

A Verified Compiler for Probability Density Functions

Manuel Eberl, Johannes Hölzl and Tobias Nipkow

March 1, 2024

Abstract

Bhat *et al.* [1] developed an inductive compiler that computes density functions for probability spaces described by programs in a probabilistic functional language. In this work, we implement such a compiler for a modified version of this language within the theorem prover Isabelle and give a formal proof of its soundness w.r.t. the semantics of the source and target language. Together with Isabelle’s code generation for inductive predicates, this yields a fully verified, executable density compiler. The proof is done in two steps: First, an abstract compiler working with abstract functions modelled directly in the theorem prover’s logic is defined and proved sound. Then, this compiler is refined to a concrete version that returns a target-language expression.

A detailed presentation of this work can be found in the first author’s master’s thesis [2].

Contents

1	Density Predicates	2
1.1	Probability Densities	3
1.2	Measure spaces with densities	3
1.3	Probability spaces with densities	4
1.4	Parametrized probability densities	5
1.5	Density in the Giry monad	6
2	Measure Space Transformations	8
3	Source Language Values	10
3.1	Values and stock measures	11
3.2	Measures on states	16
3.3	Equalities of measure embeddings	16
3.4	Monadic operations on values	17
3.5	Lifting of functions	17

4	Built-in Probability Distributions	20
4.1	Bernoulli	20
4.2	Uniform	21
4.3	Gaussian	21
4.4	Poisson	22
5	Source Language Syntax and Semantics	22
5.1	Expressions	22
5.2	Typing	25
5.3	Semantics	26
	5.3.1 Countable types	27
5.4	Semantics	28
5.5	Measurability	30
5.6	Randomfree expressions	32
6	Density Contexts	33
7	Abstract PDF Compiler	41
7.1	Density compiler predicate	41
7.2	Auxiliary lemmas	43
7.3	Soundness proof	45
8	Target Language Syntax and Semantics	46
8.1	General functions on Expressions	51
8.2	Nonnegative expressions	55
8.3	Randomfree expressions	58
8.4	Builtin density expressions	58
8.5	Integral expressions	60
9	Concrete Density Contexts	62
9.1	Definition	62
9.2	Expressions for density context operations	63
10	Concrete PDF Compiler	65
11	Tests	68

1 Density Predicates

```

theory Density-Predicates
imports HOL-Probability.Probability
begin

```

1.1 Probability Densities

definition *is-subprob-density* :: 'a measure \Rightarrow ('a \Rightarrow ennreal) \Rightarrow bool **where**
is-subprob-density M f \equiv (f \in borel-measurable M) \wedge space M \neq {} \wedge
 $(\forall x \in$ space M. f x \geq 0) \wedge ($\int^+ x.$ f x ∂ M) \leq 1

lemma *is-subprob-densityI*[intro]:

\llbracket f \in borel-measurable M; $\bigwedge x. x \in$ space M \implies f x \geq 0; space M \neq {} \rrbracket ; ($\int^+ x.$ f x ∂ M) \leq 1
 \implies *is-subprob-density* M f
 <proof>

lemma *is-subprob-densityD*[dest]:

is-subprob-density M f \implies f \in borel-measurable M
is-subprob-density M f \implies x \in space M \implies f x \geq 0
is-subprob-density M f \implies space M \neq {}
is-subprob-density M f \implies ($\int^+ x.$ f x ∂ M) \leq 1
 <proof>

1.2 Measure spaces with densities

definition *has-density* :: 'a measure \Rightarrow 'a measure \Rightarrow ('a \Rightarrow ennreal) \Rightarrow bool **where**
has-density M N f \longleftrightarrow (f \in borel-measurable N) \wedge space N \neq {} \wedge M = density N f

lemma *has-densityI*[intro]:

\llbracket f \in borel-measurable N; M = density N f; space N \neq {} \rrbracket \implies *has-density* M N f
 <proof>

lemma *has-densityD*:

assumes *has-density* M N f
shows f \in borel-measurable N M = density N f space N \neq {}
 <proof>

lemma *has-density-sets*: *has-density* M N f \implies sets M = sets N

<proof>

lemma *has-density-space*: *has-density* M N f \implies space M = space N

<proof>

lemma *has-density-emeasure*:

has-density M N f \implies X \in sets M \implies emeasure M X = $\int^+ x.$ f x * indicator X x ∂ N
 <proof>

lemma *nn-integral-cong'*: ($\bigwedge x. x \in$ space N =simp \Rightarrow f x = g x) \implies ($\int^+ x.$ f x ∂ N) = ($\int^+ x.$ g x ∂ N)

<proof>

lemma *has-density-emeasure-space*:

$has_density\ M\ N\ f \implies\ emeasure\ M\ (space\ M) = (\int^+ x. f\ x\ \partial N)$
<proof>

lemma *has-density-emeasure-space'*:

$has_density\ M\ N\ f \implies\ emeasure\ (density\ N\ f)\ (space\ (density\ N\ f)) = \int^+ x. f\ x\ \partial N$
<proof>

lemma *has-density-imp-is-subprob-density*:

$\llbracket has_density\ M\ N\ f; (\int^+ x. f\ x\ \partial N) = 1 \rrbracket \implies is_subprob_density\ N\ f$
<proof>

lemma *has-density-imp-is-subprob-density'*:

$\llbracket has_density\ M\ N\ f; prob_space\ M \rrbracket \implies is_subprob_density\ N\ f$
<proof>

lemma *has-density-equal-on-space*:

assumes $has_density\ M\ N\ f \wedge x. x \in space\ N \implies f\ x = g\ x$
shows $has_density\ M\ N\ g$
<proof>

lemma *has-density-cong*:

assumes $\wedge x. x \in space\ N \implies f\ x = g\ x$
shows $has_density\ M\ N\ f = has_density\ M\ N\ g$
<proof>

lemma *has-density-dens-AE*:

$\llbracket AE\ y\ in\ N. f\ y = f'\ y; f' \in borel_measurable\ N; \wedge x. x \in space\ M \implies f'\ x \geq 0; has_density\ M\ N\ f \rrbracket$
 $\implies has_density\ M\ N\ f'$
<proof>

1.3 Probability spaces with densities

lemma *is-subprob-density-imp-has-density*:

$\llbracket is_subprob_density\ N\ f; M = density\ N\ f \rrbracket \implies has_density\ M\ N\ f$
<proof>

lemma *has-subprob-density-imp-subprob-space'*:

$\llbracket has_density\ M\ N\ f; is_subprob_density\ N\ f \rrbracket \implies subprob_space\ M$
<proof>

lemma *has-subprob-density-imp-subprob-space[dest]*:

$is_subprob_density\ M\ f \implies subprob_space\ (density\ M\ f)$
<proof>

definition $has_subprob_density\ M\ N\ f \equiv has_density\ M\ N\ f \wedge subprob_space\ M$

lemma *subprob-space-density-not-empty*: $\text{subprob-space } (\text{density } M f) \implies \text{space } M \neq \{\}$
 ⟨proof⟩

lemma *has-subprob-densityI*:
 $\llbracket f \in \text{borel-measurable } N; M = \text{density } N f; \text{subprob-space } M \rrbracket \implies \text{has-subprob-density } M N f$
 ⟨proof⟩

lemma *has-subprob-densityI'*:
assumes $f \in \text{borel-measurable } N$ $\text{space } N \neq \{\}$
 $M = \text{density } N f$ $(\int^+ x. f x \partial N) \leq 1$
shows $\text{has-subprob-density } M N f$
 ⟨proof⟩

lemma *has-subprob-densityD*:
assumes $\text{has-subprob-density } M N f$
shows $f \in \text{borel-measurable } N$ $\bigwedge x. x \in \text{space } N \implies f x \geq 0$ $M = \text{density } N f$ $\text{subprob-space } M$
 ⟨proof⟩

lemma *has-subprob-density-measurable[measurable-dest]*:
 $\text{has-subprob-density } M N f \implies f \in N \rightarrow_M \text{borel}$
 ⟨proof⟩

lemma *has-subprob-density-imp-has-density*:
 $\text{has-subprob-density } M N f \implies \text{has-density } M N f$ ⟨proof⟩

lemma *has-subprob-density-equal-on-space*:
assumes $\text{has-subprob-density } M N f$ $\bigwedge x. x \in \text{space } N \implies f x = g x$
shows $\text{has-subprob-density } M N g$
 ⟨proof⟩

lemma *has-subprob-density-cong*:
assumes $\bigwedge x. x \in \text{space } N \implies f x = g x$
shows $\text{has-subprob-density } M N f = \text{has-subprob-density } M N g$
 ⟨proof⟩

lemma *has-subprob-density-dens-AE*:
 $\llbracket AE y \text{ in } N. f y = f' y; f' \in \text{borel-measurable } N;$
 $\bigwedge x. x \in \text{space } M \implies f' x \geq 0; \text{has-subprob-density } M N f \rrbracket$
 $\implies \text{has-subprob-density } M N f'$
 ⟨proof⟩

1.4 Parametrized probability densities

definition

has-parametrized-subprob-density $M N R f \equiv$
 $(\forall x \in \text{space } M. \text{has-subprob-density } (N x) R (f x)) \wedge \text{case-prod } f \in$
 $\text{borel-measurable } (M \otimes_M R)$

lemma *has-parametrized-subprob-densityI*:
assumes $\bigwedge x. x \in \text{space } M \implies N x = \text{density } R (f x)$
assumes $\bigwedge x. x \in \text{space } M \implies \text{subprob-space } (N x)$
assumes $\text{case-prod } f \in \text{borel-measurable } (M \otimes_M R)$
shows *has-parametrized-subprob-density* $M N R f$
 $\langle \text{proof} \rangle$

lemma *has-parametrized-subprob-densityD*:
assumes *has-parametrized-subprob-density* $M N R f$
shows $\bigwedge x. x \in \text{space } M \implies N x = \text{density } R (f x)$
and $\bigwedge x. x \in \text{space } M \implies \text{subprob-space } (N x)$
and $[\text{measurable-dest}]$: $\text{case-prod } f \in \text{borel-measurable } (M \otimes_M R)$
 $\langle \text{proof} \rangle$

lemma *has-parametrized-subprob-density-integral*:
assumes *has-parametrized-subprob-density* $M N R f$ $x \in \text{space } M$
shows $(\int^+ y. f x y \partial R) \leq 1$
 $\langle \text{proof} \rangle$

lemma *has-parametrized-subprob-density-cong*:
assumes $\bigwedge x. x \in \text{space } M \implies N x = N' x$
shows *has-parametrized-subprob-density* $M N R f = \text{has-parametrized-subprob-density}$
 $M N' R f$
 $\langle \text{proof} \rangle$

lemma *has-parametrized-subprob-density-dens-AE*:
assumes $\bigwedge x. x \in \text{space } M \implies AE y \text{ in } R. f x y = f' x y$
 $\text{case-prod } f' \in \text{borel-measurable } (M \otimes_M R)$
has-parametrized-subprob-density $M N R f$
shows *has-parametrized-subprob-density* $M N R f'$
 $\langle \text{proof} \rangle$

1.5 Density in the Giry monad

lemma *emeasure-bind-density*:
assumes $\text{space } M \neq \{\}$ $\bigwedge x. x \in \text{space } M \implies \text{has-density } (f x) N (g x)$
 $f \in \text{measurable } M (\text{subprob-algebra } N) X \in \text{sets } N$
shows $\text{emeasure } (M \gg f) X = \int^+ x. \int^+ y. g x y * \text{indicator } X y \partial N \partial M$
 $\langle \text{proof} \rangle$

lemma *bind-density*:
assumes $\text{sigma-finite-measure } M \text{ sigma-finite-measure } N$
 $\text{space } M \neq \{\}$ $\bigwedge x. x \in \text{space } M \implies \text{has-density } (f x) N (g x)$
and $[\text{measurable}]$: $\text{case-prod } g \in \text{borel-measurable } (M \otimes_M N) f \in \text{measurable}$
 $M (\text{subprob-algebra } N)$

shows $(M \ggg f) = \text{density } N (\lambda y. \int^+ x. g \ x \ y \ \partial M)$
 ⟨proof⟩

lemma *bind-has-density*:

assumes *sigma-finite-measure* M *sigma-finite-measure* N
 $\text{space } M \neq \{\}$ $\bigwedge x. x \in \text{space } M \implies \text{has-density } (f \ x) \ N \ (g \ x)$
case-prod $g \in \text{borel-measurable } (M \otimes_M N)$
 $f \in \text{measurable } M \ (\text{subprob-algebra } N)$
shows $\text{has-density } (M \ggg f) \ N \ (\lambda y. \int^+ x. g \ x \ y \ \partial M)$
 ⟨proof⟩

lemma *bind-has-density'*:

assumes *sfM*: *sigma-finite-measure* M
and *sfR*: *sigma-finite-measure* R
and *not-empty*: $\text{space } M \neq \{\}$ **and** *dens-M*: $\text{has-density } M \ N \ \delta M$
and *dens-f*: $\bigwedge x. x \in \text{space } M \implies \text{has-density } (f \ x) \ R \ (\delta f \ x)$
and *Mδf*: *case-prod* $\delta f \in \text{borel-measurable } (N \otimes_M R)$
and *Mf*: $f \in \text{measurable } N \ (\text{subprob-algebra } R)$
shows $\text{has-density } (M \ggg f) \ R \ (\lambda y. \int^+ x. \delta M \ x \ * \ \delta f \ x \ y \ \partial N)$
 ⟨proof⟩

lemma *bind-has-subprob-density*:

assumes *subprob-space* M *sigma-finite-measure* N
 $\text{space } M \neq \{\}$ $\bigwedge x. x \in \text{space } M \implies \text{has-density } (f \ x) \ N \ (g \ x)$
case-prod $g \in \text{borel-measurable } (M \otimes_M N)$
 $f \in \text{measurable } M \ (\text{subprob-algebra } N)$
shows $\text{has-subprob-density } (M \ggg f) \ N \ (\lambda y. \int^+ x. g \ x \ y \ \partial M)$
 ⟨proof⟩

lemma *bind-has-subprob-density'*:

assumes *has-subprob-density* $M \ N \ \delta M$ $\text{space } R \neq \{\}$ *sigma-finite-measure* R
 $\bigwedge x. x \in \text{space } M \implies \text{has-subprob-density } (f \ x) \ R \ (\delta f \ x)$
case-prod $\delta f \in \text{borel-measurable } (N \otimes_M R)$ $f \in \text{measurable } N \ (\text{subprob-algebra } R)$
shows $\text{has-subprob-density } (M \ggg f) \ R \ (\lambda y. \int^+ x. \delta M \ x \ * \ \delta f \ x \ y \ \partial N)$
 ⟨proof⟩

lemma *null-measure-has-subprob-density*:

$\text{space } M \neq \{\} \implies \text{has-subprob-density } (\text{null-measure } M) \ M \ (\lambda-. \ 0)$
 ⟨proof⟩

lemma *emeasure-has-parametrized-subprob-density*:

assumes *has-parametrized-subprob-density* $M \ N \ R \ f$
assumes $x \in \text{space } M \ X \in \text{sets } R$
shows $\text{emeasure } (N \ x) \ X = \int^+ y. f \ x \ y \ * \ \text{indicator } X \ y \ \partial R$
 ⟨proof⟩

lemma *emeasure-count-space-density-singleton*:

assumes $x \in A$ *has-density* M (*count-space* A) f
shows *emeasure* M $\{x\} = f x$
 ⟨*proof*⟩

lemma *subprob-count-space-density-le-1*:
assumes *has-subprob-density* M (*count-space* A) $f x \in A$
shows $f x \leq 1$
 ⟨*proof*⟩

lemma *has-density-embed-measure*:
assumes *inj*: $inj f$ **and** *inv*: $\bigwedge x. x \in space N \implies f' (f x) = x$
shows *has-density* (*embed-measure* $M f$) (*embed-measure* $N f$) $(\delta \circ f')$ \longleftrightarrow
has-density $M N \delta$
 (**is** *has-density* $?M' ?N' ?\delta' \longleftrightarrow$ *has-density* $M N \delta$)
 ⟨*proof*⟩

lemma *has-density-embed-measure'*:
assumes *inj*: $inj f$ **and** *inv*: $\bigwedge x. x \in space N \implies f' (f x) = x$ **and**
sets-M: $sets M = sets (embed-measure N f)$
shows *has-density* (*distr* $M N f'$) $N (\delta \circ f)$ \longleftrightarrow *has-density* $M (embed-measure$
 $N f) \delta$
 ⟨*proof*⟩

lemma *has-density-embed-measure''*:
assumes *inj*: $inj f$ **and** *inv*: $\bigwedge x. x \in space N \implies f' (f x) = x$ **and**
has-density $M (embed-measure N f) \delta$
shows *has-density* (*distr* $M N f'$) $N (\delta \circ f)$
 ⟨*proof*⟩

lemma *has-subprob-density-embed-measure''*:
assumes *inj*: $inj f$ **and** *inv*: $\bigwedge x. x \in space N \implies f' (f x) = x$ **and**
has-subprob-density $M (embed-measure N f) \delta$
shows *has-subprob-density* (*distr* $M N f'$) $N (\delta \circ f)$
 ⟨*proof*⟩

end

2 Measure Space Transformations

theory *PDF-Transformations*
imports *Density-Predicates*
begin

lemma *not-top-le-1-ennreal[simp]*: $\neg top \leq (1::ennreal)$
 ⟨*proof*⟩

lemma *range-int*: $range int = \{n. n \geq 0\}$
 ⟨*proof*⟩

lemma *range-exp*: $\text{range } (\text{exp} :: \text{real} \Rightarrow \text{real}) = \{x. x > 0\}$
<proof>

