \rightarrow y$ == z.

Lemma *Uplus_eq_zero_left* : $\forall$ x y:U, $x \leq$ [1-] $y \rightarrow (x + y)$ == $y \rightarrow x$ == 0.

Lemma *Uinv_le_trans* : $\forall$ x y z t, $x \leq$ [1-] $y \rightarrow z \leq x \rightarrow t \leq y \rightarrow z \leq$ [1-] t.

Lemma *Uinv_plus_left_le* : $\forall$ x y, [1-]$y \leq$ [1-]$(x+y)$ $+x$.

Lemma *Uinv_plus_right_le* : $\forall$ x y, [1-]$x \leq$ [1-]$(x+y)$ $+y$.

Hint *Resolve Uinv_plus_left_le Uinv_plus_right_le.*

4.7 Disequality

Lemma *neq_sym* : $\forall\ x\ y, \neg x{==}y \to \neg y{==}x$.
Hint Immediate *neq_sym*.

Lemma *Uinv_neq_compat* : $\forall\ x\ y, \neg x == y \to \neg$ [1-] $x ==$ [1-] y.

Lemma *Uinv_neq_simpl* : $\forall\ x\ y, \neg$ [1-] $x ==$ [1-] $y \to \neg x == y$.

Hint *Resolve* *Uinv_neq_compat*.
Hint Immediate *Uinv_neq_simpl*.

Lemma *Uinv_neq_left* : $\forall\ x\ y, \neg x ==$ [1-] $y \to \neg$ [1-] $x == y$.

Lemma *Uinv_neq_right* : $\forall\ x\ y, \neg$ [1-] $x == y \to \neg x ==$ [1-] y.

4.7.1 Properties of $<$

Lemma *Ult_antirefl* : $\forall\ x{:}U, \neg x < x$.

Lemma *Ult_0_1* : $(0 < 1)$.

Lemma *Ule_lt_trans* : $\forall\ x\ y\ z{:}U,\ x \leq y \to y < z \to x < z$.

Lemma *Ult_le_trans* : $\forall\ x\ y\ z{:}U,\ x < y \to y \leq z \to x < z$.

Hint *Resolve* *Ult_0_1* *Ult_antirefl*.

Lemma *Uplus_neq_zero_left* : $\forall\ x\ y, \neg 0 == x \to \neg 0 == x{+}y$.

Lemma *Uplus_neq_zero_right* : $\forall\ x\ y, \neg 0 == y \to \neg 0 == x{+}y$.

Lemma *not_Ult_le* : $\forall\ x\ y, \neg x < y \to y \leq x$.

Lemma *Ule_not_lt* : $\forall\ x\ y,\ x \leq y \to \neg y < x$.

Hint Immediate *not_Ult_le* *Ule_not_lt*.

Theorem *Uplus_le_simpl_left* : $\forall\ x\ y\ z :\ U,\ z \leq$ [1-] $x \to z + x \leq z + y \to x \leq y$.

Lemma *Uplus_lt_compat_left* : $\forall\ x\ y\ z{:}U,\ z \leq$ [1-] $y \to x < y \to (x + z) < (y + z)$.

Lemma *Uplus_lt_compat_right* : $\forall\ x\ y\ z{:}U,\ z \leq$ [1-] $y \to x < y \to (z + x) < (z + y)$.

Hint *Resolve* *Uplus_lt_compat_right* *Uplus_lt_compat_left*.

Lemma *Uplus_lt_compat* :
$\forall\ x\ y\ z\ t{:}U,\ z \leq$ [1-] $x \to t \leq$ [1-] $y \to x < y \to z < t \to (x + z) < (y + t)$.

Hint Immediate *Uplus_lt_compat*.

Lemma *Uplus_lt_simpl_left* : $\forall\ x\ y\ z{:}U,\ z \leq$ [1-] $y \to (z + x) < (z + y) \to x < y$.

Lemma *Uplus_lt_simpl_right* : $\forall\ x\ y\ z{:}U,\ z \leq$ [1-] $y \to (x + z) < (y + z) \to x < y$.

Lemma *Uplus_one_le* : $\forall\ x\ y,\ x + y == 1 \to$ [1-] $y \leq x$.
Hint Immediate *Uplus_one_le*.

Theorem *Uplus_eq_zero* : $\forall\ x,\ x \leq$ [1-] $x \to (x + x) == x \to x == 0$.

Lemma *Umult_zero_left* : $\forall\ x,\ 0 \times x == 0$.
Hint *Resolve* *Umult_zero_left*.

Lemma *Umult_zero_right* : $\forall\ x,\ (x \times 0) == 0$.
Hint *Resolve* *Uplus_eq_zero* *Umult_zero_right*.

4.7.2 Compatibility of operations with respect to order.

Lemma *Umult_le_simpl_right* : $\forall\ x\ y\ z,\ \neg 0 == z \rightarrow (x \times z) \leq (y \times z) \rightarrow x \leq y$.
Hint *Resolve Umult_le_simpl_right.*

Lemma *Umult_simpl_right* : $\forall\ x\ y\ z,\ \neg 0 == z \rightarrow (x \times z) == (y \times z) \rightarrow x == y$.

Lemma *Umult_simpl_left* : $\forall\ x\ y\ z,\ \neg 0 == x \rightarrow (x \times y) == (x \times z) \rightarrow y == z$.

Lemma *Umult_lt_compat_left* : $\forall\ x\ y\ z,\ \neg 0 == z \rightarrow x < y \rightarrow (x \times z) < (y \times z)$.

Lemma *Umult_lt_compat_right* : $\forall\ x\ y\ z,\ \neg 0 == z \rightarrow x < y \rightarrow (z \times x) < (z \times y)$.

Lemma *Umult_lt_simpl_right* : $\forall\ x\ y\ z,\ \neg 0 == z \rightarrow (x \times z) < (y \times z) \rightarrow x < y$.

Lemma *Umult_lt_simpl_left* : $\forall\ x\ y\ z,\ \neg 0 == z \rightarrow (z \times x) < (z \times y) \rightarrow x < y$.

Hint *Resolve Umult_lt_compat_left Umult_lt_compat_right.*

Lemma *Umult_zero_simpl_right* : $\forall\ x\ y,\ 0 == x \times y \rightarrow \neg 0 == x \rightarrow 0 == y$.

Lemma *Umult_zero_simpl_left* : $\forall\ x\ y,\ 0 == x \times y \rightarrow \neg 0 == y \rightarrow 0 == x$.

Lemma *Umult_neq_zero* : $\forall\ x\ y,\ \neg 0 == x \rightarrow \neg 0 == y \rightarrow \neg 0 == x \times y$.
Hint *Resolve Umult_neq_zero.*

Lemma *Umult_lt_zero* : $\forall\ x\ y,\ 0 < x \rightarrow 0 < y \rightarrow 0 < x \times y$.
Hint *Resolve Umult_lt_zero.*

Lemma *Umult_lt_compat* : $\forall\ x\ y\ z\ t,\ x < y \rightarrow z < t \rightarrow x \times z < y \times t$.

4.7.3 More Properties

Lemma *Uplus_one* : $\forall\ x\ y,\ [1\text{-}]\ x \leq y \rightarrow x + y == 1$.
Hint *Resolve Uplus_one.*

Lemma *Uplus_one_right* : $\forall\ x,\ x + 1 == 1$.

Lemma *Uplus_one_left* : $\forall\ x{:}U,\ 1 + x == 1$.
Hint *Resolve Uplus_one_right Uplus_one_left.*

Lemma *Uinv_mult_simpl* : $\forall\ x\ y\ z\ t,\ x \leq [1\text{-}]\ y \rightarrow (x \times z) \leq [1\text{-}]\ (y \times t)$.
Hint *Resolve Uinv_mult_simpl.*

Lemma *Umult_inv_plus* : $\forall\ x\ y,\ x \times [1\text{-}]\ y + y == x + y \times [1\text{-}]\ x$.
Hint *Resolve Umult_inv_plus.*

Lemma *Umult_inv_plus_le* : $\forall\ x\ y\ z,\ y \leq z \rightarrow x \times [1\text{-}]\ y + y \leq x \times [1\text{-}]\ z + z$.
Hint *Resolve Umult_inv_plus_le.*

Lemma *Uplus_lt_Uinv* : $\forall\ x\ y,\ x+y < 1 \rightarrow x \leq [1\text{-}]\ y$.

Lemma *Uinv_lt_perm_left*: $\forall\ x\ y\ :\ U,\ [1\text{-}]\ x < y \rightarrow [1\text{-}]\ y < x$.

Lemma *Uinv_lt_perm_right*: $\forall\ x\ y\ :\ U,\ x < [1\text{-}]\ y \rightarrow y < [1\text{-}]\ x$.

Hint Immediate *Uinv_lt_perm_left Uinv_lt_perm_right.*

Lemma *Uinv_lt_one* : $\forall\ x,\ 0 < x \rightarrow [1\text{-}]x < 1$.

Lemma *Uinv_lt_zero* : $\forall\ x,\ x < 1 \rightarrow 0 < [1\text{-}]x$.

Hint *Resolve Uinv_lt_one Uinv_lt_zero.*

Lemma *Umult_lt_right* : $\forall\ p\ q,\ p < 1 \rightarrow 0 < q \rightarrow p \times q < q$.

Lemma *Umult_lt_left* : $\forall\ p\ q,\ 0 < p \rightarrow q < 1 \rightarrow p \times q < p$.

Hint *Resolve Umult_lt_right Umult_lt_left.*

4.8 Definition of x^n

Fixpoint *Uexp* $(x{:}U)$ $(n{:}nat)$ $\{struct\ n\}$: U :=
  match n with $0 \Rightarrow 1 \mid (S\ p) \Rightarrow x \times$ *Uexp x p end.*

Infix "^" := *Uexp* : *U_scope.*

Lemma *Uexp_1* : $\forall\, x,\ x\hat{}1{=}{=}x$.

Lemma *Uexp_0* : $\forall\, x,\ x\hat{}0{=}{=}1$.

Lemma *Uexp_zero* : $\forall\, n,\ (0<n)\%nat \to 0\hat{}n{=}{=}0$.

Lemma *Uexp_one* : $\forall\, n,\ 1\hat{}n{=}{=}1$.

Lemma *Uexp_le_compat* :
  $\forall\ x\ n\ m,\ (n\leq m)\%nat \to x\hat{}m \leq x\hat{}n$.

Lemma *Uexp_Ule_compat* :
  $\forall\ x\ y\ n,\ x\leq y \to x\hat{}n \leq y\hat{}n$.

Add Morphism Uexp with signature Ueq ==> eq ==> Ueq as Uexp_eq_compat.

Lemma *Uexp_inv_S* : $\forall\ x\ n,\ ([1\text{-}]x\hat{}(S\ n)){=}{=}x*([1\text{-}]x\hat{}n)+[1\text{-}]x$.

Lemma *Uexp_lt_compat* : $\forall\ p\ q\ n,\ (O<n)\%nat\text{->}(p<q)\text{->}(p\hat{}n<q\hat{}n)$.

Hint *Resolve Uexp_lt_compat.*

Lemma *Uexp_lt_zero* : $\forall\ p\ n,\ (0<p)\text{->}(0<p\hat{}n)$.
Hint *Resolve Uexp_lt_zero.*

Lemma *Uexp_lt_one* : $\forall\ p\ n,\ (0<n)\%nat\text{->}(p<1)\text{->}(p\hat{}n<1)$.
Hint *Resolve Uexp_lt_one.*

Lemma *Uexp_lt_antimon*: $\forall\ p\ n\ m,\ (n<m)\%nat\to 0<p \to p < 1 \to p\hat{}m < p\hat{}n$.
Hint *Resolve Uexp_lt_antimon.*

4.9 Definition and properties of $x\&y$

A conjonction operation which coincides with min and mult on 0 and 1, see Morgan & McIver

Definition *Uesp* $(x\ y{:}U)$:= [1-] ([1-] x + [1-] y).

Infix "&" := *Uesp* (*left associativity, at level* 40) : *U_scope.*

Lemma *Uinv_plus_esp* : $\forall\ x\ y$, [1-] $(x + y)$ == [1-] x & [1-] y.
Hint *Resolve Uinv_plus_esp.*

Lemma *Uinv_esp_plus* : $\forall\ x\ y$, [1-] $(x\ \&\ y)$ == [1-] x + [1-] y.
Hint *Resolve Uinv_esp_plus.*

Lemma *Uesp_sym* : $\forall\ x\ y : U,\ x\ \&\ y == y\ \&\ x$.

Lemma *Uesp_one_right* : $\forall\ x : U,\ x\ \&\ 1 == x$.

Lemma *Uesp_one_left* : $\forall\ x : U,\ 1\ \&\ x == x$.

Lemma *Uesp_zero* : $\forall\ x\ y,\ x \leq [1\text{-}]\ y \to x\ \&\ y == 0$.

Hint *Resolve Uesp_sym Uesp_one_right Uesp_one_left Uesp_zero.*

Lemma *Uesp_zero_right* : $\forall\ x : U,\ x\ \&\ 0 == 0$.

Lemma *Uesp_zero_left* : $\forall\ x : U,\ 0\ \&\ x == 0$.

Hint *Resolve Uesp_zero_right Uesp_zero_left.*

Add Morphism Uesp with signature Ueq ==> Ueq ==> Ueq as Uesp_eq_compat.

Lemma *Uesp_le_compat* : $\forall\ x\ y\ z\ t,\ x\leq y \to z \leq t \to x\&z \leq y\&t$.

Hint Immediate *Uesp_le_compat Uesp_eq_compat.*

Lemma *Uesp_le_left* : $\forall\ x\ y,\ x\ \&\ y \leq x$.

Lemma *Uesp_le_right* : $\forall$ x y, x & $y \leq y$.

Hint *Resolve Uesp_le_left Uesp_le_right.*

Lemma *Uesp_plus_inv* : $\forall$ x y, [1-] $y \leq x \rightarrow x$ == x & y + [1-] y.
Hint *Resolve Uesp_plus_inv.*

Lemma *Uesp_le_plus_inv* : $\forall$ x y, $x \leq x$ & y + [1-] y.
Hint *Resolve Uesp_le_plus_inv.*

Lemma *Uplus_inv_le_esp* : $\forall$ x y z, $x \leq y$ + ([1-] z) $\rightarrow x$ & $z \leq y$.
Hint Immediate *Uplus_inv_le_esp.*

4.10 Definition and properties of $x - y$

Definition *Uminus* (x y:U) := [1-] ([1-] x + y).

Infix "-" := *Uminus* : *U_scope.*

Lemma *Uminus_le_compat_left* : $\forall$ x y z, $x \leq y \rightarrow x$ - $z \leq y$ - z.

Lemma *Uminus_le_compat_right* : $\forall$ x y z, $y \leq z \rightarrow x$ - $z \leq x$ - y.

Hint *Resolve Uminus_le_compat_left Uminus_le_compat_right.*

Lemma *Uminus_le_compat* : $\forall$ x y z t, $x \leq y \rightarrow t \leq z \rightarrow x$ - $z \leq y$ - t.

Hint Immediate *Uminus_le_compat.*

Add Morphism Uminus with signature Ueq ==> Ueq ==> Ueq as Uminus_eq_compat.
Hint Immediate *Uminus_eq_compat.*

Lemma *Uminus_zero_right* : $\forall$ x, x - 0 == x.

Lemma *Uminus_one_left* : $\forall$ x, 1 - x == [1-] x.

Lemma *Uminus_le_zero* : $\forall$ x y, $x \leq y \rightarrow x$ - y == 0.

Hint *Resolve Uminus_zero_right Uminus_one_left Uminus_le_zero.*

Lemma *Uminus_eq* : $\forall$ x, x-x == 0.
Hint *Resolve Uminus_eq.*

Lemma *Uminus_le_left* : $\forall$ x y, x - $y \leq x$.

Hint *Resolve Uminus_le_left.*

Lemma *Uminus_le_inv* : $\forall$ x y, x - $y \leq$ [1-]y.
Hint *Resolve Uminus_le_inv.*

Lemma *Uminus_plus_simpl* : $\forall$ x y, $y \leq x \rightarrow$ (x - y) + y == x.

Lemma *Uminus_plus_zero* : $\forall$ x y, $x \leq y \rightarrow$ (x - y) + y == y.

Hint *Resolve Uminus_plus_simpl Uminus_plus_zero.*

Lemma *Uesp_minus_distr_left* : $\forall$ x y z, (x & y) - z == (x - z) & y.

Lemma *Uesp_minus_distr_right* : $\forall$ x y z, (x & y) - z == x & (y - z).

Hint *Resolve Uesp_minus_distr_left Uesp_minus_distr_right.*

Lemma *Uesp_minus_distr* : $\forall$ x y z t, (x & y) - (z + t) == (x - z) & (y - t).
Hint *Resolve Uesp_minus_distr.*

Lemma *Uminus_esp_simpl_left* : $\forall$ x y, [1-]$x \leq y \rightarrow x$ - (x & y) == [1-]y.

Lemma *Uplus_esp_simpl* : $\forall$ x y, (x - (x & y))+y == x+y.
Hint *Resolve Uminus_esp_simpl_left Uplus_esp_simpl.*

Lemma *Uminus_esp_le_inv* : $\forall$ x y, x - (x & y) $\leq$ [1-]y.

Hint *Resolve Uminus_esp_le_inv.*

Lemma *Uplus_esp_inv_simpl* : $\forall$ x y, $x \leq$ [1-]$y \rightarrow$ (x + y) & [1-]y == x.
Hint *Resolve Uplus_esp_inv_simpl.*

Lemma *Uplus_inv_esp_simpl* : $\forall$ x y, $x \leq y \rightarrow$ (x + [1-]y) & y == x.
Hint *Resolve Uplus_inv_esp_simpl.*

4.11 Definition and properties of max

Definition $max\ (x\ y : U) : U := (x - y) + y.$

Lemma $max\_eq\_right : \forall\ x\ y : U, y \leq x \rightarrow max\ x\ y == x.$

Lemma $max\_eq\_left : \forall\ x\ y : U, x \leq y \rightarrow max\ x\ y == y.$

Hint $Resolve\ max\_eq\_right\ max\_eq\_left.$

Lemma $max\_eq\_case : \forall\ x\ y : U, orc\ (max\ x\ y == x)\ (max\ x\ y == y).$

Add Morphism max with signature Ueq ==> Ueq ==> Ueq as max_eq_compat.

Lemma $max\_le\_right : \forall\ x\ y : U, x \leq max\ x\ y.$

Lemma $max\_le\_left : \forall\ x\ y : U, y \leq max\ x\ y.$

Hint $Resolve\ max\_le\_right\ max\_le\_left.$

Lemma $max\_le : \forall\ x\ y\ z : U, x \leq z \rightarrow y \leq z \rightarrow max\ x\ y \leq z.$

4.12 Definition and properties of min

Definition $min\ (x\ y : U) : U :=$ [1-] $((y - x) +$ [1-]$y).$

Lemma $min\_eq\_right : \forall\ x\ y : U, x \leq y \rightarrow min\ x\ y == x.$

Lemma $min\_eq\_left : \forall\ x\ y : U, y \leq x \rightarrow min\ x\ y == y.$

Hint $Resolve\ min\_eq\_right\ min\_eq\_left.$

Lemma $min\_eq\_case : \forall\ x\ y : U, orc\ (min\ x\ y == x)\ (min\ x\ y == y).$

Add Morphism min with signature Ueq ==> Ueq ==> Ueq as min_eq_compat.

Lemma $min\_le\_right : \forall\ x\ y : U, min\ x\ y \leq x.$

Lemma $min\_le\_left : \forall\ x\ y : U, min\ x\ y \leq y.$

Hint $Resolve\ min\_le\_right\ min\_le\_left.$

Lemma $min\_le : \forall\ x\ y\ z : U, z \leq x \rightarrow z \leq y \rightarrow z \leq min\ x\ y.$

Lemma $Uinv\_min\_max : \forall\ x\ y,$ [1-]$(min\ x\ y) == max$ ([1-]x) ([1-]y).

Lemma $Uinv\_max\_min : \forall\ x\ y,$ [1-]$(max\ x\ y) == min$ ([1-]x) ([1-]y).

Lemma $min\_mult : \forall\ x\ y\ k,$
$min\ (k \times x)\ (k \times y) == k \times (min\ x\ y).$
Hint $Resolve\ min\_mult.$

Lemma $min\_plus : \forall\ x1\ x2\ y1\ y2,$
$(min\ x1\ x2) + (min\ y1\ y2) \leq min\ (x1+y1)\ (x2+y2).$
Hint $Resolve\ min\_plus.$

Lemma $min\_plus\_cte : \forall\ x\ y\ k, min\ (x + k)\ (y + k) == (min\ x\ y) + k.$
Hint $Resolve\ min\_plus\_cte.$

Lemma $min\_le\_compat : \forall\ x1\ x2\ y1\ y2,$
$x1 \leq y1 \rightarrow x2 \leq y2 \rightarrow min\ x1\ x2 \leq min\ y1\ y2.$

Lemma $min\_sym : \forall\ x\ y, min\ x\ y == min\ y\ x.$
Hint $Resolve\ min\_sym.$

Definition $incr\ (f{:}nat \rightarrow U) := \forall\ n, f\ n \leq f\ (S\ n).$

Lemma $incr\_mon : \forall\ f, incr\ f \rightarrow \forall\ n\ m, (n \leq m)\%nat \rightarrow f\ n \leq f\ m.$
Hint $Resolve\ incr\_mon.$

Lemma $incr\_decomp\_aux : \forall\ f\ g, incr\ f \rightarrow incr\ g \rightarrow$
$\forall\ n1\ n2, (\forall\ m, \neg\ ((n1 \leq m)\%nat \wedge f\ n1 \leq g\ m))$
$\rightarrow (\forall\ m, \tilde{}((n2 \leq m)\%nat \wedge g\ n2 \leq f\ m)) \rightarrow (n1 \leq n2)\%nat \rightarrow False.$

Lemma $incr\_decomp : \forall\ f\ g, incr\ f \rightarrow incr\ g \rightarrow$
$orc\ (\forall\ n, exc\ (fun\ m \Rightarrow (n \leq m)\%nat \wedge f\ n \leq g\ m))$
$(\forall\ n, exc\ (fun\ m \Rightarrow (n \leq m)\%nat \wedge g\ n \leq f\ m)).$

4.13 Other properties

Lemma *Uplus_minus_simpl_right* : $\forall$ x y, $y \leq$ [1-] $x \rightarrow (x + y)$ - $y == x$.
Hint *Resolve Uplus_minus_simpl_right*.

Lemma *Uplus_minus_simpl_left* : $\forall$ x y, $y \leq$ [1-] $x \rightarrow (x + y)$ - $x == y$.

Lemma *Uminus_assoc_left* : $\forall$ x y z, $(x$ - $y)$ - $z == x$ - $(y + z)$.

Hint *Resolve Uminus_assoc_left*.

Lemma *Uminus_perm* : $\forall$ x y z, $(x$ - $y)$ - $z == (x$ - $z)$ - y.
Hint *Resolve Uminus_perm*.

Lemma *Uminus_le_perm_left* : $\forall$ x y z, $y \leq x \rightarrow x$ - $y \leq z \rightarrow x \leq z + y$.

Lemma *Uplus_le_perm_left* : $\forall$ x y z, $y \leq x \rightarrow x \leq y + z \rightarrow x$ - $y \leq z$.

Lemma *Uminus_eq_perm_left* : $\forall$ x y z, $y \leq x \rightarrow x$ - $y == z \rightarrow x == z + y$.

Lemma *Uplus_eq_perm_left* : $\forall$ x y z, $y \leq$ [1-] $z \rightarrow x == y + z \rightarrow x$ - $y == z$.

Hint *Resolve Uminus_le_perm_left Uminus_eq_perm_left*.
Hint *Resolve Uplus_le_perm_left Uplus_eq_perm_left*.

Lemma *Uminus_le_perm_right* : $\forall$ x y z, $z \leq y \rightarrow x \leq y$ - $z \rightarrow x + z \leq y$.

Lemma *Uplus_le_perm_right* : $\forall$ x y z, $z \leq$ [1-] $x \rightarrow x + z \leq y \rightarrow x \leq y$ - z.
Hint *Resolve Uminus_le_perm_right Uplus_le_perm_right*.

Lemma *Uminus_le_perm* : $\forall$ x y z, $z \leq y \rightarrow x \leq$ [1-] $z \rightarrow x \leq y$ - $z \rightarrow z \leq y$ - x.
Hint *Resolve Uminus_le_perm*.

Lemma *Uminus_eq_perm_right* : $\forall$ x y z, $z \leq y \rightarrow x == y$ - $z \rightarrow x + z == y$.
Hint *Resolve Uminus_eq_perm_right*.

Lemma *Uminus_plus_perm* : $\forall$ x y z, $y \leq x \rightarrow z \leq$ [1-]$x \rightarrow x$ - $y + z == x + z$ - y.

Lemma *Uminus_zero_le* : $\forall$ x y, x - $y == 0 \rightarrow x \leq y$.

Lemma *Uminus_lt_non_zero* : $\forall$ x y, $x < y \rightarrow \neg 0 == y$ - x.
Hint Immediate *Uminus_zero_le Uminus_lt_non_zero*.

Lemma *Ult_le_nth* : $\forall$ x y, $x < y \rightarrow$ *exc* (*fun* $n \Rightarrow x \leq y$ - [1/]1+n).

Lemma *Uminus_distr_left* : $\forall$ x y z, $(x$ - $y) \times z == (x \times z)$ - $(y \times z)$.

Hint *Resolve Uminus_distr_left*.

Lemma *Uminus_distr_right* : $\forall$ x y z, $x \times (y$ - $z) == (x \times y)$ - $(x \times z)$.

Hint *Resolve Uminus_distr_right*.

Lemma *Uminus_assoc_right* : $\forall$ x y z, $y \leq x \rightarrow z \leq y \rightarrow x$ - $(y$ - $z) == (x$ - $y) + z$.

Lemma *Uplus_minus_assoc_right* : $\forall$ x y z, $y \leq$ [1-]$x \rightarrow z \leq y \rightarrow x + (y$ - $z) == (x + y)$ - z.

4.14 Definition and properties of generalized sums

Definition *sigma* (*alpha* : *nat* $\rightarrow$ U) (n:*nat*) := *comp Uplus* 0 *alpha* n.

Lemma *sigma_0* : $\forall$ (f : *nat* $\rightarrow$ U), *sigma* f 0 == 0.

Lemma *sigma_S* : $\forall$ (f :*nat* $\rightarrow$ U) (n:*nat*), *sigma* f (S n) = (f n) + (*sigma* f n).

Lemma *sigma_1* : $\forall$ (f : *nat* $\rightarrow$ U), *sigma* f (S 0) == f O.

Lemma *sigma_S_lift* : $\forall$ (f :*nat* $\rightarrow$ U) (n:*nat*),
    sigma f (S n) == (f O) + (*sigma* (*fun* $k \Rightarrow f$ (S k)) n).

Lemma *sigma_incr* : $\forall$ (f : *nat* $\rightarrow$ U) (n m:*nat*), ($n \leq m$)%*nat* $\rightarrow$ (*sigma* f n) $\leq$ (*sigma* f m).

Hint *Resolve sigma_incr*.

Lemma *sigma_eq_compat* : $\forall$ (f g: $nat \to U$) (n:nat),
($\forall$ k, (k < n)%nat $\to$ f k == g k) $\to$ ($sigma$ f n) == ($sigma$ g n).

Lemma *sigma_le_compat* : $\forall$ (f g: $nat \to U$) (n:nat),
($\forall$ k, (k < n)%nat $\to$ f $k \leq g$ k) $\to$ ($sigma$ f n) $\leq$ ($sigma$ g n).

Lemma *sigma_zero* : $\forall$ f n, ($\forall$ k, (k<n)%nat $\to$ f k ==0)->($sigma$ f n)==0.

Lemma *sigma_not_zero* : $\forall$ f n k, (k<n)%nat $\to$ 0 < f k $\to$ 0 < $sigma$ f n.

Lemma *sigma_zero_elim* : $\forall$ f n, ($sigma$ f n)==0$\to\forall$ k, (k<n)%nat $\to$ f k ==0.

Hint *Resolve sigma_eq_compat sigma_le_compat sigma_zero.*

Lemma *sigma_le* : $\forall$ f n k, (k<n)%nat $\to$ f $k \leq$ $sigma$ f n.

Lemma *sigma_minus_decr* : $\forall$ f n, ($\forall$ k, f (S k) $\leq$ f k) $\to$
sigma (*fun* $k \Rightarrow f$ k - f (S k)) n == f O - f n.

Lemma *sigma_minus_incr* : $\forall$ f n, ($\forall$ k, f $k \leq f$ (S k)) $\to$
sigma (*fun* $k \Rightarrow f$ (S k) - f k) n == f n - f O.

Definition *sigma_inf* (f : $nat \to U$) : U := *lub* (*sigma* f).

4.15 Definition and properties of generalized products

Definition *prod* (*alpha* : $nat \to U$) (n:nat) := *comp Umult* 1 *alpha* n.

Lemma *prod_0* : $\forall$ (f : $nat \to U$), *prod* f 0 = 1.

Lemma *prod_S* : $\forall$ (f :$nat \to U$) (n:nat), *prod* f (S n) = (f n) $\times$ (*prod* f n).

Lemma *prod_1* : $\forall$ (f : $nat \to U$), *prod* f (S 0) == f O.

Lemma *prod_S_lift* : $\forall$ (f :$nat \to U$) (n:nat),
prod f (S n) == (f O) $\times$ (*prod* (*fun* $k \Rightarrow f$ (S k)) n).

Lemma *prod_decr* : $\forall$ (f : $nat \to U$) (n m:nat), ($n \leq m$)%nat $\to$ (*prod* f m) $\leq$ (*prod* f n).

Hint *Resolve prod_decr.*

Lemma *prod_eq_compat* : $\forall$ (f g: $nat \to U$) (n:nat),
($\forall$ k, (k < n)%nat $\to$ f k == g k) $\to$ (*prod* f n) == (*prod* g n).

Lemma *prod_le_compat* : $\forall$ (f g: $nat \to U$) (n:nat),
($\forall$ k, (k < n)%nat $\to$ f $k \leq g$ k) $\to$ *prod* f n $\leq$ *prod* g n.

Lemma *prod_zero* : $\forall$ f n k, (k<n)%nat $\to$ f k ==0 $\to$ *prod* f n==0.

Lemma *prod_not_zero* : $\forall$ f n, ($\forall$ k, (k<n)%nat $\to$ 0 < f k)-> 0 < *prod* f n.

Lemma *prod_zero_elim* : $\forall$ f n, *prod* f n==0 $\to$ *exc* (*fun* $k \Rightarrow$ (k<n)%nat $\wedge$ f k ==0).

Hint *Resolve prod_eq_compat prod_le_compat prod_not_zero.*

Lemma *prod_le* : $\forall$ f n k, (k<n)%nat $\to$ *prod* f n $\leq f$ k.

Lemma *prod_minus* : $\forall$ f n, *prod* f n - *prod* f (S n) == ([1-]f n) $\times$ *prod* f n.

4.16 Properties of *Unth*

Lemma *Unth_zero* : [1/]1+0 == 1.

Notation "[1/2]" := (*Unth* 1).

Lemma *Unth_one* : [1/2] == [1-] [1/2].

Hint *Resolve Unth_zero Unth_one.*

Lemma *Unth_one_plus* : [1/2] + [1/2] == 1.
Hint *Resolve Unth_one_plus.*

Lemma *Unth_not_null* : $\forall$ n, $\neg$ (0 == [1/]1+n).

Hint *Resolve* *Unth_not_null*.

Lemma *Unth_lt_zero* : $\forall$ *n*, 0 < [1/]1+*n*.
Hint *Resolve* *Unth_lt_zero*.

Lemma *Unth_inv_lt_one* : $\forall$ *n*, [1-][1/]1+*n*<1.
Hint *Resolve* *Unth_inv_lt_one*.

Lemma *Unth_not_one* : $\forall$ *n*, $\neg$ (1 == [1-][1/]1+*n*).
Hint *Resolve* *Unth_not_one*.

Lemma *Unth_prop_sigma* : $\forall$ *n*, [1/]1+*n* == [1-] (*sigma* (*fun* *k* $\Rightarrow$ [1/]1+*n*) *n*).
Hint *Resolve* *Unth_prop_sigma*.

Lemma *Unth_sigma_n* : $\forall$ *n* : *nat*, $\neg$ (1 == *sigma* (*fun* *k* $\Rightarrow$ [1/]1+*n*) *n*).

Lemma *Unth_sigma_Sn* : $\forall$ *n* : *nat*, 1 == *sigma* (*fun* *k* $\Rightarrow$ [1/]1+*n*) (*S* *n*).

Hint *Resolve* *Unth_sigma_n* *Unth_sigma_Sn*.

Lemma *Unth_decr* : $\forall$ *n*, [1/]1+(*S* *n*) < [1/]1+*n*.
Hint *Resolve* *Unth_decr*.

Lemma *Unth_anti_mon* :
  $\forall$ *n* *m*, (*n* $\leq$ *m*)%*nat* $\rightarrow$ [1/]1+*m* $\leq$ [1/]1+*n*.
Hint *Resolve* *Unth_anti_mon*.

Lemma *Unth_le_half* : $\forall$ *n*, [1/]1+(*S* *n*) $\leq$ [1/2].
Hint *Resolve* *Unth_le_half*.

4.16.1 Mean of two numbers : $\frac{1}{2}x + \frac{1}{2}y$

Definition *mean* (*x* *y*:*U*) := [1/2] $\times$ *x* + [1/2] $\times$ *y*.

Lemma *mean_eq* : $\forall$ *x*:*U*, *mean* *x* *x* ==*x*.

Lemma *mean_le_compat_right* : $\forall$ *x* *y* *z*, *y* $\leq$ *z* $\rightarrow$ *mean* *x* *y* $\leq$ *mean* *x* *z*.

Lemma *mean_le_compat_left* : $\forall$ *x* *y* *z*, *x* $\leq$ *y* $\rightarrow$ *mean* *x* *z* $\leq$ *mean* *y* *z*.

Hint *Resolve* *mean_eq* *mean_le_compat_left* *mean_le_compat_right*.

Lemma *mean_lt_compat_right* : $\forall$ *x* *y* *z*, *y* < *z* $\rightarrow$ *mean* *x* *y* < *mean* *x* *z*.

Lemma *mean_lt_compat_left* : $\forall$ *x* *y* *z*, *x* < *y* $\rightarrow$ *mean* *x* *z* < *mean* *y* *z*.

Hint *Resolve* *mean_eq* *mean_le_compat_left* *mean_le_compat_right*.
Hint *Resolve* *mean_lt_compat_left* *mean_lt_compat_right*.

Lemma *mean_le_up* : $\forall$ *x* *y*, *x* $\leq$ *y* $\rightarrow$ *mean* *x* *y* $\leq$ *y*.

Lemma *mean_le_down* : $\forall$ *x* *y*, *x* $\leq$ *y* $\rightarrow$ *x* $\leq$ *mean* *x* *y*.

Lemma *mean_lt_up* : $\forall$ *x* *y*, *x* < *y* $\rightarrow$ *mean* *x* *y* < *y*.

Lemma *mean_lt_down* : $\forall$ *x* *y*, *x* < *y* $\rightarrow$ *x* < *mean* *x* *y*.

Hint *Resolve* *mean_le_up* *mean_le_down* *mean_lt_up* *mean_lt_down*.

4.16.2 Properties of $\frac{1}{2}$

Lemma *le_half_inv* : $\forall$ *x*, *x* $\leq$ [1/2] $\rightarrow$ *x* $\leq$ [1-] *x*.

Hint Immediate *le_half_inv*.

Lemma *ge_half_inv* : $\forall$ *x*, [1/2] $\leq$ *x* $\rightarrow$ [1-] *x* $\leq$ *x*.

Hint Immediate *ge_half_inv*.

Lemma *Uinv_le_half_left* : $\forall$ *x*, *x* $\leq$ [1/2] $\rightarrow$ [1/2] $\leq$ [1-] *x*.

Lemma *Uinv_le_half_right* : $\forall$ *x*, [1/2] $\leq$ *x* $\rightarrow$ [1-] *x* $\leq$ [1/2].

Hint *Resolve Uinv_le_half_left Uinv_le_half_right.*

Lemma *half_twice* : $\forall$ *x*, ($x \leq$ [1/2]) $\rightarrow$ ([1/2]) $\times$ (x + x) == x.

Lemma *half_twice_le* : $\forall$ *x*, ([1/2]) $\times$ (x + x) $\leq$ x.

Lemma *Uinv_half* : $\forall$ *x*, ([1/2]) $\times$ ([1-] x) + ([1/2]) == [1-] (([1/2]) $\times$ x).

Lemma *half_esp* :
 $\forall$ *x*, ([1/2] $\leq$ x) $\rightarrow$ ([1/2]) $\times$ (x & x) + [1/2] == x.

Lemma *half_esp_le* : $\forall$ *x*, $x \leq$ ([1/2]) $\times$ (x & x) + [1/2].
Hint *Resolve half_esp_le.*

Lemma *half_le* : $\forall$ *x y*, $y \leq$ [1-] y $\rightarrow$ $x \leq y + y$ $\rightarrow$ ([1/2]) $\times$ $x \leq y$.

Lemma *half_Unth*: $\forall$ *n*, ([1/2])*([1/]1+n) $\leq$ [1/]1+(S n).
Hint *Resolve half_le half_Unth.*

Lemma *half_exp* : $\forall$ *n*, [1/2]^n == [1/2]^(S n) + [1/2]^(S n).

4.17 Density

Lemma *Ule_lt_lim* : $\forall$ *x y*, ($\forall$ *t*, $t < x \rightarrow t \leq y$) $\rightarrow$ $x \leq y$.

4.18 Properties of least upper bounds

Section *lubs.*

Lemma *lub_le_stable* : $\forall$ *f g*, ($\forall$ *n*, f $n \leq g$ n) $\rightarrow$ *lub* $f \leq$ *lub* g.

Hint *Resolve lub_le_stable.*

Lemma *lub_eq_stable* : $\forall$ *f g*, ($\forall$ *n*, f n == g n) $\rightarrow$ *lub* f == *lub* g.

Hint *Resolve lub_eq_stable.*

Lemma *lub_zero* : (*lub* (*fun* $n \Rightarrow 0$)) == 0.

Lemma *lub_un* : (*lub* (*fun* $n \Rightarrow 1$)) == 1.

Lemma *lub_cte* : $\forall$ *c*:*U*, (*lub* (*fun* $n \Rightarrow c$)) == c.

Hint *Resolve lub_zero lub_un lub_cte.*

Lemma *lub_eq_plus_cte_left* : $\forall$ (f : *nat* $\rightarrow$ U) (k:U), *lub* (*fun* $n \Rightarrow k$ + (f n)) == k + (*lub* f).
Hint *Resolve lub_eq_plus_cte_left.*

Lemma *min_lub_le* : $\forall$ *f g*,
 lub (*fun* $n \Rightarrow$ *min* (f n) (g n)) $\leq$ *min* (*lub* f) (*lub* g).

Lemma *min_lub_le_incr_aux* : $\forall$ *f g*, *incr* f $\rightarrow$
 ($\forall$ *n*, *exc* (*fun* $m \Rightarrow$ ($n \leq m$)%*nat* $\wedge$ f $n \leq g$ m))
 $\rightarrow$ *min* (*lub* f) (*lub* g) $\leq$ *lub* (*fun* $n \Rightarrow$ *min* (f n) (g n)).

Lemma *min_lub_le_incr* : $\forall$ *f g*, *incr* f $\rightarrow$ *incr* g $\rightarrow$
 min (*lub* f) (*lub* g) $\leq$ *lub* (*fun* $n \Rightarrow$ *min* (f n) (g n)).

Lemma *lub_eq_esp_right* :
 $\forall$ (f : *nat* $\rightarrow$ U) (k : U), *lub* (*fun* n : *nat* $\Rightarrow$ f n & k) == *lub* f & k.
Hint *Resolve lub_eq_esp_right.*

4.19 Greatest lower bounds

Definition *glb* $(f{:}nat{\rightarrow}U)$:= [1-]*lub* (*fun* $n \Rightarrow$ [1-]$(f\ n)$).

Definition *prod_inf* $(f : nat \rightarrow U) : U :=$ *glb* (*prod* f).

Lemma *glb_le_stable*:
$\forall f\ g : nat \rightarrow U, (\forall n : nat, f\ n \leq g\ n) \rightarrow glb\ f \leq glb\ g.$
Hint *Resolve glb_le_stable.*

Lemma *glb_eq_stable*:
$\forall f\ g : nat \rightarrow U, (\forall n : nat, f\ n == g\ n) \rightarrow glb\ f == glb\ g.$
Hint *Resolve glb_eq_stable.*

Lemma *glb_cte*: $\forall c : U, glb\ (fun\ \_ : nat \Rightarrow c) == c.$
Hint *Resolve glb_cte.*

Lemma *glb_eq_plus_cte_right*:
$\forall (f : nat \rightarrow U)\ (k : U), glb\ (fun\ n : nat \Rightarrow f\ n + k) == glb\ f + k.$

Lemma *glb_eq_mult*:
$\forall (k : U)\ (f : nat \rightarrow U), glb\ (fun\ n : nat \Rightarrow k \times f\ n) == k \times glb\ f.$

Lemma *glb_le*: $\forall (f : nat \rightarrow U)\ (n : nat), glb\ f \leq (f\ n).$

Lemma *le_glb*: $\forall (f : nat \rightarrow U)\ (x{:}U), (\forall n : nat, x \leq f\ n)\text{-}{>}\ x \leq glb\ f.$
Hint *Resolve glb_le.*

Lemma *glb_le_esp* : $\forall f\ g, (glb\ f)\ \&\ (glb\ g) \leq glb\ (fun\ n \Rightarrow (f\ n)\ \&\ (g\ n)).$
Hint *Resolve glb_le_esp.*

Lemma *Uesp_min* : $\forall a1\ a2\ b1\ b2, min\ a1\ b1\ \&\ min\ a2\ b2 \leq min\ (a1\ \&\ a2)\ (b1\ \&\ b2).$

Lemma *mon_seq_Succ* : $\forall f : nat \rightarrow U, (\forall n, f\ n \leq f\ (S\ n)) \rightarrow mon\_seq\ Ule\ f.$
Hint Immediate *mon_seq_Succ.*

Variables $f\ g : nat \rightarrow U.$

Hypothesis *monf* : $\forall n, f\ n \leq f\ (S\ n).$
Hypothesis *mong* : $\forall n, g\ n \leq g\ (S\ n).$

Lemma *mon_seqf* : *mon_seq Ule f.*

Lemma *mon_seqg* : *mon_seq Ule g.*

Hint *Resolve mon_seqf mon_seqg.*

Lemma *lub_lift* : $\forall n, (lub\ f) == (lub\ (fun\ k \Rightarrow f\ (n{+}k)\%nat)).$

Hint *Resolve lub_lift.*

Let *sum* := *fun* $n \Rightarrow f\ n + g\ n.$

Lemma *mon_sum* : *mon_seq Ule sum.*

Hint *Resolve mon_sum.*

Lemma *lub_eq_plus* : $lub\ (fun\ n \Rightarrow (f\ n) + (g\ n)) == (lub\ f) + (lub\ g).$
Hint *Resolve lub_eq_plus.*

Variables $k : U.$
Let *prod* := *fun* $n \Rightarrow k \times f\ n.$

Lemma *mon_prod* : *mon_seq Ule prod.*

Let *inv*:= *fun* $n \Rightarrow$ [1-] $(g\ n).$

Lemma *lub_inv* : $(\forall n, f\ n \leq inv\ n) \rightarrow lub\ f \leq$ [1-] $(lub\ g).$

Variable $h : nat \rightarrow U.$
Hypothesis *dech* : $\forall n, h\ (S\ n) \leq h\ n.$

Lemma *dec_sech* : $\forall n\ m, (n \leq m)\%nat \rightarrow h\ m \leq h\ n.$
Hint *Resolve dec_sech.*

Lemma *glb_lift* : $\forall$ *n*, (*glb h*) == (*glb* (*fun k* $\Rightarrow$ *h* (*n*+*k*)%*nat*)).

Hint *Resolve glb_lift.*

Lemma *lub_glb_le* : ($\forall$ *n*, *f n* $\leq$ *h n*) $\rightarrow$ *lub f* $\leq$ *glb h*.

End *lubs.*

Lemma *double_lub_simpl* : $\forall$ *h* : *nat* $\rightarrow$ *nat* $\rightarrow$ *U*,
($\forall$ *n m*, *h n m* $\leq$ *h* (*S n*) *m*) $\rightarrow$ ($\forall$ *n m*, *h n m* $\leq$ *h n* (*S m*))
$\rightarrow$ *lub* (*fun n* $\Rightarrow$ *lub* (*h n*)) == *lub* (*fun n* $\Rightarrow$ *h n n*).

Lemma *double_lub_exch_le* : $\forall$ *h* : *nat* $\rightarrow$ *nat* $\rightarrow$ *U*,
lub (*fun n* $\Rightarrow$ *lub* (*fun m* $\Rightarrow$ *h n m*)) $\leq$ *lub* (*fun m* $\Rightarrow$ *lub* (*fun n* $\Rightarrow$ *h n m*)).
Hint *Resolve double_lub_exch_le.*

Hint *Resolve double_lub_exch_le.*

Lemma *double_lub_exch* : $\forall$ *h* : *nat* $\rightarrow$ *nat* $\rightarrow$ *U*,
lub (*fun n* $\Rightarrow$ *lub* (*fun m* $\Rightarrow$ *h n m*)) == *lub* (*fun m* $\Rightarrow$ *lub* (*fun n* $\Rightarrow$ *h n m*)).

Hint *Resolve double_lub_exch.*

4.19.1 Definitions

Definition *fle* (*A*:*Type*) (*f g*:*A*$\rightarrow$*U*) : *Prop* := $\forall$ *x*:*A*, (*f x*) $\leq$ (*g x*).
Definition *feq* (*A*:*Type*) (*f g*:*A*$\rightarrow$*U*) : *Prop* := $\forall$ *x*:*A*, (*f x*) == (*g x*).
Hint *Unfold fle feq.*
Definition *fplus* (*A*:*Type*) (*f g*:*A*$\rightarrow$*U*) (*x*:*A*) : *U* := (*f x*) + (*g x*).
Definition *fesp* (*A*:*Type*) (*f g*:*A*$\rightarrow$*U*) (*x*:*A*) : *U* := (*f x*) & (*g x*).
Definition *fminus* (*A*:*Type*) (*f g*:*A*$\rightarrow$*U*) (*x*:*A*) : *U* := (*f x*) - (*g x*).
Definition *finv* (*A*:*Type*) (*f*:*A*$\rightarrow$*U*) (*x*:*A*) : *U* := *Uinv* (*f x*).
Definition *fmult* (*A*:*Type*) (*k*:*U*) (*f*:*A*$\rightarrow$*U*) (*x*:*A*) : *U* := *k* $\times$ (*f x*).
Definition *f_one* (*A*:*Type*) (*x* : *A*) : *U* := *U1.*
Definition *f_zero* (*A*:*Type*) (*x* : *A*) : *U* := *U0.*
Definition *f_cte* (*A*:*Type*) (*c*:*U*) (*x* : *A*) : *U* := *c*.
Definition *flub* (*A*:*Type*) (*fn*:*nat*$\rightarrow$*A*$\rightarrow$*U*) (*x* : *A*) : *U* := *lub* (*fun n* $\Rightarrow$ *fn n x*).
Definition *fglb* (*A*:*Type*) (*fn*:*nat*$\rightarrow$*A*$\rightarrow$*U*) (*x* : *A*) : *U* := *glb* (*fun n* $\Rightarrow$ *fn n x*).
Definition *increase* (*A*:*Type*)(*fn* : *nat* $\rightarrow$ *A* $\rightarrow$ *U*) := $\forall$ *n*, *fle* (*fn n*) (*fn* (*S n*)).
Definition *decrease* (*A*:*Type*)(*fn* : *nat* $\rightarrow$ *A* $\rightarrow$ *U*) := $\forall$ *n*, *fle* (*fn* (*S n*)) (*fn n*).

Implicit *Arguments f_one* [].
Implicit *Arguments f_zero* [].
Implicit *Arguments f_cte* [].

4.19.2 Elementary properties

Lemma *feq_refl* : $\forall$ (*A*:*Type*) (*f* : *A*$\rightarrow$*U*), *feq f f*.
Hint *Resolve feq_refl.*

Lemma *feq_sym* : $\forall$ (*A*:*Type*) (*f g* : *A*$\rightarrow$*U*), *feq f g* $\rightarrow$ *feq g f*.

Lemma *feq_trans* : $\forall$ (*A*:*Type*) (*f g h* : *A*$\rightarrow$*U*), *feq f g* $\rightarrow$ *feq g h* $\rightarrow$ *feq f h*.

Lemma *fSetoid* : $\forall$ (*A*:*Type*), *Setoid_Theory* (*A*$\rightarrow$*U*) (@*feq A*).

Add Setoid (*fun A* $\Rightarrow$ *A*$\rightarrow$*U*) *feq fSetoid as f_Setoid.*

Lemma *feq_fle* : $\forall$ (*A*:*Type*) (*f g* : *A*$\rightarrow$*U*), *feq f g* $\rightarrow$ *fle f g*.

Lemma *feq_fle_sym* : $\forall$ (*A*:*Type*) (*f g* : *A*$\rightarrow$*U*), *feq f g* $\rightarrow$ *fle g f*.
Hint Immediate *feq_fle feq_fle_sym.*

Lemma *fle_le* : $\forall$ (*A*:*Type*) (*f g* : *A*$\rightarrow$*U*), *fle f g* $\rightarrow$ $\forall$ *x*, *f x* $\leq$ *g x*.

Lemma *fle_refl* : $\forall$ (*A*:*Type*) (*f*:*A*$\rightarrow$*U*), *fle f f*.

Lemma *fle_trans* : $\forall$ (*A*:*Type*) (*f g h* : *A*$\rightarrow$*U*), *fle f g* $\rightarrow$ *fle g h* $\rightarrow$ *fle f h*.

Add Relation (fun (A:Type) ⇒ A→U) fle reflexivity proved by fle_refl transitivity proved by fle_trans as fle_Relation.

Lemma *fle_feq_trans* : ∀ (*A*:*Type*) (*f g h* : *A*→*U*), *fle f g* → *feq g h* → *fle f h*.

Lemma *feq_fle_trans* : ∀ (*A*:*Type*) (*f g h* : *A*→*U*), *feq f g* → *fle g h* → *fle f h*.

Lemma *fle_antisym* : ∀ (*A*:*Type*) (*f g* : *A*→*U*), *fle f g* → *fle g f* → *feq f g*.
Hint *Resolve fle_antisym.*

Add Morphism fle with signature feq ==> feq ==> iff as fle_feq_compat.

Lemma *fle_fplus_left* : ∀ (*A*:*Type*) (*f g* : *A*→*U*), *fle f* (*fplus f g*).

Lemma *fle_fplus_right* : ∀ (*A*:*Type*) (*f g* : *A*→*U*), *fle g* (*fplus f g*).

Lemma *fle_fmult* : ∀ (*A*:*Type*) (*k*:*U*)(*f* : *A*→*U*), *fle* (*fmult k f*) *f*.

Lemma *fle_zero* : ∀ (*A*:*Type*) (*f* : *A*→*U*), *fle* (*f_zero A*) *f*.

Lemma *fle_one* : ∀ (*A*:*Type*) (*f* : *A*→*U*), *fle f* (*f_one A*).

Lemma *feq_finv_finv* : ∀ (*A*:*Type*) (*f* : *A*→*U*), *feq* (*finv* (*finv f*)) *f*.

Lemma *fle_fesp_left* : ∀ (*A*:*Type*) (*f g* : *A*→*U*), *fle* (*fesp f g*) *f*.

Lemma *fle_fesp_right* : ∀ (*A*:*Type*) (*f g* : *A*→*U*), *fle* (*fesp f g*) *g*.

4.19.3 Defining morphisms

Add Morphism fplus with signature feq ==> feq ==> feq as fplus_feq_compat.

Add Morphism fplus with signature fle ++> fle ++> fle as fplus_fle_compat.

Add Morphism finv with signature feq ==> feq as finv_feq_compat.

Add Morphism finv with signature fle --> fle as finv_fle_compat.

Add Morphism fmult with signature Ueq ==> feq ==> feq as fmult_feq_compat.

Add Morphism fmult with signature Ule ++> fle ++> fle as fmult_fle_compat.

Add Morphism fminus with signature feq ==> feq ==> feq as fminus_feq_compat.

Add Morphism fminus with signature fle ++> fle --> fle as fminus_fle_compat.

Add Morphism fesp with signature feq ==> feq ==> feq as fesp_feq_compat.

Add Morphism fesp with signature fle ++> fle ++> fle as fesp_fle_compat.

Hint Immediate *feq_sym fplus_fle_compat fplus_feq_compat fmult_fle_compat fmult_feq_compat fminus_fle_compat fminus_feq_compat.*

Hint *Resolve fle_fplus_left fle_fplus_right fle_zero fle_one feq_finv_finv finv_fle_compat fle_fmult fle_fesp_left fle_fesp_right.*

Hint *Resolve finv_feq_compat finv_fle_compat.*

4.20 Fixpoints of functions of type $A \to [0,1]$

Section *FixDef.*
Variable *A* : *Type*.

Variable *F* : (*A*→*U*) → *A* → *U*.
Definition *Fmonotonic* := ∀ *f g*, (*fle f g*) → *fle* (*F f*) (*F g*).
Definition *Fstable* := ∀ *f g*, (*feq f g*) → *feq* (*F f*) (*F g*).

Lemma *Fmonotonic_stable* : *Fmonotonic* → *Fstable*.

Lemma *Fmonotonic_fle* : *Fmonotonic* → ∀ *f g*, *fle f g* → *fle* (*F f*) (*F g*).

Lemma *Fmonotonic_le* : *Fmonotonic* → ∀ *x*, ∀ *f g*, *fle f g* → ∀ *x*, *F f x* ≤ *F g x*.

Lemma $Fstable\_feq$: $Fstable \rightarrow \forall\ f\ g,\ feq\ f\ g \rightarrow feq\ (F\ f)\ (F\ g)$.

Lemma $Fstable\_eq$: $Fstable \rightarrow \forall\ f\ g,\ feq\ f\ g \rightarrow \forall\ x,\ F\ f\ x == F\ g\ x$.

Hint $Resolve\ Fmonotonic\_fle\ Fstable\_feq\ Fmonotonic\_le\ Fstable\_eq$.

Hypothesis $Fmon$: $Fmonotonic$.

Fixpoint $muiter\ (n{:}nat)\ (x{:}A)\ \{struct\ n\}$: U :=
$match\ n\ with\ O \Rightarrow 0\ |\ S\ p \Rightarrow F\ (muiter\ p)\ x\ end$.

Fixpoint $nuiter\ (n{:}nat)\ (x{:}A)\ \{struct\ n\}$: U :=
$match\ n\ with\ O \Rightarrow 1\ |\ S\ p \Rightarrow F\ (nuiter\ p)\ x\ end$.

Definition $mufix\ (x{:}A) := lub\ (fun\ n \Rightarrow muiter\ n\ x)$.
Definition $nufix\ (x{:}A) := glb\ (fun\ n \Rightarrow nuiter\ n\ x)$.

Lemma $mufix\_inv$: $\forall\ f,\ fle\ (F\ f)\ f \rightarrow fle\ mufix\ f$.
Hint $Resolve\ mufix\_inv$.

Lemma $nufix\_inv$: $\forall\ f,\ fle\ f\ (F\ f) \rightarrow fle\ f\ nufix$.
Hint $Resolve\ nufix\_inv$.

Lemma $mufix\_le$: $fle\ mufix\ (F\ mufix)$.
Hint $Resolve\ mufix\_le$.

Lemma $nufix\_sup$: $fle\ (F\ nufix)\ nufix$.
Hint $Resolve\ nufix\_sup$.

Definition $Fcontlub := \forall\ (fn : nat \rightarrow A \rightarrow U),\ increase\ fn \rightarrow$
$fle\ (F\ (flub\ fn))\ (flub\ (fun\ n \Rightarrow F\ (fn\ n)))$.
Definition $Fcontglb := \forall\ (fn : nat \rightarrow A \rightarrow U),\ decrease\ fn \rightarrow$
$fle\ (fglb\ (fun\ n \Rightarrow F\ (fn\ n)))\ (F\ (fglb\ fn))$.

Lemma $Fcontlub\_fle$: $Fcontlub \rightarrow \forall\ (fn : nat \rightarrow A \rightarrow U),\ increase\ fn \rightarrow$
$fle\ (F\ (flub\ fn))\ (flub\ (fun\ n \Rightarrow F\ (fn\ n)))$.

Lemma $Fcontglb\_fle$: $Fcontglb \rightarrow \forall\ (fn : nat \rightarrow A \rightarrow U),\ decrease\ fn \rightarrow$
$fle\ (fglb\ (fun\ n \Rightarrow F\ (fn\ n)))\ (F\ (fglb\ fn))$.

Hypothesis $muFcont$: $\forall\ (fn : nat \rightarrow A \rightarrow U),\ increase\ fn \rightarrow$
$fle\ (F\ (flub\ fn))\ (flub\ (fun\ n \Rightarrow F\ (fn\ n)))$.

Hypothesis $nuFcont$: $\forall\ (fn : nat \rightarrow A \rightarrow U),\ decrease\ fn \rightarrow$
$fle\ (fglb\ (fun\ n \Rightarrow F\ (fn\ n)))\ (F\ (fglb\ fn))$.

Implicit $Arguments\ muFcont$ [].
Implicit $Arguments\ nuFcont$ [].

Lemma $incr\_muiter$: $increase\ muiter$.

Lemma $decr\_nuiter$: $decrease\ nuiter$.

Hint $Resolve\ incr\_muiter\ decr\_nuiter$.

Lemma $mufix\_sup$: $\forall\ x,\ F\ mufix\ x \leq mufix\ x$.
Hint $Resolve\ mufix\_sup$.

Lemma $nufix\_le$: $\forall\ x,\ nufix\ x \leq F\ nufix\ x$.
Hint $Resolve\ nufix\_le$.

Lemma $mufix\_eq$: $\forall\ x,\ mufix\ x == F\ mufix\ x$.
Hint $Resolve\ mufix\_eq$.

Lemma $nufix\_eq$: $\forall\ x,\ nufix\ x == F\ nufix\ x$.
Hint $Resolve\ nufix\_eq$.

End $FixDef$.
Hint $Unfold\ Fmonotonic$.
Hint $Resolve\ Fmonotonic\_stable$.
Hint $Resolve\ Fmonotonic\_fle\ Fstable\_feq\ Fmonotonic\_le\ Fstable\_eq$.
Hint $Resolve\ Fcontlub\_fle\ Fcontglb\_fle$.