lemma *Int-stable-Icc*: $\text{Int-stable } (\text{range } (\lambda(a, b). \{a .. b::\text{real}\}))$
<proof>

lemma *distr-mult-real*:
 assumes $c \neq 0$ *has-density* M *lborel* $(f :: \text{real} \Rightarrow \text{ennreal})$
 shows *has-density* $(\text{distr } M \text{ borel } ((*) c))$ *lborel* $(\lambda x. f (x / c) * \text{inverse } (\text{abs } c))$
 (**is** *has-density* $?M' - ?f'$)
<proof>

lemma *distr-uminus-real*:
 assumes *has-density* M *lborel* $(f :: \text{real} \Rightarrow \text{ennreal})$
 shows *has-density* $(\text{distr } M \text{ borel } \text{uminus})$ *lborel* $(\lambda x. f (- x))$
<proof>

lemma *distr-plus-real*:
 assumes *has-density* M *lborel* $(f :: \text{real} \Rightarrow \text{ennreal})$
 shows *has-density* $(\text{distr } M \text{ borel } ((+) c))$ *lborel* $(\lambda x. f (x - c))$
<proof>

lemma *count-space-uminus*:
 $\text{count-space } UNIV = \text{distr } (\text{count-space } UNIV) (\text{count-space } UNIV) (\text{uminus} ::$
 $'a :: \text{ring} \Rightarrow -)$
<proof>

lemma *count-space-plus*:
 $\text{count-space } UNIV = \text{distr } (\text{count-space } UNIV) (\text{count-space } UNIV) ((+) (c ::$
 $'a :: \text{ring}))$
<proof>

lemma *distr-uminus-ring-count-space*:
 assumes *has-density* M $(\text{count-space } UNIV)$ $(f :: - :: \text{ring} \Rightarrow \text{ennreal})$
 shows *has-density* $(\text{distr } M (\text{count-space } UNIV) \text{uminus}) (\text{count-space } UNIV)$
 $(\lambda x. f (- x))$
<proof>

lemma *distr-plus-ring-count-space*:
 assumes *has-density* M $(\text{count-space } UNIV)$ $(f :: - :: \text{ring} \Rightarrow \text{ennreal})$
 shows *has-density* $(\text{distr } M (\text{count-space } UNIV) ((+) c)) (\text{count-space } UNIV)$
 $(\lambda x. f (x - c))$
<proof>

lemma *subprob-density-distr-real-eq*:
 assumes *dens*: *has-subprob-density* M *lborel* f
 assumes *Mh*: $h \in \text{borel-measurable borel}$
 assumes *Mg*: $g \in \text{borel-measurable borel}$

assumes *measure-eq*:
 $\bigwedge a b. a \leq b \implies \text{emeasure } (\text{distr } (\text{density } \text{lborel } f) \text{ lborel } h) \{a..b\} =$
 $\text{emeasure } (\text{density } \text{lborel } g) \{a..b\}$
shows *has-subprob-density* (*distr M borel (h :: real \Rightarrow real)*) *lborel g*
 $\langle \text{proof} \rangle$

lemma *subprob-density-distr-real-exp*:
assumes *dens: has-subprob-density M lborel f*
shows *has-subprob-density* (*distr M borel exp*) *lborel*
 $(\lambda x. \text{if } x > 0 \text{ then } f (\ln x) * \text{ennreal } (\text{inverse } x) \text{ else } 0)$
 $(\text{is } \text{has-subprob-density} \text{ - - } ?g)$
 $\langle \text{proof} \rangle$

lemma *subprob-density-distr-real-inverse-aux*:
assumes *dens: has-subprob-density M lborel f*
shows *has-subprob-density* (*distr M borel ($\lambda x. - \text{inverse } x$)*) *lborel*
 $(\lambda x. f (-\text{inverse } x) * \text{ennreal } (\text{inverse } (x * x)))$
 $(\text{is } \text{has-subprob-density} \text{ - - } ?g)$
 $\langle \text{proof} \rangle$

lemma *subprob-density-distr-real-inverse*:
assumes *dens: has-subprob-density M lborel f*
shows *has-subprob-density* (*distr M borel inverse*) *lborel* ($\lambda x. f (\text{inverse } x) * \text{ennreal } (\text{inverse } (x * x))$)
 $\langle \text{proof} \rangle$

lemma *distr-convolution-real*:
assumes *has-density M lborel (f :: (real \times real) \Rightarrow ennreal)*
shows *has-density* (*distr M borel (case-prod (+))*) *lborel* ($\lambda z. \int^+ x. f (x, z - x) \partial \text{lborel}$)
 $(\text{is } \text{has-density } ?M' \text{ - } ?f')$
 $\langle \text{proof} \rangle$

lemma *distr-convolution-ring-count-space*:
assumes *C: countable (UNIV :: 'a set)*
assumes *has-density M (count-space UNIV) (f :: (('a :: ring) \times 'a) \Rightarrow ennreal)*
shows *has-density* (*distr M (count-space UNIV) (case-prod (+))*) (*count-space UNIV*)
 $(\lambda z. \int^+ x. f (x, z - x) \partial \text{count-space UNIV})$
 $(\text{is } \text{has-density } ?M' \text{ - } ?f')$
 $\langle \text{proof} \rangle$

end

3 Source Language Values

theory *PDF-Values*
imports *Density-Predicates*
begin

3.1 Values and stock measures

datatype *pdf-type* = *UNIT* | *BOOL* | *INTEG* | *REAL* | *PRODUCT pdf-type pdf-type*

datatype *val* = *UnitVal*
 | *BoolVal* (*extract-bool*: *bool*)
 | *IntVal* (*extract-int*: *int*)
 | *RealVal* (*extract-real*: *real*)
 | *PairVal* (*extract-fst*: *val*) (*extract-snd*: *val*) (<|-, -|> [0, 61] 1000)

where

extract-bool UnitVal = *False*
 | *extract-bool (IntVal i)* = *False*
 | *extract-bool (RealVal r)* = *False*
 | *extract-bool (PairVal x y)* = *False*
 | *extract-int UnitVal* = 0
 | *extract-int (BoolVal b)* = 0
 | *extract-int (RealVal r)* = 0
 | *extract-int (PairVal x y)* = 0
 | *extract-real UnitVal* = 0
 | *extract-real (BoolVal b)* = 0
 | *extract-real (IntVal i)* = 0
 | *extract-real (PairVal x y)* = 0

primrec *extract-pair'* **where**

extract-pair' f s <| *x, y* |> = (*f x, s y*)

definition *map-int-pair* **where**

map-int-pair f g x = (*case x of* <| *IntVal a, IntVal b* |> \Rightarrow *f a b* | - \Rightarrow *g x*)

definition *map-real-pair* **where**

map-real-pair f g x = (*case x of* <| *RealVal a, RealVal b* |> \Rightarrow *f a b* | - \Rightarrow *g x*)

abbreviation *TRUE* \equiv *BoolVal True*

abbreviation *FALSE* \equiv *BoolVal False*

type-synonym *vname* = *nat*

type-synonym *state* = *vname* \Rightarrow *val*

lemma *map-int-pair[simp]*: *map-int-pair f g* <| *IntVal i, IntVal j* |> = *f i j*
 <proof>

lemma *map-int-pair-REAL[simp]*: *map-int-pair f g* <| *RealVal i, RealVal j* |> =
g <| *RealVal i, RealVal j* |>
 <proof>

lemma *map-real-pair[simp]*: *map-real-pair f g* <| *RealVal i, RealVal j* |> = *f i j*
 <proof>

abbreviation *extract-pair* \equiv *extract-pair' id id*

abbreviation $extract\text{-}real\text{-}pair \equiv extract\text{-}pair' \ extract\text{-}real \ extract\text{-}real$
abbreviation $extract\text{-}int\text{-}pair \equiv extract\text{-}pair' \ extract\text{-}int \ extract\text{-}int$

definition $RealPairVal \equiv \lambda(x,y). \langle |RealVal \ x, \ RealVal \ y| \rangle$

definition $IntPairVal \equiv \lambda(x,y). \langle |IntVal \ x, \ IntVal \ y| \rangle$

lemma $inj\text{-}RealPairVal: \ inj \ RealPairVal \ \langle proof \rangle$

lemma $inj\text{-}IntPairVal: \ inj \ IntPairVal \ \langle proof \rangle$

fun $val\text{-}type :: val \Rightarrow pdf\text{-}type$ **where**

$val\text{-}type \ (BoolVal \ b) = \ \mathit{BOOL}$
 $| \ val\text{-}type \ (IntVal \ i) = \ \mathit{INTEG}$
 $| \ val\text{-}type \ UnitVal = \ \mathit{UNIT}$
 $| \ val\text{-}type \ (RealVal \ r) = \ \mathit{REAL}$
 $| \ val\text{-}type \ (\langle |v1 \ , \ v2| \rangle) = \ (\mathit{PRODUCT} \ (val\text{-}type \ v1) \ (val\text{-}type \ v2))$

lemma $val\text{-}type\text{-}eq\text{-}REAL: \ val\text{-}type \ x = \ \mathit{REAL} \ \longleftrightarrow \ x \in \ \mathit{RealVal}'UNIV$
 $\langle proof \rangle$

lemma $val\text{-}type\text{-}eq\text{-}INTEG: \ val\text{-}type \ x = \ \mathit{INTEG} \ \longleftrightarrow \ x \in \ \mathit{IntVal}'UNIV$
 $\langle proof \rangle$

definition $type\text{-}universe \ t = \ \{v. \ val\text{-}type \ v = \ t\}$

lemma $type\text{-}universe\text{-}nonempty[simp]: \ type\text{-}universe \ t \neq \ \{\}$
 $\langle proof \rangle$

lemma $val\text{-}in\text{-}type\text{-}universe[simp]:$
 $v \in \ type\text{-}universe \ (val\text{-}type \ v)$
 $\langle proof \rangle$

lemma $BoolVal\text{-}in\text{-}type\text{-}universe[simp]: \ BoolVal \ v \in \ type\text{-}universe \ \mathit{BOOL}$
 $\langle proof \rangle$

lemma $IntVal\text{-}in\text{-}type\text{-}universe[simp]: \ IntVal \ v \in \ type\text{-}universe \ \mathit{INTEG}$
 $\langle proof \rangle$

lemma $type\text{-}universe\text{-}type[simp]:$
 $w \in \ type\text{-}universe \ t \ \longleftrightarrow \ val\text{-}type \ w = \ t$
 $\langle proof \rangle$

lemma $type\text{-}universe\text{-}REAL: \ type\text{-}universe \ \mathit{REAL} = \ \mathit{RealVal}'UNIV$
 $\langle proof \rangle$

lemma $type\text{-}universe\text{-}eq\text{-}imp\text{-}type\text{-}eq:$
assumes $type\text{-}universe \ t1 = \ type\text{-}universe \ t2$
shows $t1 = \ t2$
 $\langle proof \rangle$

lemma *type-universe-eq-iff*[simp]: *type-universe* $t1 = \text{type-universe } t2 \iff t1 = t2$
 ⟨proof⟩

primrec *stock-measure* :: *pdf-type* \Rightarrow *val measure* **where**
stock-measure *UNIT* = *count-space* {*UnitVal*}
 | *stock-measure* *INTEG* = *count-space* (*range IntVal*)
 | *stock-measure* *BOOL* = *count-space* (*range BoolVal*)
 | *stock-measure* *REAL* = *embed-measure* *lborel RealVal*
 | *stock-measure* (*PRODUCT* $t1\ t2$) =
embed-measure (*stock-measure* $t1 \otimes_M \text{stock-measure } t2$) ($\lambda(a, b). \langle |a, b| \rangle$)

declare [[*coercion stock-measure*]]

lemma *sigma-finite-stock-measure*[simp]: *sigma-finite-measure* (*stock-measure* t)
 ⟨proof⟩

lemma *val-case-stock-measurable*:
assumes $t = \text{UNIT} \implies c \in \text{space } M$
assumes $\bigwedge b. t = \text{BOOL} \implies g\ b \in \text{space } M$
assumes $\bigwedge i. t = \text{INTEG} \implies h\ i \in \text{space } M$
assumes $t = \text{REAL} \implies j \in \text{measurable borel } M$
assumes *: $\bigwedge t1\ t2. t = \text{PRODUCT } t1\ t2 \implies \text{case-prod } k \in \text{measurable } (\text{stock-measure } t1 \otimes_M \text{stock-measure } t2)\ M$
shows ($\lambda x. \text{case } x \text{ of } \text{UnitVal} \Rightarrow c \mid \text{BoolVal } b \Rightarrow g\ b \mid \text{IntVal } i \Rightarrow h\ i \mid \text{RealVal } r \Rightarrow j\ r$
 | *PairVal* $y\ z \Rightarrow k\ y\ z$) $\in \text{measurable } t\ M$
 ⟨proof⟩

lemma *space-stock-measure*[simp]: *space* (*stock-measure* t) = *type-universe* t
 ⟨proof⟩

lemma *type-universe-stock-measure*[measurable]: *type-universe* $t \in \text{sets } (\text{stock-measure } t)$
 ⟨proof⟩

lemma *inj-RealVal*[simp]: *inj* *RealVal* ⟨proof⟩

lemma *inj-IntVal*[simp]: *inj* *IntVal* ⟨proof⟩

lemma *inj-BoolVal*[simp]: *inj* *BoolVal* ⟨proof⟩

lemma *inj-PairVal*[simp]: *inj* ($\lambda(x, y). \langle |x, y| \rangle$) ⟨proof⟩

lemma *measurable-PairVal*[measurable]:

fixes $t1\ t2 :: \text{pdf-type}$

shows *case-prod* *PairVal* $\in \text{measurable } (t1 \otimes_M t2)\ (\text{PRODUCT } t1\ t2)$

⟨proof⟩

lemma *measurable-RealVal*[measurable]: *RealVal* $\in \text{measurable borel } \text{REAL}$
 ⟨proof⟩

lemma *nn-integral-BoolVal*:

assumes $\bigwedge x. f \text{ (BoolVal } x) \geq 0$

shows $(\int^{+x}. f \ x \ \partial \text{BOOL}) = f \text{ (BoolVal True)} + f \text{ (BoolVal False)}$

<proof>

lemma *nn-integral-RealVal*:

$f \in \text{borel-measurable REAL} \implies (\int^{+x}. f \ x \ \partial \text{REAL}) = (\int^{+x}. f \ \text{(RealVal } x) \ \partial \text{borel})$

<proof>

lemma *nn-integral-IntVal*: $(\int^{+x}. f \ x \ \partial \text{INTEG}) = (\int^{+x}. f \ \text{(IntVal } x) \ \partial \text{count-space UNIV})$

<proof>

lemma *nn-integral-PairVal*:

$f \in \text{borel-measurable (PRODUCT } t1 \ t2) \implies$

$(\int^{+x}. f \ x \ \partial \text{PRODUCT } t1 \ t2) = (\int^{+x}. f \ \text{(PairVal (fst } x) \ \text{(snd } x)) \ \partial (t1 \otimes_M t2))$

<proof>

lemma *BOOL-E*: $\llbracket \text{val-type } v = \text{BOOL}; \bigwedge b. v = \text{BoolVal } b \implies P \rrbracket \implies P$

<proof>

lemma *PROD-E*: $\llbracket \text{val-type } v = \text{PRODUCT } t1 \ t2 ;$

$\bigwedge a \ b. \text{val-type } a = t1 \implies \text{val-type } b = t2 \implies v = \langle | a, b | \rangle \implies P \rrbracket \implies P$

<proof>

lemma *REAL-E*: $\llbracket \text{val-type } v = \text{REAL}; \bigwedge b. v = \text{RealVal } b \implies P \rrbracket \implies P$

<proof>

lemma *INTEG-E*: $\llbracket \text{val-type } v = \text{INTEG}; \bigwedge i. v = \text{IntVal } i \implies P \rrbracket \implies P$

<proof>

lemma *measurable-extract-pair'*[*measurable (raw)*]:

fixes $t1 \ t2 :: \text{pdf-type}$

assumes [*measurable*]: $f \in \text{measurable } t1 \ M$

assumes [*measurable*]: $g \in \text{measurable } t2 \ N$

assumes $h: h \in \text{measurable } K \ \text{(PRODUCT } t1 \ t2)$

shows $(\lambda x. \text{extract-pair}' f \ g \ (h \ x)) \in \text{measurable } K \ (M \otimes_M N)$

<proof>

lemma *measurable-extract-pair*[*measurable*]: $\text{extract-pair} \in \text{measurable (PRODUCT } t1 \ t2) \ (t1 \otimes_M t2)$

<proof>

lemma *measurable-extract-real*[*measurable*]: $\text{extract-real} \in \text{measurable REAL borel}$

<proof>

lemma *measurable-extract-int*[*measurable*]: *extract-int* \in *measurable* *INTEG* (*count-space* *UNIV*)

<proof>

lemma *measurable-extract-int-pair*[*measurable*]:

extract-int-pair \in *measurable* (*PRODUCT* *INTEG* *INTEG*) (*count-space* *UNIV*

\otimes_M *count-space* *UNIV*)

<proof>

lemma *measurable-extract-real-pair*[*measurable*]:

extract-real-pair \in *measurable* (*PRODUCT* *REAL* *REAL*) (*borel* \otimes_M *borel*)

<proof>

lemma *measurable-extract-real-pair'*[*measurable*]:

extract-real-pair \in *measurable* (*PRODUCT* *REAL* *REAL*) *borel*

<proof>

lemma *measurable-extract-bool*[*measurable*]: *extract-bool* \in *measurable* *BOOL* (*count-space* *UNIV*)