Definition $Fcte\ (A{:}Type)\ (f{:}A{\rightarrow}U) := fun\ (\_{:}A{\rightarrow}U) \Rightarrow f.$
Lemma $Fcte\_mon : \forall\ (A{:}Type)\ (f{:}A{\rightarrow}U),\ Fmonotonic\ (Fcte\ f).$

Lemma $mufix\_cte : \forall\ (A{:}Type)\ (f{:}A{\rightarrow}U),\ feq\ (mufix\ (Fcte\ f))\ f.$

Lemma $nufix\_cte : \forall\ (A{:}Type)\ (f{:}A{\rightarrow}U),\ feq\ (nufix\ (Fcte\ f))\ f.$

Hint $Resolve\ mufix\_cte\ nufix\_cte.$

4.21 Properties of barycenter of two points

Section $Barycenter.$
Variables $a\ b\ :\ U.$
Hypothesis $sum\_le\_one\ :\ a \leq [1\text{-}]\ b.$

Lemma $Uinv\_bary$:
$\forall\ x\ y\ :\ U,\ [1\text{-}]\ (a \times x + b \times y) == a \times [1\text{-}]\ x + b \times [1\text{-}]\ y + [1\text{-}]\ (a + b).$

Lemma $Uinv\_bary\_le$:
$\forall\ x\ y\ :\ U,\ a \times [1\text{-}]\ x + b \times [1\text{-}]\ y \leq [1\text{-}]\ (a \times x + b \times y).$

End $Barycenter.$
Hint $Resolve\ Uinv\_bary\_le.$

Lemma $Uinv\_half\_bary$:
$\forall\ x\ y\ :\ U,\ [1\text{-}]\ ([1/2] \times x + [1/2] \times y) == [1/2] \times [1\text{-}]\ x + [1/2] \times [1\text{-}]\ y.$
Hint $Resolve\ Uinv\_half\_bary.$

4.22 Properties of generalized sums *sigma*

Lemma $sigma\_plus : \forall\ (f\ g\ :\ nat \rightarrow U)\ (n{:}nat),$
$(sigma\ (fun\ k \Rightarrow (f\ k) + (g\ k))\ n) == (sigma\ f\ n) + (sigma\ g\ n).$

Definition $retract\ (f\ :\ nat \rightarrow U)\ (n\ :\ nat) := \forall\ k,\ (k < n)\%nat \rightarrow (f\ k) \leq [1\text{-}]\ (sigma\ f\ k).$

Lemma $retract0 : \forall\ (f\ :\ nat \rightarrow U),\ retract\ f\ 0.$

Lemma $retract\_pred : \forall\ (f\ :\ nat \rightarrow U)\ (n\ :\ nat),\ retract\ f\ (S\ n) \rightarrow retract\ f\ n.$

Lemma $retractS$: $\forall\ (f\ :\ nat \rightarrow U)\ (n\ :\ nat),\ retract\ f\ (S\ n) \rightarrow f\ n \leq [1\text{-}]\ (sigma\ f\ n).$

Lemma $retractS\_intro$: $\forall\ (f\ :\ nat \rightarrow U)\ (n\ :\ nat),$
$retract\ f\ n \rightarrow f\ n \leq [1\text{-}]\ (sigma\ f\ n){-}{>}retract\ f\ (S\ n).$

Hint $Resolve\ retract0\ retractS\_intro.$
Hint Immediate $retract\_pred\ retractS.$

Lemma $retract\_lt : \forall\ (f\ :\ nat \rightarrow U)\ (n\ :\ nat),\ (sigma\ f\ n) < 1 \rightarrow retract\ f\ n.$

Lemma $sigma\_mult$:
$\forall\ (f\ :\ nat \rightarrow U)\ n\ c,\ retract\ f\ n \rightarrow (sigma\ (fun\ k \Rightarrow c \times (f\ k))\ n) == c \times (sigma\ f\ n).$
Hint $Resolve\ sigma\_mult.$

Lemma $sigma\_prod\_maj : \forall\ (f\ g\ :\ nat \rightarrow U)\ n,$
$(sigma\ (fun\ k \Rightarrow (f\ k) \times (g\ k))\ n) \leq (sigma\ f\ n).$

Hint $Resolve\ sigma\_prod\_maj.$

Lemma $sigma\_prod\_le : \forall\ (f\ g\ :\ nat \rightarrow U)\ (c{:}U),\ (\forall\ k,\ (f\ k) \leq c)$
$\rightarrow \forall\ n,\ (retract\ g\ n) \rightarrow (sigma\ (fun\ k \Rightarrow (f\ k) \times (g\ k))\ n) \leq c \times (sigma\ g\ n).$

Lemma $sigma\_prod\_ge : \forall\ (f\ g\ :\ nat \rightarrow U)\ (c{:}U),\ (\forall\ k,\ c \leq (f\ k))$
$\rightarrow \forall\ n,\ (retract\ g\ n) \rightarrow c \times (sigma\ g\ n) \leq (sigma\ (fun\ k \Rightarrow (f\ k) \times (g\ k))\ n).$

Hint $Resolve\ sigma\_prod\_maj\ sigma\_prod\_le\ sigma\_prod\_ge.$

Lemma $sigma\_inv : \forall\ (f\ g\ :\ nat \rightarrow U)\ (n{:}nat),\ (retract\ f\ n) \rightarrow$
$[1\text{-}]\ (sigma\ (fun\ k \Rightarrow f\ k \times g\ k)\ n) == (sigma\ (fun\ k \Rightarrow f\ k \times [1\text{-}]\ (g\ k))\ n) + [1\text{-}]\ (sigma\ f\ n).$

4.23 Product by an integer

4.23.1 Definition of *Nmult n x* written *n */ x*

Fixpoint *Nmult* (n: *nat*) (x : U) {*struct* n} : U :=
match n *with* $O \Rightarrow 0$ | (S O) $\Rightarrow x$ | S $p \Rightarrow x$ + (*Nmult* p x) *end.*

4.23.2 Condition for *n */ x* to be exact : $n = 0$ or $x \leq \frac{1}{n}$

Definition *Nmult_def* (n: *nat*) (x : U) :=
match n *with* $O \Rightarrow$ *True* | S $p \Rightarrow x \leq$ [1/]1+p *end.*

Lemma *Nmult_def_O* : ∀ x, *Nmult_def* O x.
Hint *Resolve Nmult_def_O.*

Lemma *Nmult_def_1* : ∀ x, *Nmult_def* (S O) x.
Hint *Resolve Nmult_def_1.*

Lemma *Nmult_def_intro* : ∀ n x , $x \leq$ [1/]1+n → *Nmult_def* (S n) x.
Hint *Resolve Nmult_def_intro.*

Lemma *Nmult_def_Unth*: ∀ n , *Nmult_def* (S n) ([1/]1+n).
Hint *Resolve Nmult_def_Unth.*

Lemma *Nmult_def_pred* : ∀ n x, *Nmult_def* (S n) x → *Nmult_def* n x.

Hint Immediate *Nmult_def_pred.*

Lemma *Nmult_defS* : ∀ n x, *Nmult_def* (S n) x → $x \leq$ [1/]1+n.
Hint Immediate *Nmult_defS.*

Lemma *Nmult_def_class* : ∀ n p, *class* (*Nmult_def* n p).
Hint *Resolve Nmult_def_class.*

Add Morphism Nmult_def with signature eq ==> Ueq ==> iff as Nmult_def_eq_compat.

Infix "*/" := *Nmult* (*at level* 60) : *U_scope.*

Lemma *Nmult_def_zero* : ∀ n, *Nmult_def* n 0.
Hint *Resolve Nmult_def_zero.*

4.23.3 Properties of *n */ x*

Lemma *Nmult_0* : ∀ (x:U), O*/x = 0.

Lemma *Nmult_1* : ∀ (x:U), (S O)*/x = x.

Lemma *Nmult_zero* : ∀ n, n */ 0 == 0.

Lemma *Nmult_SS* : ∀ (n:*nat*) (x:U), S (S n) */x = x + (S n */ x).

Lemma *Nmult_2* : ∀ (x:U), 2*/x = x + x.

Lemma *Nmult_S* : ∀ (n:*nat*) (x:U), S n */ x == x + (n*/x).

Hint *Resolve Nmult_1 Nmult_SS Nmult_2 Nmult_S.*

Add Morphism Nmult with signature eq ==> Ueq ==> Ueq as Nmult_eq_compat.
Hint *Resolve Nmult_eq_compat.*

Lemma *Nmult_eq_compat_right* : ∀ (n m:*nat*) (x:U), (n = m)%*nat* → n */ x == m */ x.
Hint *Resolve Nmult_eq_compat_right.*

Lemma *Nmult_le_compat_right* : ∀ n x y, $x \leq y$ → n */ $x \leq n$ */ y.

Lemma *Nmult_le_compat_left* : ∀ n m x, ($n \leq m$)%*nat* → n */ $x \leq m$ */ x.

Lemma *Nmult_sigma* : ∀ (n:*nat*) (x:U), n */ x == *sigma* (*fun* $k \Rightarrow x$) n.

Hint *Resolve Nmult_eq_compat_right Nmult_le_compat_right*

Nmult_le_compat_left Nmult_sigma.

Lemma *Nmult_Unth_prop* : ∀ n:nat, [1/]1+n == [1-] (n*/ ([1/]1+n)).
Hint *Resolve Nmult_Unth_prop.*

Lemma *Nmult_n_Unth*: ∀ n:nat, n */ [1/]1+n == [1-] ([1/]1+n).

Lemma *Nmult_Sn_Unth*: ∀ n:nat, S n */ [1/]1+n == 1.

Hint *Resolve Nmult_n_Unth Nmult_Sn_Unth.*

Lemma *Nmult_ge_Sn_Unth*: ∀ n k, (S n ≤ k)%nat → k */ [1/]1+n == 1.

Lemma *Nmult_le_n_Unth*: ∀ n k, (k ≤ n)%nat → k */ [1/]1+n ≤ [1-] ([1/]1+n).

Hint *Resolve Nmult_ge_Sn_Unth Nmult_le_n_Unth.*

Lemma *Nmult_Umult_assoc_left* : ∀ n x y, Nmult_def n x → n*/(x×y) == (n*/x)*y.

Hint *Resolve Nmult_Umult_assoc_left.*

Lemma *Nmult_Umult_assoc_right* : ∀ n x y, Nmult_def n y → n*/(x×y) == x*(n*/y).

Hint *Resolve Nmult_Umult_assoc_right.*

Lemma *plus_Nmult_distr* : ∀ n m x, (n + m) */ x== (n */ x) + (m */ x).

Lemma *Nmult_Uplus_distr* : ∀ n x y, n */ (x + y) == (n */ x) + (n */ y).

Lemma *Nmult_mult_assoc* : ∀ n m x, (n × m) */ x == n */ (m */ x).

Lemma *Nmult_Unth_simpl_left* : ∀ n x, (S n) */ ([1/]1+n × x) == x.

Lemma *Nmult_Unth_simpl_right* : ∀ n x, (S n) */ (x × [1/]1+n) == x.

Hint *Resolve Nmult_Unth_simpl_left Nmult_Unth_simpl_right.*

Lemma *Uinv_Nmult* : ∀ k n, [1-] (k */ [1/]1+n) == ((S n) - k) */ [1/]1+n.

Lemma *Nmult_neq_zero* : ∀ n x, ¬0==x → ¬0==S n */ x.
Hint *Resolve Nmult_neq_zero.*

Lemma *Nmult_le_simpl* : ∀ (n:nat) (x y:U),
  Nmult_def (S n) x → Nmult_def (S n) y → (S n */ x) ≤ (S n */ y) → x ≤ y.

Lemma *Nmult_Unth_le* : ∀ (n1 n2 m1 m2:nat),
  (n2 × S n1≤ m2 × S m1)%nat → n2 */ [1/]1+m1 ≤ m2 */ [1/]1+n1.

Lemma *Nmult_Unth_eq* :
  ∀ (n1 n2 m1 m2:nat),
  (n2 × S n1= m2 × S m1)%nat → n2 */ [1/]1+m1 == m2 */ [1/]1+n1.

Hint *Resolve Nmult_Unth_le Nmult_Unth_eq.*

Lemma *Nmult_def_lt* : ∀ n x, n */ x <1 → Nmult_def n x.

Hint Immediate *Nmult_def_lt.*

4.24 Conversion from booleans to U

Definition *B2U* (b:bool) :U := if b then 1 else 0.
Definition *NB2U* (b:bool) :U := if b then 0 else 1.

Lemma *B2Uinv* : feq NB2U (finv B2U).

Lemma *NB2Uinv* : feq B2U (finv NB2U).

Hint *Resolve B2Uinv NB2Uinv.*

4.25 Particular sequences

$pmin(p)(n) = p - \frac{1}{2^n}$

Definition *pmin* (p:U) (n:nat) := p - ([1/2]^n).

Add Morphism pmin with signature Ueq ==> eq ==> Ueq as pmin_eq_compat.

4.25.1 Properties of the invariant

Lemma *pmin_esp_S* : $\forall$ *p n*, *pmin* (*p* & *p*) *n* == *pmin p* (*S n*) & *pmin p* (*S n*).

Lemma *pmin_esp_le* : $\forall$ *p n*, *pmin p* (*S n*) $\leq$ [1/2] $\times$ (*pmin* (*p* & *p*) *n*) + [1/2].

Lemma *pmin_plus_eq* : $\forall$ *p n*, *p* $\leq$ [1/2] $\rightarrow$ *pmin p* (*S n*) == [1/2] $\times$ (*pmin* (*p* + *p*) *n*).

Lemma *pmin_0* : $\forall$ *p*:*U*, *pmin p O* == 0.

Lemma *pmin_le* : $\forall$ (*p*:*U*) (*n*:*nat*), *p* - ([1/]1+*n*) $\leq$ *pmin p n*.

Hint *Resolve pmin_0 pmin_le*.

Lemma *le_p_lim_pmin* : $\forall$ *p*, *p* $\leq$ *lub* (*pmin p*).

Lemma *le_lim_pmin_p* : $\forall$ *p*, *lub* (*pmin p*) $\leq$ *p*.
Hint *Resolve le_p_lim_pmin le_lim_pmin_p*.

Lemma *eq_lim_pmin_p* : $\forall$ *p*, *lub* (*pmin p*) == *p*.

Hint *Resolve eq_lim_pmin_p*.

Particular case where p = 1

Definition *U1min* := *pmin* 1.

Lemma *eq_lim_U1min* : *lub U1min* == 1.

Lemma *U1min_S* : $\forall$ *n*, *U1min* (*S n*) == [1/2]*(*U1min n*) + [1/2].

Lemma *U1min_0* : *U1min O* == 0.

Hint *Resolve eq_lim_U1min U1min_S U1min_0*.

4.26 Tactic for simplification of goals

Ltac *Usimpl* := *match goal with*
  $\vdash$ *context* [(Uplus 0 ?x)] $\Rightarrow$ *setoid_rewrite* (*Uplus_zero_left x*)
| $\vdash$ *context* [(Uplus ?x 0)] $\Rightarrow$ *setoid_rewrite* (*Uplus_zero_right x*)
| $\vdash$ *context* [(Uplus 1 ?x)] $\Rightarrow$ *setoid_rewrite* (*Uplus_one_left x*)
| $\vdash$ *context* [(Uplus ?x 1)] $\Rightarrow$ *setoid_rewrite* (*Uplus_one_right x*)
| $\vdash$ *context* [(Umult 0 ?x)] $\Rightarrow$ *setoid_rewrite* (*Umult_zero_left x*)
| $\vdash$ *context* [(Umult ?x 0)] $\Rightarrow$ *setoid_rewrite* (*Umult_zero_right x*)
| $\vdash$ *context* [(Umult 1 ?x)] $\Rightarrow$ *setoid_rewrite* (*Umult_one_left x*)
| $\vdash$ *context* [(Umult ?x 1)] $\Rightarrow$ *setoid_rewrite* (*Umult_one_right x*)
| $\vdash$ *context* [(Uesp 0 ?x)] $\Rightarrow$ *setoid_rewrite* (*Uesp_zero_left x*)
| $\vdash$ *context* [(Uesp ?x 0)] $\Rightarrow$ *setoid_rewrite* (*Uesp_zero_right x*)
| $\vdash$ *context* [(Uesp 1 ?x)] $\Rightarrow$ *setoid_rewrite* (*Uesp_one_left x*)
| $\vdash$ *context* [(Uesp ?x 1)] $\Rightarrow$ *setoid_rewrite* (*Uesp_one_right x*)
| $\vdash$ *context* [(Uminus 0 ?x)] $\Rightarrow$ *setoid_rewrite* (*Uminus_le_zero* 0 *x*);
  [apply (Upos x)| idtac]
| $\vdash$ *context* [(Uminus ?x 0)] $\Rightarrow$ *setoid_rewrite* (*Uminus_zero_right x*)
| $\vdash$ *context* [(Uminus ?x 1)] $\Rightarrow$ *setoid_rewrite* (*Uminus_le_zero x* 1);
  [apply (Unit x)| idtac]
| $\vdash$ *context* [([1-] ([1-] ?*x*))] $\Rightarrow$ *setoid_rewrite* (*Uinv_inv x*)
| $\vdash$ *context* [([1-] 1)] $\Rightarrow$ *setoid_rewrite Uinv_one*
| $\vdash$ *context* [([1-] 0)] $\Rightarrow$ *setoid_rewrite Uinv_zero*
| $\vdash$ *context* [([1/]1+*O*)] $\Rightarrow$ *setoid_rewrite Unth_zero*
| $\vdash$ *context* [?x^O] $\Rightarrow$ *setoid_rewrite* (*Uexp_0 x*)
| $\vdash$ *context* [?x^(S O)] $\Rightarrow$ *setoid_rewrite* (*Uexp_1 x*)
| $\vdash$ *context* [0^(?n)] $\Rightarrow$ *setoid_rewrite Uexp_zero*; [omega|idtac]
| $\vdash$ *context* [U1^(?n)] $\Rightarrow$ *setoid_rewrite Uexp_one*
| $\vdash$ *context* [(Nmult 0 ?x)] $\Rightarrow$ *setoid_rewrite* (*Nmult_0 x*)
| $\vdash$ *context* [(Nmult 1 ?x)] $\Rightarrow$ *setoid_rewrite* (*Nmult_1 x*)
| $\vdash$ *context* [(Nmult ?n 0)] $\Rightarrow$ *setoid_rewrite* (*Nmult_zero n*)
| $\vdash$ *context* [(sigma ?f O)] $\Rightarrow$ *setoid_rewrite* (*sigma_0 f*)

$\mid \vdash$ *context* [(sigma ?f (S O))] $\Rightarrow$ *setoid_rewrite* (*sigma_1 f*)
$\mid \vdash$ (*Ule* (*Uplus* ?*x* ?*y*) (*Uplus* ?*x* ?*z*)) $\Rightarrow$ *apply Uplus_le_compat_right*
$\mid \vdash$ (*Ule* (*Uplus* ?*x* ?*z*) (*Uplus* ?*y* ?*z*)) $\Rightarrow$ *apply Uplus_le_compat_left*
$\mid \vdash$ (*Ule* (*Uplus* ?*x* ?*z*) (*Uplus* ?*z* ?*y*)) $\Rightarrow$ *setoid_rewrite* (*Uplus_sym z y*);
apply Uplus_le_compat_left
$\mid \vdash$ (*Ule* (*Uplus* ?*x* ?*y*) (*Uplus* ?*z* ?*x*)) $\Rightarrow$ *setoid_rewrite* (*Uplus_sym x y*);
apply Uplus_le_compat_left
$\mid \vdash$ (*Ule* (*Uinv* ?*y*) (*Uinv* ?*x*)) $\Rightarrow$ *apply Uinv_le_compat*
$\mid \vdash$ (*Ule* (*Uminus* ?*x* ?*y*) (*Uplus* ?*x* ?*z*)) $\Rightarrow$ *apply Uminus_le_compat_right*
$\mid \vdash$ (*Ule* (*Uminus* ?*x* ?*z*) (*Uplus* ?*y* ?*z*)) $\Rightarrow$ *apply Uminus_le_compat_left*
$\mid \vdash$ (*Ueq* (*Uinv* ?*x*) (*Uinv* ?*y*)) $\Rightarrow$ *apply Uinv_eq_compat*
$\mid \vdash$ (*Ueq* (*Uplus* ?*x* ?*y*) (*Uplus* ?*x* ?*z*)) $\Rightarrow$ *apply Uplus_eq_compat_right*
$\mid \vdash$ (*Ueq* (*Uplus* ?*x* ?*z*) (*Uplus* ?*y* ?*z*)) $\Rightarrow$ *apply Uplus_eq_compat_left*
$\mid \vdash$ (*Ueq* (*Uplus* ?*x* ?*z*) (*Uplus* ?*z* ?*y*)) $\Rightarrow$ *setoid_rewrite* (*Uplus_sym z y*);
apply Uplus_eq_compat_left
$\mid \vdash$ (*Ueq* (*Uplus* ?*x* ?*y*) (*Uplus* ?*z* ?*x*)) $\Rightarrow$ *setoid_rewrite* (*Uplus_sym x y*);
apply Uplus_eq_compat_left
$\mid \vdash$ (*Ueq* (*Uminus* ?*x* ?*y*) (*Uplus* ?*x* ?*z*)) $\Rightarrow$ *apply Uminus_eq_compat*;[apply Ueq_refl|idtac]
$\mid \vdash$ (*Ueq* (*Uminus* ?*x* ?*z*) (*Uplus* ?*y* ?*z*)) $\Rightarrow$ *apply Uminus_eq_compat*;[idtac|apply Ueq_refl]
$\mid \vdash$ (*Ule* (*Umult* ?*x* ?*y*) (*Umult* ?*x* ?*z*)) $\Rightarrow$ *apply Umult_le_compat_right*
$\mid \vdash$ (*Ule* (*Umult* ?*x* ?*z*) (*Umult* ?*y* ?*z*)) $\Rightarrow$ *apply Umult_le_compat_left*
$\mid \vdash$ (*Ule* (*Umult* ?*x* ?*z*) (*Umult* ?*z* ?*y*)) $\Rightarrow$ *setoid_rewrite* (*Umult_sym z y*);
apply Umult_le_compat_left
$\mid \vdash$ (*Ule* (*Umult* ?*x* ?*y*) (*Umult* ?*z* ?*x*)) $\Rightarrow$ *setoid_rewrite* (*Umult_sym x y*);
apply Umult_le_compat_left
$\mid \vdash$ (*Ueq* (*Umult* ?*x* ?*y*) (*Umult* ?*x* ?*z*)) $\Rightarrow$ *apply Umult_eq_compat_right*
$\mid \vdash$ (*Ueq* (*Umult* ?*x* ?*z*) (*Umult* ?*y* ?*z*)) $\Rightarrow$ *apply Umult_eq_compat_left*
$\mid \vdash$ (*Ueq* (*Umult* ?*x* ?*z*) (*Umult* ?*z* ?*y*)) $\Rightarrow$ *setoid_rewrite* (*Umult_sym z y*);
apply Umult_eq_compat_left
$\mid \vdash$ (*Ueq* (*Umult* ?*x* ?*y*) (*Umult* ?*z* ?*x*)) $\Rightarrow$ *setoid_rewrite* (*Umult_sym x y*);
apply Umult_eq_compat_left
end.

4.27 Intervals

4.27.1 Definition

Record *IU* : *Type* := *mk_IU* {*low*:*U*; *up*:*U*; *proper*:*low* $\leq$ *up*}.

Hint *Resolve proper.*

the all set : [0,1]
Definition *full* := *mk_IU* (*Upos* 1).
singleton : [x]
Definition *singl* (x:U) := *mk_IU* (*Ule_refl* x).
down segment : [0,x]
Definition *inf* (x:U) := *mk_IU* (*Upos* x).
up segment : [x,1]
Definition *sup* (x:U) := *mk_IU* (*Unit* x).

4.27.2 Relations

Definition *Iin* (x:U) (I:*IU*) := *low* $I \leq x \land x \leq$ *up* I.

Definition *Iincl* I J := *low* $J \leq$ *low* $I \land$ *up* $I \leq$ *up* J.

Definition *Ieq* I J := *low* I == *low* $J \land$ *up* I == *up* J.
Hint *Unfold Iin Iincl Ieq.*

4.27.3 Properties

Lemma *Iin_low* : $\forall$ *I*, *Iin* (*low I*) *I*.

Lemma *Iin_up* : $\forall$ *I*, *Iin* (*up I*) *I*.

Hint *Resolve Iin_low Iin_up*.

Lemma *Iin_singl_elim* : $\forall$ *x y*, *Iin x* (*singl y*) $\rightarrow$ *x* == *y*.

Lemma *Iin_inf_elim* : $\forall$ *x y*, *Iin x* (*inf y*) $\rightarrow x \leq y$.

Lemma *Iin_sup_elim* : $\forall$ *x y*, *Iin x* (*sup y*) $\rightarrow y \leq x$.

Lemma *Iin_singl_intro* : $\forall$ *x y*, *x* == *y* $\rightarrow$ *Iin x* (*singl y*).

Lemma *Iin_inf_intro* : $\forall$ *x y*, $x \leq y \rightarrow$ *Iin x* (*inf y*).

Lemma *Iin_sup_intro* : $\forall$ *x y*, $y \leq x \rightarrow$ *Iin x* (*sup y*).

Hint Immediate *Iin_inf_elim Iin_sup_elim Iin_singl_elim*.
Hint *Resolve Iin_inf_intro Iin_sup_intro Iin_singl_intro*.

Lemma *Iin_class* : $\forall$ *I x*, *class* (*Iin x I*).

Lemma *Iincl_class* : $\forall$ *I J*, *class* (*Iincl I J*).

Lemma *Ieq_class* : $\forall$ *I J*, *class* (*Ieq I J*).
Hint *Resolve Iin_class Iincl_class Ieq_class*.

Lemma *Iincl_in* : $\forall$ *I J*, *Iincl I J* $\rightarrow \forall$ *x*, *Iin x I* $\rightarrow$ *Iin x J*.

Lemma *Iincl_low* : $\forall$ *I J*, *Iincl I J* $\rightarrow$ *low J* $\leq$ *low I*.

Lemma *Iincl_up* : $\forall$ *I J*, *Iincl I J* $\rightarrow$ *up I* $\leq$ *up J*.

Hint Immediate *Iincl_low Iincl_up*.

Lemma *Iincl_refl* : $\forall$ *I*, *Iincl I I*.
Hint *Resolve Iincl_refl*.

Lemma *Iincl_trans* : $\forall$ *I J K*, *Iincl I J* $\rightarrow$ *Iincl J K* $\rightarrow$ *Iincl I K*.

Lemma *Ieq_incl* : $\forall$ *I J*, *Ieq I J* $\rightarrow$ *Iincl I J*.

Lemma *Ieq_incl_sym* : $\forall$ *I J*, *Ieq I J* $\rightarrow$ *Iincl J I*.
Hint Immediate *Ieq_incl Ieq_incl_sym*.

Lemma *lincl_eq_compat* : $\forall$ *I J K L*,
  Ieq I J $\rightarrow$ *Iincl J K* $\rightarrow$ *Ieq K L* $\rightarrow$ *Iincl I L*.

Lemma *lincl_eq_trans* : $\forall$ *I J K*,
  Iincl I J $\rightarrow$ *Ieq J K* $\rightarrow$ *Iincl I K*.

Lemma *Ieq_incl_trans* : $\forall$ *I J K*,
  Ieq I J $\rightarrow$ *Iincl J K* $\rightarrow$ *Iincl I K*.

Lemma *Iincl_antisym* : $\forall$ *I J*, *Iincl I J* $\rightarrow$ *Iincl J I* $\rightarrow$ *Ieq I J*.
Hint Immediate *Iincl_antisym*.

Lemma *Ieq_refl* : $\forall$ *I*, *Ieq I I*.
Hint *Resolve Ieq_refl*.

Lemma *Ieq_sym* : $\forall$ *I J*, *Ieq I J* $\rightarrow$ *Ieq J I*.
Hint Immediate *Ieq_sym*.

Lemma *Ieq_trans* : $\forall$ *I J K*, *Ieq I J* $\rightarrow$ *Ieq J K* $\rightarrow$ *Ieq I K*.

Lemma *Isingl_eq* : $\forall$ *x y*, *Iincl* (*singl x*) (*singl y*) $\rightarrow$ *x*==*y*.
Hint Immediate *Isingl_eq*.

Lemma *Iincl_full* : $\forall$ *I*, *Iincl I full*.
Hint *Resolve Iincl_full*.

4.27.4 Operations on intervals

Definition *Iplus I J* := *mk_IU* (*Uplus_le_compat* (*proper I*) (*proper J*)).

Lemma *low_Iplus* : $\forall$ *I J*, *low* (*Iplus I J*)=*low I* + *low J*.

Lemma *up_Iplus* : $\forall$ *I J*, *up* (*Iplus I J*)=*up I* + *up J*.

Lemma *Iplus_in* : $\forall$ *I J x y*, *Iin x I* $\rightarrow$ *Iin y J* $\rightarrow$ *Iin* (*x*+*y*) (*Iplus I J*).

Lemma *lplus_in_elim* :
  $\forall$ *I J z*, *low I* $\leq$ [1-]*up J* $\rightarrow$ *Iin z* (*Iplus I J*)
    $\rightarrow$ *exc* (*fun x* $\Rightarrow$ *Iin x I* $\wedge$
      exc (*fun y* $\Rightarrow$ *Iin y J* $\wedge$ *z*==*x*+*y*)).

Definition *Imult I J* := *mk_IU* (*Umult_le_compat* (*proper I*) (*proper J*)).

Lemma *low_Imult* : $\forall$ *I J*, *low* (*Imult I J*) = *low I* $\times$ *low J*.

Lemma *up_Imult* : $\forall$ *I J*, *up* (*Imult I J*) = *up I* $\times$ *up J*.

Definition *Imultk p I* := *mk_IU* (*Umult_le_compat_right p* (*proper I*)).

Lemma *low_Imultk* : $\forall$ *p I*, *low* (*Imultk p I*) = *p* $\times$ *low I*.

Lemma *up_Imultk* : $\forall$ *p I*, *up* (*Imultk p I*) = *p* $\times$ *up I*.

Lemma *Imult_in* : $\forall$ *I J x y*, *Iin x I* $\rightarrow$ *Iin y J* $\rightarrow$ *Iin* (*x*$\times$*y*) (*Imult I J*).

Lemma *Imultk_in* : $\forall$ *p I x* , *Iin x I* $\rightarrow$ *Iin* (*p*$\times$*x*) (*Imultk p I*).

4.27.5 limits

Definition *lim* : $\forall$ *I*:*nat*$\rightarrow$*IU*, ($\forall$ *n*, *Iincl* (*I* (*S n*)) (*I n*)) $\rightarrow$ *IU*.

Lemma *low_lim* : $\forall$ (*I*:*nat*$\rightarrow$*IU*) (*Idec* : $\forall$ *n*, *Iincl* (*I* (*S n*)) (*I n*)),
  low (*lim I Idec*) = *lub* (*fun n* $\Rightarrow$ *low* (*I n*)).

Lemma *up_lim* : $\forall$ (*I*:*nat*$\rightarrow$*IU*) (*Idec* : $\forall$ *n*, *Iincl* (*I* (*S n*)) (*I n*)),
  up (*lim I Idec*) = *glb* (*fun n* $\Rightarrow$ *up* (*I n*)).

Lemma *lim_Iincl* : $\forall$ (*I*:*nat*$\rightarrow$*IU*) (*Idec* : $\forall$ *n*, *Iincl* (*I* (*S n*)) (*I n*)),
  $\forall$ *n*, *Iincl* (*lim I Idec*) (*I n*).
Hint *Resolve lim_Iincl*.

Lemma *Iincl_lim* : $\forall$ *J* (*I*:*nat*$\rightarrow$*IU*) (*Idec* : $\forall$ *n*, *Iincl* (*I* (*S n*)) (*I n*)),
  ($\forall$ *n*, *Iincl J* (*I n*)) $\rightarrow$ *Iincl J* (*lim I Idec*).

Lemma *Iim_incl_stable* : $\forall$ *I J* (*Idec* : $\forall$ *n*, *Iincl* (*I* (*S n*)) (*I n*))
  (*Jdec* : $\forall$ *n*, *Iincl* (*J* (*S n*)) (*J n*)),
  ($\forall$ *n*, *Iincl* (*I n*) (*J n*)) $\rightarrow$ *Iincl* (*lim I Idec*) (*lim J Jdec*).
Hint *Resolve Iim_incl_stable*.

4.27.6 Fixpoints

Section *Ifixpoint*.
Variable *A* : *Type*.
Variable *F* : (*A* $\rightarrow$ *IU*) $\rightarrow$ *A* $\rightarrow$ *IU*.
Hypothesis *Fmon* : $\forall$ *I J*, ($\forall$ *x*, *Iincl* (*I x*) (*J x*)) $\rightarrow$ $\forall$ *x*, *Iincl* (*F I x*) (*F J x*).

Fixpoint *Iiter* (*n*:*nat*) : *A* $\rightarrow$ *IU* :=
  match n with O $\Rightarrow$ *fun x* $\Rightarrow$ *full* | *S m* $\Rightarrow$ *F* (*Iiter m*) *end*.

Lemma *Iiter_decr* : $\forall$ *x n*, *Iincl* (*Iiter* (*S n*) *x*) (*Iiter n x*).
Hint *Resolve Iiter_decr*.

Definition *Ifix* (*x*:*A*) := *lim* (*fun n* $\Rightarrow$ *Iiter n x*) (*Iiter_decr x*).

Lemma *Iincl_fix* : $\forall$ (*x*:*A*), *Iincl* (*F Ifix x*) (*Ifix x*).

Lemma *Iincl_inv* : $\forall$ *f*, ($\forall$ *x*, *Iincl* (*f x*) (*F f x*)) $\rightarrow$ $\forall$ *x*, *Iincl* (*f x*) (*Ifix x*).

End *Ifixpoint*.
End *Univ_prop*.

5 Monads.v: Monads for randomized constructions

Require Export *Uprop*.

Module *Monad* (*Univ*:*Universe*).
Module *UP* := (*Univ_prop Univ*).

5.1 Definition of monadic operators

Definition M (A:*Type*) := $(A \to U) \to U$.

Definition *unit* (A:*Type*) (x:A) : $M\ A$:= *fun* $f \Rightarrow f\ x$.

Definition *star* ($A\ B$:*Type*) (a:$M\ A$) (F:$A \to M\ B$) : $M\ B$:= *fun* $f \Rightarrow a$ (*fun* $x \Rightarrow F\ x\ f$).

5.2 Properties of monadic operators

Lemma *law1* : $\forall$ ($A\ B$:*Type*) (x:A) (F:$A \to M\ B$) (f:$B \to U$), *star* (*unit* x) $F\ f = F\ x\ f$.

Lemma *law2* :
$\forall$ (A:*Type*) (a:$M\ A$) (f:$A \to U$), *star* a (*fun* x:$A \Rightarrow$ *unit* x) $f = a$ (*fun* x:$A \Rightarrow f\ x$).

Lemma *law3* :
$\forall$ ($A\ B\ C$:*Type*) (a:$M\ A$) (F:$A\to M\ B$) (G:$B\to M\ C$)
(f:$C\to U$), *star* (*star* $a\ F$) $G\ f$ = *star* a (*fun* x:$A \Rightarrow$ *star* ($F\ x$) G) f.

5.3 Properties of distributions

5.3.1 Expected properties of measures

Definition *monotonic* (A:*Type*) (m:$M\ A$) : *Prop* := $\forall\ f\ g$:$A\to U$, *fle* $f\ g \to (m\ f) \le (m\ g)$.

Definition *stable_eq* (A:*Type*) (m:$M\ A$) : *Prop* := $\forall\ f\ g$:$A\to U$, *feq* $f\ g \to (m\ f) == (m\ g)$.

Definition *stable_inv* (A:*Type*) (m:$M\ A$) : *Prop* := $\forall\ f$:$A\to U$, m (*finv* f) $\le$ *Uinv* ($m\ f$).

Definition *continuous* (A:*Type*) (m:$M\ A$) := $\forall$ *fn* : *nat* $\to A \to U$,
(*increase fn*) $\to m$ (*flub fn*) $\le$ *lub* (*fun* $n \Rightarrow m$ (*fn* n)).

Definition *fplusok* (A:*Type*) ($f\ g$: $A \to U$) := *fle* f (*finv* g).
Hint *Unfold fplusok*.

Lemma *fplusok_sym* : $\forall$ (A:*Type*) ($f\ g$: $A \to U$) , *fplusok* $f\ g \to$ *fplusok* $g\ f$.
Hint Immediate *fplusok_sym*.

Definition *stable_plus* (A:*Type*) (m:$M\ A$) : *Prop* :=
$\forall\ f\ g$:$A\to U$, *fplusok* $f\ g \to m$ (*fplus* $f\ g$) == ($m\ f$) + ($m\ g$).

Definition *le_plus* (A:*Type*) (m:$M\ A$) : *Prop* :=
$\forall\ f\ g$:$A\to U$, *fplusok* $f\ g \to (m\ f) + (m\ g) \le m$ (*fplus* $f\ g$).

Definition *le_esp* (A:*Type*) (m:$M\ A$) : *Prop* :=
$\forall\ f\ g$:$A\to U$, $(m\ f)$ & $(m\ g) \le m$ (*fesp* $f\ g$).

Definition *le_plus_cte* (A:*Type*) (m:$M\ A$) : *Prop* :=
$\forall$ (f :$A\to U$) (k:U), m (*fplus* f (*f_cte* $A\ k$)) $\le m\ f + k$.

Definition *stable_mult* (A:*Type*) (m:$M\ A$) : *Prop* :=
$\forall$ (k:U) (f:$A \to U$), m (*fmult* $k\ f$) == $k \times (m\ f)$.

5.3.2 Stability for equality

Lemma $monotonic\_stable\_eq : \forall\ (A{:}Type)\ (m{:}M\ A), (monotonic\ m) \rightarrow (stable\_eq\ m)$.
Hint *Resolve monotonic_stable_eq.*

Lemma $stable\_minus\_distr : \forall\ (A{:}Type)\ (m{:}M\ A)$,
$stable\_plus\ m \rightarrow stable\_inv\ m \rightarrow monotonic\ m \rightarrow$
$\forall\ (f\ g : A \rightarrow U), fle\ g\ f \rightarrow m\ (fminus\ f\ g) == m\ f - m\ g$.

Hint *Resolve stable_minus_distr.*

Lemma $inv\_minus\_distr : \forall\ (A{:}Type)\ (m{:}M\ A)$,
$stable\_plus\ m \rightarrow stable\_inv\ m \rightarrow monotonic\ m \rightarrow$
$\forall\ (f : A \rightarrow U), m\ (finv\ f) == m\ (f\_one\ A) - m\ f$.
Hint *Resolve inv_minus_distr.*

Lemma $le\_minus\_distr : \forall\ (A : Type)(m{:}M\ A)$,
$monotonic\ m \rightarrow \forall\ (f\ g{:}A \rightarrow U), m\ (fminus\ f\ g) \leq m\ f$.
Hint *Resolve le_minus_distr.*

Lemma $le\_plus\_distr : \forall\ (A : Type)(m{:}M\ A)$,
$stable\_plus\ m \rightarrow stable\_inv\ m \rightarrow monotonic\ m \rightarrow$
$\forall\ (f\ g{:}A \rightarrow U), m\ (fplus\ f\ g) \leq m\ f + m\ g$.
Hint *Resolve le_plus_distr.*

Lemma $le\_esp\_distr : \forall\ (A : Type)\ (m{:}M\ A)$,
$stable\_plus\ m \rightarrow stable\_inv\ m \rightarrow monotonic\ m \rightarrow le\_esp\ m$.

5.3.3 Monotonicity

Lemma $unit\_monotonic : \forall\ (A{:}Type)\ (x{:}A), monotonic\ (unit\ x)$.

Lemma $star\_monotonic : \forall\ (A\ B{:}Type)\ (m{:}M\ A)\ (F{:}A \rightarrow M\ B)$,
$monotonic\ m \rightarrow (\forall\ a{:}A, monotonic\ (F\ a)) \rightarrow monotonic\ (star\ m\ F)$.

Lemma $unit\_stable\_eq : \forall\ (A{:}Type)\ (x{:}A), stable\_eq\ (unit\ x)$.

Lemma $star\_stable\_eq : \forall\ (A\ B{:}Type)\ (m{:}M\ A)\ (F{:}A \rightarrow M\ B)$,
$stable\_eq\ m \rightarrow (\forall\ a{:}A, stable\_eq\ (F\ a)) \rightarrow stable\_eq\ (star\ m\ F)$.

5.3.4 Stability for inversion

Lemma $unit\_stable\_inv : \forall\ (A{:}Type)\ (x{:}A), stable\_inv\ (unit\ x)$.

Lemma $star\_stable\_inv : \forall\ (A\ B{:}Type)\ (m{:}M\ A)\ (F{:}A \rightarrow M\ B)$,
$stable\_inv\ m \rightarrow monotonic\ m$
$\rightarrow (\forall\ a{:}A, stable\_inv\ (F\ a)) \rightarrow (\forall\ a{:}A, monotonic\ (F\ a))$
$\rightarrow stable\_inv\ (star\ m\ F)$.

5.3.5 Stability for addition

Lemma $unit\_stable\_plus : \forall\ (A{:}Type)\ (x{:}A), stable\_plus\ (unit\ x)$.

Lemma $star\_stable\_plus : \forall\ (A\ B{:}Type)\ (m{:}M\ A)\ (F{:}A \rightarrow M\ B)$,
$stable\_plus\ m \rightarrow stable\_eq\ m \rightarrow$
$(\forall\ a{:}A, \forall\ f\ g, fplusok\ f\ g \rightarrow (F\ a\ f) \leq Uinv\ (F\ a\ g))$
$\rightarrow (\forall\ a{:}A, stable\_plus\ (F\ a)) \rightarrow stable\_plus\ (star\ m\ F)$.

Lemma $unit\_le\_plus : \forall\ (A{:}Type)\ (x{:}A), le\_plus\ (unit\ x)$.

Lemma $star\_le\_plus : \forall\ (A\ B{:}Type)\ (m{:}M\ A)\ (F{:}A \rightarrow M\ B)$,
$le\_plus\ m \rightarrow monotonic\ m \rightarrow$
$(\forall\ a{:}A, \forall\ f\ g, fplusok\ f\ g \rightarrow (F\ a\ f) \leq Uinv\ (F\ a\ g))$
$\rightarrow (\forall\ a{:}A, le\_plus\ (F\ a)) \rightarrow le\_plus\ (star\ m\ F)$.

5.3.6 Stability for product

Lemma *unit_stable_mult* : $\forall$ (A:*Type*) (x:A), *stable_mult* (*unit* x).

Lemma *star_stable_mult* : $\forall$ (A B:*Type*) (m:M A) (F:$A \to M$ B),
stable_mult $m \to$ *stable_eq* $m \to$ ($\forall$ a:A, *stable_mult* (F a)) $\to$ *stable_mult* (*star* m F).

5.3.7 Continuity

Lemma *unit_continuous* : $\forall$ (A:*Type*) (x:A), *continuous* (*unit* x).

Lemma *star_continuous* : $\forall$ (A B : *Type*) (m : M A)(F: $A \to M$ B),
monotonic $m \to$ *continuous* $m \to$
($\forall$ x, *continuous* (F x)) $\to$ ($\forall$ x, *monotonic* (F x)) $\to$ *continuous* (*star* m F).

End *Monad*.

6 Probas.v: The monad for distributions

Require Export *Uprop*.
Require Export *Monads*.
Module *Proba* (*Univ*:*Universe*).
Module *MP* := (*Monad* *Univ*).

6.1 Definition of distribution

Distributions are measure functions such that

- $\mu(1-f) \leq 1-\mu(f)$
- $f \leq 1-g \Rightarrow \mu(f+g) = \mu(f)+\mu(g)$
- $\mu(k \times f) = k \times \mu(f)$
- $f \leq g \Rightarrow \mu(f) \leq \mu(g)$

Record *distr* (A:*Type*) : *Type* :=
{*mu* : M A;
mu_stable_inv : *stable_inv* *mu*;
mu_stable_plus : *stable_plus* *mu*;
mu_stable_mult : *stable_mult* *mu*;
mu_monotonic : *monotonic* *mu*}.

Hint *Resolve* *mu_stable_plus* *mu_stable_inv* *mu_stable_mult* *mu_monotonic*.

6.2 Properties of measures

Lemma *mu_stable_eq* : $\forall$ (A : *Type*)(m: *distr* A), *stable_eq* (*mu* m).
Hint *Resolve* *mu_stable_eq*.
Implicit *Arguments* *mu_stable_eq* [A].

Lemma *mu_zero* : $\forall$ (A : *Type*)(m: *distr* A), *mu* m (*f_zero* A) == 0.
Hint *Resolve* *mu_zero*.

Lemma *mu_one_inv* : $\forall$ (A : *Type*)(m:*distr* A),
mu m (*f_one* A) == 1 $\to$ $\forall$ f, *mu* m (*finv* f) == [1-] (*mu* m f).
Hint *Resolve* *mu_one_inv*.

Lemma *mu_le_minus* : $\forall$ (A : *Type*)(m:*distr* A) (f g:$A \to U$),
mu m (*fminus* f g) $\leq$ *mu* m f.
Hint *Resolve* *mu_le_minus*.

Lemma *mu_le_plus* : $\forall$ (A : *Type*)(m:*distr* A) (f g:$A \to U$),

$mu\ m\ (fplus\ f\ g) \leq mu\ m\ f + mu\ m\ g.$
Hint *Resolve mu_le_plus.*

Lemma mu_cte : $\forall\ (A : Type)(m{:}(distr\ A))\ (c{:}U),$
$mu\ m\ (f\_cte\ A\ c) == c \times mu\ m\ (f\_one\ A).$
Hint *Resolve mu_cte.*

Lemma mu_cte_le : $\forall\ (A : Type)(m{:}(distr\ A))\ (c{:}U),$
$mu\ m\ (f\_cte\ A\ c) \leq c.$

Lemma mu_cte_eq : $\forall\ (A : Type)(m{:}(distr\ A))\ (c{:}U),$
$mu\ m\ (f\_one\ A) == 1 \rightarrow mu\ m\ (f\_cte\ A\ c) == c.$

Hint *Resolve mu_cte_le mu_cte_eq.*

Lemma $mu\_stable\_mult\_right$: $\forall\ (A : Type)(m{:}(distr\ A))\ (c{:}U)\ (f : A \rightarrow U),$
$mu\ m\ (fun\ x \Rightarrow (f\ x) \times c) == (mu\ m\ f) \times c.$

Lemma mu_stable_minus : $\forall\ (A{:}Type)\ (m{:}distr\ A)(f\ g : A \rightarrow U),$
$fle\ g\ f \rightarrow mu\ m\ (fun\ x \Rightarrow f\ x\ \text{-}\ g\ x) == mu\ m\ f\ \text{-}\ mu\ m\ g.$

Lemma mu_inv_minus :
$\forall\ (A{:}Type)\ (m{:}distr\ A)(f{:}\ A \rightarrow U),\ mu\ m\ (finv\ f) == mu\ m\ (f\_one\ A)\ \text{-}\ mu\ m\ f.$

Lemma $mu\_inv\_minus\_inv$: $\forall\ (A{:}Type)\ (m{:}distr\ A)(f{:}\ A \rightarrow U),$
$mu\ m\ (finv\ f) + [1\text{-}](mu\ m\ (f\_one\ A)) == [1\text{-}](mu\ m\ f).$

Lemma $mu\_le\_esp\_inv$: $\forall\ (A{:}Type)\ (m{:}distr\ A)(f\ g : A \rightarrow U),$
$([1\text{-}]mu\ m\ (finv\ f))\ \&\ mu\ m\ g \leq mu\ m\ (fesp\ f\ g).$
Hint *Resolve mu_le_esp_inv.*

Lemma $mu\_stable\_inv\_inv$: $\forall\ (A{:}Type)\ (m{:}distr\ A)(f : A \rightarrow U),$
$mu\ m\ f \leq [1\text{-}]\ mu\ m\ (finv\ f).$
Hint *Resolve mu_stable_inv_inv.*

Lemma mu_le_esp : $\forall\ (A{:}Type)\ (m{:}distr\ A)(f\ g : A \rightarrow U),$
$mu\ m\ f\ \&\ mu\ m\ g \leq mu\ m\ (fesp\ f\ g).$
Hint *Resolve mu_le_esp.*

6.3 Monadic operators for distributions

Definition $Munit$: $\forall\ A{:}Type,\ A \rightarrow distr\ A.$

Definition $Mlet$: $\forall\ A\ B{:}Type,\ (distr\ A) \rightarrow (A \rightarrow distr\ B) \rightarrow distr\ B.$

6.4 Operations on distributions

Definition $le\_distr\ (A{:}Type)\ (m1\ m2{:}distr\ A) := \forall\ f,\ mu\ m1\ f \leq mu\ m2\ f.$

Definition $eq\_distr\ (A{:}Type)\ (m1\ m2{:}distr\ A) := \forall\ f,\ mu\ m1\ f == mu\ m2\ f.$

Lemma $le\_distr\_antisym$: $\forall\ (A{:}Type)\ (m1\ m2{:}distr\ A),$
$le\_distr\ m1\ m2 \rightarrow le\_distr\ m2\ m1 \rightarrow eq\_distr\ m1\ m2.$

Lemma le_distr_refl : $\forall\ (A{:}Type)\ (m\ {:}distr\ A),\ le\_distr\ m\ m.$

Lemma eq_distr_sym : $\forall\ A\ (m1\ m2{:}distr\ A),\ eq\_distr\ m1\ m2 \rightarrow eq\_distr\ m2\ m1.$

Lemma eq_distr_refl : $\forall\ A\ (m{:}distr\ A),\ eq\_distr\ m\ m.$

Lemma eq_distr_trans : $\forall\ A\ (m1\ m2\ m3{:}distr\ A),$
$eq\_distr\ m1\ m2 \rightarrow eq\_distr\ m2\ m3 \rightarrow eq\_distr\ m1\ m3.$

Hint *Resolve eq_distr_refl.*
Hint Immediate *eq_distr_sym.*

Lemma $distr\_setoid$: $\forall\ (A{:}Type),\ Setoid\_Theory\ (distr\ A)\ (eq\_distr\ (A{:=}A)).$

Lemma le_distr_trans : $\forall\ (A{:}Type)\ (m1\ m2\ m3{:}distr\ A),$

le_distr m1 m2 → le_distr m2 m3 → le_distr m1 m3.

Hint *Resolve le_distr_refl.*
Hint *Unfold le_distr.*

Add Setoid distr eq_distr distr_setoid as Distr_Setoid.

Lemma *Munit_compat* : ∀ *A* (*x y* : *A*), *x*=*y* → *eq_distr* (*Munit x*) (*Munit y*).

Lemma *Mlet_compat* : ∀ (*A B* : *Type*) (*m1 m2*:*distr A*) (*M1 M2* : *A*→ *distr B*),
eq_distr m1 m2 → (∀ *x*, *eq_distr* (*M1 x*) (*M2 x*)) →
eq_distr (*Mlet m1 M1*) (*Mlet m2 M2*).

Lemma *Munit_eq* : ∀ (*A*:*Type*) (*q*:*A*→*U*) *x*, *mu* (*Munit x*) *q* == *q x*.

Lemma *le_distr_gen* : ∀ (*A*:*Type*) (*m1 m2*:*distr A*),
le_distr m1 m2 → ∀ *f g*, *fle f g* → *mu m1 f* ≤ *mu m2 g*.

6.5 Properties of monadic operators

Lemma *Mlet_unit* : ∀ (*A B*:*Type*) (*x*:*A*) (*m*:*A* → *distr B*), *eq_distr* (*Mlet* (*Munit x*) *m*) (*m x*).

Lemma *M_ext* : ∀ (*A*:*Type*) (*m*:*distr A*), *eq_distr* (*Mlet m* (*fun x* ⇒ (*Munit x*))) *m*.

Lemma *Mcomp* : ∀ (*A B C*:*Type*) (*m1*:(*distr A*)) (*m2*:*A* → *distr B*) (*m3*:*B* → *distr C*),
eq_distr (*Mlet* (*Mlet m1 m2*) *m3*) (*Mlet m1* (*fun x*:*A* ⇒ (*Mlet* (*m2 x*) *m3*))).

Lemma *Mlet_mon* : ∀ (*A B*:*Type*) (*m1 m2*: *distr A*) (*f1 f2* : *A* → *distr B*),
le_distr m1 m2 → (∀ *x*, *le_distr* (*f1 x*) (*f2 x*)) → *le_distr* (*Mlet m1 f1*) (*Mlet m2 f2*).

6.6 A specific distribution

Definition *distr_null* : ∀ *A* : *Type*, *distr A*.

Lemma *le_distr_null* : ∀ (*A*:*Type*) (*m* : *distr A*), *le_distr* (*distr_null A*) *m*.
Hint *Resolve le_distr_null.*

6.7 Least upper bound of increasing sequences of distributions

Section *Lubs.*
Variable *A* : *Type.*
Variable *muf* : *nat* → (*distr A*).
Hypothesis *muf_mon* : ∀ *n m*:*nat*, (*n* ≤ *m*)%*nat* → *le_distr* (*muf n*) (*muf m*).

Definition *mu_lub_* : *M A* := *fun f* ⇒ *lub* (*fun n* ⇒ *mu* (*muf n*) *f*).

Definition *mu_lub*: *distr A*.

Lemma *mu_lub_le* : ∀ *n*:*nat*, *le_distr* (*muf n*) *mu_lub*.

Lemma *mu_lub_sup* : ∀ *m*:(*distr A*), (∀ *n*:*nat*, *le_distr* (*muf n*) *m*) → *le_distr mu_lub m*.

End *Lubs.*

6.8 Distribution for *flip*

The distribution associated to *flip* () is $f \mapsto \frac{1}{2}f(true) + \frac{1}{2}f(false)$
Definition *flip* : (*M bool*) := *fun* (*f* : *bool* → *U*) ⇒ [1/2] × (*f true*) + [1/2] × (*f false*).

Lemma *flip_stable_inv* : *stable_inv flip.*

Lemma *flip_stable_plus* : *stable_plus flip.*

Lemma *flip_stable_mult* : *stable_mult flip.*

Lemma *flip_monotonic* : *monotonic flip.*

Definition *ctrue* (*b*:*bool*) := *if b then* 1 *else* 0.
Definition *cfalse* (*b*:*bool*) := *if b then* 0 *else* 1.

Lemma *flip_ctrue* : *flip ctrue* == [1/2].

Lemma *flip_cfalse* : *flip cfalse* == [1/2].

Hint *Resolve flip_ctrue flip_cfalse.*

Definition *Flip* : *distr bool.*

6.9 Uniform distribution beween 0 and n

Require *Arith.*

6.9.1 Definition of *fnth*

fnth n k is defined as $\frac{1}{n+1}$

Definition *fnth* (*n*:*nat*) : *nat* → *U* := *fun k* ⇒ ([1/]1+*n*).

6.9.2 Basic properties of *fnth*

Lemma *Unth_eq* : ∀ *n*, *Unth n* == [1-] (*sigma* (*fnth n*) *n*).
Hint *Resolve Unth_eq.*

Lemma *sigma_fnth_one* : ∀ *n*, *sigma* (*fnth n*) (*S n*) == 1.
Hint *Resolve sigma_fnth_one.*

Lemma *Unth_inv_eq* : ∀ *n*, [1-] ([1/]1+*n*) == *sigma* (*fnth n*) *n.*

Lemma *sigma_fnth_sup* : ∀ *n m*, (*m* > *n*) → *sigma* (*fnth n*) *m* == *sigma* (*fnth n*) (*S n*).

Lemma *sigma_fnth_le* : ∀ *n m*, (*sigma* (*fnth n*) *m*) ≤ (*sigma* (*fnth n*) (*S n*)).

Hint *Resolve sigma_fnth_le.*

fnth is a retract
Lemma *fnth_retract* : ∀ *n*:*nat*,(*retract* (*fnth n*) (*S n*)).

Implicit *Arguments fnth_retract* [].

6.9.3 Distribution for *random n*

The distribution associated to *random n* is $f \mapsto \Sigma_{i=0}^{n} \frac{f(i)}{n+1}$ we cannot factorize $\frac{1}{n+1}$ because of possible overflow

Definition *random* (*n*:*nat*):*M nat*:= *fun* (*f*:*nat*→*U*) ⇒ *sigma* (*fun k* ⇒ *Unth n* × *f k*) (*S n*).

6.9.4 Properties of *random*

Lemma *random_stable_inv* : ∀ *n*, *stable_inv* (*random n*).

Lemma *random_stable_plus* : ∀ *n*, *stable_plus* (*random n*).

Lemma *random_stable_mult* : ∀ *n*, *stable_mult* (*random n*).

Lemma *random_monotonic* : ∀ *n*, *monotonic* (*random n*).

Definition *Random* (*n*:*nat*) : (*distr nat*).

Lemma *random_total* : ∀ *n* : *nat*, *mu* (*Random n*) (*f_one nat*) == 1.

7 Nondeterministic choice

Record *Ndistr* (*A*:*Type*): *Type* :=
{*nu* : *M A*; *nu_monotonic* : *monotonic nu*; *nu_continuous* : *continuous nu*; *nu_le_esp*: *le_esp nu*}.

Hint *Resolve nu_monotonic nu_continuous nu_le_esp.*

Definition *Nunit* (*A*:*Type*) (*x*:*A*) : *Ndistr A*.

Definition *Nlet* (*A B*:*Type*)(*n*:*Ndistr A*) (*N*:*A*→*Ndistr B*): *Ndistr B*.

Definition *Nif* (*A*:*Type*) (*nb*: *Ndistr bool*) (*n1 n2* :*Ndistr A*): *Ndistr A* :=
Nlet nb (*fun b* ⇒ *if b then n1 else n2*).

Definition *Ndistr_cte* : ∀ *A*, ∀ *x*:*U*, *Ndistr A*.

Definition *Nmin* : ∀ *A*, *Ndistr A* → *Ndistr A* → *Ndistr A*.

Lemma *Ndistr_eq_esp* : ∀ (*A*:*Type*) (*n*:*Ndistr A*) *f g*, 1 ≤ *nu n f* → *nu n g* == *nu n* (*fesp f g*).
Hint *Resolve Ndistr_eq_esp.*

Definition *le_ndistr* (*A*:*Type*)(*m1 m2* : *Ndistr A*) := ∀ *f*, *nu m1 f* ≤ *nu m2 f*.

End *Proba.*

8 Prog.v: Composition of distributions

Require Export *Probas.*

Module *Rules* (*Univ*:*Universe*).
Module *PP* := (*Proba Univ*).

8.1 Conditional

Definition *Mif* (*A*:*Type*) (*b*:*distr bool*) (*m1 m2*: *distr A*)
:= *Mlet b* (*fun x*:*bool* ⇒ *if x then m1 else m2*).