<proof>

lemma *map-int-pair-measurable*[*measurable*]:

assumes *f*: *case-prod* *f* \in *measurable* (*count-space* *UNIV* \otimes_M *count-space* *UNIV*) *M*

shows *map-int-pair* *f g* \in *measurable* (*PRODUCT* *INTEG* *INTEG*) *M*

<proof>

lemma *map-int-pair-measurable-REAL*[*measurable*]:

assumes *g* \in *measurable* (*PRODUCT* *REAL* *REAL*) *M*

shows *map-int-pair* *f g* \in *measurable* (*PRODUCT* *REAL* *REAL*) *M*

<proof>

lemma *map-real-pair-measurable*[*measurable*]:

assumes *f*: *case-prod* *f* \in *measurable* (*borel* \otimes_M *borel*) *M*

shows *map-real-pair* *f g* \in *measurable* (*PRODUCT* *REAL* *REAL*) *M*

<proof>

lemma *count-space-IntVal-prod*[*simp*]: *INTEG* \otimes_M *INTEG* = *count-space* (*range* *IntVal* \times *range* *IntVal*)

<proof>

lemma *count-space-BoolVal-prod*[*simp*]: *BOOL* \otimes_M *BOOL* = *count-space* (*range* *BoolVal* \times *range* *BoolVal*)

<proof>

lemma *measurable-stock-measure-val-type*:

assumes *f* \in *measurable* *M* (*stock-measure* *t*) *x* \in *space* *M*

shows *val-type* (*f x*) = *t*

<proof>

lemma *singleton-in-stock-measure*[simp]: $\text{val-type } v = t \implies \{v\} \in \text{sets } t$
 ⟨proof⟩

lemma *emeasure-stock-measure-singleton-finite*[simp]:
 $\text{emeasure } (\text{stock-measure } (\text{val-type } v)) \{v\} \neq \infty$
 ⟨proof⟩

3.2 Measures on states

definition *state-measure* :: $\text{vname set} \Rightarrow (\text{vname} \Rightarrow \text{pdf-type}) \Rightarrow \text{state measure}$
where

$$\text{state-measure } V \Gamma \equiv \Pi_M y \in V. \Gamma y$$

lemma *state-measure-nonempty*[simp]: $\text{space } (\text{state-measure } V \Gamma) \neq \{\}$
 ⟨proof⟩

lemma *space-state-measure*: $\text{space } (\text{state-measure } V \Gamma) = (\Pi_E y \in V. \text{type-universe } (\Gamma y))$
 ⟨proof⟩

lemma *state-measure-var-type*:
 $\sigma \in \text{space } (\text{state-measure } V \Gamma) \implies x \in V \implies \text{val-type } (\sigma x) = \Gamma x$
 ⟨proof⟩

lemma *merge-in-state-measure*:
 $x \in \text{space } (\text{state-measure } A \Gamma) \implies y \in \text{space } (\text{state-measure } B \Gamma) \implies$
 $\text{merge } A B (x, y) \in \text{space } (\text{state-measure } (A \cup B) \Gamma)$ ⟨proof⟩

lemma *measurable-merge-stock*[measurable (raw)]:
 $f \in N \rightarrow_M \text{state-measure } V \Gamma \implies g \in N \rightarrow_M \text{state-measure } V' \Gamma \implies$
 $(\lambda x. \text{merge } V V' (f x, g x)) \in N \rightarrow_M \text{state-measure } (V \cup V') \Gamma$
 ⟨proof⟩

lemma *comp-in-state-measure*:
assumes $\sigma \in \text{space } (\text{state-measure } V \Gamma)$
shows $\sigma \circ f \in \text{space } (\text{state-measure } (f \text{ - ' } V) (\Gamma \circ f))$
 ⟨proof⟩

lemma *sigma-finite-state-measure*[intro]:
 $\text{finite } V \implies \text{sigma-finite-measure } (\text{state-measure } V \Gamma)$ ⟨proof⟩

3.3 Equalities of measure embeddings

lemma *embed-measure-RealPairVal*:
 $\text{stock-measure } (\text{PRODUCT REAL REAL}) = \text{embed-measure } \text{lborel } \text{RealPairVal}$
 ⟨proof⟩

lemma *embed-measure-IntPairVal*:
 $\text{stock-measure } (\text{PRODUCT INTEG INTEG}) = \text{count-space } (\text{range } \text{IntPairVal})$

<proof>

3.4 Monadic operations on values

definition *return-val* $x = \text{return } (\text{stock-measure } (\text{val-type } x)) x$

lemma *sets-return-val*[*measurable-cong*]: $\text{sets } (\text{return-val } x) = \text{sets } (\text{stock-measure } (\text{val-type } x))$
<proof>

lemma *measurable-return-val*[*simp*]:
 $\text{return-val} \in \text{measurable } (\text{stock-measure } t) (\text{subprob-algebra } (\text{stock-measure } t))$
<proof>

lemma *bind-return-val*:
assumes $\text{space } M \neq \{\}$ $f \in \text{measurable } M (\text{stock-measure } t')$
shows $M \ggg (\lambda x. \text{return-val } (f x)) = \text{distr } M (\text{stock-measure } t') f$
<proof>

lemma *bind-return-val'*:
assumes $\text{val-type } x = t$ $f \in \text{measurable } (\text{stock-measure } t) (\text{stock-measure } t')$
shows $\text{return-val } x \ggg (\lambda x. \text{return-val } (f x)) = \text{return-val } (f x)$
<proof>

lemma *bind-return-val''*:
assumes $f \in \text{measurable } (\text{stock-measure } (\text{val-type } x)) (\text{subprob-algebra } M)$
shows $\text{return-val } x \ggg f = f x$
<proof>

lemma *bind-assoc-return-val*:
assumes *sets-M*: $\text{sets } M = \text{sets } (\text{stock-measure } t)$
assumes *Mf*: $f \in \text{measurable } (\text{stock-measure } t) (\text{stock-measure } t')$
assumes *Mg*: $g \in \text{measurable } (\text{stock-measure } t') (\text{stock-measure } t'')$
shows $(M \ggg (\lambda x. \text{return-val } (f x))) \ggg (\lambda x. \text{return-val } (g x)) =$
 $M \ggg (\lambda x. \text{return-val } (g (f x)))$
<proof>

lemma *bind-return-val-distr*:
assumes *sets-M*: $\text{sets } M = \text{sets } (\text{stock-measure } t)$
assumes *Mf*: $f \in \text{measurable } (\text{stock-measure } t) (\text{stock-measure } t')$
shows $M \ggg \text{return-val} \circ f = \text{distr } M (\text{stock-measure } t') f$
<proof>

3.5 Lifting of functions

definition *lift-RealVal* **where**
 $\text{lift-RealVal } f \equiv \lambda \text{RealVal } v \Rightarrow \text{RealVal } (f v) \mid - \Rightarrow \text{RealVal } (f 0)$

definition *lift-IntVal* **where**
 $\text{lift-IntVal } f \equiv \lambda \text{IntVal } v \Rightarrow \text{IntVal } (f v) \mid - \Rightarrow \text{IntVal } (f 0)$

definition *lift-RealIntVal* **where**

$lift\text{-}RealIntVal\ f\ g \equiv \lambda\ IntVal\ v \Rightarrow IntVal\ (f\ v) \mid RealVal\ v \Rightarrow RealVal\ (g\ v)$

definition *lift-RealIntVal2* **where**

$lift\text{-}RealIntVal2\ f\ g \equiv$
 $\quad map\text{-}int\text{-}pair\ (\lambda a\ b.\ IntVal\ (f\ a\ b))$
 $\quad (map\text{-}real\text{-}pair\ (\lambda a\ b.\ RealVal\ (g\ a\ b))$
 $\quad id)$

definition *lift-Comp* **where**

$lift\text{-}Comp\ f\ g \equiv map\text{-}int\text{-}pair\ (\lambda a\ b.\ BoolVal\ (f\ a\ b))$
 $\quad (map\text{-}real\text{-}pair\ (\lambda a\ b.\ BoolVal\ (g\ a\ b))$
 $\quad (\lambda \cdot.\ FALSE))$

lemma *lift-RealVal-eq*: $lift\text{-}RealVal\ f\ (RealVal\ x) = RealVal\ (f\ x)$

<proof>

lemma *lift-RealIntVal-Real*:

$x \in space\ (stock\text{-}measure\ REAL) \implies lift\text{-}RealIntVal\ f\ g\ x = lift\text{-}RealVal\ g\ x$
<proof>

lemma *lift-RealIntVal-Int*:

$x \in space\ (stock\text{-}measure\ INTEG) \implies lift\text{-}RealIntVal\ f\ g\ x = lift\text{-}IntVal\ f\ x$
<proof>

declare *stock-measure.simps*[*simp del*]

lemma *measurable-lift-RealVal*[*measurable*]:

assumes [*measurable*]: $f \in borel\text{-}measurable\ borel$
shows $lift\text{-}RealVal\ f \in measurable\ REAL\ REAL$
<proof>

lemma *measurable-lift-IntVal*[*simp*]: $lift\text{-}IntVal\ f \in range\ IntVal \rightarrow range\ IntVal$

<proof>

lemma *measurable-lift-IntVal'*[*measurable*]: $lift\text{-}IntVal\ f \in measurable\ INTEG\ INTEG$

<proof>

lemma *split-apply*: $(case\ x\ of\ (a,\ b) \Rightarrow f\ a\ b)\ y = (case\ x\ of\ (a,\ b) \Rightarrow f\ a\ b\ y)$

<proof>

lemma *measurable-lift-Comp-RealVal*[*measurable*]:

assumes [*measurable*]: $Measurable.pred\ (borel \otimes_M borel)\ (case\text{-}prod\ g)$
shows $lift\text{-}Comp\ f\ g \in measurable\ (PRODUCT\ REAL\ REAL)\ BOOL$
<proof>

lemma *measurable-lift-Comp-IntVal*[*simp*]:

$lift\text{-}Comp\ f\ g \in measurable\ (PRODUCT\ INTEG\ INTEG)\ BOOL$
<proof>

lemma *measurable-lift-RealIntVal-IntVal[simp]*: *lift-RealIntVal* $f g \in \text{range IntVal}$
 $\rightarrow \text{range IntVal}$
 ⟨*proof*⟩

lemma *measurable-lift-RealIntVal-IntVal'[measurable]*:
lift-RealIntVal $f g \in \text{measurable INTEG INTEG}$
 ⟨*proof*⟩

lemma *measurable-lift-RealIntVal-RealVal[measurable]*:
assumes [*measurable*]: $g \in \text{borel-measurable borel}$
shows *lift-RealIntVal* $f g \in \text{measurable REAL REAL}$
 ⟨*proof*⟩

lemma *measurable-lift-RealIntVal2-IntVal[measurable]*:
lift-RealIntVal2 $f g \in \text{measurable (PRODUCT INTEG INTEG) INTEG}$
 ⟨*proof*⟩

lemma *measurable-lift-RealIntVal2-RealVal[measurable]*:
assumes [*measurable*]: *case-prod* $g \in \text{borel-measurable (borel } \otimes_M \text{ borel)}$
shows *lift-RealIntVal2* $f g \in \text{measurable (PRODUCT REAL REAL) REAL}$
 ⟨*proof*⟩

lemma *distr-lift-RealVal*:
fixes f
assumes $Mf[\text{measurable}]$: $f \in \text{borel-measurable borel}$
assumes $pdens$: *has-subprob-density* M (*stock-measure REAL*) δ
assumes $dens'$: $\bigwedge M \delta. \text{has-subprob-density } M \text{ lborel } \delta \implies \text{has-density (distr } M$
borel f) *lborel* ($g \delta$)
defines $N \equiv \text{distr } M$ (*stock-measure REAL*) (*lift-RealVal f*)
shows *has-density* N (*stock-measure REAL*) ($g (\lambda x. \delta (\text{RealVal } x)) \circ \text{extract-real}$)
 ⟨*proof*⟩

lemma *distr-lift-IntVal*:
fixes f
assumes $pdens$: *has-density* M (*stock-measure INTEG*) δ
assumes $dens'$: $\bigwedge M \delta. \text{has-density } M$ (*count-space UNIV*) $\delta \implies$
has-density (*distr* M (*count-space UNIV*) f) (*count-space*
UNIV) ($g \delta$)
defines $N \equiv \text{distr } M$ (*stock-measure INTEG*) (*lift-IntVal f*)
shows *has-density* N (*stock-measure INTEG*) ($g (\lambda x. \delta (\text{IntVal } x)) \circ \text{extract-int}$)
 ⟨*proof*⟩

lemma *distr-lift-RealPairVal*:
fixes $f f' g$
assumes $Mf[\text{measurable}]$: *case-prod* $f \in \text{borel-measurable borel}$
assumes $pdens$: *has-subprob-density* M (*stock-measure (PRODUCT REAL REAL)*)
 δ
assumes $dens'$: $\bigwedge M \delta. \text{has-subprob-density } M \text{ lborel } \delta \implies \text{has-density (distr } M$

borel (case-prod f)) lborel (g δ)
defines $N \equiv \text{distr } M \text{ (stock-measure } REAL) \text{ (lift-RealIntVal2 } f' f)$
shows $\text{has-density } N \text{ (stock-measure } REAL) \text{ (g } (\lambda x. \delta \text{ (RealPairVal } x)) \circ \text{extract-real})}$
<proof>

lemma *distr-lift-IntPairVal:*

fixes $f f'$
assumes $\text{pdens: has-density } M \text{ (stock-measure (PRODUCT INTEG INTEG)) } \delta$
assumes $\text{dens': } \bigwedge M \delta. \text{has-density } M \text{ (count-space UNIV) } \delta \implies$
 $\text{has-density (distr } M \text{ (count-space UNIV) (case-prod } f))$
 $\text{(count-space UNIV) (g } \delta)$
defines $N \equiv \text{distr } M \text{ (stock-measure INTEG) (lift-RealIntVal2 } f f')$
shows $\text{has-density } N \text{ (stock-measure INTEG) (g } (\lambda x. \delta \text{ (IntPairVal } x)) \circ \text{extract-int})}$
<proof>

end

theory *PDF-Semantics*

imports *PDF-Values*

begin

lemma *measurable-subprob-algebra-density:*

assumes $\text{sigma-finite-measure } N$
assumes $\text{space } N \neq \{\}$
assumes $[\text{measurable}]: \text{case-prod } f \in \text{borel-measurable } (M \otimes_M N)$
assumes $\bigwedge x. x \in \text{space } M \implies (\int^+ y. f x y \partial N) \leq 1$
shows $(\lambda x. \text{density } N (f x)) \in \text{measurable } M \text{ (subprob-algebra } N)$
<proof>

4 Built-in Probability Distributions

4.1 Bernoulli

definition *bernoulli-density* :: $\text{real} \Rightarrow \text{bool} \Rightarrow \text{ennreal}$ **where**

$\text{bernoulli-density } p b = (\text{if } p \in \{0..1\} \text{ then (if } b \text{ then } p \text{ else } 1 - p) \text{ else } 0)$

definition *bernoulli* :: $\text{val} \Rightarrow \text{val measure}$ **where**

$\text{bernoulli } p = \text{density } BOOL \text{ (bernoulli-density (extract-real } p) \circ \text{extract-bool})}$

lemma *measurable-bernoulli-density[measurable]:*

$\text{case-prod } \text{bernoulli-density} \in \text{borel-measurable } (\text{borel } \otimes_M \text{ count-space UNIV})$

<proof>

lemma *measurable-bernoulli[measurable]:* $\text{bernoulli} \in \text{measurable } REAL \text{ (subprob-algebra } BOOL)$

<proof>

4.2 Uniform

definition *uniform-real-density* :: $real \times real \Rightarrow real \Rightarrow ennreal$ **where**

uniform-real-density $\equiv \lambda(a,b) x. ennreal (if\ a < b \wedge x \in \{a..b\}\ then\ inverse\ (b - a)\ else\ 0)$

definition *uniform-int-density* :: $int \times int \Rightarrow int \Rightarrow ennreal$ **where**

uniform-int-density $\equiv \lambda(a,b) x. (if\ x \in \{a..b\}\ then\ inverse\ (nat\ (b - a + 1))\ else\ 0)$

lemma *measurable-uniform-density-int*[measurable]:

(case-prod *uniform-int-density*)
 \in borel-measurable ((count-space UNIV \otimes_M count-space UNIV) \otimes_M count-space UNIV)
 ⟨proof⟩

lemma *measurable-uniform-density-real*[measurable]:

(case-prod *uniform-real-density*) \in borel-measurable (borel \otimes_M borel)
 ⟨proof⟩

definition *uniform-int* :: $val \Rightarrow val\ measure$ **where**

uniform-int = map-int-pair ($\lambda l\ u. density\ INTEG\ (uniform-int-density\ (l,u)\ o\ extract-int)$) ($\lambda-. undefined$)

definition *uniform-real* :: $val \Rightarrow val\ measure$ **where**

uniform-real = map-real-pair ($\lambda l\ u. density\ REAL\ (uniform-real-density\ (l,u)\ o\ extract-real)$) ($\lambda-. undefined$)

lemma *if-bounded*: ($if\ a \leq i \wedge i \leq b\ then\ v\ else\ 0$) = ($v::real$) * indicator {a .. b} i

⟨proof⟩

lemma *measurable-uniform-int*[measurable]:

uniform-int \in measurable (PRODUCT INTEG INTEG) (subprob-algebra INTEG)
 ⟨proof⟩

lemma *density-cong'*:

($\bigwedge x. x \in space\ M \implies f\ x = g\ x$) $\implies density\ M\ f = density\ M\ g$
 ⟨proof⟩

lemma *measurable-uniform-real*[measurable]:

uniform-real \in measurable (PRODUCT REAL REAL) (subprob-algebra REAL)
 ⟨proof⟩

4.3 Gaussian

definition *gaussian-density* :: $real \times real \Rightarrow real \Rightarrow ennreal$ **where**

gaussian-density $\equiv \lambda(m,s) x. (if\ s > 0\ then\ exp\ (-(x - m)^2 / (2 * s^2)) / sqrt\ (2 * pi * s^2)\ else$

0)

lemma *measurable-gaussian-density*[*measurable*]:
 case-prod gaussian-density \in *borel-measurable* (*borel* \otimes_M *borel*)
 ⟨*proof*⟩

definition *gaussian* :: *val* \Rightarrow *val measure* **where**
 gaussian = *map-real-pair* (λm *s. density REAL (gaussian-density (m,s) o extract-real)*) *undefined*

lemma *measurable-gaussian*[*measurable*]: *gaussian* \in *measurable* (*PRODUCT REAL REAL*) (*subprob-algebra REAL*)
 ⟨*proof*⟩

4.4 Poisson

definition *poisson-density'* :: *real* \Rightarrow *int* \Rightarrow *ennreal* **where**
 poisson-density' rate k = *pmf (poisson-pmf rate) (nat k) * indicator ({0 <..} \times {0..}) (rate, k)*

lemma *measurable-poisson-density'*[*measurable*]:
 case-prod poisson-density' \in *borel-measurable* (*borel* \otimes_M *count-space UNIV*)
 ⟨*proof*⟩

definition *poisson* :: *val* \Rightarrow *val measure* **where**
 poisson rate = *density INTEG (poisson-density' (extract-real rate) o extract-int)*

lemma *measurable-poisson*[*measurable*]: *poisson* \in *measurable REAL* (*subprob-algebra INTEG*)
 ⟨*proof*⟩

5 Source Language Syntax and Semantics

5.1 Expressions

class *expr* = **fixes** *free-vars* :: '*a* \Rightarrow *vname set*

datatype *pdf-dist* = *Bernoulli* | *UniformInt* | *UniformReal* | *Poisson* | *Gaussian*

datatype *pdf-operator* = *Fst* | *Snd* | *Add* | *Mult* | *Minus* | *Less* | *Equals* | *And* | *Not* | *Or* | *Pow* |
 Sqrt | *Exp* | *Ln* | *Fact* | *Inverse* | *Pi* | *Cast pdf-type*

datatype *expr* =
 Var vname
 | *Val val*
 | *LetVar expr expr (LET - IN - [0, 60] 61)*
 | *Operator pdf-operator expr (infixl \$\$ 999)*
 | *Pair expr expr (<- , -> [0, 60] 1000)*

```

| Random pdf-dist expr
| IfThenElse expr expr expr (IF - THEN - ELSE - [0, 0, 70] 71)
| Fail pdf-type

```

type-synonym *tyenv* = *vname* \Rightarrow *pdf-type*

instantiation *expr* :: *expr*

begin

primrec *free-vars-expr* :: *expr* \Rightarrow *vname set* **where**

```

  free-vars-expr (Var x) = {x}
| free-vars-expr (Val -) = {}
| free-vars-expr (LetVar e1 e2) = free-vars-expr e1  $\cup$  Suc - ' free-vars-expr e2
| free-vars-expr (Operator - e) = free-vars-expr e
| free-vars-expr (<e1, e2>) = free-vars-expr e1  $\cup$  free-vars-expr e2
| free-vars-expr (Random - e) = free-vars-expr e
| free-vars-expr (IF b THEN e1 ELSE e2) =
  free-vars-expr b  $\cup$  free-vars-expr e1  $\cup$  free-vars-expr e2
| free-vars-expr (Fail -) = {}

```

instance \langle *proof* \rangle

end

primrec *free-vars-expr-code* :: *expr* \Rightarrow *vname set* **where**

```

  free-vars-expr-code (Var x) = {x}
| free-vars-expr-code (Val -) = {}
| free-vars-expr-code (LetVar e1 e2) =
  free-vars-expr-code e1  $\cup$  ( $\lambda x. x - 1$ ) ' (free-vars-expr-code e2 - {0})
| free-vars-expr-code (Operator - e) = free-vars-expr-code e
| free-vars-expr-code (<e1, e2>) = free-vars-expr-code e1  $\cup$  free-vars-expr-code e2
| free-vars-expr-code (Random - e) = free-vars-expr-code e
| free-vars-expr-code (IF b THEN e1 ELSE e2) =
  free-vars-expr-code b  $\cup$  free-vars-expr-code e1  $\cup$  free-vars-expr-code e2
| free-vars-expr-code (Fail -) = {}

```

lemma *free-vars-expr-code*[*code*]:

```

  free-vars (e::expr) = free-vars-expr-code e
 $\langle$ proof $\rangle$ 

```

primrec *dist-param-type* **where**

```

  dist-param-type Bernoulli = REAL
| dist-param-type Poisson = REAL
| dist-param-type Gaussian = PRODUCT REAL REAL
| dist-param-type UniformInt = PRODUCT INTEG INTEG
| dist-param-type UniformReal = PRODUCT REAL REAL

```

primrec *dist-result-type* **where**

```

  dist-result-type Bernoulli = BOOL

```

| *dist-result-type UniformInt* = *INTEG*
| *dist-result-type UniformReal* = *REAL*
| *dist-result-type Poisson* = *INTEG*
| *dist-result-type Gaussian* = *REAL*

primrec *dist-measure* :: *pdf-dist* \Rightarrow *val* \Rightarrow *val measure* **where**

dist-measure Bernoulli = *bernoulli*
| *dist-measure UniformInt* = *uniform-int*
| *dist-measure UniformReal* = *uniform-real*
| *dist-measure Poisson* = *poisson*
| *dist-measure Gaussian* = *gaussian*

lemma *measurable-dist-measure*[*measurable*]:

dist-measure d \in *measurable* (*dist-param-type d*) (*subprob-algebra* (*dist-result-type d*))
 ⟨*proof*⟩

lemma *sets-dist-measure*[*simp*]:

val-type x = *dist-param-type dst* \implies
 sets (*dist-measure dst x*) = *sets* (*stock-measure* (*dist-result-type dst*))
 ⟨*proof*⟩

lemma *space-dist-measure*[*simp*]:

val-type x = *dist-param-type dst* \implies
 space (*dist-measure dst x*) = *type-universe* (*dist-result-type dst*)
 ⟨*proof*⟩

primrec *dist-dens* :: *pdf-dist* \Rightarrow *val* \Rightarrow *val* \Rightarrow *ennreal* **where**

dist-dens Bernoulli x y = *bernoulli-density* (*extract-real x*) (*extract-bool y*)
| *dist-dens UniformInt x y* = *uniform-int-density* (*extract-int-pair x*) (*extract-int y*)
| *dist-dens UniformReal x y* = *uniform-real-density* (*extract-real-pair x*) (*extract-real y*)
| *dist-dens Gaussian x y* = *gaussian-density* (*extract-real-pair x*) (*extract-real y*)
| *dist-dens Poisson x y* = *poisson-density'* (*extract-real x*) (*extract-int y*)

lemma *measurable-dist-dens*[*measurable*]:

assumes *f* \in *measurable M* (*stock-measure* (*dist-param-type dst*)) (**is** - \in *measurable M ?N*)
 assumes *g* \in *measurable M* (*stock-measure* (*dist-result-type dst*)) (**is** - \in *measurable M ?R*)
 shows ($\lambda x.$ *dist-dens dst* (*f x*) (*g x*)) \in *borel-measurable M*
 ⟨*proof*⟩

lemma *dist-measure-has-density*:

v \in *type-universe* (*dist-param-type dst*) \implies
 has-density (*dist-measure dst v*) (*stock-measure* (*dist-result-type dst*)) (*dist-dens dst v*)
 ⟨*proof*⟩

lemma *subprob-space-dist-measure*:

$v \in \text{type-universe } (\text{dist-param-type } \text{dst}) \implies \text{subprob-space } (\text{dist-measure } \text{dst } v)$
{proof}

lemma *dist-measure-has-subprob-density*:

$v \in \text{type-universe } (\text{dist-param-type } \text{dst}) \implies$
 $\text{has-subprob-density } (\text{dist-measure } \text{dst } v) (\text{stock-measure } (\text{dist-result-type } \text{dst}))$
(*dist-dens dst v*)
{proof}

lemma *dist-dens-integral-space*:

assumes $v \in \text{type-universe } (\text{dist-param-type } \text{dst})$
shows $(\int^+ u. \text{dist-dens } \text{dst } v \ u \ \partial \text{stock-measure } (\text{dist-result-type } \text{dst})) \leq 1$
{proof}

5.2 Typing

primrec *op-type* :: *pdf-operator* \Rightarrow *pdf-type* \Rightarrow *pdf-type option* **where**

op-type Add $x =$
 (case x of
 PRODUCT INTEG INTEG \Rightarrow *Some INTEG*
 | *PRODUCT REAL REAL* \Rightarrow *Some REAL*
 | - \Rightarrow *None*)
| *op-type Mult* $x =$
 (case x of
 PRODUCT INTEG INTEG \Rightarrow *Some INTEG*
 | *PRODUCT REAL REAL* \Rightarrow *Some REAL*
 | - \Rightarrow *None*)
| *op-type Minus* $x =$
 (case x of
 INTEG \Rightarrow *Some INTEG*
 | *REAL* \Rightarrow *Some REAL*
 | - \Rightarrow *None*)
| *op-type Equals* $x =$
 (case x of
 PRODUCT t1 t2 \Rightarrow if $t1 = t2$ then *Some BOOL* else *None*
 | - \Rightarrow *None*)
| *op-type Less* $x =$
 (case x of
 PRODUCT INTEG INTEG \Rightarrow *Some BOOL*
 | *PRODUCT REAL REAL* \Rightarrow *Some BOOL*
 | - \Rightarrow *None*)
| *op-type (Cast t)* $x =$
 (case (x, t) of
 (*BOOL, INTEG*) \Rightarrow *Some INTEG*
 | (*BOOL, REAL*) \Rightarrow *Some REAL*
 | (*INTEG, REAL*) \Rightarrow *Some REAL*
 | (*REAL, INTEG*) \Rightarrow *Some INTEG*

$| - \Rightarrow \text{None}$
 $| \text{op-type Or } x = (\text{case } x \text{ of } \text{PRODUCT } \text{BOOL } \text{BOOL} \Rightarrow \text{Some } \text{BOOL} \mid - \Rightarrow \text{None})$
 $| \text{op-type And } x = (\text{case } x \text{ of } \text{PRODUCT } \text{BOOL } \text{BOOL} \Rightarrow \text{Some } \text{BOOL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Not } x = (\text{case } x \text{ of } \text{BOOL} \Rightarrow \text{Some } \text{BOOL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Inverse } x = (\text{case } x \text{ of } \text{REAL} \Rightarrow \text{Some } \text{REAL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Fact } x = (\text{case } x \text{ of } \text{INTEG} \Rightarrow \text{Some } \text{INTEG} \mid - \Rightarrow \text{None})$
 $| \text{op-type Sqrt } x = (\text{case } x \text{ of } \text{REAL} \Rightarrow \text{Some } \text{REAL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Exp } x = (\text{case } x \text{ of } \text{REAL} \Rightarrow \text{Some } \text{REAL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Ln } x = (\text{case } x \text{ of } \text{REAL} \Rightarrow \text{Some } \text{REAL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Pi } x = (\text{case } x \text{ of } \text{UNIT} \Rightarrow \text{Some } \text{REAL} \mid - \Rightarrow \text{None})$
 $| \text{op-type Pow } x = (\text{case } x \text{ of}$
 $\quad \text{PRODUCT } \text{REAL } \text{INTEG} \Rightarrow \text{Some } \text{REAL}$
 $\quad \mid \text{PRODUCT } \text{INTEG } \text{INTEG} \Rightarrow \text{Some } \text{INTEG}$
 $\quad \mid - \Rightarrow \text{None})$
 $| \text{op-type Fst } x = (\text{case } x \text{ of } \text{PRODUCT } t - \Rightarrow \text{Some } t \mid - \Rightarrow \text{None})$
 $| \text{op-type Snd } x = (\text{case } x \text{ of } \text{PRODUCT } - t \Rightarrow \text{Some } t \mid - \Rightarrow \text{None})$

5.3 Semantics

abbreviation (*input*) *de-bruijn-insert* (**infixr** · 65) **where**

de-bruijn-insert $x f \equiv \text{case-nat } x f$

inductive *expr-typing* :: *tyenv* \Rightarrow *expr* \Rightarrow *pdf-type* \Rightarrow *bool* ((1-/ \vdash / (- \cdot / -))

[50,0,50] 50) **where**

$| \text{et-var}: \Gamma \vdash \text{Var } x : \Gamma x$
 $| \text{et-val}: \Gamma \vdash \text{Val } v : \text{val-type } v$
 $| \text{et-let}: \Gamma \vdash e1 : t1 \Longrightarrow t1 \cdot \Gamma \vdash e2 : t2 \Longrightarrow \Gamma \vdash \text{LetVar } e1 e2 : t2$
 $| \text{et-op}: \Gamma \vdash e : t \Longrightarrow \text{op-type oper } t = \text{Some } t' \Longrightarrow \Gamma \vdash \text{Operator oper } e : t'$
 $| \text{et-pair}: \Gamma \vdash e1 : t1 \Longrightarrow \Gamma \vdash e2 : t2 \Longrightarrow \Gamma \vdash \langle e1, e2 \rangle : \text{PRODUCT } t1 t2$
 $| \text{et-rand}: \Gamma \vdash e : \text{dist-param-type } \text{dst} \Longrightarrow \Gamma \vdash \text{Random } \text{dst } e : \text{dist-result-type } \text{dst}$
 $| \text{et-if}: \Gamma \vdash b : \text{BOOL} \Longrightarrow \Gamma \vdash e1 : t \Longrightarrow \Gamma \vdash e2 : t \Longrightarrow \Gamma \vdash \text{IF } b \text{ THEN } e1 \text{ ELSE } e2 : t$
 $| \text{et-fail}: \Gamma \vdash \text{Fail } t : t$

lemma *expr-typing-cong'*:

$\Gamma \vdash e : t \Longrightarrow (\bigwedge x. x \in \text{free-vars } e \Longrightarrow \Gamma x = \Gamma' x) \Longrightarrow \Gamma' \vdash e : t$
<proof>

lemma *expr-typing-cong*:

$(\bigwedge x. x \in \text{free-vars } e \Longrightarrow \Gamma x = \Gamma' x) \Longrightarrow \Gamma \vdash e : t \longleftrightarrow \Gamma' \vdash e : t$
<proof>

inductive-cases *expr-typing-valE*[*elim*]: $\Gamma \vdash \text{Val } v : t$

inductive-cases *expr-typing-varE*[*elim*]: $\Gamma \vdash \text{Var } x : t$

inductive-cases *expr-typing-letE*[*elim*]: $\Gamma \vdash \text{LetVar } e1 e2 : t$

inductive-cases *expr-typing-ifE*[*elim*]: $\Gamma \vdash \text{IfThenElse } b e1 e2 : t$

inductive-cases *expr-typing-opE*[*elim*]: $\Gamma \vdash \text{Operator oper } e : t$

inductive-cases *expr-typing-pairE*[*elim*]: $\Gamma \vdash \langle e1, e2 \rangle : t$

inductive-cases *expr-typing-randE*[elim]: $\Gamma \vdash \text{Random } \text{dst } e : t$
inductive-cases *expr-typing-failE*[elim]: $\Gamma \vdash \text{Fail } t : t'$

lemma *expr-typing-unique*:

$\Gamma \vdash e : t \implies \Gamma \vdash e : t' \implies t = t'$
 ⟨proof⟩

fun *expr-type* :: *tyenv* \Rightarrow *expr* \Rightarrow *pdf-type option* **where**

expr-type Γ (Var *x*) = Some (Γ *x*)
 | *expr-type* Γ (Val *v*) = Some (*val-type* *v*)
 | *expr-type* Γ (LetVar *e1 e2*) =
 (case *expr-type* Γ *e1* of
 Some *t* \Rightarrow *expr-type* (case-nat *t* Γ) *e2*
 | None \Rightarrow None)
 | *expr-type* Γ (Operator *oper e*) =
 (case *expr-type* Γ *e* of Some *t* \Rightarrow *op-type* *oper t* | None \Rightarrow None)
 | *expr-type* Γ (<*e1*, *e2*>) =
 (case (*expr-type* Γ *e1*, *expr-type* Γ *e2*) of
 (Some *t1*, Some *t2*) \Rightarrow Some (PRODUCT *t1 t2*)
 | - \Rightarrow None)
 | *expr-type* Γ (Random *dst e*) =
 (if *expr-type* Γ *e* = Some (*dist-param-type* *dst*) then
 Some (*dist-result-type* *dst*)
 else None)
 | *expr-type* Γ (IF *b THEN e1 ELSE e2*) =
 (if *expr-type* Γ *b* = Some *BOOL* then
 (case (*expr-type* Γ *e1*, *expr-type* Γ *e2*) of
 (Some *t*, Some *t'*) \Rightarrow if *t* = *t'* then Some *t* else None
 | - \Rightarrow None) else None)
 | *expr-type* Γ (Fail *t*) = Some *t*

lemma *expr-type-Some-iff*: *expr-type* Γ *e* = Some *t* \longleftrightarrow $\Gamma \vdash e : t$
 ⟨proof⟩

lemmas *expr-typing-code*[code-unfold] = *expr-type-Some-iff*[symmetric]