Lemma *Mif_mon* : ∀ (*A*:*Type*) (*b b'*:*distr bool*) (*m1 m2 m1' m2'*: *distr A*),
le_distr b b' → *le_distr m1 m1'* → *le_distr m2 m2'*
→ *le_distr* (*Mif b m1 m2*) (*Mif b' m1' m2'*).

8.2 Fixpoints

Section *Fixpoints.*

8.2.1 Hypotheses

Variables *A B* : *Type.*

Variable *F* : (*A* → *distr B*) → *A* → *distr B*.

Hypothesis *F_mon* : ∀ *f g* : *A* → *distr B*,
(∀ *x*, *le_distr* (*f x*) (*g x*)) → ∀ *x*, *le_distr* (*F f x*) (*F g x*).

8.2.2 Iteration of the functional F from the 0-distribution

Fixpoint *iter* (*n*:*nat*) : *A* → (*distr B*)
:= *match n with* | *O* ⇒ *fun x* ⇒ (*distr_null B*)
| *S n* ⇒ *fun x* ⇒ *F* (*iter n*) *x*
end.

Definition *Flift* (*dn*:*A*→*nat*→*distr B*)(*x*:*A*)(*n*:*nat*):(*distr B*)
:= *F* (*fun y* ⇒ *dn y n*) *x*.

Lemma *Flift_mon* : ∀ *dn* : *A* → *nat* → *distr B*,

$(\forall\ (x{:}A)\ (n\ m{:}nat),\ (n \leq m)\%nat \rightarrow le\_distr\ (dn\ x\ n)\ (dn\ x\ m))$
$\rightarrow \forall\ (x{:}A)\ (n\ m{:}nat),\ (n \leq m)\%nat \rightarrow le\_distr\ (Flift\ dn\ x\ n)\ (Flift\ dn\ x\ m).$

Hypothesis $F\_continuous : \forall\ (dn{:}A{\rightarrow}nat{\rightarrow}distr\ B)$
$(dnmon : \forall\ x\ n\ m,(n \leq m)\%nat \rightarrow le\_distr\ (dn\ x\ n)\ (dn\ x\ m))$
$(x{:}A),$
$(le\_distr\ (F\ (fun\ y \Rightarrow mu\_lub\ (dn\ y)\ (dnmon\ y))\ x)$
$(mu\_lub\ (Flift\ dn\ x)\ (Flift\_mon\ dn\ dnmon\ x))).$

Let $muf\ (x{:}A)\ (n{:}nat) := (iter\ n\ x).$

Lemma $muf\_mon\_succ : \forall\ (n{:}nat)\ (x{:}A),\ le\_distr\ (muf\ x\ n)\ (muf\ x\ (S\ n)).$

Lemma $muf\_mon : \forall\ (x{:}A)\ (n\ m{:}nat),\ (n \leq m)\%nat \rightarrow le\_distr\ (muf\ x\ n)\ (muf\ x\ m).$

8.2.3 Definition

Definition $Mfix\ (x{:}A) := mu\_lub\ (fun\ n \Rightarrow iter\ n\ x)\ (muf\_mon\ x).$

8.2.4 Properties

Lemma $Mfix\_le\_iter : \forall\ (x : A)\ (n{:}nat),\ le\_distr\ (iter\ n\ x)\ (Mfix\ x).$
Hint *Resolve Mfix_le_iter.*

Lemma $Mfix\_iter\_le : \forall\ (m{:}A{\rightarrow}distr\ B),$
$(\forall\ (x : A)\ (n{:}nat),\ le\_distr\ (iter\ n\ x)\ (m\ x)) \rightarrow \forall\ x,\ le\_distr\ (Mfix\ x)\ (m\ x).$
Hint *Resolve Mfix_iter_le.*

Lemma $Mfix\_le : \forall\ x : A,\ le\_distr\ (Mfix\ x)\ (F\ Mfix\ x).$

Lemma $Mfix\_sup : \forall\ x : A,\ le\_distr\ (F\ Mfix\ x)\ (Mfix\ x).$

Lemma $Mfix\_eq : \forall\ x : A,\ eq\_distr\ (Mfix\ x)\ (F\ Mfix\ x).$

End *Fixpoints.*

Lemma $Mfix\_le\_stable : \forall\ (A\ B{:}Type)\ F\ G$
$(Fmon{:}\forall\ f\ g : A \rightarrow distr\ B,\ (\forall\ x : A,\ le\_distr\ (f\ x)\ (g\ x)) \rightarrow$
$\forall\ x : A,\ le\_distr\ (F\ f\ x)\ (F\ g\ x))$
$(Gmon{:}\forall\ f\ g : A \rightarrow distr\ B,\ (\forall\ x : A,\ le\_distr\ (f\ x)\ (g\ x)) \rightarrow$
$\forall\ x : A,\ le\_distr\ (G\ f\ x)\ (G\ g\ x)),$
$(\forall\ f\ x,\ le\_distr\ (F\ f\ x)\ (G\ f\ x)) \rightarrow \forall\ x,\ le\_distr\ (Mfix\ F\ Fmon\ x)\ (Mfix\ G\ Gmon\ x).$

8.3 Continuity

Section *Continuity.*

Variables $A\ B{:}Type.$
Variable $mun : nat \rightarrow distr\ A.$
Hypothesis $mun\_incr : \forall\ n\ m,\ ((n \leq m)\%nat) \rightarrow le\_distr\ (mun\ n)\ (mun\ m).$
Hypothesis $mun\_cont : \forall\ n,\ continuous\ (mu\ (mun\ n)).$
Variable $Mn : A \rightarrow nat \rightarrow distr\ B.$
Hypothesis $Mn\_incr : \forall\ x\ n\ m,\ ((n \leq m)\%nat) \rightarrow le\_distr\ (Mn\ x\ n)\ (Mn\ x\ m).$
Lemma $Mlet\_incr :$
$\forall\ n\ m,(n \leq m)\%nat \rightarrow$
$le\_distr\ (Mlet\ (mun\ n)\ (fun\ x \Rightarrow Mn\ x\ n))\ (Mlet\ (mun\ m)\ (fun\ x \Rightarrow Mn\ x\ m)).$

Lemma $Mlet\_continuous :$
$le\_distr\ (Mlet\ (mu\_lub\ mun\ mun\_incr)\ (fun\ x \Rightarrow mu\_lub\ (Mn\ x)\ (Mn\_incr\ x)))$
$(mu\_lub\ (fun\ n \Rightarrow Mlet\ (mun\ n)\ (fun\ x \Rightarrow Mn\ x\ n))\ Mlet\_incr).$

Variable $Fn : nat \rightarrow (A \rightarrow distr\ B) \rightarrow A \rightarrow distr\ B.$

Hypothesis $Fn\_mon : \forall\ n\ (f\ g : A \rightarrow distr\ B),$
$(\forall\ x,\ le\_distr\ (f\ x)\ (g\ x)) \rightarrow \forall\ x,\ le\_distr\ (Fn\ n\ f\ x)\ (Fn\ n\ g\ x).$

Hypothesis Fn_incr : $\forall\ f\ x\ n\ m,\ (n \leq m)\%nat \rightarrow le\_distr\ (Fn\ n\ f\ x)\ (Fn\ m\ f\ x)$.

Hypothesis $Fn\_continuous$: $\forall\ n\ (dn{:}A{\rightarrow}nat{\rightarrow}distr\ B)$
$(dnmon : \forall\ x\ n\ m,(n \leq m)\%nat \rightarrow le\_distr\ (dn\ x\ n)\ (dn\ x\ m))$
$(x{:}A)$,
$(le\_distr\ (Fn\ n\ (fun\ y \Rightarrow mu\_lub\ (dn\ y)\ (dnmon\ y))\ x)$
$(mu\_lub\ (Flift\ (Fn\ n)\ dn\ x)\ (Flift\_mon\ (Fn\ n)\ (Fn\_mon\ n)\ dn\ dnmon\ x)))$.

Lemma $Mfix\_incr$: $\forall\ x$,
$\forall\ n\ m,(n \leq m)\%nat \rightarrow$
$le\_distr\ (Mfix\ (Fn\ n)\ (Fn\_mon\ n)\ x)\ (Mfix\ (Fn\ m)\ (Fn\_mon\ m)\ x)$.

Lemma mu_lub_mon :
$\forall\ (f\ g : A \rightarrow distr\ B),\ (\forall\ x,\ le\_distr\ (f\ x)\ (g\ x))$
$\rightarrow \forall\ x,\ le\_distr\ (mu\_lub\ (fun\ n \Rightarrow Fn\ n\ f\ x)\ (Fn\_incr\ f\ x))$
$(mu\_lub\ (fun\ n \Rightarrow Fn\ n\ g\ x)\ (Fn\_incr\ g\ x))$.

Lemma $iter\_incr$: $\forall\ k\ x$,
$\forall\ n\ m,(n \leq m)\%nat \rightarrow$
$le\_distr\ (iter\ (Fn\ n)\ k\ x)\ (iter\ (Fn\ m)\ k\ x)$.
Hint $Resolve\ iter\_incr$.

Lemma $iter\_continuous$:
$\forall\ k\ x,\ le\_distr\ (iter\ (fun\ f\ x \Rightarrow mu\_lub\ (fun\ n \Rightarrow Fn\ n\ f\ x)\ (Fn\_incr\ f\ x))\ k\ x)$
$(mu\_lub\ (fun\ n \Rightarrow iter\ (Fn\ n)\ k\ x)\ (iter\_incr\ k\ x))$.
Hint $Resolve\ iter\_continuous$.

Lemma $MFix\_continuous$:
$\forall\ x$,
$le\_distr\ (Mfix\ (fun\ f\ x \Rightarrow mu\_lub\ (fun\ n \Rightarrow Fn\ n\ f\ x)\ (Fn\_incr\ f\ x))\ mu\_lub\_mon\ x)$
$(mu\_lub\ (fun\ n \Rightarrow Mfix\ (Fn\ n)\ (Fn\_mon\ n)\ x)\ (Mfix\_incr\ x))$.

End $Continuity$.

9 Prog.v: Axiomatic semantics

9.1 Definition of correctness judgements

$p \leq \langle e \rangle(q)$ is defined as $p \leq \mu(e)(q)$ $\langle e \rangle(q) \leq p$ is defined as $\mu(e)(q) \leq p$

Definition $ok\ (A{:}Type)\ (p{:}U)\ (e{:}distr\ A)\ (q{:}A{\rightarrow}U) := p \leq mu\ e\ q$.

Definition $okfun\ (A\ B{:}Type)(p{:}A{\rightarrow}U)(e{:}A{\rightarrow}distr\ B)(q{:}A{\rightarrow}B{\rightarrow}U)$
$:= \forall\ x{:}A,\ ok\ (p\ x)\ (e\ x)\ (q\ x)$.

Definition $okup\ (A{:}Type)\ (p{:}U)\ (e{:}distr\ A)\ (q{:}A{\rightarrow}U) := mu\ e\ q \leq p$.

Definition $upfun\ (A\ B{:}Type)(p{:}A{\rightarrow}U)(e{:}A{\rightarrow}distr\ B)(q{:}A{\rightarrow}B{\rightarrow}U)$
$:= \forall\ x{:}A,\ okup\ (p\ x)\ (e\ x)\ (q\ x)$.

9.2 Stability properties

Lemma ok_le_compat : $\forall\ (A{:}Type)\ (p\ p'{:}U)\ (e{:}distr\ A)\ (q\ q'{:}A{\rightarrow}U)$,
$p' \leq p \rightarrow fle\ q\ q' \rightarrow ok\ p\ e\ q \rightarrow ok\ p'\ e\ q'$.

Lemma ok_eq_compat : $\forall\ (A{:}Type)\ (p\ p'{:}U)\ (e\ e'{:}distr\ A)\ (q\ q'{:}A{\rightarrow}U)$,
$p' == p \rightarrow (feq\ q\ q') \rightarrow eq\_distr\ e\ e' \rightarrow ok\ p\ e\ q \rightarrow ok\ p'\ e'\ q'$.

Lemma $okfun\_le\_compat$: $\forall\ (A\ B{:}Type)\ (p\ p'{:}A \rightarrow U)\ (e{:}A \rightarrow distr\ B)\ (q\ q'{:}A{\rightarrow}B{\rightarrow}U)$,
$fle\ p'\ p \rightarrow (\forall\ x, fle\ (q\ x)\ (q'\ x)) \rightarrow okfun\ p\ e\ q \rightarrow okfun\ p'\ e\ q'$.

Lemma ok_mult : $\forall\ (A{:}Type)(k\ p{:}U)(e{:}distr\ A)(f : A \rightarrow U),\ ok\ p\ e\ f \rightarrow ok\ (k{\times}p)\ e\ (fmult\ k\ f)$.

Lemma ok_inv : $\forall\ (A{:}Type)(p{:}U)(e{:}distr\ A)(f : A \rightarrow U),\ ok\ p\ e\ f \rightarrow mu\ e\ (finv\ f) \leq [1\text{-}]p$.

Lemma *okup_le_compat* : ∀ (*A*:*Type*) (*p p'*:*U*) (*e*:*distr A*) (*q q'*:*A*→*U*),
p ≤ *p'* → *fle q' q* → *okup p e q* → *okup p' e q'*.

Lemma *okup_eq_compat* : ∀ (*A*:*Type*) (*p p'*:*U*) (*e e'*:*distr A*) (*q q'*:*A*→*U*),
p == *p'* → (*feq q q'*) → *eq_distr e e'* → *okup p e q* → *okup p' e' q'*.

Lemma *upfun_le_compat* : ∀ (*A B*:*Type*) (*p p'*:*A* → *U*) (*e*:*A* → *distr B*) (*q q'*:*A*→*B*→*U*),
fle p p' → (∀ *x*,*fle* (*q' x*) (*q x*)) → *upfun p e q* → *upfun p' e q'*.

Lemma *okup_mult* : ∀ (*A*:*Type*)(*k p*:*U*)(*e*:*distr A*)(*f* : *A* → *U*), *okup p e f* → *okup* (*k*×*p*) *e* (*fmult k f*).

9.3 Basic rules

9.3.1 Rules for application

$$\frac{r \le \langle a\rangle(p) \quad \forall x, p(x) \le \langle f(x)\rangle(q)}{r \le \langle f(a)\rangle(q)} \qquad \frac{\langle a\rangle(p) \le r \quad \forall x, \langle f(x)\rangle(q) \le p(x)}{\langle f(a)\rangle(q) \le r}$$

Lemma *apply_rule* : ∀ (*A B*:*Type*)(*a*:(*distr A*))(*f*:*A*→*distr B*)(*r*:*U*)(*p*:*A*→*U*)(*q*:*B*→*U*),
(*ok r a p*) → (*okfun p f* (*fun x* ⇒ *q*)) → *ok r* (*Mlet a f*) *q*.

Lemma *okup_apply_rule* : ∀ (*A B*:*Type*)(*a*:*distr A*)(*f*:*A*→*distr B*)(*r*:*U*)(*p*:*A*→*U*)(*q*:*B*→*U*),
(*okup r a p*) → (*upfun p f* (*fun x* ⇒ *q*)) → *okup r* (*Mlet a f*) *q*.

9.3.2 Rules for abstraction

Lemma *lambda_rule* : ∀ (*A B*:*Type*)(*f*:*A*→*distr B*)(*p*:*A*→*U*)(*q*:*A* → *B*→*U*),
(∀ *x*:*A*, *ok* (*p x*) (*f x*) (*q x*)) → *okfun p f q*.

Lemma *okup_lambda_rule* : ∀ (*A B*:*Type*)(*f*:*A*→*distr B*)(*p*:*A*→*U*)(*q*:*A* → *B*→*U*),
(∀ *x*:*A*, *okup* (*p x*) (*f x*) (*q x*)) → *upfun p f q*.

9.3.3 Rule for conditional

$$\frac{p_1 \le \langle e_1\rangle(q) \quad p_2 \le \langle e_2\rangle(q)}{p_1 \times \mu(b)(1_{true}) + p_2 \times \mu(b)(1_{false}) \le \langle if\ b\ then\ e_1\ else\ e_2\rangle(q)}$$

$$\frac{\langle e_1\rangle(q) \le p_1 \quad \langle e_2\rangle(q) \le p_2}{\langle if\ b\ then\ e_1\ else\ e_2\rangle(q) \le p_1 \times \mu(b)(1_{true}) + p_2 \times \mu(b)(1_{false})}$$

Lemma *combiok* : ∀ (*A*:*Type*) *p q* (*f1 f2* : *A* → *U*), *p* ≤ [1-]*q* → *fplusok* (*fmult p f1*) (*fmult q f2*).
Hint *Resolve combiok*.

Lemma *fmult_fplusok* : ∀ (*A*:*Type*) *p q* (*f1 f2* : *A* → *U*), *fplusok f1 f2* → *fplusok* (*fmult p f1*) (*fmult q f2*).
Hint *Resolve fmult_fplusok*.

Lemma *ifok* : ∀ *f1 f2*, *fplusok* (*fmult f1 ctrue*) (*fmult f2 cfalse*).
Hint *Resolve ifok*.

Lemma *Mif_eq* : ∀ (*A*:*Type*)(*b*:(*distr bool*))(*f1 f2*:*distr A*)(*q*:*A*→*U*),
(*mu* (*Mif b f1 f2*) *q*) == (*mu f1 q*) × (*mu b ctrue*) + (*mu f2 q*) × (*mu b cfalse*).

Lemma *ifrule* :
∀ (*A*:*Type*)(*b*:(*distr bool*))(*f1 f2*:*distr A*)(*p1 p2*:*U*)(*q*:*A*→*U*),
ok p1 f1 q → *ok p2 f2 q*
→ *ok* (*p1* × (*mu b ctrue*) + *p2* × (*mu b cfalse*)) (*Mif b f1 f2*) *q*.

Lemma *okup_ifrule* :
∀ (*A*:*Type*)(*b*:(*distr bool*))(*f1 f2*:*distr A*)(*p1 p2*:*U*)(*q*:*A*→*U*),
okup p1 f1 q → *okup p2 f2 q*
→ *okup* (*p1* × (*mu b ctrue*) + *p2* × (*mu b cfalse*)) (*Mif b f1 f2*) *q*.

9.3.4 Rule for fixpoints

with $\phi(x) = F(\phi)(x)$, p_i an increasing sequence of functions starting from 0

$$\frac{\forall f\ i, (\forall x, p_i(x) \leq \langle f\rangle(q)) \Rightarrow \forall x, p_{i+1}(x) \leq \langle F(f)(x)\rangle(q)}{\forall x, \bigcup_i p_i\ x \leq \langle \phi(x)\rangle(q)}$$

Section *Fixrule.*
Variables $A\ B$: *Type.*

Variable $F : (A \rightarrow distr\ B) \rightarrow A \rightarrow distr\ B.$

Hypothesis $F\_mon : \forall\ f\ g\ :\ A \rightarrow (distr\ B),$
$(\forall\ x,\ le\_distr\ (f\ x)\ (g\ x)) \rightarrow \forall\ x,\ le\_distr\ (F\ f\ x)\ (F\ g\ x).$

Section *Ruleseq.*
Variable $q\ :\ A \rightarrow B \rightarrow U.$
Variable $p\ :\ A \rightarrow nat \rightarrow U.$

Lemma *fixrule* :
$(\forall\ x{:}A,\ p\ x\ O == 0)$->
$(\forall\ (i{:}nat)\ (f{:}A{\rightarrow}distr\ B),$
$(okfun\ (fun\ x \Rightarrow p\ x\ i)\ f\ q) \rightarrow okfun\ (fun\ x \Rightarrow p\ x\ (S\ i))\ (fun\ x \Rightarrow F\ f\ x)\ q)$
$\rightarrow okfun\ (fun\ x \Rightarrow lub\ (p\ x))\ (Mfix\ F\ F\_mon)\ q.$

Lemma *fixrule_up_lub* :
$(\forall\ (i{:}nat)\ (f{:}A{\rightarrow}distr\ B),$
$(upfun\ (fun\ x \Rightarrow p\ x\ i)\ f\ q) \rightarrow upfun\ (fun\ x \Rightarrow p\ x\ (S\ i))\ (fun\ x \Rightarrow F\ f\ x)\ q)$
$\rightarrow upfun\ (fun\ x \Rightarrow lub\ (p\ x))\ (Mfix\ F\ F\_mon)\ q.$

Lemma *okup_fixrule_glb* :
$(\forall\ (x{:}A)\ n,\ p\ x\ (S\ n) \leq p\ x\ n)$->
$(\forall\ (i{:}nat)\ (f{:}A{\rightarrow}distr\ B),$
$(upfun\ (fun\ x \Rightarrow p\ x\ i)\ f\ q) \rightarrow upfun\ (fun\ x \Rightarrow p\ x\ (S\ i))\ (fun\ x \Rightarrow F\ f\ x)\ q)$
$\rightarrow upfun\ (fun\ x \Rightarrow glb\ (p\ x))\ (Mfix\ F\ F\_mon)\ q.$
End *Ruleseq.*

Lemma *okup_fixrule_inv* :
$\forall\ (q\ :\ A \rightarrow B \rightarrow U)\ (p\ :\ A \rightarrow U),$
$(\forall\ (f{:}A{\rightarrow}distr\ B),\ upfun\ p\ f\ q \rightarrow upfun\ p\ (fun\ x \Rightarrow F\ f\ x)\ q)$
$\rightarrow upfun\ p\ (Mfix\ F\ F\_mon)\ q.$

9.3.5 Rules using commutation properties

Section *TransformFix.*

Section *Fix_muF.*
Variable $q\ :\ A \rightarrow B \rightarrow U.$
Variable $muF : (A \rightarrow U) \rightarrow A \rightarrow U.$
Hypothesis muF_mon : *Fmonotonic muF.*

Lemma muF_stable : *Fstable muF.*

Definition $mu\_muF\_commute\_le :=$
$\forall\ f\ x,\ (\forall\ y,\ le\_distr\ (f\ y)\ (Mfix\ F\ F\_mon\ y)) \rightarrow$
$mu\ (F\ f\ x)\ (q\ x) \leq muF\ (fun\ y \Rightarrow mu\ (f\ y)\ (q\ y))\ x.$
Hint *Unfold mu_muF_commute_le.*

Section *F_muF_results.*
Hypothesis F_muF_le : $mu\_muF\_commute\_le.$

Lemma $mu\_mufix\_le : \forall\ x,\ mu\ (Mfix\ F\ F\_mon\ x)\ (q\ x) \leq mufix\ muF\ x.$
Hint *Resolve mu_mufix_le.*

Lemma $muF\_le : \forall\ f,\ (fle\ (muF\ f)\ f)$
$\rightarrow \forall\ x,\ mu\ (Mfix\ F\ F\_mon\ x)\ (q\ x) \leq f\ x.$

Hypothesis muF_F_le :

$\forall$ *f x*, ($\forall$ *y*, *le_distr* (*f y*) (*Mfix F F_mon y*)) $\rightarrow$
muF (*fun y* $\Rightarrow$ *mu* (*f y*) (*q y*)) *x* $\leq$ *mu* (*F f x*) (*q x*).

Lemma *mufix_mu_le* : $\forall$ *x*, *mufix muF x* $\leq$ *mu* (*Mfix F F_mon x*) (*q x*).

End *F_muF_results*.
Hint *Resolve mu_mufix_le mufix_mu_le*.

Lemma *mufix_mu* :
($\forall$ *f x*, ($\forall$ *y*, *le_distr* (*f y*) (*Mfix F F_mon y*))
$\rightarrow$ *mu* (*F f x*) (*q x*) == *muF* (*fun y* $\Rightarrow$ *mu* (*f y*) (*q y*)) *x*)
$\rightarrow$ $\forall$ *x*, *mufix muF x* == *mu* (*Mfix F F_mon x*) (*q x*).
Hint *Resolve mufix_mu*.
End *Fix_muF*.

Section *Fix_Term*.

Definition *pterm* (*x*:*A*) := *mu* (*Mfix F F_mon x*) (*f_one B*).
Variable *muFone* : ($A \rightarrow U$) $\rightarrow A \rightarrow U$.

Hypothesis *muF_mon* : *Fmonotonic muFone*.

Hypothesis *F_muF_eq_one* :
$\forall$ *f x*, ($\forall$ *y*, *le_distr* (*f y*) (*Mfix F F_mon y*))
$\rightarrow$ *mu* (*F f x*) (*f_one B*) == *muFone* (*fun y* $\Rightarrow$ *mu* (*f y*) (*f_one B*)) *x*.

Hypothesis *muF_cont* : *Fcontlub muFone*.

Lemma *muF_pterm* : *feq pterm* (*muFone pterm*).
Hint *Resolve muF_pterm*.
End *Fix_Term*.

Section *Fix_muF_Term*.

Variable *q* : $A \rightarrow B \rightarrow U$.
Definition *qinv x y* := [1-]*q x y*.

Variable *muFqinv* : ($A \rightarrow U$) $\rightarrow A \rightarrow U$.

Hypothesis *muF_mon_inv* : *Fmonotonic muFqinv*.

Hypothesis *F_muF_le_inv* : *mu_muF_commute_le qinv muFqinv*.

Lemma *muF_le_term* : $\forall$ *f*, *fle* (*muFqinv* (*finv f*)) (*finv f*) $\rightarrow$
$\forall$ *x*, *f x* & *pterm x* $\leq$ *mu* (*Mfix F F_mon x*) (*q x*).

Lemma *muF_le_term_minus* :
$\forall$ *f*, *fle f pterm* $\rightarrow$ *fle* (*muFqinv* (*fminus pterm f*)) (*fminus pterm f*) $\rightarrow$
$\forall$ *x*, *f x* $\leq$ *mu* (*Mfix F F_mon x*) (*q x*).

Variable *muFq* : ($A \rightarrow U$) $\rightarrow A \rightarrow U$.

Hypothesis *muF_mon* : *Fmonotonic muFq*.

Hypothesis *F_muF_le* : *mu_muF_commute_le q muFq*.

Lemma *muF_eq* : $\forall$ *f*, *fle* (*muFq f*) *f* $\rightarrow$ *fle* (*muFqinv* (*finv f*)) (*finv f*)->
$\forall$ *x*, *pterm x* == 1 $\rightarrow$ *mu* (*Mfix F F_mon x*) (*q x*) == *f x*.

End *Fix_muF_Term*.
End *TransformFix*.

Section *LoopRule*.
Variable *q* : $A \rightarrow B \rightarrow U$.
Variable *stop* : $A \rightarrow$ *distr bool*.
Variable *step* : $A \rightarrow$ *distr A*.
Variable *a* : *U*.

Definition *Loop* (*f*:$A \rightarrow U$) (*x*:*A*) : *U*:=
mu (*stop x*) (*fun b* $\Rightarrow$ *if b then a else mu* (*step x*) *f*).

Fixpoint *loopn* (*n*:*nat*)(*x*:*A*){*struct n*} : *U* :=

match n with O $\Rightarrow$ 0
| *S p* $\Rightarrow$ *Loop* (*loopn p*) *x*
end.

Definition *loop* (*x*:*A*) : *U* := *lub* (*fun n* $\Rightarrow$ *loopn n x*).

Lemma *Mfixvar* :
($\forall$ (*f*:*A*$\rightarrow$*distr B*),
okfun (*fun x* $\Rightarrow$ *Loop* (*fun y* $\Rightarrow$ *mu* (*f y*) (*q y*)) *x*) (*fun x* $\Rightarrow$ *F f x*) *q*)
$\rightarrow$ *okfun loop* (*Mfix F F_mon*) *q.*

Fixpoint *up_loopn* (*n*:*nat*)(*x*:*A*){*struct n*} : *U* :=
match n with O $\Rightarrow$ 1
| *S p* $\Rightarrow$ *Loop* (*up_loopn p*) *x*
end.

Definition *up_loop* (*x*:*A*) : *U* := *glb* (*fun n* $\Rightarrow$ *up_loopn n x*).

Lemma *Mfixvar_up* :
($\forall$ (*f*:*A*$\rightarrow$*distr B*),
upfun (*fun x* $\Rightarrow$ *Loop* (*fun y* $\Rightarrow$ *mu* (*f y*) (*q y*)) *x*) (*fun x* $\Rightarrow$ *F f x*) *q*)
$\rightarrow$ *upfun up_loop* (*Mfix F F_mon*) *q.*

End *LoopRule.*

End *Fixrule.*

9.4 Rules for intervals

Distributions operates on intervals

Definition *Imu* : $\forall$ *A*:*Type*, *distr A*$\rightarrow$ (*A*$\rightarrow$*IU*) $\rightarrow$ *IU.*

Lemma *low_Imu* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*F*: *A* $\rightarrow$ *IU*),
low (*Imu e F*) = *mu e* (*fun x* $\Rightarrow$ *low* (*F x*)).

Lemma *up_Imu* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*F*: *A* $\rightarrow$ *IU*),
up (*Imu e F*) = *mu e* (*fun x* $\Rightarrow$ *up* (*F x*)).

Lemma *Imu_monotonic* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*F G* : *A* $\rightarrow$ *IU*),
($\forall$ *x*, *Iincl* (*F x*) (*G x*)) $\rightarrow$ *Iincl* (*Imu e F*) (*Imu e G*).

Lemma *Imu_stable_eq* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*F G* : *A* $\rightarrow$ *IU*),
($\forall$ *x*, *Ieq* (*F x*) (*G x*)) $\rightarrow$ *Ieq* (*Imu e F*) (*Imu e G*).
Hint *Resolve Imu_monotonic Imu_stable_eq.*

Lemma *Imu_singl* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*f*:*A*$\rightarrow$*U*),
Ieq (*Imu e* (*fun x* $\Rightarrow$ *singl* (*f x*))) (*singl* (*mu e f*)).

Lemma *Imu_inf* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*f*:*A*$\rightarrow$*U*),
Ieq (*Imu e* (*fun x* $\Rightarrow$ *inf* (*f x*))) (*inf* (*mu e f*)).

Lemma *Imu_sup* : $\forall$ (*A*:*Type*) (*e*:*distr A*) (*f*:*A*$\rightarrow$*U*),
Iincl (*Imu e* (*fun x* $\Rightarrow$ *sup* (*f x*))) (*sup* (*mu e f*)).

Lemma *Iin_mu_Imu* :
$\forall$ (*A*:*Type*) (*e*:*distr A*) (*F*:*A*$\rightarrow$*IU*) (*f*:*A*$\rightarrow$*U*),
($\forall$ *x*, *Iin* (*f x*) (*F x*)) $\rightarrow$ *Iin* (*mu e f*) (*Imu e F*).
Hint *Resolve Iin_mu_Imu.*

Definition *Iok* (*A*:*Type*) (*I*:*IU*) (*e*:*distr A*) (*F*:*A*$\rightarrow$*IU*) := *Iincl* (*Imu e F*) *I.*
Definition *Iokfun* (*A B*:*Type*)(*I*:*A*$\rightarrow$*IU*) (*e*:*A*$\rightarrow$*distr B*) (*F*:*A*$\rightarrow$*B*$\rightarrow$*IU*)
:= $\forall$ *x*, *Iok* (*I x*) (*e x*) (*F x*).

Lemma *Iin_mu_Iok* :
$\forall$ (*A*:*Type*) (*I*:*IU*) (*e*:*distr A*) (*F*:*A*$\rightarrow$*IU*) (*f*:*A*$\rightarrow$*U*),
($\forall$ *x*, *Iin* (*f x*) (*F x*)) $\rightarrow$ *Iok I e F* $\rightarrow$ *Iin* (*mu e f*) *I.*

9.4.1 Stability

Lemma $Iok\_le\_compat : \forall\ (A{:}Type)\ (I\ J{:}IU)\ (e{:}distr\ A)\ (F\ G{:}A{\rightarrow}IU),$
$Iincl\ I\ J \rightarrow (\forall\ x,\ Iincl\ (G\ x)\ (F\ x)) \rightarrow Iok\ I\ e\ F \rightarrow Iok\ J\ e\ G.$

Lemma $Iokfun\_le\_compat : \forall\ (A\ B{:}Type)\ (I\ J{:}A{\rightarrow}IU)\ (e{:}A{\rightarrow}distr\ B)\ (F\ G{:}A{\rightarrow}B{\rightarrow}IU),$
$(\forall\ x,\ Iincl\ (I\ x)\ (J\ x)) \rightarrow (\forall\ x\ y,\ Iincl\ (G\ x\ y)\ (F\ x\ y)) \rightarrow Iokfun\ I\ e\ F \rightarrow Iokfun\ J\ e\ G.$

9.4.2 Rule for values

Lemma $Iunit\_eq : \forall\ (A{:}\ Type)\ (a{:}A)\ (F{:}A{\rightarrow}IU),\ Ieq\ (Imu\ (Munit\ a)\ F)\ (F\ a).$

9.4.3 Rule for application

Lemma $Ilet\_eq : \forall\ (A\ B : Type)\ (a{:}distr\ A)\ (f{:}A{\rightarrow}distr\ B)(I{:}IU)(G{:}B{\rightarrow}IU),$
$Ieq\ (Imu\ (Mlet\ a\ f)\ G)\ (Imu\ a\ (fun\ x \Rightarrow Imu\ (f\ x)\ G)).$
Hint $Resolve\ Ilet\_eq.$

Lemma $Iapply\_rule : \forall\ (A\ B : Type)\ (a{:}distr\ A)\ (f{:}A{\rightarrow}distr\ B)(I{:}IU)(F{:}A{\rightarrow}IU)(G{:}B{\rightarrow}IU),$
$Iok\ I\ a\ F \rightarrow Iokfun\ F\ f\ (fun\ x \Rightarrow G) \rightarrow Iok\ I\ (Mlet\ a\ f)\ G.$

9.4.4 Rule for abstraction

Lemma $Ilambda\_rule : \forall\ (A\ B{:}Type)(f{:}A{\rightarrow}distr\ B)(F{:}A{\rightarrow}IU)(G{:}A \rightarrow B{\rightarrow}IU),$
$(\forall\ x{:}A,\ Iok\ (F\ x)\ (f\ x)\ (G\ x)) \rightarrow Iokfun\ F\ f\ G.$

9.4.5 Rule for conditional

Lemma $Imu\_Mif\_eq : \forall\ (A{:}Type)(b{:}distr\ bool)(f1\ f2{:}distr\ A)(F{:}A{\rightarrow}IU),$
$Ieq\ (Imu\ (Mif\ b\ f1\ f2)\ F)\ (Iplus\ (Imultk\ (mu\ b\ ctrue)\ (Imu\ f1\ F))\ (Imultk\ (mu\ b\ cfalse)\ (Imu\ f2\ F))).$

Lemma $Iifrule$:
$\forall\ (A{:}Type)(b{:}(distr\ bool))(f1\ f2{:}distr\ A)(I1\ I2{:}IU)(F{:}A{\rightarrow}IU),$
$Iok\ I1\ f1\ F \rightarrow Iok\ I2\ f2\ F$
$\rightarrow Iok\ (Iplus\ (Imultk\ (mu\ b\ ctrue)\ I1)\ (Imultk\ (mu\ b\ cfalse)\ I2))\ (Mif\ b\ f1\ f2)\ F.$

9.4.6 Rule for fixpoints

with $\phi(x) = F(\phi)(x)$, p_i an decreasing sequence of intervals functions ($p_{i+1}(x) \subseteq p_i(x)$) such that $p_0(x)$ contains 0 for all x.

$$\frac{\forall f\ i, (\forall x, \langle f\rangle(q\ x) \sqsubseteq p_i(x)) \Rightarrow \forall x, \langle F(f)(x)\rangle(q\ x) \sqsubseteq p_{i+1}(x)}{\forall x, \langle\phi(x)\rangle(q\ x) \sqsubseteq \bigcap_i p_i\ x}$$

Section $IFixrule.$
Variables $A\ B : Type.$

Variable $F : (A \rightarrow distr\ B) \rightarrow A \rightarrow distr\ B.$

Hypothesis $F\_mon : \forall\ f\ g :\ A \rightarrow (distr\ B),$
$(\forall\ x,\ le\_distr\ (f\ x)\ (g\ x)) \rightarrow \forall\ x,\ le\_distr\ (F\ f\ x)\ (F\ g\ x).$

Section $IRuleseq.$
Variable $Q : A \rightarrow B \rightarrow IU.$

Variable $I : A \rightarrow nat \rightarrow IU.$
Hypothesis $decrp : \forall\ x\ n,\ Iincl\ (I\ x\ (S\ n))\ (I\ x\ n).$

Lemma $Ifixrule$:
$(\forall\ x{:}A,\ Iin\ 0\ (I\ x\ O)) \rightarrow$
$(\forall\ (i{:}nat)\ (f{:}A{\rightarrow}distr\ B),$
$(Iokfun\ (fun\ x \Rightarrow I\ x\ i)\ f\ Q) \rightarrow Iokfun\ (fun\ x \Rightarrow I\ x\ (S\ i))\ (fun\ x \Rightarrow F\ f\ x)\ Q)$
$\rightarrow Iokfun\ (fun\ x \Rightarrow lim\ (I\ x)\ (decrp\ x))\ (Mfix\ F\ F\_mon)\ Q.$

End *IRuleseq.*

Section *ITransformFix.*

Section *IFix_muF.*
Variable $Q : A \to B \to IU.$
Variable $ImuF : (A \to IU) \to A \to IU.$
Hypothesis $ImuF\_mon : \forall\ I\ J,$
$(\forall\ x,\ Iincl\ (I\ x)\ (J\ x)) \to \forall\ x,\ Iincl\ (ImuF\ I\ x)\ (ImuF\ J\ x).$

Lemma $ImuF\_stable : \forall\ I\ J,$
$(\forall\ x,\ Ieq\ (I\ x)\ (J\ x)) \to \forall\ x,\ Ieq\ (ImuF\ I\ x)\ (ImuF\ J\ x).$

Section *IF_muF_results.*
Hypothesis $Iincl\_F\_ImuF :$
$\forall\ f\ x,\ (\forall\ y,\ le\_distr\ (f\ y)\ (Mfix\ F\ F\_mon\ y)) \to$
$Iincl\ (Imu\ (F\ f\ x)\ (Q\ x))\ (ImuF\ (fun\ y \Rightarrow Imu\ (f\ y)\ (Q\ y))\ x).$

Lemma $Iincl\_fix\_ifix : \forall\ x,\ Iincl\ (Imu\ (Mfix\ F\ F\_mon\ x)\ (Q\ x))\ (Ifix\ ImuF\ ImuF\_mon\ x).$
Hint *Resolve Iincl_fix_ifix.*

End *IF_muF_results.*

End *IFix_muF.*
End *ITransformFix.*
End *IFixrule.*

9.5 Rules for *Flip*

Lemma $Flip\_ctrue : mu\ Flip\ ctrue == [1/2].$

Lemma $Flip\_cfalse : mu\ Flip\ cfalse == [1/2].$

Lemma $ok\_Flip : \forall\ q : bool \to U,\ ok\ ([1/2] \times q\ true + [1/2] \times q\ false)\ Flip\ q.$

Lemma $okup\_Flip : \forall\ q : bool \to U,\ okup\ ([1/2] \times q\ true + [1/2] \times q\ false)\ Flip\ q.$

Hint *Resolve ok_Flip okup_Flip Flip_ctrue Flip_cfalse.*

Lemma $Flip\_eq : \forall\ q : bool \to U,\ mu\ Flip\ q == [1/2] \times q\ true + [1/2] \times q\ false.$
Hint *Resolve Flip_eq.*

Lemma $IFlip\_eq : \forall\ Q : bool \to IU,\ Ieq\ (Imu\ Flip\ Q)\ (Iplus\ (Imultk\ [1/2]\ (Q\ true))\ (Imultk\ [1/2]\ (Q\ false))).$
Hint *Resolve IFlip_eq.*

9.6 Rules for total (well-founded) fixpoints

Section *Wellfounded.*
Variables $A\ B : Type.$
Variable $R : A \to A \to Prop.$
Hypothesis $Rwf : well\_founded\ R.$
Variable $F : \forall\ x,\ (\forall\ y,\ R\ y\ x \to distr\ B) \to distr\ B.$

Definition $WfFix : A \to distr\ B := \mathsf{Fix}\ Rwf\ (fun\ \_ \Rightarrow distr\ B)\ F.$

Hypothesis $Fext : \forall\ x\ f\ g,\ (\forall\ y\ (p{:}R\ y\ x),\ eq\_distr\ (f\ y\ p)\ (g\ y\ p)) \to eq\_distr\ (F\ f)\ (F\ g).$

Lemma $Acc\_iter\_distr :$
$\forall\ x, \forall\ r\ s : Acc\ R\ x,\ eq\_distr\ (Acc\_iter\ (fun\ \_\Rightarrow distr\ B)\ F\ r)\ (Acc\_iter\ (fun\ \_\Rightarrow distr\ B)\ F\ s).$

Lemma $WfFix\_eq : \forall\ x,\ eq\_distr\ (WfFix\ x)\ (F\ (fun\ (y{:}A)\ (p{:}R\ y\ x) \Rightarrow WfFix\ y)).$

Variable $P : distr\ B \to Prop.$
Hypothesis $Pext : \forall\ m1\ m2,\ eq\_distr\ m1\ m2 \to P\ m1 \to P\ m2.$

Lemma $WfFix\_ind :$
$(\forall\ x\ f,\ (\forall\ y\ (p{:}R\ y\ x),\ P\ (f\ y\ p)) \to P\ (F\ f))$

$\rightarrow \forall\ x,\ P\ (WfFix\ x).$

End *Wellfounded.*

End *Rules.*
Require Export *Setoid.*
Require *Omega.*

10 Sets.v: Definition of sets as predicates over a type A

Section *sets.*
Variable A : *Type.*
Variable $decA : \forall\ x\ y : A,\ \{x{=}y\}{+}\{x{\neq}y\}.$

Definition $set := A{\rightarrow}Prop.$
Definition $full : set := fun\ (x{:}A) \Rightarrow True.$
Definition $empty : set := fun\ (x{:}A) \Rightarrow False.$
Definition $add\ (a{:}A)\ (P{:}set) : set := fun\ (x{:}A) \Rightarrow x{=}a \lor (P\ x).$
Definition $singl\ (a{:}A) : set := fun\ (x{:}A) \Rightarrow x{=}a.$
Definition $union\ (P\ Q{:}set) : set := fun\ (x{:}A) \Rightarrow (P\ x) \lor (Q\ x).$
Definition $compl\ (P{:}set) : set := fun\ (x{:}A) \Rightarrow \neg P\ x.$
Definition $inter\ (P\ Q{:}set) : set := fun\ (x{:}A) \Rightarrow (P\ x) \land (Q\ x).$
Definition $rem\ (a{:}A)\ (P{:}set) : set := fun\ (x{:}A) \Rightarrow x{\neq}a \land (P\ x).$

10.1 Equivalence

Definition $equiv\ (P\ Q{:}set) := \forall\ (x{:}A),\ P\ x \leftrightarrow Q\ x.$

Implicit *Arguments full* [].
Implicit *Arguments empty* [].

Lemma $equiv\_refl : \forall\ P{:}set,\ equiv\ P\ P.$

Lemma $equiv\_sym : \forall\ P\ Q{:}set,\ equiv\ P\ Q \rightarrow equiv\ Q\ P.$

Lemma $equiv\_trans : \forall\ P\ Q\ R{:}set,$
  $equiv\ P\ Q \rightarrow equiv\ Q\ R \rightarrow equiv\ P\ R.$

Hint *Resolve equiv_refl.*
Hint Immediate *equiv_sym.*

10.2 Setoid structure

Lemma $set\_setoid : Setoid\_Theory\ set\ equiv.$

Add Setoid set equiv set_setoid as Set_setoid.

Add Morphism add : equiv_add.

Add Morphism rem : equiv_rem.
Hint *Resolve equiv_add equiv_rem.*

Add Morphism union : equiv_union.
Hint Immediate *equiv_union.*

Lemma $equiv\_union\_left$:
 $\forall\ P1\ Q\ P2,$
  $equiv\ P1\ P2 \rightarrow equiv\ (union\ P1\ Q)\ (union\ P2\ Q).$

Lemma $equiv\_union\_right$:
 $\forall\ P\ Q1\ Q2$,
  $equiv\ Q1\ Q2 \rightarrow equiv\ (union\ P\ Q1)\ (union\ P\ Q2).$

Hint *Resolve equiv_union_left equiv_union_right.*

Add Morphism inter : equiv_inter.
Hint Immediate *equiv_inter.*

Add Morphism compl : equiv_compl.
Hint *Resolve equiv_compl.*

Lemma *equiv_add_empty* : ∀ (*a*:*A*) (*P*:*set*), ¬*equiv* (*add a P*) *empty*.

10.3 Finite sets given as an enumeration of elements

```
Inductive finite (P: set) : Type :=
    fin_eq_empty : equiv P empty → finite P
  | fin_eq_add : ∀ (x:A)(Q:set),
                 ¬ Q x→ finite Q → equiv P (add x Q) → finite P.
Hint Constructors finite.
```

Lemma *fin_empty* : (*finite empty*).

```
Lemma fin_add : ∀ (x:A)(P:set),
                ¬ P x → finite P → finite (add x P).
```

Lemma *fin_equiv*: ∀ (*P Q* : *set*), (*equiv P Q*)->(*finite P*)->(*finite Q*).

Hint *Resolve fin_empty fin_add.*

10.3.1 Emptyness is decidable for finite sets

Definition *isempty* (*P*:*set*) := *equiv P empty*.
Definition *notempty* (*P*:*set*) := *not* (*equiv P empty*).

Lemma *isempty_dec* : ∀ *P*, *finite P* → {*isempty P*}+{*notempty P*}.

10.3.2 Size of a finite set

```
Fixpoint size (P:set) (f:finite P) {struct f}: nat :=
    match f with fin_eq_empty _ ⇒ 0%nat
                | fin_eq_add _ Q _ f' _ ⇒ S (size f')
    end.
```

```
Lemma size_equiv : ∀ P Q (f:finite P) (e:equiv P Q),
     (size (fin_equiv e f)) = (size f).
```

10.4 Inclusion

Definition *incl* (*P Q*:*set*) := ∀ *x*, *P x* → *Q x*.

Lemma *incl_refl* : ∀ (*P*:*set*), *incl P P*.

```
Lemma incl_trans : ∀ (P Q R:set),
    incl P Q → incl Q R → incl P R.
```

Lemma *equiv_incl* : ∀ (*P Q* : *set*), *equiv P Q* → *incl P Q*.

Lemma *equiv_incl_sym* : ∀ (*P Q* : *set*), *equiv P Q* → *incl Q P*.

```
Lemma equiv_incl_intro :
   ∀ (P Q : set), incl P Q → incl Q P → equiv P Q.
```

Hint *Resolve incl_refl incl_trans equiv_incl_intro.*
Hint Immediate *equiv_incl equiv_incl_sym.*

10.5 Properties of operations on sets

Lemma *incl_empty* : $\forall$ *P*, *incl empty P*.

Lemma *incl_empty_false* : $\forall$ *P a*, *incl P empty* $\rightarrow \neg$ *P a*.

Lemma *incl_add_empty* : $\forall$ (*a*:*A*) (*P*:*set*), $\neg$ *incl* (*add a P*) *empty*.

Lemma *equiv_empty_false* : $\forall$ *P a*, *equiv P empty* $\rightarrow$ *P a* $\rightarrow$ *False*.

Hint Immediate *incl_empty_false equiv_empty_false incl_add_empty*.

Lemma *incl_rem_stable* : $\forall$ *a P Q*, *incl P Q* $\rightarrow$ *incl* (*rem a P*) (*rem a Q*).

Lemma *incl_add_stable* : $\forall$ *a P Q*, *incl P Q* $\rightarrow$ *incl* (*add a P*) (*add a Q*).

Lemma *incl_rem_add_iff* :
$\forall$ *a P Q*, *incl* (*rem a P*) *Q* $\leftrightarrow$ *incl P* (*add a Q*).

Lemma *incl_rem_add*:
$\forall$ (*a*:*A*) (*P Q*:*set*),
(*P a*) $\rightarrow$ *incl Q* (*rem a P*) $\rightarrow$ *incl* (*add a Q*) *P*.

Lemma *incl_add_rem* :
$\forall$ (*a*:*A*) (*P Q*:*set*),
$\neg$ *Q a* $\rightarrow$ *incl* (*add a Q*) *P* $\rightarrow$ *incl Q* (*rem a P*) .

Hint Immediate *incl_rem_add incl_add_rem*.

Lemma *equiv_rem_add* :
$\forall$ (*a*:*A*) (*P Q*:*set*),
(*P a*) $\rightarrow$ *equiv Q* (*rem a P*) $\rightarrow$ *equiv* (*add a Q*) *P*.

Lemma *equiv_add_rem* :
$\forall$ (*a*:*A*) (*P Q*:*set*),
$\neg$ *Q a* $\rightarrow$ *equiv* (*add a Q*) *P* $\rightarrow$ *equiv Q* (*rem a P*).

Hint Immediate *equiv_rem_add equiv_add_rem*.

Lemma *add_rem_eq_equiv* :
$\forall$ *x* (*P*:*set*), *equiv* (*add x* (*rem x P*)) (*add x P*).

Lemma *add_rem_diff_equiv* :
$\forall$ *x y* (*P*:*set*),
$x \neq y$ $\rightarrow$ *equiv* (*add x* (*rem y P*)) (*rem y* (*add x P*)).

Lemma *add_equiv_in* :
$\forall$ *x* (*P*:*set*), *P x* $\rightarrow$ *equiv* (*add x P*) *P*.

Hint *Resolve add_rem_eq_equiv add_rem_diff_equiv add_equiv_in*.

Lemma *add_rem_equiv_in* :
$\forall$ *x* (*P*:*set*), *P x* $\rightarrow$ *equiv* (*add x* (*rem x P*)) *P*.

Hint *Resolve add_rem_equiv_in*.

Lemma *rem_add_eq_equiv* :
$\forall$ *x* (*P*:*set*), *equiv* (*rem x* (*add x P*)) (*rem x P*).

Lemma *rem_add_diff_equiv* :
$\forall$ *x y* (*P*:*set*),
$x \neq y$ $\rightarrow$ *equiv* (*rem x* (*add y P*)) (*add y* (*rem x P*)).

Lemma *rem_equiv_notin* :
$\forall$ *x* (*P*:*set*), $\neg$*P x* $\rightarrow$ *equiv* (*rem x P*) *P*.

Hint *Resolve rem_add_eq_equiv rem_add_diff_equiv rem_equiv_notin*.

Lemma *rem_add_equiv_notin* :
$\forall$ *x* (*P*:*set*), $\neg$*P x* $\rightarrow$ *equiv* (*rem x* (*add x P*)) *P*.

Hint *Resolve rem_add_equiv_notin*.

Lemma *rem_not_in* : $\forall$ *x* (*P*:*set*), $\neg$ *rem x P x*.

Lemma *add_in* : $\forall$ *x* (*P*:*set*), *add x P x*.

Lemma *add_in_eq* : $\forall$ *x y P*, *x*=*y* $\rightarrow$ *add x P y*.

Lemma *add_intro* : $\forall$ *x* (*P*:*set*) *y*, *P y* $\rightarrow$ *add x P y*.

Lemma *add_incl* : $\forall$ *x* (*P*:*set*), *incl P* (*add x P*).

Lemma *add_incl_intro* : $\forall$ *x* (*P Q*:*set*), (*Q x*) $\rightarrow$ (*incl P Q*) $\rightarrow$ (*incl* (*add x P*) *Q*).

Lemma *rem_incl* : $\forall$ *x* (*P*:*set*), *incl* (*rem x P*) *P*.

Hint *Resolve rem_not_in add_in rem_incl add_incl*.

Lemma *union_sym* : $\forall$ *P Q* : *set*,
equiv (*union P Q*) (*union Q P*).

Lemma *union_empty_left* : $\forall$ *P* : *set*,
equiv P (*union P empty*).

Lemma *union_empty_right* : $\forall$ *P* : *set*,
equiv P (*union empty P*).

Lemma *union_add_left* : $\forall$ (*a*:*A*) (*P Q*: *set*),
equiv (*add a* (*union P Q*)) (*union P* (*add a Q*)).

Lemma *union_add_right* : $\forall$ (*a*:*A*) (*P Q*: *set*),
equiv (*add a* (*union P Q*)) (*union* (*add a P*) *Q*).

Hint *Resolve union_sym union_empty_left union_empty_right*
union_add_left union_add_right.

Lemma *union_incl_left* : $\forall$ *P Q*, *incl P* (*union P Q*).

Lemma *union_incl_right* : $\forall$ *P Q*, *incl Q* (*union P Q*).

Lemma *union_incl_intro* : $\forall$ *P Q R*, *incl P R* $\rightarrow$ *incl Q R* $\rightarrow$ *incl* (*union P Q*) *R*.

Hint *Resolve union_incl_left union_incl_right union_incl_intro*.

Lemma *incl_union_stable* : $\forall$ *P1 P2 Q1 Q2*,
incl P1 P2 $\rightarrow$ *incl Q1 Q2* $\rightarrow$ *incl* (*union P1 Q1*) (*union P2 Q2*).
Hint Immediate *incl_union_stable*.

Lemma *inter_sym* : $\forall$ *P Q* : *set*,
equiv (*inter P Q*) (*inter Q P*).

Lemma *inter_empty_left* : $\forall$ *P* : *set*,
equiv empty (*inter P empty*).

Lemma *inter_empty_right* : $\forall$ *P* : *set*,
equiv empty (*inter empty P*).

Lemma *inter_add_left_in* : $\forall$ (*a*:*A*) (*P Q*: *set*),
(*P a*) $\rightarrow$ *equiv* (*add a* (*inter P Q*)) (*inter P* (*add a Q*)).

Lemma *inter_add_left_out* : $\forall$ (*a*:*A*) (*P Q*: *set*),
$\neg$ *P a* $\rightarrow$ *equiv* (*inter P Q*) (*inter P* (*add a Q*)).

Lemma *inter_add_right_in* : $\forall$ (*a*:*A*) (*P Q*: *set*),
Q a $\rightarrow$ *equiv* (*add a* (*inter P Q*)) (*inter* (*add a P*) *Q*).

Lemma *inter_add_right_out* : $\forall$ (*a*:*A*) (*P Q*: *set*),
$\neg$ *Q a* $\rightarrow$ *equiv* (*inter P Q*) (*inter* (*add a P*) *Q*).

Hint *Resolve inter_sym inter_empty_left inter_empty_right*
inter_add_left_in inter_add_left_out inter_add_right_in inter_add_right_out.

10.6 Removing an element from a finite set

Lemma $\mathit{finite\_rem} : \forall\ (P{:}\mathit{set})\ (a{:}A),$
$(\mathit{finite}\ P) \to (\mathit{finite}\ (\mathit{rem}\ a\ P)).$

Lemma $\mathit{size\_finite\_rem}:$
$\forall\ (P{:}\mathit{set})\ (a{:}A)\ (f{:}\mathit{finite}\ P),$
$(P\ a) \to \mathit{size}\ f = S\ (\mathit{size}\ (\mathit{finite\_rem}\ a\ f)).$

Require Import $\mathit{Arith}.$

Lemma $\mathit{size\_incl} :$
$\forall\ (P{:}\mathit{set})(f{:}\mathit{finite}\ P)\ (Q{:}\mathit{set})(g{:}\mathit{finite}\ Q),$
$(\mathit{incl}\ P\ Q)\text{->}\ \mathit{size}\ f \le \mathit{size}\ g.$

Lemma $\mathit{size\_unique} :$
$\forall\ (P{:}\mathit{set})(f{:}\mathit{finite}\ P)\ (Q{:}\mathit{set})(g{:}\mathit{finite}\ Q),$
$(\mathit{equiv}\ P\ Q)\text{->}\ \mathit{size}\ f = \mathit{size}\ g.$

10.7 Decidable sets

Definition $\mathit{dec}\ (P{:}\mathit{set}) := \forall\ x, \{P\ x\}+\{\tilde{}\ P\ x\}.$

Lemma $\mathit{finite\_incl} : \forall\ P{:}\mathit{set},$
$\mathit{finite}\ P \to \forall\ Q{:}\mathit{set},\ \mathit{dec}\ Q \to \mathit{incl}\ Q\ P \to \mathit{finite}\ Q.$

Lemma $\mathit{finite\_dec} : \forall\ P{:}\mathit{set},\ \mathit{finite}\ P \to \mathit{dec}\ P.$

Lemma $\mathit{fin\_add\_in} : \forall\ (a{:}A)\ (P{:}\mathit{set}),\ \mathit{finite}\ P \to \mathit{finite}\ (\mathit{add}\ a\ P).$

Lemma $\mathit{finite\_union} :$
$\forall\ P\ Q,\ \mathit{finite}\ P \to \mathit{finite}\ Q \to \mathit{finite}\ (\mathit{union}\ P\ Q).$

Lemma $\mathit{finite\_full\_dec} : \forall\ P{:}\mathit{set},\ \mathit{finite}\ \mathit{full} \to \mathit{dec}\ P \to \mathit{finite}\ P.$

Require Import $\mathit{Lt}.$

10.7.1 Filter operation

Lemma $\mathit{finite\_inter} : \forall\ P\ Q,\ \mathit{dec}\ P \to \mathit{finite}\ Q \to \mathit{finite}\ (\mathit{inter}\ P\ Q).$

Lemma $\mathit{size\_inter\_empty} : \forall\ P\ Q\ (\mathit{decP}{:}\mathit{dec}\ P)\ (e{:}\mathit{equiv}\ Q\ \mathit{empty}),$
$\mathit{size}\ (\mathit{finite\_inter}\ \mathit{decP}\ (\mathit{fin\_eq\_empty}\ e)){=}O.$

Lemma $\mathit{size\_inter\_add\_in} :$
$\forall\ P\ Q\ R\ (\mathit{decP}{:}\mathit{dec}\ P)(x{:}A)(\mathit{nq}{:}\tilde{}Q\ x)(\mathit{FQ}{:}\mathit{finite}\ Q)(e{:}\mathit{equiv}\ R\ (\mathit{add}\ x\ Q)),$
$P\ x \to \mathit{size}\ (\mathit{finite\_inter}\ \mathit{decP}\ (\mathit{fin\_eq\_add}\ \mathit{nq}\ \mathit{FQ}\ e)){=}S\ (\mathit{size}\ (\mathit{finite\_inter}\ \mathit{decP}\ \mathit{FQ})).$

Lemma $\mathit{size\_inter\_add\_notin} :$
$\forall\ P\ Q\ R\ (\mathit{decP}{:}\mathit{dec}\ P)(x{:}A)(\mathit{nq}{:}\tilde{}Q\ x)(\mathit{FQ}{:}\mathit{finite}\ Q)(e{:}\mathit{equiv}\ R\ (\mathit{add}\ x\ Q)),$
$\neg\ P\ x \to \mathit{size}\ (\mathit{finite\_inter}\ \mathit{decP}\ (\mathit{fin\_eq\_add}\ \mathit{nq}\ \mathit{FQ}\ e)){=}\mathit{size}\ (\mathit{finite\_inter}\ \mathit{decP}\ \mathit{FQ}).$

Lemma $\mathit{size\_inter\_incl} : \forall\ P\ Q\ (\mathit{decP}{:}\mathit{dec}\ P)(\mathit{FP}{:}\mathit{finite}\ P)(\mathit{FQ}{:}\mathit{finite}\ Q),$
$(\mathit{incl}\ P\ Q) \to \mathit{size}\ (\mathit{finite\_inter}\ \mathit{decP}\ \mathit{FQ}){=}\mathit{size}\ \mathit{FP}.$

10.7.2 Selecting elements in a finite set

Fixpoint $\mathit{nth\_finite}\ (P{:}\mathit{set})\ (k{:}\mathit{nat})\ (\mathit{PF} : \mathit{finite}\ P)\ \{\mathit{struct}\ \mathit{PF}\}\text{:}\ (k < \mathit{size}\ \mathit{PF}) \to A :=$
$\mathit{match}\ \mathit{PF}\ \mathit{as}\ F\ \mathit{return}\ (k < \mathit{size}\ F) \to A\ \mathit{with}$
$\quad \mathit{fin\_eq\_empty}\ H \Rightarrow (\mathit{fun}\ (e : k{<}0) \Rightarrow \mathit{match}\ \mathit{lt\_n\_O}\ k\ e\ \mathit{with}\ \mathit{end})$
$\mid \mathit{fin\_eq\_add}\ x\ Q\ \mathit{nqx}\ \mathit{fq}\ \mathit{eqq} \Rightarrow$
$\quad \mathit{match}\ k\ \mathit{as}\ \mathit{k0}\ \mathit{return}\ \mathit{k0}{<}S\ (\mathit{size}\ \mathit{fq})\text{->}A\ \mathit{with}$
$\qquad O \Rightarrow \mathit{fun}\ e \Rightarrow x$
$\quad \mid (S\ \mathit{k1}) \Rightarrow \mathit{fun}\ (e{:}S\ \mathit{k1}{<}S\ (\mathit{size}\ \mathit{fq})) \Rightarrow \mathit{nth\_finite}\ \mathit{fq}\ (\mathit{lt\_S\_n}\ \mathit{k1}\ (\mathit{size}\ \mathit{fq})\ e)$
$\quad \mathit{end}$

end.