5.3.1 Countable types

primrec *countable-type* :: *pdf-type* \Rightarrow *bool* **where**

countable-type *UNIT* = True
 | *countable-type* *BOOL* = True
 | *countable-type* *INTEG* = True
 | *countable-type* *REAL* = False
 | *countable-type* (PRODUCT *t1 t2*) = (*countable-type* *t1* \wedge *countable-type* *t2*)

lemma *countable-type-countable*[dest]:

countable-type *t* \implies *countable* (*space* (*stock-measure* *t*))
 ⟨proof⟩

lemma *countable-type-imp-count-space*:
countable-type $t \implies \text{stock-measure } t = \text{count-space } (\text{type-universe } t)$
 ⟨*proof*⟩

lemma *return-val-countable*:
assumes *countable-type* (*val-type* v)
shows *return-val* $v = \text{density } (\text{stock-measure } (\text{val-type } v)) (\text{indicator } \{v\})$ (**is**
 $?M1 = ?M2$)
 ⟨*proof*⟩

5.4 Semantics

definition *bool-to-int* :: *bool* \Rightarrow *int* **where**
bool-to-int $b = (\text{if } b \text{ then } 1 \text{ else } 0)$

lemma *measurable-bool-to-int*[*measurable*]:
bool-to-int $\in \text{measurable } (\text{count-space } UNIV) (\text{count-space } UNIV)$
 ⟨*proof*⟩

definition *bool-to-real* :: *bool* \Rightarrow *real* **where**
bool-to-real $b = (\text{if } b \text{ then } 1 \text{ else } 0)$

lemma *measurable-bool-to-real*[*measurable*]:
bool-to-real $\in \text{borel-measurable } (\text{count-space } UNIV)$
 ⟨*proof*⟩

definition *safe-ln* :: *real* \Rightarrow *real* **where**
safe-ln $x = (\text{if } x > 0 \text{ then } \ln x \text{ else } 0)$

lemma *safe-ln-gt-0*[*simp*]: $x > 0 \implies \text{safe-ln } x = \ln x$
 ⟨*proof*⟩

lemma *borel-measurable-safe-ln*[*measurable*]: *safe-ln* $\in \text{borel-measurable borel}$
 ⟨*proof*⟩

definition *safe-sqrt* :: *real* \Rightarrow *real* **where**
safe-sqrt $x = (\text{if } x \geq 0 \text{ then } \text{sqrt } x \text{ else } 0)$

lemma *safe-sqrt-ge-0*[*simp*]: $x \geq 0 \implies \text{safe-sqrt } x = \text{sqrt } x$
 ⟨*proof*⟩

lemma *borel-measurable-safe-sqrt*[*measurable*]: *safe-sqrt* $\in \text{borel-measurable borel}$
 ⟨*proof*⟩

fun *op-sem* :: *pdf-operator* \Rightarrow *val* \Rightarrow *val* **where**
op-sem *Add* = *lift-RealIntVal2* (+) (+)
 | *op-sem* *Mult* = *lift-RealIntVal2* (*) (*)

$| \text{op-sem Minus} = \text{lift-RealIntVal } \text{uminus } \text{uminus}$
 $| \text{op-sem Equals} = (\lambda \langle |v1, v2| \rangle \Rightarrow \text{BoolVal } (v1 = v2))$
 $| \text{op-sem Less} = \text{lift-Comp } (<) (<)$
 $| \text{op-sem Or} = (\lambda \langle | \text{BoolVal } a, \text{BoolVal } b | \rangle \Rightarrow \text{BoolVal } (a \vee b))$
 $| \text{op-sem And} = (\lambda \langle | \text{BoolVal } a, \text{BoolVal } b | \rangle \Rightarrow \text{BoolVal } (a \wedge b))$
 $| \text{op-sem Not} = (\lambda \text{BoolVal } a \Rightarrow \text{BoolVal } (\neg a))$
 $| \text{op-sem (Cast } t) = (\text{case } t \text{ of}$
 $\quad \text{INTEG} \Rightarrow (\lambda \text{BoolVal } b \Rightarrow \text{IntVal } (\text{bool-to-int } b))$
 $\quad \quad | \text{RealVal } r \Rightarrow \text{IntVal } (\text{floor } r))$
 $\quad | \text{REAL} \Rightarrow (\lambda \text{BoolVal } b \Rightarrow \text{RealVal } (\text{bool-to-real } b))$
 $\quad \quad | \text{IntVal } i \Rightarrow \text{RealVal } (\text{real-of-int } i))$
 $| \text{op-sem Inverse} = \text{lift-RealVal } \text{inverse}$
 $| \text{op-sem Fact} = \text{lift-IntVal } (\lambda i::\text{int. fact } (\text{nat } i))$
 $| \text{op-sem Sqrt} = \text{lift-RealVal } \text{safe-sqrt}$
 $| \text{op-sem Exp} = \text{lift-RealVal } \text{exp}$
 $| \text{op-sem Ln} = \text{lift-RealVal } \text{safe-ln}$
 $| \text{op-sem Pi} = (\lambda \cdot. \text{RealVal } \text{pi})$
 $| \text{op-sem Pow} = (\lambda \langle | \text{RealVal } x, \text{IntVal } n | \rangle \Rightarrow \text{if } n < 0 \text{ then RealVal } 0 \text{ else RealVal } (x \wedge \text{nat } n))$
 $\quad | \langle | \text{IntVal } x, \text{IntVal } n | \rangle \Rightarrow \text{if } n < 0 \text{ then IntVal } 0 \text{ else IntVal } (x \wedge \text{nat } n))$
 $| \text{op-sem Fst} = \text{fst} \circ \text{extract-pair}$
 $| \text{op-sem Snd} = \text{snd} \circ \text{extract-pair}$

The semantics of expressions. Assumes that the expression given is well-typed.

primrec $\text{expr-sem} :: \text{state} \Rightarrow \text{expr} \Rightarrow \text{val measure}$ **where**

$\text{expr-sem } \sigma (\text{Var } x) = \text{return-val } (\sigma x)$
 $| \text{expr-sem } \sigma (\text{Val } v) = \text{return-val } v$
 $| \text{expr-sem } \sigma (\text{LET } e1 \text{ IN } e2) =$
 $\quad \text{do } \{$
 $\quad \quad v \leftarrow \text{expr-sem } \sigma e1;$
 $\quad \quad \text{expr-sem } (v \cdot \sigma) e2$
 $\quad \}$
 $| \text{expr-sem } \sigma (\text{oper } \$\$ e) =$
 $\quad \text{do } \{$
 $\quad \quad x \leftarrow \text{expr-sem } \sigma e;$
 $\quad \quad \text{return-val } (\text{op-sem oper } x)$
 $\quad \}$
 $| \text{expr-sem } \sigma \langle v, w \rangle =$
 $\quad \text{do } \{$
 $\quad \quad x \leftarrow \text{expr-sem } \sigma v;$
 $\quad \quad y \leftarrow \text{expr-sem } \sigma w;$
 $\quad \quad \text{return-val } \langle |x, y| \rangle$
 $\quad \}$
 $| \text{expr-sem } \sigma (\text{IF } b \text{ THEN } e1 \text{ ELSE } e2) =$
 $\quad \text{do } \{$
 $\quad \quad b' \leftarrow \text{expr-sem } \sigma b;$
 $\quad \quad \text{if } b' = \text{TRUE} \text{ then expr-sem } \sigma e1 \text{ else expr-sem } \sigma e2$

$$\begin{aligned}
& \} \\
| \text{expr-sem } \sigma \text{ (Random dst e)} &= \\
& \text{do } \{ \\
& \quad x \leftarrow \text{expr-sem } \sigma \text{ e}; \\
& \quad \text{dist-measure dst x} \\
& \} \\
| \text{expr-sem } \sigma \text{ (Fail t)} &= \text{null-measure (stock-measure t)}
\end{aligned}$$

lemma *expr-sem-pair-vars*: $\text{expr-sem } \sigma \text{ } \langle \text{Var } x, \text{Var } y \rangle = \text{return-val } \langle | \sigma \text{ x}, \sigma \text{ y} | \rangle$
<proof>

Well-typed expressions produce a result in the measure space that corresponds to their type

lemma *op-sem-val-type*:
 $\text{op-type oper (val-type v)} = \text{Some } t' \implies \text{val-type (op-sem oper v)} = t'$
<proof>

lemma *sets-expr-sem*:
 $\Gamma \vdash w : t \implies (\forall x \in \text{free-vars } w. \text{val-type } (\sigma \text{ x}) = \Gamma \text{ x}) \implies$
 $\text{sets (expr-sem } \sigma \text{ w)} = \text{sets (stock-measure t)}$
<proof>

lemma *space-expr-sem*:
 $\Gamma \vdash w : t \implies (\forall x \in \text{free-vars } w. \text{val-type } (\sigma \text{ x}) = \Gamma \text{ x})$
 $\implies \text{space (expr-sem } \sigma \text{ w)} = \text{type-universe t}$
<proof>

lemma *measurable-expr-sem-eq*:
 $\Gamma \vdash e : t \implies \sigma \in \text{space (state-measure V } \Gamma) \implies \text{free-vars } e \subseteq V \implies$
 $\text{measurable (expr-sem } \sigma \text{ e)} = \text{measurable (stock-measure t)}$
<proof>

lemma *measurable-expr-semI*:
 $\Gamma \vdash e : t \implies \sigma \in \text{space (state-measure V } \Gamma) \implies \text{free-vars } e \subseteq V \implies$
 $f \in \text{measurable (stock-measure t)} \text{ } M \implies f \in \text{measurable (expr-sem } \sigma \text{ e)} \text{ } M$
<proof>

lemma *expr-sem-eq-on-vars*:
 $(\bigwedge x. x \in \text{free-vars } e \implies \sigma_1 \text{ x} = \sigma_2 \text{ x}) \implies \text{expr-sem } \sigma_1 \text{ e} = \text{expr-sem } \sigma_2 \text{ e}$
<proof>

5.5 Measurability

lemma *borel-measurable-eq[measurable (raw)]*:
assumes [*measurable*]: $f \in \text{borel-measurable } M \text{ } g \in \text{borel-measurable } M$
shows $\text{Measurable.pred } M \text{ } (\lambda x. f \text{ x} = (g \text{ x} :: \text{real}))$
<proof>

lemma *measurable-equals*:

$(\lambda(x,y). x = y) \in \text{measurable } (\text{stock-measure } t \otimes_M \text{stock-measure } t) \text{ (count-space UNIV)}$
 $\langle \text{proof} \rangle$

lemma *measurable-equals-stock-measure*[*measurable (raw)*]:

assumes $f \in \text{measurable } M \text{ (stock-measure } t) \text{ } g \in \text{measurable } M \text{ (stock-measure } t)$
shows $\text{Measurable.pred } M \ (\lambda x. f \ x = g \ x)$
 $\langle \text{proof} \rangle$

lemma *measurable-op-sem*:

assumes *op-type* $\text{oper } t = \text{Some } t'$
shows $\text{op-sem } \text{oper} \in \text{measurable } (\text{stock-measure } t) \text{ (stock-measure } t')$
 $\langle \text{proof} \rangle$

definition *shift-var-set* :: *vname set* \Rightarrow *vname set* **where**

shift-var-set $V = \text{insert } 0 \ (\text{Suc } ' V)$

lemma *shift-var-set-0*[*simp*]: $0 \in \text{shift-var-set } V$

$\langle \text{proof} \rangle$

lemma *shift-var-set-Suc*[*simp*]: $\text{Suc } x \in \text{shift-var-set } V \longleftrightarrow x \in V$

$\langle \text{proof} \rangle$

lemma *case-nat-update-0*[*simp*]: $(\text{case-nat } x \ \sigma)(0 := y) = \text{case-nat } y \ \sigma$

$\langle \text{proof} \rangle$

lemma *case-nat-delete-var-1*[*simp*]:

$\text{case-nat } x \ (\text{case-nat } y \ \sigma) \circ \text{case-nat } 0 \ (\lambda x. \text{Suc } (\text{Suc } x)) = \text{case-nat } x \ \sigma$
 $\langle \text{proof} \rangle$

lemma *delete-var-1-vimage*[*simp*]:

$\text{case-nat } 0 \ (\lambda x. \text{Suc } (\text{Suc } x)) - ' (\text{shift-var-set } (\text{shift-var-set } V)) = \text{shift-var-set } V$
 $\langle \text{proof} \rangle$

lemma *measurable-case-nat*[*measurable*]:

assumes $g \in \text{measurable } R \ N \ h \in \text{measurable } R \ (Pi_M \ V \ M)$
shows $(\lambda x. \text{case-nat } (g \ x) \ (h \ x)) \in \text{measurable } R \ (Pi_M \ (\text{shift-var-set } V) \ (\text{case-nat } N \ M))$
 $\langle \text{proof} \rangle$

lemma *measurable-case-nat'*[*measurable*]:

assumes $g \in \text{measurable } R \ (\text{stock-measure } t) \ h \in \text{measurable } R \ (\text{state-measure } V \ \Gamma)$
shows $(\lambda x. \text{case-nat } (g \ x) \ (h \ x)) \in \text{measurable } R \ (\text{state-measure } (\text{shift-var-set } V) \ (\text{case-nat } t \ \Gamma))$

<proof>

lemma *case-nat-in-state-measure*[intro]:

assumes $x \in \text{type-universe } t1 \ \sigma \in \text{space } (\text{state-measure } V \ \Gamma)$

shows $\text{case-nat } x \ \sigma \in \text{space } (\text{state-measure } (\text{shift-var-set } V) (\text{case-nat } t1 \ \Gamma))$

<proof>

lemma *subset-shift-var-set*:

$\text{Suc } - ' A \subseteq V \implies A \subseteq \text{shift-var-set } V$

<proof>

lemma *measurable-expr-sem*[measurable]:

assumes $\Gamma \vdash e : t$ **and** $\text{free-vars } e \subseteq V$

shows $(\lambda\sigma. \text{expr-sem } \sigma \ e) \in \text{measurable } (\text{state-measure } V \ \Gamma)$

$(\text{subprob-algebra } (\text{stock-measure } t))$

<proof>

5.6 Randomfree expressions

fun *randomfree* :: $\text{expr} \Rightarrow \text{bool}$ **where**

$\text{randomfree } (\text{Val } -) = \text{True}$

| $\text{randomfree } (\text{Var } -) = \text{True}$

| $\text{randomfree } (\text{Pair } e1 \ e2) = (\text{randomfree } e1 \ \wedge \ \text{randomfree } e2)$

| $\text{randomfree } (\text{Operator } - \ e) = \text{randomfree } e$

| $\text{randomfree } (\text{LetVar } e1 \ e2) = (\text{randomfree } e1 \ \wedge \ \text{randomfree } e2)$

| $\text{randomfree } (\text{IfThenElse } b \ e1 \ e2) = (\text{randomfree } b \ \wedge \ \text{randomfree } e1 \ \wedge \ \text{randomfree } e2)$

| $\text{randomfree } (\text{Random } -) = \text{False}$

| $\text{randomfree } (\text{Fail } -) = \text{False}$

primrec *expr-sem-rf* :: $\text{state} \Rightarrow \text{expr} \Rightarrow \text{val}$ **where**

$\text{expr-sem-rf } - (\text{Val } v) = v$

| $\text{expr-sem-rf } \sigma (\text{Var } x) = \sigma \ x$

| $\text{expr-sem-rf } \sigma (<e1, e2>) = <|\text{expr-sem-rf } \sigma \ e1, \text{expr-sem-rf } \sigma \ e2|>$

| $\text{expr-sem-rf } \sigma (\text{Operator } \text{oper } e) = \text{op-sem } \text{oper } (\text{expr-sem-rf } \sigma \ e)$

| $\text{expr-sem-rf } \sigma (\text{LetVar } e1 \ e2) = \text{expr-sem-rf } (\text{expr-sem-rf } \sigma \ e1 \cdot \sigma) \ e2$

| $\text{expr-sem-rf } \sigma (\text{IfThenElse } b \ e1 \ e2) =$

$(\text{if } \text{expr-sem-rf } \sigma \ b = \text{BoolVal } \text{True} \text{ then } \text{expr-sem-rf } \sigma \ e1 \text{ else } \text{expr-sem-rf } \sigma \ e2)$

| $\text{expr-sem-rf } - (\text{Random } -) = \text{undefined}$

| $\text{expr-sem-rf } - (\text{Fail } -) = \text{undefined}$

lemma *measurable-expr-sem-rf*[measurable]:

$\Gamma \vdash e : t \implies \text{randomfree } e \implies \text{free-vars } e \subseteq V \implies$

$(\lambda\sigma. \text{expr-sem-rf } \sigma \ e) \in \text{measurable } (\text{state-measure } V \ \Gamma) (\text{stock-measure } t)$

<proof>

lemma *expr-sem-rf-sound*:

$\Gamma \vdash e : t \implies \text{randomfree } e \implies \text{free-vars } e \subseteq V \implies \sigma \in \text{space } (\text{state-measure } V \Gamma) \implies$
 $\text{return-val } (\text{expr-sem-rf } \sigma e) = \text{expr-sem } \sigma e$
 <proof>

lemma *val-type-expr-sem-rf*:

assumes $\Gamma \vdash e : t$ *randomfree* e *free-vars* $e \subseteq V$ $\sigma \in \text{space } (\text{state-measure } V \Gamma)$
shows $\text{val-type } (\text{expr-sem-rf } \sigma e) = t$
 <proof>

lemma *expr-sem-rf-eq-on-vars*:

$(\bigwedge x. x \in \text{free-vars } e \implies \sigma 1 x = \sigma 2 x) \implies \text{expr-sem-rf } \sigma 1 e = \text{expr-sem-rf } \sigma 2 e$
 <proof>

end

6 Density Contexts

theory *PDF-Density-Contexts*

imports *PDF-Semantics*

begin

lemma *measurable-proj-state-measure*[*measurable (raw)*]:

$i \in V \implies (\lambda x. x i) \in \text{measurable } (\text{state-measure } V \Gamma) (\Gamma i)$
 <proof>

lemma *measurable-dens-ctxt-fun-upd*[*measurable (raw)*]:

$f \in N \rightarrow_M \text{state-measure } V' \Gamma \implies V = V' \cup \{x\} \implies$
 $g \in N \rightarrow_M \text{stock-measure } (\Gamma x) \implies$
 $(\lambda \omega. (f \omega)(x := g \omega)) \in N \rightarrow_M \text{state-measure } V \Gamma$
 <proof>

lemma *measurable-case-nat-Suc-PiM*:

$(\lambda \sigma. \sigma \circ \text{Suc}) \in \text{measurable } (\text{PiM } (\text{Suc } 'A) (\text{case-nat } M N)) (\text{PiM } A N)$
 <proof>

lemma *measurable-case-nat-Suc*:

$(\lambda \sigma. \sigma \circ \text{Suc}) \in \text{measurable } (\text{state-measure } (\text{Suc } 'A) (\text{case-nat } t \Gamma)) (\text{state-measure } A \Gamma)$
 <proof>

A density context holds a set of variables V , their types (using Γ), and a common density function δ of the finite product space of all the variables in V . δ takes a state $\sigma \in (\prod_E x \in V. \text{type-universe } (\Gamma x))$ and returns the common density of these variables.

type-synonym $\text{dens-ctxt} = \text{vname set} \times \text{vname set} \times (\text{vname} \Rightarrow \text{pdf-type}) \times (\text{state} \Rightarrow \text{ennreal})$

type-synonym $\text{expr-density} = \text{state} \Rightarrow \text{val} \Rightarrow \text{ennreal}$

definition $\text{empty-dens-ctxt} :: \text{dens-ctxt}$ **where**

$\text{empty-dens-ctxt} = (\{\}, \{\}, \lambda-. \text{undefined}, \lambda-. 1)$

definition $\text{state-measure}'$

$:: \text{vname set} \Rightarrow \text{vname set} \Rightarrow (\text{vname} \Rightarrow \text{pdf-type}) \Rightarrow \text{state} \Rightarrow \text{state measure}$

where

$\text{state-measure}' V V' \Gamma \varrho =$

$\text{distr} (\text{state-measure } V \Gamma) (\text{state-measure } (V \cup V') \Gamma) (\lambda\sigma. \text{merge } V V' (\sigma, \varrho))$

The marginal density of a variable x is obtained by integrating the common density δ over all the remaining variables.

definition $\text{marg-dens} :: \text{dens-ctxt} \Rightarrow \text{vname} \Rightarrow \text{expr-density}$ **where**

$\text{marg-dens} = (\lambda(V, V', \Gamma, \delta) x \varrho v. \int^{+\sigma}. \delta (\text{merge } V V' (\sigma(x := v), \varrho)) \partial \text{state-measure } (V - \{x\}) \Gamma)$

definition $\text{marg-dens2} :: \text{dens-ctxt} \Rightarrow \text{vname} \Rightarrow \text{vname} \Rightarrow \text{expr-density}$ **where**

$\text{marg-dens2} \equiv (\lambda(V, V', \Gamma, \delta) x y \varrho v.$

$\int^{+\sigma}. \delta (\text{merge } V V' (\sigma(x := \text{fst} (\text{extract-pair } v), y := \text{snd} (\text{extract-pair } v)), \varrho))$

$\partial \text{state-measure } (V - \{x, y\}) \Gamma)$

definition $\text{dens-ctxt-measure} :: \text{dens-ctxt} \Rightarrow \text{state} \Rightarrow \text{state measure}$ **where**

$\text{dens-ctxt-measure} \equiv \lambda(V, V', \Gamma, \delta) \varrho. \text{density} (\text{state-measure}' V V' \Gamma \varrho) \delta$

definition $\text{branch-prob} :: \text{dens-ctxt} \Rightarrow \text{state} \Rightarrow \text{ennreal}$ **where**

$\text{branch-prob } \mathcal{Y} \varrho = \text{emeasure} (\text{dens-ctxt-measure } \mathcal{Y} \varrho) (\text{space} (\text{dens-ctxt-measure } \mathcal{Y} \varrho))$

lemma $\text{dens-ctxt-measure-nonempty[simp]}$:

$\text{space} (\text{dens-ctxt-measure } \mathcal{Y} \varrho) \neq \{\}$

$\langle \text{proof} \rangle$

lemma $\text{sets-dens-ctxt-measure-eq[measurable-cong]}$:

$\text{sets} (\text{dens-ctxt-measure } (V, V', \Gamma, \delta) \varrho) = \text{sets} (\text{state-measure } (V \cup V') \Gamma)$

$\langle \text{proof} \rangle$

lemma $\text{measurable-dens-ctxt-measure-eq}$:

$\text{measurable} (\text{dens-ctxt-measure } (V, V', \Gamma, \delta) \varrho) = \text{measurable} (\text{state-measure } (V \cup V') \Gamma)$

$\langle \text{proof} \rangle$

lemma $\text{space-dens-ctxt-measure}$:

$\text{space} (\text{dens-ctxt-measure } (V, V', \Gamma, \delta) \varrho) = \text{space} (\text{state-measure } (V \cup V') \Gamma)$

$\langle \text{proof} \rangle$

definition *apply-dist-to-dens* :: *pdf-dist* \Rightarrow (*state* \Rightarrow *val* \Rightarrow *ennreal*) \Rightarrow (*state* \Rightarrow *val* \Rightarrow *ennreal*) **where**
apply-dist-to-dens *dst* *f* = ($\lambda \varrho y. \int^+ x. f \varrho x * \text{dist-dens } \text{dst } x y \partial \text{stock-measure } (\text{dist-param-type } \text{dst})$)

definition *remove-var* :: *state* \Rightarrow *state* **where**
remove-var $\sigma = (\lambda x. \sigma (\text{Suc } x))$

lemma *measurable-remove-var*[*measurable*]:
remove-var \in *measurable* (*state-measure* (*shift-var-set* *V*) (*case-nat* *t* Γ)) (*state-measure* *V* Γ)
 $\langle \text{proof} \rangle$

lemma *measurable-case-nat-undefined*[*measurable*]:
case-nat undefined \in *measurable* (*state-measure* *A* Γ) (*state-measure* (*Suc* 'A) (*case-nat* *t* Γ)) (*is -* \in ?*M*)
 $\langle \text{proof} \rangle$

definition *insert-dens*
:: *vname set* \Rightarrow *vname set* \Rightarrow *expr-density* \Rightarrow (*state* \Rightarrow *ennreal*) \Rightarrow *state* \Rightarrow *ennreal* **where**
insert-dens *V* *V'* *f* $\delta \equiv \lambda \sigma. \delta (\text{remove-var } \sigma) * f (\text{remove-var } \sigma) (\sigma \ 0)$

definition *if-dens* :: (*state* \Rightarrow *ennreal*) \Rightarrow (*state* \Rightarrow *val* \Rightarrow *ennreal*) \Rightarrow *bool* \Rightarrow (*state* \Rightarrow *ennreal*) **where**
if-dens δ *f* *b* $\equiv \lambda \sigma. \delta \sigma * f \sigma (\text{BoolVal } b)$

definition *if-dens-det* :: (*state* \Rightarrow *ennreal*) \Rightarrow *expr* \Rightarrow *bool* \Rightarrow (*state* \Rightarrow *ennreal*) **where**
if-dens-det δ *e* *b* $\equiv \lambda \sigma. \delta \sigma * (\text{if } \text{expr-sem-rf } \sigma \ e = \text{BoolVal } b \text{ then } 1 \text{ else } 0)$

lemma *measurable-if-dens*:
assumes [*measurable*]: $\delta \in \text{borel-measurable } M$
assumes [*measurable*]: *case-prod* *f* \in *borel-measurable* ($M \otimes_M \text{count-space } (\text{range } \text{BoolVal})$)
shows *if-dens* δ *f* *b* \in *borel-measurable* *M*
 $\langle \text{proof} \rangle$

lemma *measurable-if-dens-det*:
assumes *e*: $\Gamma \vdash e : \text{BOOL randomfree } e \text{ free-vars } e \subseteq V$
assumes [*measurable*]: $\delta \in \text{borel-measurable } (\text{state-measure } V \ \Gamma)$
shows *if-dens-det* δ *e* *b* \in *borel-measurable* (*state-measure* *V* Γ)
 $\langle \text{proof} \rangle$

locale *density-context* =
fixes *V* *V'* Γ δ
assumes *subprob-space-dens*:
 $\bigwedge \varrho. \varrho \in \text{space } (\text{state-measure } V' \ \Gamma) \implies \text{subprob-space } (\text{dens-ctxt-measure } \varrho)$

$(V, V', \Gamma, \delta) \varrho$
and *finite-vars*[*simp*]: $\text{finite } V \text{ finite } V'$
and *measurable-dens*[*measurable*]:
 $\delta \in \text{borel-measurable } (\text{state-measure } (V \cup V') \Gamma)$
and *disjoint*: $V \cap V' = \{\}$
begin

abbreviation $\mathcal{Y} \equiv (V, V', \Gamma, \delta)$

lemma *branch-prob-altdef*:
assumes $\varrho: \varrho \in \text{space } (\text{state-measure } V' \Gamma)$
shows $\text{branch-prob } \mathcal{Y} \varrho = \int^+ x. \delta (\text{merge } V V' (x, \varrho)) \partial \text{state-measure } V \Gamma$
 $\langle \text{proof} \rangle$

lemma *measurable-branch-prob*[*measurable*]:
 $\text{branch-prob } \mathcal{Y} \in \text{borel-measurable } (\text{state-measure } V' \Gamma)$
 $\langle \text{proof} \rangle$

lemma *measurable-marg-dens'*:
assumes [*simp*]: $x \in V$
shows $\text{case-prod } (\text{marg-dens } \mathcal{Y} x) \in \text{borel-measurable } (\text{state-measure } V' \Gamma \otimes_M \text{stock-measure } (\Gamma x))$
 $\langle \text{proof} \rangle$

lemma *insert-Diff*: $\text{insert } x (A - B) = \text{insert } x A - (B - \{x\})$
 $\langle \text{proof} \rangle$

lemma *measurable-marg-dens2'*:
assumes $x \in V y \in V$
shows $\text{case-prod } (\text{marg-dens2 } \mathcal{Y} x y) \in \text{borel-measurable } (\text{state-measure } V' \Gamma \otimes_M \text{stock-measure } (\text{PRODUCT } (\Gamma x) (\Gamma y)))$
 $\langle \text{proof} \rangle$

lemma *measurable-marg-dens*:
assumes $x \in V \varrho \in \text{space } (\text{state-measure } V' \Gamma)$
shows $\text{marg-dens } \mathcal{Y} x \varrho \in \text{borel-measurable } (\text{stock-measure } (\Gamma x))$
 $\langle \text{proof} \rangle$

lemma *measurable-marg-dens2*:
assumes $x \in V y \in V x \neq y \varrho \in \text{space } (\text{state-measure } V' \Gamma)$
shows $\text{marg-dens2 } \mathcal{Y} x y \varrho \in \text{borel-measurable } (\text{stock-measure } (\text{PRODUCT } (\Gamma x) (\Gamma y)))$
 $\langle \text{proof} \rangle$

lemma *measurable-state-measure-component*:
 $x \in V \implies (\lambda \sigma. \sigma x) \in \text{measurable } (\text{state-measure } V \Gamma) (\text{stock-measure } (\Gamma x))$
 $\langle \text{proof} \rangle$

lemma *measurable-dens-ctxt-measure-component*:

$x \in V \implies (\lambda\sigma. \sigma x) \in \text{measurable} (\text{dens-ctxt-measure} (V, V', \Gamma, \delta) \varrho) (\text{stock-measure} (\Gamma x))$
 ⟨proof⟩

lemma *space-dens-ctxt-measure-dens-ctxt-measure'*:

assumes $x \in V$
shows $\text{space} (\text{state-measure} V \Gamma) = \{\sigma(x := y) \mid \sigma y. \sigma \in \text{space} (\text{state-measure} (V - \{x\}) \Gamma) \wedge y \in \text{type-universe} (\Gamma x)\}$
 ⟨proof⟩

lemma *state-measure-integral-split*:

assumes $x \in A$ *finite* A
assumes $f \in \text{borel-measurable} (\text{state-measure} A \Gamma)$
shows $(\int^+ \sigma. f \sigma \partial \text{state-measure} A \Gamma) = (\int^+ y. \int^+ \sigma. f (\sigma(x := y)) \partial \text{state-measure} (A - \{x\}) \Gamma) \partial \text{stock-measure} (\Gamma x)$
 ⟨proof⟩

lemma *fun-upd-in-state-measure*:

$[\sigma \in \text{space} (\text{state-measure} A \Gamma); y \in \text{space} (\text{stock-measure} (\Gamma x))]$
 $\implies \sigma(x := y) \in \text{space} (\text{state-measure} (\text{insert } x A) \Gamma)$
 ⟨proof⟩

lemma *marg-dens-integral*:

fixes $X :: \text{val set}$ **assumes** $x \in V$ **and** $[\text{measurable}]: X \in \text{sets} (\text{stock-measure} (\Gamma x))$
assumes $\varrho \in \text{space} (\text{state-measure} V' \Gamma)$
defines $X' \equiv (\lambda\sigma. \sigma x) - ' X \cap \text{space} (\text{state-measure} V \Gamma)$
shows $(\int^+ y. \text{marg-dens} \mathcal{Y} x \varrho y * \text{indicator } X y \partial \text{stock-measure} (\Gamma x)) = (\int^+ \sigma. \delta (\text{merge } V V' (\sigma, \varrho)) * \text{indicator } X' \sigma \partial \text{state-measure} V \Gamma)$
 ⟨proof⟩

lemma *marg-dens2-integral*:

fixes $X :: \text{val set}$
assumes $x \in V$ $y \in V$ $x \neq y$ **and** $[\text{measurable}]: X \in \text{sets} (\text{stock-measure} (\text{PRODUCT} (\Gamma x) (\Gamma y)))$
assumes $\varrho \in \text{space} (\text{state-measure} V' \Gamma)$
defines $X' \equiv (\lambda\sigma. \langle \sigma x, \sigma y \rangle) - ' X \cap \text{space} (\text{state-measure} V \Gamma)$
shows $(\int^+ z. \text{marg-dens2} \mathcal{Y} x y \varrho z * \text{indicator } X z \partial \text{stock-measure} (\text{PRODUCT} (\Gamma x) (\Gamma y))) = (\int^+ \sigma. \delta (\text{merge } V V' (\sigma, \varrho)) * \text{indicator } X' \sigma \partial \text{state-measure} V \Gamma)$
 ⟨proof⟩

The space described by the marginal density is the same as the space obtained by projecting x (resp. x and y) out of the common distribution of all variables.

lemma *density-marg-dens-eq*:

assumes $x \in V \ \varrho \in \text{space } (\text{state-measure } V' \ \Gamma)$
shows $\text{density } (\text{stock-measure } (\Gamma \ x)) \ (\text{marg-dens } \mathcal{Y} \ x \ \varrho) =$
 $\text{distr } (\text{dens-ctxt-measure } (V, V', \Gamma, \delta) \ \varrho) \ (\text{stock-measure } (\Gamma \ x)) \ (\lambda\sigma. \ \sigma \ x)$
(is ?M1 = ?M2)
 $\langle \text{proof} \rangle$

lemma *density-marg-dens2-eq*:

assumes $x \in V \ y \in V \ x \neq y \ \varrho \in \text{space } (\text{state-measure } V' \ \Gamma)$
defines $M \equiv \text{stock-measure } (\text{PRODUCT } (\Gamma \ x) \ (\Gamma \ y))$
shows $\text{density } M \ (\text{marg-dens2 } \mathcal{Y} \ x \ y \ \varrho) =$
 $\text{distr } (\text{dens-ctxt-measure } (V, V', \Gamma, \delta) \ \varrho) \ M \ (\lambda\sigma. \ \langle \sigma \ x, \sigma \ y \rangle)$ **(is ?M1**
 $= ?M2)$
 $\langle \text{proof} \rangle$

lemma *measurable-insert-dens[measurable]*:

assumes $Mf[\text{measurable}]$: $\text{case-prod } f \in \text{borel-measurable } (\text{state-measure } (V \cup V') \ \Gamma \ \otimes_M \ \text{stock-measure } t)$
shows $\text{insert-dens } V \ V' \ f \ \delta$
 $\in \text{borel-measurable } (\text{state-measure } (\text{shift-var-set } (V \cup V')) \ (\text{case-nat } t \ \Gamma))$
 $\langle \text{proof} \rangle$

lemma *nn-integral-dens-ctxt-measure*:

assumes $\varrho \in \text{space } (\text{state-measure } V' \ \Gamma)$
 $f \in \text{borel-measurable } (\text{state-measure } (V \cup V') \ \Gamma)$
shows $(\int^+ x. f \ x \ \partial \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \ \varrho) =$
 $\int^+ x. \ \delta \ (\text{merge } V \ V' \ (x, \varrho)) * f \ (\text{merge } V \ V' \ (x, \varrho)) \ \partial \text{state-measure } V \ \Gamma$
 $\langle \text{proof} \rangle$

lemma *shift-var-set-Un[simp]*: $\text{shift-var-set } V \cup \text{Suc } 'V' = \text{shift-var-set } (V \cup V')$
 $\langle \text{proof} \rangle$

lemma *emeasure-dens-ctxt-measure-insert*:

fixes $t \ f \ \varrho$
defines $M \equiv \text{dens-ctxt-measure } (\text{shift-var-set } V, \text{Suc } 'V', \text{case-nat } t \ \Gamma, \text{insert-dens } V \ V' \ f \ \delta) \ \varrho$
assumes dens : $\text{has-parametrized-subprob-density } (\text{state-measure } (V \cup V') \ \Gamma) \ F$
 $(\text{stock-measure } t) \ f$
assumes ϱ : $\varrho \in \text{space } (\text{state-measure } (\text{Suc } 'V') \ (\text{case-nat } t \ \Gamma))$
assumes X : $X \in \text{sets } M$
shows $\text{emeasure } M \ X =$
 $\int^+ x. \ \text{insert-dens } V \ V' \ f \ \delta \ (\text{merge } (\text{shift-var-set } V) \ (\text{Suc } 'V') \ (x, \varrho)) * \text{indicator } X \ (\text{merge } (\text{shift-var-set } V) \ (\text{Suc } 'V') \ (x, \varrho))$
 $\partial \text{state-measure } (\text{shift-var-set } V) \ (\text{case-nat } t \ \Gamma)$ **(is - = ?I)**
 $\langle \text{proof} \rangle$

lemma *merge-Suc-aux'*:

$\varrho \in \text{space } (\text{state-measure } (\text{Suc } 'V') \ (\text{case-nat } t \ \Gamma)) \implies$
 $(\lambda\sigma. \ \text{merge } V \ V' \ (\sigma, \varrho \circ \text{Suc})) \in \text{measurable } (\text{state-measure } V \ \Gamma) \ (\text{state-measure } V' \ \Gamma)$

$(V \cup V') \Gamma$
 $\langle \text{proof} \rangle$

lemma *merge-Suc-aux*:

$\varrho \in \text{space } (\text{state-measure } (Suc' V') (\text{case-nat } t \Gamma)) \implies$
 $(\lambda \sigma. \delta (\text{merge } V V' (\sigma, \varrho \circ Suc))) \in \text{borel-measurable } (\text{state-measure } V \Gamma)$
 $\langle \text{proof} \rangle$

lemma *nn-integral-PiM-Suc*:

assumes *fin*: $\bigwedge i. \text{sigma-finite-measure } (N i)$
assumes *Mf*: $f \in \text{borel-measurable } (Pi_M V N)$
shows $(\int^{+x}. f x \partial \text{distr } (Pi_M (Suc' V)) (\text{case-nat } M N)) (Pi_M V N) (\lambda \sigma. \sigma \circ Suc) =$
 $(\int^{+x}. f x \partial Pi_M V N)$
 $(\text{is nn-integral } (?M1 V) - = -)$
 $\langle \text{proof} \rangle$

lemma *PiM-Suc*:

assumes $\bigwedge i. \text{sigma-finite-measure } (N i)$
shows $\text{distr } (Pi_M (Suc' V)) (\text{case-nat } M N) (Pi_M V N) (\lambda \sigma. \sigma \circ Suc) = Pi_M$
 $V N$ **(is ?M1 = ?M2)**
 $\langle \text{proof} \rangle$

lemma *distr-state-measure-Suc*:

$\text{distr } (\text{state-measure } (Suc' V)) (\text{case-nat } t \Gamma) (\text{state-measure } V \Gamma) (\lambda \sigma. \sigma \circ Suc)$
 $=$
 $\text{state-measure } V \Gamma$ **(is ?M1 = ?M2)**
 $\langle \text{proof} \rangle$

lemma *emeasure-dens-ctxt-measure-insert'*:

fixes $t f \varrho$
defines $M \equiv \text{dens-ctxt-measure } (\text{shift-var-set } V, Suc' V', \text{case-nat } t \Gamma, \text{insert-dens } V V' f \delta) \varrho$
assumes *dens*: *has-parametrized-subprob-density* $(\text{state-measure } (V \cup V') \Gamma) F$
 $(\text{stock-measure } t) f$
assumes ϱ : $\varrho \in \text{space } (\text{state-measure } (Suc' V') (\text{case-nat } t \Gamma))$
assumes *X*: $X \in \text{sets } M$
shows $\text{emeasure } M X = \int^{+\sigma}. \delta (\text{merge } V V' (\sigma, \varrho \circ Suc)) * \int^{+y}. f (\text{merge } V$
 $V' (\sigma, \varrho \circ Suc)) y *$
 $\text{indicator } X (\text{merge } (\text{shift-var-set } V) (Suc' V') (\text{case-nat } y \sigma, \varrho))$
 $\partial \text{stock-measure } t \partial \text{state-measure } V \Gamma$ **(is - = ?I)**
 $\langle \text{proof} \rangle$

lemma *density-context-insert*:

assumes *dens*: *has-parametrized-subprob-density* $(\text{state-measure } (V \cup V') \Gamma) F$
 $(\text{stock-measure } t) f$
shows *density-context* $(\text{shift-var-set } V) (Suc' V') (\text{case-nat } t \Gamma) (\text{insert-dens } V$
 $V' f \delta)$

(**is density-context** ?V ?V' ?Γ' ?δ')

<proof>

lemma *dens-ctxt-measure-insert*:

assumes ϱ : $\varrho \in \text{space } (\text{state-measure } V' \Gamma)$

assumes *meas-M*: $M \in \text{measurable } (\text{state-measure } (V \cup V') \Gamma)$ (*subprob-algebra* (*stock-measure* t))

assumes *meas-f*[*measurable*]: *case-prod* $f \in \text{borel-measurable } (\text{state-measure } (V \cup V') \Gamma \otimes_M \text{stock-measure } t)$

assumes *has-dens*: $\bigwedge \varrho. \varrho \in \text{space } (\text{state-measure } (V \cup V') \Gamma) \implies \text{has-subprob-density } (M \varrho) (\text{stock-measure } t) (f \varrho)$

shows *do* { $\sigma \leftarrow \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \varrho$;
 $y \leftarrow M \sigma$;
 $\text{return } (\text{state-measure } (\text{shift-var-set } (V \cup V'))) (\text{case-nat } t \Gamma) (\text{case-nat } y \sigma)$ } =
 $\text{dens-ctxt-measure } (\text{shift-var-set } V, \text{Suc}'V', \text{case-nat } t \Gamma, \text{insert-dens } V V' f \delta)$
 $(\text{case-nat undefined } \varrho)$
(**is** *bind* ?N (λ -. *bind* - (λ -. *return* ?R -)) = *dens-ctxt-measure* (?V, ?V', ?Γ', ?δ')
-)
<proof>

lemma *density-context-if-dens*:

assumes *has-parametrized-subprob-density* (*state-measure* $(V \cup V') \Gamma$) M
(*count-space* (*range* *BoolVal*)) f

shows *density-context* $V V' \Gamma$ (*if-dens* $\delta f b$)

<proof>

lemma *density-context-if-dens-det*:

assumes $e: \Gamma \vdash e : \text{BOOL randomfree } e \text{ free-vars } e \subseteq V \cup V'$

shows *density-context* $V V' \Gamma$ (*if-dens-det* $\delta e b$)

<proof>

lemma *density-context-empty[simp]*: *density-context* {} $(V \cup V') \Gamma$ (λ -. 1)

<proof>

lemma *dens-ctxt-measure-bind-const*:

assumes $\varrho \in \text{space } (\text{state-measure } V' \Gamma)$ *subprob-space* N

shows *dens-ctxt-measure* $\mathcal{Y} \varrho \gg (\lambda$ -. $N) = \text{density } N (\lambda$ -. *branch-prob* $\mathcal{Y} \varrho)$ (**is** ?M1 = ?M2)

<proof>

lemma *nn-integral-dens-ctxt-measure-restrict*:

assumes $\varrho \in \text{space } (\text{state-measure } V' \Gamma)$ $f \varrho \geq 0$

assumes $f \in \text{borel-measurable } (\text{state-measure } V' \Gamma)$

shows $(\int^+ x. f (\text{restrict } x V') \partial \text{dens-ctxt-measure } \mathcal{Y} \varrho) = \text{branch-prob } \mathcal{Y} \varrho * f \varrho$

<proof>

lemma *expr-sem-op-eq-distr*:

assumes $\Gamma \vdash \text{oper } \text{\$ \$ } e : t' \text{ free-vars } e \subseteq V \cup V' \varrho \in \text{space } (\text{state-measure } V' \Gamma)$

defines $M \equiv \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \varrho$

shows $M \gg \gg (\lambda \sigma. \text{expr-sem } \sigma (\text{oper } \text{\$ \$ } e)) =$

$\text{distr } (M \gg \gg (\lambda \sigma. \text{expr-sem } \sigma e)) (\text{stock-measure } t') (\text{op-sem oper})$

<proof>

end

lemma *density-context-equiv*:

assumes $\bigwedge \sigma. \sigma \in \text{space } (\text{state-measure } (V \cup V') \Gamma) \implies \delta \sigma = \delta' \sigma$

assumes [*simp, measurable*]: $\delta' \in \text{borel-measurable } (\text{state-measure } (V \cup V') \Gamma)$

assumes *density-context* $V V' \Gamma \delta$

shows *density-context* $V V' \Gamma \delta'$

<proof>

end

7 Abstract PDF Compiler

theory *PDF-Compiler-Pred*

imports *PDF-Semantics PDF-Density-Contexts PDF-Transformations Density-Predicates*

begin

7.1 Density compiler predicate

Predicate version of the probability density compiler that compiles an expression to a probability density function of its distribution. The density is a HOL function of type $\text{val} \Rightarrow \text{ennreal}$.

inductive *expr-has-density* :: $\text{dens-ctxt} \Rightarrow \text{expr} \Rightarrow (\text{state} \Rightarrow \text{val} \Rightarrow \text{ennreal}) \Rightarrow \text{bool}$

$((1 \vdash_d / (- \Rightarrow / -)) [50, 0, 50] 50)$ **where**

hd-AE: $\llbracket (V, V', \Gamma, \delta) \vdash_d e \Rightarrow f; \Gamma \vdash e : t;$

$\bigwedge \varrho. \varrho \in \text{space } (\text{state-measure } V' \Gamma) \implies$

$\text{AE } x \text{ in } \text{stock-measure } t. f \varrho x = f' \varrho x;$

$\text{case-prod } f' \in \text{borel-measurable } (\text{state-measure } V' \Gamma \otimes_M \text{stock-measure } t) \rrbracket$

$t \rrbracket$

$\implies (V, V', \Gamma, \delta) \vdash_d e \Rightarrow f'$

| *hd-dens-ctxt-cong*:

$(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies (\bigwedge \sigma. \sigma \in \text{space } (\text{state-measure } (V \cup V') \Gamma)$

$\implies \delta \sigma = \delta' \sigma)$

$\implies (V, V', \Gamma, \delta') \vdash_d e \Rightarrow f$

| *hd-val*: *countable-type* (*val-type* v) \implies

$(V, V', \Gamma, \delta) \vdash_d \text{Val } v \Rightarrow (\lambda \varrho x. \text{branch-prob } (V, V', \Gamma, \delta) \varrho * \text{indicator}$

$\{v\} x)$

$|$ *hd-var*: $x \in V \implies (V, V', \Gamma, \delta) \vdash_d \text{Var } x \Rightarrow \text{marg-dens } (V, V', \Gamma, \delta) x$
 $|$ *hd-let*: $\llbracket (\{\}, V \cup V', \Gamma, \lambda \cdot 1) \vdash_d e1 \Rightarrow f;$
 $(\text{shift-var-set } V, \text{Suc } V', \text{the}(\text{expr-type } \Gamma \ e1) \cdot \Gamma, \text{insert-dens } V \ V' \ f \ \delta) \vdash_d$
 $e2 \Rightarrow g \rrbracket$
 $\implies (V, V', \Gamma, \delta) \vdash_d \text{LetVar } e1 \ e2 \Rightarrow (\lambda \varrho. g \ (\text{case-nat } \text{undefined } \varrho))$
 $|$ *hd-rand*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies (V, V', \Gamma, \delta) \vdash_d \text{Random } \text{dst } e \Rightarrow \text{apply-dist-to-dens}$
 $\text{dst } f$
 $|$ *hd-rand-det*: $\text{randomfree } e \implies \text{free-vars } e \subseteq V' \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Random } \text{dst } e \Rightarrow$
 $(\lambda \varrho x. \text{branch-prob } (V, V', \Gamma, \delta) \ \varrho * \text{dist-dens } \text{dst } (\text{expr-sem-rf } \varrho \ e)$
 $x)$
 $|$ *hd-fail*: $(V, V', \Gamma, \delta) \vdash_d \text{Fail } t \Rightarrow (\lambda \cdot 0)$
 $|$ *hd-pair*: $x \in V \implies y \in V \implies x \neq y \implies (V, V', \Gamma, \delta) \vdash_d \langle \text{Var } x, \text{Var } y \rangle \Rightarrow$
 $\text{marg-dens2 } (V, V', \Gamma, \delta) \ x \ y$
 $|$ *hd-if*: $\llbracket (\{\}, V \cup V', \Gamma, \lambda \cdot 1) \vdash_d b \Rightarrow f;$
 $(V, V', \Gamma, \text{if-dens } \delta \ f \ \text{True}) \vdash_d e1 \Rightarrow g1; (V, V', \Gamma, \text{if-dens } \delta \ f \ \text{False}) \vdash_d e2$
 $\Rightarrow g2 \rrbracket$
 $\implies (V, V', \Gamma, \delta) \vdash_d \text{IF } b \ \text{THEN } e1 \ \text{ELSE } e2 \Rightarrow (\lambda \varrho x. g1 \ \varrho \ x + g2 \ \varrho \ x)$
 $|$ *hd-if-det*: $\llbracket \text{randomfree } b; (V, V', \Gamma, \text{if-dens-det } \delta \ b \ \text{True}) \vdash_d e1 \Rightarrow g1;$
 $(V, V', \Gamma, \text{if-dens-det } \delta \ b \ \text{False}) \vdash_d e2 \Rightarrow g2 \rrbracket$
 $\implies (V, V', \Gamma, \delta) \vdash_d \text{IF } b \ \text{THEN } e1 \ \text{ELSE } e2 \Rightarrow (\lambda \varrho x. g1 \ \varrho \ x + g2 \ \varrho \ x)$
 $|$ *hd-fst*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Fst } \$\$ \ e \Rightarrow$
 $(\lambda \varrho x. \int^+ y. f \ \varrho \ \langle |x, y| \rangle \ \partial \text{stock-measure } (\text{the } (\text{expr-type } \Gamma \ (\text{Snd } \$\$$
 $e))))$
 $|$ *hd-snd*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Snd } \$\$ \ e \Rightarrow$
 $(\lambda \varrho y. \int^+ x. f \ \varrho \ \langle |x, y| \rangle \ \partial \text{stock-measure } (\text{the } (\text{expr-type } \Gamma \ (\text{Fst } \$\$$
 $e))))$
 $|$ *hd-op-discr*: $\text{countable-type } (\text{the } (\text{expr-type } \Gamma \ (\text{oper } \$\$ \ e))) \implies (V, V', \Gamma, \delta) \vdash_d e$
 $\Rightarrow f \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{oper } \$\$ \ e \Rightarrow (\lambda \varrho y. \int^+ x. (\text{if } \text{op-sem } \text{oper } \ x = y$
 $\text{then } 1 \ \text{else } 0) * f \ \varrho \ x$
 $\partial \text{stock-measure } (\text{the } (\text{expr-type } \Gamma \ e)))$
 $|$ *hd-neg*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Minus } \$\$ \ e \Rightarrow (\lambda \sigma x. f \ \sigma \ (\text{op-sem } \text{Minus } \ x))$
 $|$ *hd-addc*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies \text{randomfree } e' \implies \text{free-vars } e' \subseteq V' \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Add } \$\$ \ \langle e, e' \rangle \Rightarrow$
 $(\lambda \varrho x. f \ \varrho \ (\text{op-sem } \text{Add } \ \langle |x, \text{expr-sem-rf } \varrho \ (\text{Minus } \$\$ \ e') | \rangle))$
 $|$ *hd-multc*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies \text{val-type } c = \text{REAL} \implies c \neq \text{RealVal } 0 \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Mult } \$\$ \ \langle e, \text{Val } c \rangle \Rightarrow$
 $(\lambda \varrho x. f \ \varrho \ (\text{op-sem } \text{Mult } \ \langle |x, \text{op-sem } \text{Inverse } \ c | \rangle) * \text{inverse } (\text{abs } (\text{extract-real } \ c)))$
 $|$ *hd-exp*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Exp } \$\$ \ e \Rightarrow$
 $(\lambda \sigma x. \text{if } \text{extract-real } \ x > 0 \ \text{then}$
 $f \ \sigma \ (\text{lift-RealVal } \ \text{safe-ln } \ x) * \text{inverse } (\text{extract-real } \ x) \ \text{else } 0)$
 $|$ *hd-inv*: $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f \implies$
 $(V, V', \Gamma, \delta) \vdash_d \text{Inverse } \$\$ \ e \Rightarrow (\lambda \sigma x. f \ \sigma \ (\text{op-sem } \text{Inverse } \ x) *$

inverse (extract-real x) ^ 2)

| *hd-add*: $(V, V', \Gamma, \delta) \vdash_a e \Rightarrow f \Longrightarrow$
 $(V, V', \Gamma, \delta) \vdash_a \text{Add } \$\$ e \Rightarrow (\lambda \sigma z. \int^{+x}. f \sigma <|x, \text{op-sem Add } <|z,$
op-sem Minus x|>|>
∂stock-measure (val-type z)

lemmas *expr-has-density-intros* =
hd-val hd-var hd-let hd-rand hd-rand-det hd-fail hd-pair hd-if hd-if-det
hd-fst hd-snd hd-op-discr hd-neg hd-addc hd-multc hd-exp hd-inv hd-add