A set with size > 1 contains at least 2 different elements

Lemma *select_non_empty* : $\forall$ (*P*:*set*), *finite P* $\rightarrow$ *notempty P* $\rightarrow$ *sigT P*.

Lemma *select_diff* : $\forall$ (*P*:*set*) (*FP*:*finite P*),
(1 $<$ *size FP*)%*nat* $\rightarrow$ *sigT* (*fun x* $\Rightarrow$ *sigT* (*fun y* $\Rightarrow$ *P x* $\wedge$ *P y* $\wedge$ *x*$\neq$*y*)).

End *sets.*

Hint *Resolve equiv_refl.*
Hint *Resolve equiv_add equiv_rem.*
Hint Immediate *equiv_sym finite_dec finite_full_dec equiv_incl equiv_incl_sym equiv_incl_intro.*

Hint *Resolve incl_refl.*
Hint Immediate *incl_union_stable.*
Hint *Resolve union_incl_left union_incl_right union_incl_intro incl_empty rem_incl incl_rem_stable incl_add_stable.*

Hint *Constructors finite.*
Hint *Resolve add_in add_in_eq add_intro add_incl add_incl_intro union_sym union_empty_left union_empty_right union_add_left union_add_right finite_union equiv_union_left equiv_union_right.*
Implicit *Arguments full* [].
Implicit *Arguments empty* [].

11 Carac.v: Characteristic functions

Require Export *Prog.*
Require Export *Sets.*
Require Export *Arith.*
Module *CaracFun* (*Univ*:*Universe*).

Module *RP* := (*Rules Univ*).

Definition *carac* (*A*:*Type*)(*P*:*A* $\rightarrow$ *Prop*)(*Pdec*:*dec P*)(*z*:*A*): *U* := *if* (*Pdec z*) *then* 1 *else* 0.

Lemma *carac_one* : $\forall$ (*A*:*Type*)(*P*:*A* $\rightarrow$ *Prop*)(*Pdec*:*dec P*)(*z*:*A*),
P z $\rightarrow$ *carac Pdec z* == 1.

Lemma *carac_zero* : $\forall$ (*A*:*Type*)(*P*:*A* $\rightarrow$ *Prop*)(*Pdec*:*dec P*)(*z*:*A*),
$\neg$*P z* $\rightarrow$ *carac Pdec z* == 0.

Lemma *carac_unit* : $\forall$ (*A*:*Type*)(*P*:*A* $\rightarrow$ *Prop*)(*Pdec* : *dec P*)(*a*:*A*),
(*P a*) $\rightarrow$ 1 $\leq$ (*mu* (*Munit a*)) (*carac Pdec*).

Lemma *carac_let_one* : $\forall$ (*A B*:*Type*)(*m1*: *distr A*)(*m2*: *A* $\rightarrow$ *distr B*) (*P*:*B* $\rightarrow$ *Prop*)(*Pdec*: *dec P*),
mu m1 (*f_one A*) == 1 $\rightarrow$ ($\forall$ *x*:*A*, 1 $\leq$ *mu* (*m2 x*) (*carac Pdec*)) $\rightarrow$ 1 $\leq$ *mu* (*Mlet m1 m2*) (*carac Pdec*)
.

Lemma *carac_let* : $\forall$ (*A B*:*Type*)(*m1*: *distr A*)(*m2*: *A*$\rightarrow$*distr B*) (*P*:*A*$\rightarrow$*Prop*)(*Pdec*: *dec P*)(*f*:*B*$\rightarrow$*U*)(*p*:*U*),
1 $\leq$ *mu m1* (*carac Pdec*) $\rightarrow$ ($\forall$ *x*:*A*, *P x* $\rightarrow$ *p* $\leq$ *mu* (*m2 x*) *f*)
$\rightarrow$ *p* $\leq$ *mu* (*Mlet m1 m2*) *f*.

Lemma *carac_incl*: $\forall$ (*A*:*Type*)(*P Q*:*A* $\rightarrow$ *Prop*)(*Pdec*: *dec P*)(*Qdec*: *dec Q*),
incl P Q $\rightarrow$ *fle* (*carac Pdec*) (*carac Qdec*).

Definition *equiv_dec* : $\forall$ (*A*:*Type*)(*P Q*:*A* $\rightarrow$ *Prop*),*dec P* $\rightarrow$ *equiv P Q* $\rightarrow$ *dec Q*.

Lemma *carac_equiv* : $\forall$ (*A*:*Type*)(*P Q*:*A* $\rightarrow$ *Prop*)(*Pdec* : *dec P*)(*EQ* : *equiv P Q*),
feq (*carac* (*equiv_dec Pdec EQ*)) (*carac Pdec*).

Definition *union_dec* : $\forall$ (*A*:*Type*)(*P Q*:*A* $\rightarrow$ *Prop*), *dec P* $\rightarrow$ *dec Q* $\rightarrow$ *dec* (*union P Q*).

Lemma *carac_union* : $\forall$ (*A*:*Type*)(*P Q*:*A* $\rightarrow$ *Prop*)(*Pdec* : *dec P*)(*Qdec* : *dec Q*),
feq (*carac* (*union_dec Pdec Qdec*)) (*fun a* $\Rightarrow$ (*carac Pdec a*) + (*carac Qdec a*)).

Definition *inter_dec* : $\forall$ (*A*:*Type*)(*P Q*:*A* $\rightarrow$ *Prop*),(*dec P*)->(*dec Q*) $\rightarrow$ *dec* (*inter P Q*).

Lemma *carac_inter* : $\forall$ $(A{:}Type)(P\ Q{:}A \rightarrow Prop)(Pdec : dec\ P)(Qdec : dec\ Q)$,
  $feq\ (carac\ (inter\_dec\ Pdec\ Qdec))\ (fun\ a \Rightarrow (carac\ Pdec\ a) \times (carac\ Qdec\ a))$.

Definition *compl_dec* : $\forall$ $(A{:}Type)(P{:}A \rightarrow Prop)$, $dec\ P \rightarrow dec\ (compl\ P)$.

Lemma *carac_compl* : $\forall$ $(A{:}Type)(P{:}A \rightarrow Prop)(Pdec : dec\ P)$,
  $feq\ (carac\ (compl\_dec\ Pdec))\ (fun\ a \Rightarrow [1-](carac\ Pdec\ a))$.

Definition *empty_dec* : $\forall$ $(A{:}Type)(P{:}A{\rightarrow}Prop)$, $equiv\ P\ (empty\ A) \rightarrow dec\ P$.

Lemma *carac_empty* : $\forall$ $(A{:}Type)(P{:}A{\rightarrow}Prop)$
  $(empP{:}equiv\ P\ (empty\ A)), feq\ (carac\ (empty\_dec\ empP))\ (f\_zero\ A)$.

Lemma *carac_mult_fun* : $\forall$ $(A{:}Type)(P{:}A \rightarrow Prop)(Pdec : dec\ P)(f\ g{:}A{\rightarrow}U)$,
  $(\forall\ x, P\ x \rightarrow f\ x{==}g\ x) \rightarrow \forall\ x, carac\ Pdec\ x \times f\ x == carac\ Pdec\ x \times g\ x$.

Lemma *carac_esp_fun* : $\forall$ $(A{:}Type)(P{:}A \rightarrow Prop)(Pdec : dec\ P)(f\ g{:}A{\rightarrow}U)$,
  $(\forall\ x, P\ x \rightarrow f\ x{==}g\ x) \rightarrow \forall\ x, carac\ Pdec\ x\ \&\ f\ x == carac\ Pdec\ x\ \&\ g\ x$.
Hint *Resolve carac_esp_fun*.

Lemma *carac_esp_fun_le* : $\forall$ $(A{:}Type)(P{:}A \rightarrow Prop)(Pdec : dec\ P)(f\ g{:}A{\rightarrow}U)$,
  $(\forall\ x, P\ x \rightarrow f\ x{\leq}g\ x) \rightarrow \forall\ x, carac\ Pdec\ x\ \&\ f\ x \leq carac\ Pdec\ x\ \&\ g\ x$.
Hint *Resolve carac_esp_fun_le*.

Lemma *carac_ok* : $\forall$ $(A{:}Type)(P\ Q{:}A \rightarrow Prop)(Pdec : dec\ P)(Qdec : dec\ Q)$,
  $(\forall\ x, P\ x \rightarrow \neg\ Q\ x) \rightarrow fplusok\ (carac\ Pdec)\ (carac\ Qdec)$.
Hint *Resolve carac_ok*.

11.1 Modular reasoning on programs

Lemma *mu_carac_esp* : $\forall$ $(A{:}Type)(m{:}distr\ A)(P{:}A \rightarrow Prop)(Pdec : dec\ P)(f{:}A{\rightarrow}U)$,
  $1{\leq}mu\ m\ (carac\ Pdec) \rightarrow mu\ m\ f == mu\ m\ (fun\ x \Rightarrow carac\ Pdec\ x\ \&\ f\ x)$.

Lemma *mu_cut* : $\forall$ $(A{:}Type)(m{:}distr\ A)(P{:}A \rightarrow Prop)(Pdec : dec\ P)(f\ g{:}A{\rightarrow}U)$,
  $(\forall\ x, P\ x \rightarrow f\ x{==}g\ x) \rightarrow 1{\leq}mu\ m\ (carac\ Pdec) \rightarrow mu\ m\ f == mu\ m\ g$.

count the number of elements between 0 and n-1 which satisfy P
Fixpoint *nb_elts* $(P{:}nat \rightarrow Prop)(Pdec : dec\ P)(n{:}nat)\ \{struct\ n\} : nat :=$
  match n with
  $0 \Rightarrow 0\%nat$
  $|\ S\ n \Rightarrow if\ Pdec\ n\ then\ (S\ (nb\_elts\ Pdec\ n))\ else\ (nb\_elts\ Pdec\ n)$
  end.

the probability for a random number between 0 and n to satisfy P is equal to the number of elements below n which satisfy P divided by n+1

Lemma *random_carac* : $\forall$ $(P{:}nat \rightarrow Prop)(Pdec : dec\ P)(n{:}nat)$,
  $random\ n\ (carac\ Pdec) == (nb\_elts\ Pdec\ (S\ n))\ */\ [1/]1{+}n$.

Lemma *mu_carac_inv* : $\forall$ $(A{:}Type)(P{:}A{\rightarrow}Prop)(Pdec{:}dec\ P)(notPdec : dec\ (fun\ x \Rightarrow \neg\ (P\ x)))$
  $(m : distr\ A)$, $mu\ m\ (carac\ Pdec) == mu\ m\ (finv\ (carac\ notPdec))$.

Section *SigmaFinite*.
Variable $A{:}Type$.
Variable $decA : \forall\ x\ y{:}A, \{x{=}y\}{+}\{\tilde{}\,x{=}y\}$.

Section *RandomFinite*.

11.2 Uniform measure on finite sets

11.2.1 Distribution for *random_fin P* over $\{k : nat | k \leq n\}$

The distribution associated to *random_fin P* is $f \mapsto \Sigma_{a \in P} \frac{f(a)}{n+1}$ with $n+1$ the size of P we cannot factorize $\frac{1}{n+1}$ because of possible overflow

Fixpoint *sigma_fin* $(f{:}A{\to}U)(P{:}A{\to}Prop)(FP{:}finite\ P)\{struct\ FP\}{:}U :=$
match FP with
| $(fin\_eq\_empty\ eq) \Rightarrow 0$
| $(fin\_eq\_add\ x\ Q\ nQx\ FQ\ eq) \Rightarrow (f\ x) + sigma\_fin\ f\ FQ$
end.

Definition *retract_fin* $(P{:}A{\to}Prop)\ (f{:}A{\to}U) :=$
$\forall\ Q\ (FQ{:}\ finite\ Q),\ incl\ Q\ P \to \forall\ x,\ \tilde{}(Q\ x) \to (P\ x) \to f\ x \leq [1\text{-}](sigma\_fin\ f\ FQ).$

Lemma *retract_fin_incl* : $\forall\ P\ Q\ f,\ retract\_fin\ P\ f \to incl\ Q\ P \to retract\_fin\ Q\ f.$

Lemma *sigma_fin_monotonic* : $\forall\ (f\ g : A \to U)(P{:}A{\to}Prop)(FP{:}finite\ P),$
$(\forall\ x,\ P\ x \to (f\ x) {<=} (g\ x)) {->}\ sigma\_fin\ f\ FP \leq sigma\_fin\ g\ FP.$

Lemma *sigma_fin_eq_compat* :
$\forall\ (f\ g : A \to U)(P{:}A{\to}Prop)(FP{:}finite\ P),$
$(\forall\ x,\ P\ x \to (f\ x) {==} (g\ x)) {->}\ sigma\_fin\ f\ FP == sigma\_fin\ g\ FP.$

Lemma *retract_fin_le* : $\forall\ (P{:}A{\to}Prop)\ (f\ g{:}A{\to}U),$
$(\forall\ x,\ P\ x \to f\ x \leq g\ x) \to retract\_fin\ P\ g \to retract\_fin\ P\ f.$

Lemma *sigma_fin_mult* : $\forall\ (f : A \to U)\ c\ (P{:}A{\to}Prop)(FP{:}finite\ P),$
$retract\_fin\ P\ f \to sigma\_fin\ (fun\ k \Rightarrow c \times f\ k)\ FP == c \times sigma\_fin\ f\ FP.$

Lemma *sigma_fin_plus* : $\forall\ (f\ g{:}\ A \to U)\ (P{:}A{\to}Prop)(FP{:}finite\ P),$
$sigma\_fin\ (fun\ k \Rightarrow f\ k + g\ k)\ FP == sigma\_fin\ f\ FP + sigma\_fin\ g\ FP.$

Lemma *sigma_fin_prod_maj* :
$\forall\ (f\ g : A \to U)(P{:}A{\to}Prop)(FP{:}finite\ P),$
$sigma\_fin\ (fun\ k \Rightarrow f\ k \times g\ k)\ FP \leq sigma\_fin\ f\ FP.$

Lemma *sigma_fin_prod_le* :
$\forall\ (f\ g : A \to U)\ (c{:}U)\ ,\ (\forall\ k,\ f\ k \leq c) \to \forall\ (P{:}A{\to}Prop)(FP{:}finite\ P),$
$retract\_fin\ P\ g \to sigma\_fin\ (fun\ k \Rightarrow f\ k \times g\ k)\ FP \leq c \times sigma\_fin\ g\ FP.$

Lemma *sigma_fin_prod_ge* :
$\forall\ (f\ g : A \to U)\ (c{:}U)\ ,\ (\forall\ k,\ c \leq f\ k) \to \forall\ (P{:}A{\to}Prop)(FP{:}finite\ P),$
$retract\_fin\ P\ g \to c \times sigma\_fin\ g\ FP \leq sigma\_fin\ (fun\ k \Rightarrow f\ k \times g\ k)\ FP.$
Hint *Resolve sigma_fin_prod_maj sigma_fin_prod_ge sigma_fin_prod_le.*

Lemma *sigma_fin_inv* : $\forall\ (f\ g : A \to U)(P{:}A{\to}Prop)(FP{:}finite\ P),$
$retract\_fin\ P\ f \to$
$[1\text{-}]\ sigma\_fin\ (fun\ k \Rightarrow f\ k \times g\ k)\ FP ==$
$sigma\_fin\ (fun\ k \Rightarrow f\ k \times [1\text{-}]\ g\ k)\ FP + [1\text{-}]\ sigma\_fin\ f\ FP.$

Lemma *sigma_fin_equiv* : $\forall\ f\ P\ Q\ (FP{:}finite\ P)\ (e{:}equiv\ P\ Q),$
$(sigma\_fin\ f\ (fin\_equiv\ e\ FP)) = (sigma\_fin\ f\ FP).$

Lemma *sigma_fin_rem* : $\forall\ f\ P\ (FP{:}finite\ P)\ a,$
$P\ a \to sigma\_fin\ f\ FP == f\ a + sigma\_fin\ f\ (finite\_rem\ decA\ a\ FP).$

Lemma *sigma_fin_incl* : $\forall\ f\ P\ (FP{:}finite\ P)\ Q\ (FQ{:}finite\ Q),$
$(incl\ P\ Q) \to sigma\_fin\ f\ FP \leq sigma\_fin\ f\ FQ.$

Lemma *sigma_fin_unique* : $\forall\ f\ P\ Q\ (FP{:}finite\ P)\ (FQ{:}finite\ Q),\ (equiv\ P\ Q) \to sigma\_fin\ f\ FP == sigma\_fin\ f\ FQ.$

Lemma *sigma_fin_cte* : $\forall\ c\ P\ (FP{:}finite\ P),$
$sigma\_fin\ (fun\ \_ \Rightarrow c)\ FP == (size\ FP)\ {*}/\ c.$

11.2.2 Definition and Properties of *random_fin*

Variable P : $A \rightarrow Prop$.
Variable FP : *finite* P.
Let s:= (*size* FP - 1)%*nat*.
Lemma *pred_size_le* : (*size* FP $\leq S$ s)%*nat*.
Hint *Resolve pred_size_le*.

Lemma *pred_size_eq* : *notempty* P $\rightarrow$ *size* FP $=S$ s.

Definition *random_fin* :M A := *fun* (f:$A \rightarrow U$) $\Rightarrow$ *sigma_fin* (*fun* k $\Rightarrow$ *Unth* s $\times$ f k) *FP*.

Lemma *fnth_retract_fin*:
    $\forall$ n, (*size* $FP \leq S$ n)%*nat* $\rightarrow$ (*retract_fin* P (*fun* _ $\Rightarrow$ [1/]1+n)).

Lemma *random_fin_stable_inv* : *stable_inv random_fin*.

Lemma *random_fin_stable_plus* : *stable_plus random_fin*.

Lemma *random_fin_stable_mult* : *stable_mult random_fin*.

Lemma *random_fin_monotonic* : *monotonic random_fin*.

Definition *Random_fin* : (*distr* A).

Lemma *random_fin_total* : *notempty* P $\rightarrow$ *mu Random_fin* (*f_one* A) == 1.
End *RandomFinite*.

Lemma *random_fin_carac* :
  $\forall$ P Q (FP:*finite* P) (*decQ*:*dec* Q),
    mu (*Random_fin* FP) (*carac decQ*) == *size* (*finite_inter decQ FP*) */ [1/]1+(*size* FP-1)%*nat*.

Lemma *random_fin_P* : $\forall$ P (FP:*finite* P) (*decP*:*dec* P),
    notempty P $\rightarrow$ *mu* (*Random_fin* FP) (*carac decP*) ==1.

End *SigmaFinite*.
End *CaracFun*.

12 Libwp.v : Partial correctness

Require Export *Carac*.

Module *Liberal* (*Univ*:*Universe*).
Import *Univ*.

Module *CP* := (*CaracFun Univ*).
Import *CP*.
Import *CP.RP*.
Import *CP.RP.PP*.
Import *CP.RP.PP.MP*.
Import *CP.RP.PP.MP.UP*.

Section *LibDefProp*.
Variable A : *Type*.

12.1 Definition and basic properties

Definition *lib* (m:*distr* A) : M A := *fun* f $\Rightarrow$ [1-] (*mu* m (*finv* f)).

Lemma *le_mu_lib* : $\forall$ m f, *mu* m f $\leq$ *lib* m f.

Lemma *lib_one* : $\forall$ m, 1 $\leq$ *lib* m (*f_one* A).

Lemma *lib_inv* : $\forall$ m f, *lib* m (*finv* f) == [1-]*mu* m f.

Lemma *lib_monotonic* : $\forall$ m, *monotonic* (*lib* m).

Hint *Resolve lib_one lib_inv lib_monotonic le_mu_lib*.

Lemma *lib_stable_eq* : $\forall$ *m*, *stable_eq* (*lib m*).
Hint *Resolve lib_stable_eq.*

Lemma *mu_lib_le_esp* : $\forall$ *m f g*, *lib m f* & *mu m g* $\leq$ *mu m* (*fesp f g*).
Hint *Resolve mu_lib_le_esp.*

Lemma *le_lib_mu* : $\forall$ *m f*, *lib m f* & *mu m* (*f_one A*) $\leq$ *mu m f*.
Hint *Resolve le_lib_mu.*

Lemma *lib_le_esp* : $\forall$ *m f g*, *lib m f* & *lib m g* $\leq$ *lib m* (*fesp f g*).
Hint *Resolve lib_le_esp.*

Lemma *lib_plus_left* : $\forall$ *m f g*, *fplusok f g* $\rightarrow$ *lib m* (*fplus f g*) == *lib m f* + *mu m g*.

Lemma *lib_plus_right* : $\forall$ *m f g*, *fplusok f g* $\rightarrow$ *lib m* (*fplus f g*) == *mu m f* + *lib m g*.

Definition *okl* (*p* : *U*) (*m* : *distr A*) (*q* : *A* $\rightarrow$ *U*) := *p* $\leq$ *lib m q*.

End *LibDefProp.*
Hint *Resolve lib_one lib_inv lib_monotonic le_mu_lib lib_stable_eq*
mu_lib_le_esp le_lib_mu lib_le_esp lib_plus_left lib_plus_right.

12.2 Rules for liberal constructions of programs

Section *Programs.*
Variables *A B*: *Type.*

Lemma *lib_unit* : $\forall$ (*x*:*A*) (*p* : *A* $\rightarrow$ *U*), *lib* (*Munit x*) *p* == *p x*.

Lemma *lib_let* : $\forall$ (*m* : *distr A*) (*M* : *A* $\rightarrow$ *distr B*) (*p* : *B* $\rightarrow$ *U*),
lib (*Mlet m M*) *p* == *lib m* (*fun x* $\Rightarrow$ *lib* (*M x*) *p*).

Lemma *lib_if* : $\forall$ (*mb*:*distr bool*) (*m1 m2* : *distr A*) (*p* : *A* $\rightarrow$ *U*),
lib (*Mif mb m1 m2*) *p* == *lib mb* (*fun b* $\Rightarrow$ *if b then lib m1 p else lib m2 p*).

12.2.1 Rules for liberal fixpoints

with $\phi(x) = F(\phi)(x)$,

$$\frac{\forall f, (\forall x, p(x) \leq [f](q)) \Rightarrow \forall x, p(x) \leq [F(f)(x)](q)}{\forall x, p\ x \leq [\phi(x)](q)}$$

Section *Fixrules.*

Definition *oklfun* (*p* : *A*$\rightarrow$*U*) (*m* : *A*$\rightarrow$*distr B*) (*q* : *A* $\rightarrow$ *B*$\rightarrow$ *U*) :=
$\forall$ *x*, *p x* $\leq$ *lib* (*m x*) (*q x*).

Definition *uplfun* (*p* : *A*$\rightarrow$*U*) (*m* : *A*$\rightarrow$*distr B*) (*q* : *A* $\rightarrow$ *B*$\rightarrow$ *U*) :=
$\forall$ *x*, *lib* (*m x*) (*q x*) $\leq$ *p x*.

Variable *F* : (*A* $\rightarrow$ *distr B*) $\rightarrow$ *A* $\rightarrow$ *distr B*.

Hypothesis *F_mon* : $\forall$ *f g* : *A* $\rightarrow$ (*distr B*),
($\forall$ *x*, *le_distr* (*f x*) (*g x*)) $\rightarrow$ $\forall$ *x*, *le_distr* (*F f x*) (*F g x*).

Lemma *libfixrule* :
$\forall$ *p q*,
($\forall$ (*f*:*A*$\rightarrow$*distr B*), *oklfun p f q* $\rightarrow$ *oklfun p* (*fun x* $\Rightarrow$ *F f x*) *q*)
$\rightarrow$ *oklfun p* (*Mfix F F_mon*) *q*.

Section *UplibFixRule.*
Variable *p* : *A* $\rightarrow$ *nat* $\rightarrow$ *U*.
Hypothesis *p1* : $\forall$ *x*, *p x O* == 1.
Variable *q* : *A* $\rightarrow$ *B* $\rightarrow$ *U*.

Lemma *up_libfixrule* :
($\forall$ (*i*:*nat*)(*f*:*A*$\rightarrow$*distr B*), *uplfun* (*fun x* $\Rightarrow$ *p x i*) *f q*
$\rightarrow$ *uplfun* (*fun x* $\Rightarrow$ *p x* (*S i*)) (*fun x* $\Rightarrow$ *F f x*) *q*)
$\rightarrow$ *uplfun* (*fun x* $\Rightarrow$ *glb* (*p x*)) (*Mfix F F_mon*) *q*.

End *UplibFixRule.*

12.3 Case the post-expectation is transformed in a functorial way

12.3.1 Invariant rules

Section *Fix_nuF*.

Variable $nuF : (A \to U) \to A \to U$.

Hypothesis *nuF_mon* : *Fmonotonic nuF*.

Variable $q : A \to B \to U$.

Lemma *nuF_stable* : *Fstable nuF*.

Hypothesis *F_nuF_eq* :
$\forall$ *f x*, *lib* (*F f x*) (*q x*) == *nuF* (*fun y* ⇒ *lib* (*f y*) (*q y*)) *x*.

Lemma *nufix_lib* : $\forall$ *x*, *nufix nuF x* == *lib* (*Mfix F F_mon x*) (*q x*).
Hint *Resolve nufix_lib*.

Lemma *nuF_le* : $\forall$ *f*, *fle f* (*nuF f*)
→ $\forall$ *x*, *f x* ≤ *lib* (*Mfix F F_mon x*) (*q x*).

Lemma *nuF_muF_le* : $\forall$ *f*, *fle f* (*nuF f*)
→ $\forall$ *x*, *f x* & *pterm F F_mon x* ≤ *mu* (*Mfix F F_mon x*) (*q x*).
Hint *Resolve nuF_muF_le*.

Lemma *muF_pterm_le* :
$\forall$ *f*, *fle* (*fplus f* (*finv* (*pterm F F_mon*))) (*nuF* (*fplus f* (*finv* (*pterm F F_mon*))))
→ *fle f* (*pterm F F_mon*) → $\forall$ *x*, *f x* ≤ *mu* (*Mfix F F_mon x*) (*q x*).

End *Fix_nuF*.

12.3.2 Case nuF is parametric in q

Variable $nuF : (A \to B \to U) \to (A \to U) \to A \to U$.
Hypothesis *nuF_mon* : $\forall$ *q*, *Fmonotonic* (*nuF q*).

Hypothesis *nuF_q_monotonic* :
$\forall$ *q1 q2 f*, ($\forall$ *x y*, *q1 x y* ≤ *q2 x y*) → *fle* (*nuF q1 f*) (*nuF q2 f*).

Lemma *nuF_q_eq_stable* :
$\forall$ *q1 q2 f*, ($\forall$ *x y*, *q1 x y* == *q2 x y*) → *feq* (*nuF q1 f*) (*nuF q2 f*).

Variable $muF : (A \to B \to U) \to (A \to U) \to A \to U$.

Hypothesis *muF_mon* : $\forall$ *q*, *Fmonotonic* (*muF q*).

Hypothesis *nuF_plus* : $\forall$ *q1 q2 f1 f2*,
feq (*nuF* (*fun x y* ⇒ *q1 x y* + *q2 x y*) (*fplus f1 f2*)) (*fplus* (*muF q1 f1*) (*nuF q2 f2*)).

Hypothesis *nuF_mult* : $\forall$ *a q f*,
feq (*nuF* (*fun x y* ⇒ *a* × (*q x y*)) (*fmult a f*)) (*fmult a* (*nuF q f*)).

Hypothesis *nuF_inv* : $\forall$ *q f*,
feq (*nuF* (*fun x y* ⇒ [1-](*q x y*)) (*finv f*)) (*finv* (*muF q f*)).

Hypothesis *muF_mult* : $\forall$ *a q f*,
feq (*muF* (*fun x y* ⇒ *a* × (*q x y*)) (*fmult a f*)) (*fmult a* (*muF q f*)).

Hypothesis *muF_q_monotonic* :
$\forall$ *q1 q2 f*, ($\forall$ *x y*, *q1 x y* ≤ *q2 x y*) → *fle* (*muF q1 f*) (*muF q2 f*).

Hypothesis *F_muF_eq_one* :
$\forall$ *f x*, ($\forall$ *y*, *le_distr* (*f y*) (*Mfix F F_mon y*)) → *mu* (*F f x*) (*f_one B*) == *muF* (*fun* (*x*:*A*) ⇒ *f_one B*) (*fun y* ⇒ *mu* (*f y*) (*f_one B*)) *x*.

Hypothesis *F_nuF_eq_one* :

$\forall f\ x, (\forall y, le\_distr\ (f\ y)\ (Mfix\ F\ F\_mon\ y)) \rightarrow lib\ (F\ f\ x)\ (f\_one\ B) == nuF\ (fun\ (x{:}A) \Rightarrow f\_one\ B)\ (fun\ y \Rightarrow lib\ (f\ y)\ (f\_one\ B))\ x.$

Hypothesis $muF\_cont : Fcontlub\ (muF\ (fun\ (x{:}A) \Rightarrow f\_one\ B)).$

Section $InvariantTerm.$

Variable $q : A \rightarrow B \rightarrow U.$

Hypothesis F_nuF_eq :
$\forall f\ x, lib\ (F\ f\ x)\ (q\ x) == nuF\ q\ (fun\ y \Rightarrow lib\ (f\ y)\ (q\ y))\ x.$

Lemma $muF\_pterm\_le\_inv$:
$\forall f, fle\ f\ (muF\ q\ f)$
$\rightarrow fle\ f\ (pterm\ F\ F\_mon) \rightarrow \forall x, f\ x \leq mu\ (Mfix\ F\ F\_mon\ x)\ (q\ x).$

End $InvariantTerm.$

Lemma $muF\_pterm\_le\_mult$:
$\forall a\ f, fle\ f\ (muF\ (fun\ (x{:}A)\ (y{:}B) \Rightarrow 1)\ f) \rightarrow$
$(\forall f\ x, lib\ (F\ f\ x)\ (fun\ \_{:}\ B \Rightarrow a \times 1) ==$
$nuF\ (fun\ (x{:}A)\ (y{:}B) \Rightarrow a \times 1)\ (fun\ y \Rightarrow lib\ (f\ y)\ (fun\ \_{:}\ B \Rightarrow a \times 1))\ x)$
$\rightarrow \neg\ 0{==}a \rightarrow fle\ (fmult\ a\ f)\ (pterm\ F\ F\_mon) \rightarrow fle\ f\ (pterm\ F\ F\_mon).$

Lemma $muF\_pterm\_le\_inv\_mult$:
$\forall q\ a\ f, fle\ f\ (muF\ q\ f) \rightarrow$
$(\forall f\ x, lib\ (F\ f\ x)\ (q\ x) == nuF\ q\ (fun\ y \Rightarrow lib\ (f\ y)\ (q\ y))\ x) \rightarrow$
$(\forall f\ x, lib\ (F\ f\ x)\ (fun\ \_{:}\ B \Rightarrow a \times 1) ==$
$nuF\ (fun\ (x{:}A)\ (y{:}B) \Rightarrow a \times 1)\ (fun\ y \Rightarrow lib\ (f\ y)\ (fun\ \_{:}\ B \Rightarrow a \times 1))\ x) \rightarrow$
$\neg\ 0{==}a \rightarrow$
$fle\ (fmult\ a\ f)\ (pterm\ F\ F\_mon) \rightarrow \forall x, f\ x \leq mu\ (Mfix\ F\ F\_mon\ x)\ (q\ x).$

12.4 Termination

Section $Termination.$

Variable $next : A \rightarrow Ndistr\ A.$

Definition $Facc\ (t{:}A{\rightarrow}U) := fun\ (x{:}A) \Rightarrow nu\ (next\ x)\ t.$

Lemma $Facc\_monotonic : Fmonotonic\ Facc.$
Hint $Resolve\ Facc\_monotonic.$

Lemma $Facc\_continuous : Fcontlub\ Facc.$
Hint $Resolve\ Facc\_continuous.$

Definition $acc := mufix\ Facc.$

Lemma $acc\_sup : \forall x, nu\ (next\ x)\ acc \leq acc\ x.$

Lemma $prob\_acc : \forall f : A \rightarrow U,$
$(\forall x, nu\ (next\ x)\ f \leq f\ x) \rightarrow fle\ acc\ f.$

Lemma $distr\_acc : \forall (f : A \rightarrow distr\ B)\ (q{:}A \rightarrow B \rightarrow U),$
$okfun\ (fun\ x \Rightarrow nu\ (next\ x)\ (fun\ y \Rightarrow mu\ (f\ y)\ (q\ y)))\ f\ q \rightarrow okfun\ acc\ f\ q.$

12.5 Results on termination

Section $Wfterm.$

Variable $R : A \rightarrow A \rightarrow Prop.$
Hypothesis $term\_next : \forall x, 1{\leq}\ nu\ (next\ x)\ (f\_one\ A).$

12.5.1 First result

The distribution (next x) always gives values such that (R x y)
Section *Result1*.
Hypothesis *support_next* :
$\forall\ x\ f\ g, (\forall\ y, R\ y\ x \rightarrow f\ y \leq g\ y) \rightarrow nu\ (next\ x)\ f \leq nu\ (next\ x)\ g.$

Lemma *acc_next_term* : $\forall\ x, Acc\ R\ x \rightarrow 1 \leq acc\ x.$

Lemma *wf_next_term* : $(well\_founded\ R) \rightarrow \forall\ x, 1 \leq acc\ x.$
End *Result1*.

12.5.2 Second result

The probability (next x) gives values such that (R x y) is greater than 1

Hypothesis *Rdec* : $\forall\ x, dec\ (fun\ y \Rightarrow R\ y\ x).$
Lemma *acc_almost_term* :
$\forall\ x, Acc\ R\ x \rightarrow (\forall\ x, 1 \leq nu\ (next\ x)\ (carac\ (Rdec\ x))) \rightarrow 1 \leq acc\ x.$

Lemma *wf_almost_term* :
$(well\_founded\ R) \rightarrow (\forall\ x, 1 \leq nu\ (next\ x)\ (carac\ (Rdec\ x))) \rightarrow \forall\ x, 1 \leq acc\ x.$

End *Wfterm*.
End *Termination*.

End *Fixrules*.
End *Programs*.
Hint *Resolve lib_unit lib_let lib_if*.

End *Liberal*.
Require Export *Carac*.
Require *Arith*.

Module *Ycart* (*Univ*:*Universe*).

Section *UniformSec*.

13 Ycart.v: Axiomatisation of the uniform measure

13.1 Interval [0,x]

Hypothesis *Ule_dec* : $\forall\ a\ b, \{a \leq b\}+\{b<a\}.$

Definition *inf* $(a : U) :=$ *carac* $(fun\ x \Rightarrow Ule\_dec\ x\ a).$

Variable *uniform* : *distr U*.
Hypothesis *uniform_inf* : $\forall\ a, mu\ uniform\ (inf\ a) == a.$

Lemma *uniform_one* : $mu\ uniform\ (f\_one\ U) == 1.$

Lemma *uniform_inv_inf* : $\forall\ a, mu\ uniform\ (finv\ (inf\ a)) == [1\text{-}]\ a.$

Hint *Resolve uniform_inf uniform_inv_inf uniform_one*.

14 Ycart.v: An exemple of partial termination

14.1 Program giving an example of partiality

given a function F : int -> U

```
let rec ycart x = if uniform < F x then x else ycart (x+1)
```

The probability of termination is $1 - \prod_{k=x}^{\infty}(1 - F(k))$
Variable $F : nat \to U$.

Definition *FYcart* (*f*:*nat* → *distr nat*) *n* :=
Mlet uniform (*fun x* ⇒ *if Ule_dec x* (*F n*) *then Munit n else f* (*S n*)).

Lemma *FYcart_mon* : ∀ *f g* : *nat* → *distr nat*,
(∀ *n*, *le_distr* (*f n*) (*g n*)) → ∀ *n*, *le_distr* (*FYcart f n*) (*FYcart g n*).

Definition *Ycart* : *nat* → *distr nat* := *Mfix FYcart FYcart_mon*.

14.2 Properties of Ycart

Lemma *FYcart_val* : ∀ *q*:*nat*→*U*, ∀ *f x*,
mu (*FYcart f x*) *q* == *F x* × *q x* + [1-](*F x*) × *mu* (*f* (*S x*)) *q*.

Definition *P* (*x k* : *nat*) := *prod* (*fun i* ⇒ [1-]*F* (*x*+*i*)) *k*.

Definition *p* (*x*:*nat*) (*n*:*nat*) := *sigma* (*fun k* ⇒ *F* (*x*+*k*) × *P x k*) *n*.

Lemma *P_prod* : ∀ *x k*, *F* (*x*+*k*) × *P x k* == *P x k* - *P x* (*S k*).
Hint *Resolve P_prod*.

Lemma *p_diff* : ∀ *x n*, *p x n* == [1-] *P x n*.
Hint *Resolve p_diff*.

Lemma *p_lub* : ∀ *x*, *lub* (*p x*) == [1-] *prod_inf* (*fun i* ⇒ [1-]*F* (*x*+*i*)).
Hint *Resolve p_lub*.

Lemma *p_equation* : ∀ *x n*, *p x* (*S n*) == *F x* + [1-](*F x*) × *p* (*S x*) *n*.
Hint *Resolve p_equation*.

Lemma *Ycart_term1* : ∀ *x*, *mu* (*Ycart x*) (*f_one nat*) == [1-] *prod_inf* (*fun i* ⇒ [1-]*F* (*x*+*i*)).

A shorter proof using mu (Ycart x) (f_one nat) = mu h. muYcart h x

Lemma *Ycart_term2* : ∀ *x*, *mu* (*Ycart x*) (*f_one nat*) == [1-] *prod_inf* (*fun i* ⇒ [1-]*F* (*x*+*i*)).

Lemma *le_dec* : ∀ *x*, *dec* (*fun y* ⇒ *le y x*).

Lemma *lt_dec* : ∀ *x*, *dec* (*fun y* ⇒ *lt y x*).

Lemma *gt_dec* : ∀ *x*, *dec* (*lt x*).

Lemma *Ycart_ltx* : ∀ *x*, *mu* (*Ycart x*) (*carac* (*lt_dec x*)) ≤ 0.

Lemma *Ycart_eqx* : ∀ *x*, *mu* (*Ycart x*) (*carac* (*eq_nat_dec x*)) == *F x*.

End *UniformSec*.
End *Ycart*.

15 Nelist.v: A general theory of non empty lists on Type

Section *NELIST*.

Variable *A* : *Type*.

Inductive *nelist* : *Type* :=
singl : *A* → *nelist* | *add* : *A* → *nelist* → *nelist*.

Definition *hd* (*l*:*nelist*) : *A* :=
match l with (*singl a*) ⇒ *a* | (*add a _*) ⇒ *a end*.

Fixpoint *app* (l m : *nelist*) {*struct* l} : *nelist* :=
match l *with* (*singl* a) ⇒ *add* a m | (*add* a *l1*) ⇒ *add* a (*app* *l1* m) *end*.

Fixpoint *rev_app* (l m : *nelist*) {*struct* l} : *nelist* :=
match l *with* (*singl* a) ⇒ *add* a m | (*add* a *l1*) ⇒ *rev_app* *l1* (*add* a m) *end*.

Definition *rev* (l:*nelist*) : *nelist* :=
match l *with* (*singl* a) ⇒ l | (*add* a *l1*) ⇒ *rev_app* *l1* (*singl* a) *end*.

Lemma *app_assoc* : ∀ *l1* *l2* *l3*, *app* *l1* (*app* *l2* *l3*) = *app* (*app* *l1* *l2*) *l3*.

Hint *Resolve* *app_assoc*.

Lemma *rev_app_rev* : ∀ l m, *rev_app* l m = *app* (*rev* l) m.

Hint *Resolve* *rev_app_rev*.

Lemma *rev_app_app_rev* : ∀ l m, *rev* (*rev_app* l m) = *app* (*rev* m) l.

Lemma *rev_rev* : ∀ l, *rev* (*rev* l) = l.

Lemma *rev_app_distr* : ∀ l m, *rev* (*app* l m) = *app* (*rev* m) (*rev* l).

Hint *Resolve* *rev_rev* *rev_app_distr*.

Lemma *hd_app* : ∀ l m, *hd* (*app* l m) = *hd* l.

Hint *Resolve* *hd_app*.

Lemma *hd_rev_add* : ∀ a l, *hd* (*rev* (*add* a l)) = *hd* (*rev* l).
Hint *Resolve* *hd_rev_add*.

End *NELIST*.

16 Transitions.v: Probabilistic Deterministic Transition System

Require Export *Prog*.

Module *PTS*(*Univ*:*Universe*).
Module *RP* := (*Rules* *Univ*).
Section *TRANSITIONS*.

Variable A : *Type*.

16.1 One step of probabilistic transition

Variable *step* : A → *distr* A.

16.2 Extension to distributions on sequences of length k

Require Export *Nelist*.

Definition *add_step* (*start* : *distr* (*nelist* A)) : M (*nelist* A) :=
fun f ⇒ *mu* *start* (*fun* l ⇒ (*mu* (*step* (*hd* l)) (*fun* x ⇒ (f (*add* x l))))).

Lemma *add_step_stable_inv* : ∀ (*start* : *distr* (*nelist* A)), *stable_inv* (*add_step* *start*).

Lemma *add_step_stable_plus* : ∀ (*start* : *distr* (*nelist* A)), *stable_plus* (*add_step* *start*).

Lemma *add_step_stable_mult* : ∀ (*start* : *distr* (*nelist* A)), *stable_mult* (*add_step* *start*).

Lemma *add_step_monotonic* : ∀ (*start* : *distr* (*nelist* A)), *monotonic* (*add_step* *start*).

Definition *Add_step* : (*distr* (*nelist* A)) → (*distr* (*nelist* A)).

Definition of the measure
Fixpoint *path* (k:*nat*) (s : A) {*struct* k} : *distr* (*nelist* A) :=
match k *with*

$O \Rightarrow Munit\ (singl\ s)$
$|(S\ p) \Rightarrow Add\_step\ (path\ p\ s)$
end.

The opposite view of composition starting from one step

Lemma $path\_unfold : \forall\ k\ s\ f,$
$mu\ (path\ (S\ k)\ s)\ f == mu\ (step\ s)\ (fun\ x \Rightarrow mu\ (path\ k\ x)\ (fun\ l \Rightarrow f\ (app\ l\ (singl\ s))))$.

End *TRANSITIONS.*

End *PTS.*

17 Bernoulli.v: Simulating Bernoulli and Binomial distributions

Require Export *Prog.*
Require Export *Prelude.*

Module *Bernoulli* (*Univ*:*Universe*).
Module *RP* := (*Rules Univ*).

17.1 Program for computing a Bernoulli distribution

bernoulli p gives true with probability p and false with probability (1-p)

```
let rec bernoulli x = 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
```

Hypothesis $dec\_demi : \forall\ x :\ U, \{x < [1/2]\}+\{[1/2] \leq x\}$.

Definition $Fbern\ (f\colon\ U \to distr\ bool)\ (p\colon U) :=$
$Mif\ Flip$
$(if\ dec\_demi\ p\ then\ Munit\ false\ else\ f\ (p\ \&\ p))$
$(if\ dec\_demi\ p\ then\ f\ (p + p)\ else\ Munit\ true)$.

Lemma $Fbern\_mon : \forall\ f\ g :\ U \to distr\ bool,$
$(\forall\ n,\ le\_distr\ (f\ n)\ (g\ n)) \to \forall\ n,\ le\_distr\ (Fbern\ f\ n)\ (Fbern\ g\ n)$.

Definition $bernoulli :\ U \to distr\ bool := Mfix\ Fbern\ Fbern\_mon$.

17.2 Properties of the Bernoulli program

17.2.1 Proofs using fixpoint rules

Definition $Mubern\ (q\colon\ bool \to U)\ (bern :\ U \to U)\ (p\colon U) :=$
$if\ dec\_demi\ p\ then\ [1/2]^*(q\ false)+[1/2]^*(bern\ (p+p))$
$else\ [1/2]^*(bern\ (p\&p)) + [1/2]^*(q\ true)$.

Lemma $Mubern\_eq : \forall\ (q\colon\ bool \to U)\ (f\colon U \to distr\ bool)\ (p\colon U),$
$mu\ (Fbern\ f\ p)\ q == Mubern\ q\ (fun\ y \Rightarrow mu\ (f\ y)\ q)\ p$.

Lemma $Mubern\_mon : \forall\ (q\colon\ bool \to U),\ Fmonotonic\ (Mubern\ q)$.
Hint *Resolve Mubern_mon Mubern_eq.*

Lemma $Bern\_eq$:
$\forall\ q :\ bool \to U, \forall\ p,\ mu\ (bernoulli\ p)\ q == mufix\ (Mubern\ q)\ p$.
Hint *Resolve Bern_eq.*

Lemma $Bern\_commute : \forall\ q :\ bool \to U,$
$mu\_muF\_commute\_le\ Fbern\ Fbern\_mon\ (fun\ (x\colon U) \Rightarrow q)\ (Mubern\ q)$.
Hint *Resolve Bern_commute.*

Lemma $Bern\_term : \forall\ p,\ mu\ (bernoulli\ p)\ (f\_one\ bool) == 1$.
Hint *Resolve Bern_term.*

17.2.2 p is an invariant of Mubern qtrue

Lemma *MuBern_true* : $\forall$ *p*, *Mubern B2U* (*fun q* $\Rightarrow$ *q*) *p* == *p*.
Hint *Resolve MuBern_true*.

Lemma *MuBern_false* : $\forall$ *p*, *Mubern* (*finv B2U*) (*finv* (*fun q* $\Rightarrow$ *q*)) *p* == [1-]*p*.
Hint *Resolve MuBern_false*.

Lemma *Bern_true* : $\forall$ *p*, *mu* (*bernoulli p*) *B2U* == *p*.

Lemma *Bern_false* : $\forall$ *p*, *mu* (*bernoulli p*) *NB2U* == [1-]*p*.

Lemma *Mubern_inv* : $\forall$ (*q*: *bool* $\rightarrow$ *U*) (*f*:*U* $\rightarrow$ *U*) (*p*:*U*),
Mubern (*finv q*) (*finv f*) *p* == [1-] *Mubern q f p*.

17.2.3 Proofs using lubs

Invariant *pmin p* $pmin(p)(n) = p - \frac{1}{2^n}$

Property : $\forall p, p \leq \langle \text{bernoulli } p \rangle (\textbf{result} = \text{true})$

Definition *qtrue* (*p*:*U*) := *B2U*.
Definition *qfalse* (*p*:*U*) := *NB2U*.

Lemma *bernoulli_true* : *okfun* (*fun p* $\Rightarrow$ *p*) *bernoulli qtrue*.

Property : $\forall p, 1 - p \leq \langle \text{bernoulli } p \rangle (\textbf{result} = \text{false})$

Lemma *bernoulli_false* : *okfun* (*fun p* $\Rightarrow$ [1-] *p*) *bernoulli qfalse*.

Probability for the result of (bernoulli p) to be true is exactly p

Lemma *qtrue_qfalse_inv* : $\forall$ (*b*:*bool*) (*x*:*U*), *qtrue x b* == [1-] (*qfalse x b*).

Lemma *bernoulli_eq_true* : $\forall$ *p*, *mu* (*bernoulli p*) (*qtrue p*) == *p*.

Lemma *bernoulli_eq_false* : $\forall$ *p*, *mu* (*bernoulli p*) (*qfalse p*)== [1-]*p*.

Lemma *bernoulli_eq* : $\forall$ *p f*, *mu* (*bernoulli p*) *f* == *p* $\times$ *f true* + ([1-]*p*) $\times$ *f false*.

Lemma *bernoulli_total* : $\forall$ *p* , *mu* (*bernoulli p*) (*f_one bool*)==1.

17.3 Binomial distribution

(binomial p n) gives k with probability $C_k^n p^k (1-p)^{n-k}$

17.3.1 Definition and properties of binomial coefficients

Fixpoint *comb* (*k n*:*nat*) {*struct n*} : *nat* :=
match n with O $\Rightarrow$ *match k with O* $\Rightarrow$ (1%*nat*) | (*S l*) $\Rightarrow$ *O end*
| (*S m*) $\Rightarrow$ *match k with O* $\Rightarrow$ (1%*nat*)
| (*S l*) $\Rightarrow$ ((*comb l m*) + (*comb k m*))%*nat end*
end.

Lemma *comb_0_n* : $\forall$ *n*, *comb* 0 *n* = 1%*nat*.

Lemma *comb_not_le* : $\forall$ *n k*, *le* (*S n*) *k* $\rightarrow$ *comb k n*=0%*nat*.

Lemma *comb_Sn_n* : $\forall$ *n*, *comb* (*S n*) *n* =0%*nat*.

Lemma *comb_n_n* : $\forall$ *n*, *comb n n* = (1%*nat*).

Lemma *comb_1_Sn* : $\forall$ *n*, *comb* 1 (*S n*) = (*S n*).

Lemma *comb_inv* : $\forall$ *n k*, (*k*$\leq$*n*)%*nat* $\rightarrow$ *comb k n* = *comb* (*n*-*k*) *n*.

Lemma *comb_n_Sn* : $\forall$ *n*, *comb n* (*S n*) = (*S n*).

Definition *fc* (*p*:*U*)(*n k*:*nat*) := (*comb k n*) */ (*p*^*k* $\times$ ([1-]*p*)^(*n*-*k*)).

Lemma *fcp_0* : $\forall$ *p n*, *fc p n O* == ([1-]*p*)^*n*.

Lemma *fcp_n* : $\forall$ *p n*, *fc p n n* == *p*^*n*.

Lemma *fcp_not_le* : $\forall$ *p n k*, (*S n* $\leq$ *k*)%*nat* $\rightarrow$ *fc p n k* == 0.

Lemma *fc0* : $\forall$ *n k*, *fc* 0 *n* (*S k*) == 0.
Hint *Resolve fc0*.

Add Morphism fc : *fc_eq_compat*.

Hint *Resolve fc_eq_compat*.

Lemma *sigma_fc0* : $\forall$ *n k*, *sigma* (*fc* 0 *n*) (*S k*) ==1.

Lemma *retract_class* : $\forall$ *f n*, *class* (*retract f n*).
Hint *Resolve retract_class*.

Lemma *fc_retract* :
  $\forall$ *p n*, ([1-]*p*^*n* == *sigma* (*fc p n*) *n*) $\rightarrow$ *retract* (*fc p n*) (*S n*).
Hint *Resolve fc_retract*.

Lemma *fc_Nmult_def* :
  $\forall$ *p n k*, ([1-]*p*^*n* == *sigma* (*fc p n*) *n*) $\rightarrow$ *Nmult_def* (*comb k n*) (*p*^*k* $\times$ ([1-]*p*) ^(*n*-*k*)).
Hint *Resolve fc_Nmult_def*.

Lemma *fcp_S* :
  $\forall$ *p n k*, ([1-]*p*^*n* == *sigma* (*fc p n*) *n*) $\rightarrow$ *fc p* (*S n*) (*S k*) == *p* $\times$ (*fc p n k*) + ([1-]*p*) $\times$ (*fc p n* (*S k*)).

Lemma *sigma_fc_1* : $\forall$ *p n*, ([1-]*p*^*n* == *sigma* (*fc p n*) *n*) $\rightarrow$1==*sigma* (*fc p n*) (*S n*).
Hint *Resolve sigma_fc_1*.

Lemma *Uinv_exp* : $\forall$ *p n*, [1-](*p*^*n*)==*sigma* (*fc p n*) *n*.

Hint *Resolve Uinv_exp*.

Lemma *Nmult_comb* : $\forall$ *p n k*, *Nmult_def* (*comb k n*) (*p* ^ *k* $\times$ ([1-] *p*) ^ (*n* - *k*)).
Hint *Resolve Nmult_comb*.

Definition *qk* (*k n*:*nat*) : *U* := *if eq_nat_dec k n then* 1 *else* 0.

17.3.2 Definition of binomial distribution

Fixpoint *binomial* (*p*:*U*)(*n*:*nat*) {*struct n*}: *distr nat* :=
  match n with O $\Rightarrow$ (*Munit O*)
    | *S m* $\Rightarrow$ *Mlet* (*binomial p m*)
      (*fun x* $\Rightarrow$ *Mif* (*bernoulli p*) (*Munit* (*S x*)) (*Munit x*))
  end.

17.3.3 Properties of binomial distribution

Lemma *binomial_eq_k* :
  $\forall$ *p n k*, *mu* (*binomial p n*) (*qk k*) == *fc p n k*.

End *Bernoulli*.

18 Choice.v: An example of probabilistic choice

Require Export *Prog*.
Module *Choice* (*Univ*:*Universe*).
Module *RP* := (*Rules Univ*).

18.1 Definition of a probabilistic choice

We interpret the probabilistic program p which executes two probabilistic programs p_1 and p_2 and then make a choice between the two computed results.

```
let rec p () = let x = p1 () in let y = p2 () in choice x y
```

Section *CHOICE*.
Variable A : *Type*.
Variables *p1 p2* : *distr A*.
Variable *choice* : $A \rightarrow A \rightarrow A$.
Definition p : *distr A* := *Mlet p1* (*fun x* $\Rightarrow$ *Mlet p2* (*fun y* $\Rightarrow$ *Munit* (*choice x y*))).

18.2 Main result

We estimate the probability for p to satisfy Q given estimations for both p_1 and p_2.

18.2.1 Assumptions

We need extra properties on p_1, p_2 and *choice*.

- p_1 and p_2 terminate with probability 1
- Q value on *choice* is not less than the sum of values of Q on separate elements. If Q is a boolean function it means than if one of x or y satisfies Q then (*choice* x y) will also satisfy Q

Hypothesis *p1_terminates* : (*mu p1* (*f_one A*))==1.
Hypothesis *p2_terminates* : (*mu p2* (*f_one A*))==1.

Variable Q : $A \rightarrow U$.
Hypothesis *choiceok* : $\forall$ x y, Q x + Q y $\leq$ Q (*choice x y*).

18.2.2 Proof of estimation

$$\frac{k_1 \leq \langle p_1 \rangle(Q) \quad k_2 \leq \langle p_2 \rangle(Q)}{k_1(1-k_2)+k_2 \leq \langle p \rangle(Q)}$$

Lemma *choicerule* : $\forall$ *k1 k2*,
 k1 $\leq$ *mu p1 Q* $\rightarrow$ *k2* $\leq$ *mu p2 Q* $\rightarrow$ (*k1* $\times$ ([1-] *k2*) + *k2*) $\leq$ *mu p Q*.

End *CHOICE*.
End *Choice*.

19 IterFlip.v: An example of probabilistic termination

Require Export *Prog*.

Module *IterFlip* (*Univ*:*Universe*).
Module *RP* := (*Rules Univ*).

19.1 Definition of a random walk

We interpret the probabilistic program

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

Require Import *ZArith*.

Definition *Fiter* (*f*: $Z \rightarrow$ (*distr Z*)) (*x*:*Z*) := *Mif Flip* (*f* (*Zsucc x*)) (*Munit x*).

Lemma *Fiter_mon* : $\forall$ *f g* : $Z \rightarrow$ *distr Z*,
 ($\forall$ *n*, *le_distr* (*f n*) (*g n*)) $\rightarrow$ $\forall$ *n*, *le_distr* (*Fiter f n*) (*Fiter g n*).

Definition *iterflip* : $Z \rightarrow$ (*distr Z*) := *Mfix Fiter Fiter_mon*.

19.2 Main result

Probability for *iter* to terminate is 1

19.2.1 Auxiliary function p_n

Definition $p_n = 1 - \frac{1}{2^n}$

Fixpoint $p\ (n : nat) : U := match\ n\ with\ O \Rightarrow 0 \mid (S\ n) \Rightarrow [1/2] \times p\ n + [1/2]\ end.$

Lemma $p\_eq : \forall\ n{:}nat,\ p\ n == [1\text{-}]([1/2]\hat{}\,n).$
Hint *Resolve p_eq.*

Lemma $p\_le : \forall\ n{:}nat,\ [1\text{-}]([1/]1{+}n) \leq p\ n.$

Hint *Resolve p_le.*

Lemma $lim\_p\_one : 1 \leq lub\ p.$

Hint *Resolve lim_p_one.*

19.2.2 Proof of probabilistic termination

Definition $q1\ (z1\ z2{:}Z) := 1.$

Lemma $iterflip\_term : okfun\ (fun\ k \Rightarrow 1)\ iterflip\ q1.$

End *IterFlip.*

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 *Lecture Notes in Computer Science*, Kuressaare, Estonia, July 2006. Springer.

[2] Philippe Audebaud and Christine Paulin-Mohring. Proofs of randomized algorithms in Coq. submitted to Science of Computer Programming, 2007. Extended version of [1].

[3] Dexter Kozen. Semantics of probabilistic programs. *Journal of Computer and System Sciences*, 1981.

[4] Dexter Kozen. A probabilistic PDL. In *15th ACM Symposium on Theory of Computing*, 1983.

[5] Annabelle McIver and Carroll Morgan. *Abstraction, Refinement and Proof for Probabilistic Systems.* Monographs in Computer Science. Springer, 2005.

[6] The Coq Development Team. *The Coq Proof Assistant Reference Manual – Version V8.0*, April 2004. .