7.2 Auxiliary lemmas

lemma *has-subprob-density-distr-Fst*:

fixes *t1 t2 f*
defines $N \equiv \text{stock-measure } (\text{PRODUCT } t1 \ t2)$
defines $N' \equiv \text{stock-measure } t1$
defines $\text{fst}' \equiv \text{op-sem } Fst$
defines $f' \equiv \lambda x. \int^{+y}. f <|x, y|> \partial \text{stock-measure } t2$
assumes *dens*: *has-subprob-density M N f*
shows *has-subprob-density (distr M N' fst') N' f'*
<proof>

lemma *has-subprob-density-distr-Snd*:

fixes *t1 t2 f*
defines $N \equiv \text{stock-measure } (\text{PRODUCT } t1 \ t2)$
defines $N' \equiv \text{stock-measure } t2$
defines $\text{snd}' \equiv \text{op-sem } Snd$
defines $f' \equiv \lambda y. \int^{+x}. f <|x, y|> \partial \text{stock-measure } t1$
assumes *dens*: *has-subprob-density M N f*
shows *has-subprob-density (distr M N' snd') N' f'*
<proof>

lemma *dens-ctxt-measure-empty-bind*:

assumes $\varrho \in \text{space } (\text{state-measure } V' \ \Gamma)$
assumes $f[\text{measurable}]$: $f \in \text{measurable } (\text{state-measure } V' \ \Gamma)$ (*subprob-algebra N*)
shows *dens-ctxt-measure* $(\{\}, V', \Gamma, \lambda \cdot. 1) \varrho \ggg f = f \varrho$ (**is bind ?M - = ?R**)
<proof>

lemma (**in density-context**) *bind-dens-ctxt-measure-cong*:

assumes fg : $\bigwedge \sigma. (\bigwedge x. x \in V' \Longrightarrow \sigma x = \varrho x) \Longrightarrow f \sigma = g \sigma$
assumes $\varrho[\text{measurable}]$: $\varrho \in \text{space } (\text{state-measure } V' \ \Gamma)$
assumes $Mf[\text{measurable}]$: $f \in \text{measurable } (\text{state-measure } (V \cup V') \ \Gamma)$ (*subprob-algebra N*)
assumes $Mg[\text{measurable}]$: $g \in \text{measurable } (\text{state-measure } (V \cup V') \ \Gamma)$ (*subprob-algebra N*)
defines $M \equiv \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \ \varrho$
shows $M \ggg f = M \ggg g$
<proof>

lemma (in *density-context*) *bin-op-randomfree-restructure*:
assumes $t1: \Gamma \vdash e : t$ **and** $t2: \Gamma \vdash e' : t'$ **and** $t3: \text{op-type oper (PRODUCT } t \text{ } t') = \text{Some tr}$
assumes $\text{rf: randomfree } e'$ **and** $\text{vars1: free-vars } e \subseteq V \cup V'$ **and** $\text{vars2: free-vars } e' \subseteq V'$
assumes $\varrho: \varrho \in \text{space (state-measure } V' \Gamma)$
defines $M \equiv \text{dens-ctxt-measure (V, V', \Gamma, \delta) } \varrho$
defines $v \equiv \text{expr-sem-rf } \varrho \text{ } e'$
shows $M \ggg (\lambda \sigma. \text{expr-sem } \sigma (\text{oper } \$\$ \langle e, e' \rangle)) =$
 $\text{distr (M } \ggg (\lambda \sigma. \text{expr-sem } \sigma \text{ } e)) (\text{stock-measure tr}) (\lambda w. \text{op-sem oper } \langle |w, v| \rangle)$
 $\langle \text{proof} \rangle$

lemma *addc-density-measurable*:
assumes $\text{Mf: case-prod } f \in \text{borel-measurable (state-measure } V' \Gamma \otimes_M \text{stock-measure } t)$
assumes $t\text{-disj: } t = \text{REAL} \vee t = \text{INTEG}$ **and** $t: \Gamma \vdash e' : t$
assumes $\text{rf: randomfree } e'$ **and** $\text{vars: free-vars } e' \subseteq V'$
defines $f' \equiv (\lambda \varrho \text{ } x. f \ \varrho \ (\text{op-sem Add } \langle |x, \text{expr-sem-rf } \varrho \text{ (Minus } \$\$ \text{ } e') | \rangle))$
shows $\text{case-prod } f' \in \text{borel-measurable (state-measure } V' \Gamma \otimes_M \text{stock-measure } t)$
 $\langle \text{proof} \rangle$

lemma (in *density-context*) *emeasure-bind-if-dens-ctxt-measure*:
assumes $\varrho: \varrho \in \text{space (state-measure } V' \Gamma)$
defines $M \equiv \text{dens-ctxt-measure } \mathcal{Y} \ \varrho$
assumes $\text{Mf[measurable]: } f \in \text{measurable } M \ (\text{subprob-algebra (stock-measure } \text{BOOL}))$
assumes $\text{Mg[measurable]: } g \in \text{measurable } M \ (\text{subprob-algebra } R)$
assumes $\text{Mh[measurable]: } h \in \text{measurable } M \ (\text{subprob-algebra } R)$
assumes $\text{densf: has-parametrized-subprob-density (state-measure (V } \cup \text{V') } \Gamma)$
 $f \ (\text{stock-measure } \text{BOOL}) \ \delta f$
assumes $\text{densg: has-parametrized-subprob-density (state-measure } V' \Gamma)$
 $(\lambda \varrho. \text{dens-ctxt-measure (V, V', \Gamma, } \lambda \sigma. \delta \ \sigma * \delta f \ \sigma \ (\text{BoolVal True}))$
 $\varrho \ggg g) \ R \ \delta g$
assumes $\text{densh: has-parametrized-subprob-density (state-measure } V' \Gamma)$
 $(\lambda \varrho. \text{dens-ctxt-measure (V, V', \Gamma, } \lambda \sigma. \delta \ \sigma * \delta f \ \sigma \ (\text{BoolVal False}))$
 $\varrho \ggg h) \ R \ \delta h$
defines $P \equiv \lambda b. b = \text{BoolVal True}$
shows $M \ggg (\lambda x. f \ x \ggg (\lambda b. \text{if } P \ b \ \text{then } g \ x \ \text{else } h \ x)) = \text{density } R \ (\lambda x. \delta g \ \varrho \ x + \delta h \ \varrho \ x)$
 $(\text{is ?lhs} = \text{?rhs})$
 $\langle \text{proof} \rangle$

lemma (in *density-context*) *emeasure-bind-if-det-dens-ctxt-measure*:
fixes f
assumes $\varrho: \varrho \in \text{space (state-measure } V' \Gamma)$
defines $M \equiv \text{dens-ctxt-measure } \mathcal{Y} \ \varrho$

defines $P \equiv \lambda b. f b = \text{BoolVal True}$ **and** $P' \equiv \lambda b. f b = \text{BoolVal False}$
assumes $dc1$: *density-context* $V V' \Gamma (\lambda \sigma. \delta \sigma * (\text{if } P \sigma \text{ then } 1 \text{ else } 0))$
assumes $dc2$: *density-context* $V V' \Gamma (\lambda \sigma. \delta \sigma * (\text{if } P' \sigma \text{ then } 1 \text{ else } 0))$
assumes $Mf[\text{measurable}]$: $f \in \text{measurable } M$ (*stock-measure* BOOL)
assumes $Mg[\text{measurable}]$: $g \in \text{measurable } M$ (*subprob-algebra* R)
assumes $Mh[\text{measurable}]$: $h \in \text{measurable } M$ (*subprob-algebra* R)
assumes $densg$: *has-parametrized-subprob-density* (*state-measure* $V' \Gamma$)
 $(\lambda \rho. \text{dens-ctxt-measure } (V, V', \Gamma, \lambda \sigma. \delta \sigma * (\text{if } P \sigma \text{ then } 1 \text{ else } 0))$
 $\rho \ggg g) R \delta g$
assumes $densh$: *has-parametrized-subprob-density* (*state-measure* $V' \Gamma$)
 $(\lambda \rho. \text{dens-ctxt-measure } (V, V', \Gamma, \lambda \sigma. \delta \sigma * (\text{if } P' \sigma \text{ then } 1 \text{ else } 0))$
 $\rho \ggg h) R \delta h$
shows $M \ggg (\lambda x. \text{if } P x \text{ then } g x \text{ else } h x) = \text{density } R (\lambda x. \delta g \rho x + \delta h \rho x)$
(is ?lhs = ?rhs)
 $\langle \text{proof} \rangle$

7.3 Soundness proof

lemma *restrict-state-measure[measurable]*:
 $(\lambda x. \text{restrict } x V') \in \text{measurable } (\text{state-measure } (V \cup V') \Gamma)$ (*state-measure* V'
 Γ)
 $\langle \text{proof} \rangle$

lemma *expr-has-density-sound-op*:
assumes $dens\text{-}ctxt$: *density-context* $V V' \Gamma \delta$
assumes $dens$: *has-parametrized-subprob-density* (*state-measure* $V' \Gamma$)
 $(\lambda \rho. \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \rho \ggg (\lambda \sigma. \text{expr-sem } \sigma e))$
(stock-measure t) f
assumes Mg : *case-prod* $g \in \text{borel-measurable } (\text{state-measure } V' \Gamma \otimes_M \text{stock-measure}$
 $t')$
assumes $dens'$: $\bigwedge M \rho. \text{has-subprob-density } M$ (*stock-measure* t) $(f \rho) \implies$
 $\text{has-density } (\text{distr } M (\text{stock-measure } t') (\text{op-sem } \text{oper}))$
 $(\text{stock-measure } t') (g \rho)$
assumes $t1$: $\Gamma \vdash e : t$ **and** $t2$: *op-type* $\text{oper } t = \text{Some } t'$
assumes free-vars : $\text{free-vars } (\text{oper } \$\$ e) \subseteq V \cup V'$
shows *has-parametrized-subprob-density* (*state-measure* $V' \Gamma$)
 $(\lambda \rho. \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \rho \ggg (\lambda \sigma. \text{expr-sem } \sigma (\text{oper } \$\$ e)))$
(stock-measure $t')$ g
 $\langle \text{proof} \rangle$

lemma *expr-has-density-sound-aux*:
assumes $(V, V', \Gamma, \delta) \vdash_d e \implies f \Gamma \vdash e : t$
 $\text{density-context } V V' \Gamma \delta \text{ free-vars } e \subseteq V \cup V'$
shows *has-parametrized-subprob-density* (*state-measure* $V' \Gamma$)
 $(\lambda \rho. \text{do } \{\sigma \leftarrow \text{dens-ctxt-measure } (V, V', \Gamma, \delta) \rho; \text{expr-sem } \sigma e\})$
(stock-measure t)
 $(\lambda \rho x. f \rho x)$
 $\langle \text{proof} \rangle$

lemma *hd-cong*:
assumes $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f$ *density-context* $V V' \Gamma \delta \Gamma \vdash e : t$ *free-vars* $e \subseteq V \cup V'$
assumes $\bigwedge \rho x. \rho \in \text{space}(\text{state-measure } V' \Gamma) \Longrightarrow x \in \text{space}(\text{stock-measure } t)$
 $\Longrightarrow f \rho x = f' \rho x$
shows $(V, V', \Gamma, \delta) \vdash_d e \Rightarrow f'$
 $\langle \text{proof} \rangle$

lemma *prob-space-empty-dens-ctxt[simp]*:
prob-space $(\text{dens-ctxt-measure } (\{\}, \{\}, \Gamma, (\lambda-. 1))) (\lambda-. \text{undefined})$
 $\langle \text{proof} \rangle$

lemma *branch-prob-empty-ctxt[simp]*: *branch-prob* $(\{\}, \{\}, \Gamma, (\lambda-. 1)) (\lambda-. \text{undefined})$
 $= 1$
 $\langle \text{proof} \rangle$

lemma *expr-has-density-sound*:
assumes $(\{\}, \{\}, \Gamma, (\lambda-. 1)) \vdash_d e \Rightarrow f$ $\Gamma \vdash e : t$ *free-vars* $e = \{\}$
shows *has-subprob-density* $(\text{expr-sem } \sigma e)$ $(\text{stock-measure } t)$ $(f (\lambda-. \text{undefined}))$
 $\langle \text{proof} \rangle$

end

8 Target Language Syntax and Semantics

theory *PDF-Target-Semantics*
imports *PDF-Semantics*
begin

datatype *cexpr* =
 $CVar$ *vname*
 $CVal$ *val*
 $CPair$ *cexpr cexpr* $(<-, ->_c [0, 0] 1000)$
 $COperator$ *pdf-operator cexpr* **(infixl** $\$_{\$}_c 999$ **)**
 CIf *cexpr cexpr cexpr* $(IF_c - THEN - ELSE - [0, 0, 10] 10)$
 $CIntegral$ *cexpr pdf-type* $(\int_c - \partial - [61] 110)$

abbreviation $(input)$ *cexpr-fun* $:: (cexpr \Rightarrow cexpr) \Rightarrow cexpr$ **(binder** $\lambda_c 10$ **)**
where

cexpr-fun $f \equiv f (CVar 0)$

abbreviation *cexpr-Add* **(infixl** $+_c 65$ **)** **where**

cexpr-Add $a b \equiv Add \$_{\$}_c <a, b>_c$

abbreviation *cexpr-Minus* $(-_c - [81] 80)$ **where**

cexpr-Minus $a \equiv Minus \$_{\$}_c a$

abbreviation *cexpr-Sub* **(infixl** $-_c 65$ **)** **where**

cexpr-Sub $a b \equiv a +_c -_c b$

abbreviation *cexpr-Mult* **(infixl** $*_c 70$ **)** **where**

cexpr-Mult $a b \equiv Mult \$_{\$}_c <a, b>_c$

abbreviation $inverse_c e \equiv Inverse \ \$\$_c e$
abbreviation $cepr-Div$ (**infixl** $'/_c$ 70) **where**
 $cepr-Div a b \equiv a *_c inverse_c b$
abbreviation $fact_c e \equiv Fact \ \$\$_c e$
abbreviation $sqrt_c e \equiv Sqrt \ \$\$_c e$
abbreviation $exp_c e \equiv Exp \ \$\$_c e$
abbreviation $ln_c e \equiv Ln \ \$\$_c e$
abbreviation $fst_c e \equiv Fst \ \$\$_c e$
abbreviation $snd_c e \equiv Snd \ \$\$_c e$
abbreviation $cepr-Pow$ (**infixl** $\hat{\ }_c$ 75) **where**
 $cepr-Pow a b \equiv Pow \ \$\$_c <a, b>_c$
abbreviation $cepr-And$ (**infixl** \wedge_c 35) **where**
 $cepr-And a b \equiv And \ \$\$_c <a, b>_c$
abbreviation $cepr-Or$ (**infixl** \vee_c 30) **where**
 $cepr-Or a b \equiv Or \ \$\$_c <a, b>_c$
abbreviation $cepr-Not$ (\neg_c - [40] 40) **where**
 $cepr-Not a \equiv Not \ \$\$_c a$
abbreviation $cepr-Equals$ (**infixl** $=_c$ 70) **where**
 $cepr-Equals a b \equiv Equals \ \$\$_c <a, b>_c$
abbreviation $cepr-Less$ (**infixl** $<_c$ 70) **where**
 $cepr-Less a b \equiv Less \ \$\$_c <a, b>_c$
abbreviation $cepr-LessEq$ (**infixl** \leq_c 70) **where**
 $cepr-LessEq a b \equiv a =_c b \vee_c a <_c b$
abbreviation $cepr-RealCast$ ($\langle _ \rangle_c$ [0] 90) **where**
 $cepr-RealCast a \equiv Cast \ REAL \ \$\$_c a$
abbreviation $CRReal$ **where**
 $CRReal x \equiv CVal (RealVal x)$
abbreviation $CInt$ **where**
 $CInt x \equiv CVal (IntVal x)$
abbreviation π_c **where**
 $\pi_c \equiv Pi \ \$\$_c (CVal UnitVal)$

instantiation $cepr :: expr$
begin

primrec $free-vars-cepr :: cepr \Rightarrow vname \ set$ **where**
 $free-vars-cepr (CVar x) = \{x\}$
 $| free-vars-cepr (CVal _) = \{\}$
 $| free-vars-cepr (oper \ \$\$_c e) = free-vars-cepr e$
 $| free-vars-cepr (<e1, e2>_c) = free-vars-cepr e1 \cup free-vars-cepr e2$
 $| free-vars-cepr (IF_c b THEN e1 ELSE e2) =$
 $free-vars-cepr b \cup free-vars-cepr e1 \cup free-vars-cepr e2$
 $| free-vars-cepr (\int_c e \partial t) = Suc \ -' free-vars-cepr e$

instance $\langle proof \rangle$
end

inductive $cepr-typing :: tyenv \Rightarrow cepr \Rightarrow pdf-type \Rightarrow bool$ ($(1-/_c \vdash_c / (- :/ -))$
 $[50,0,50] 50$) **where**

cet-val: $\Gamma \vdash_c CVal\ v : \text{val-type } v$
| *cet-var*: $\Gamma \vdash_c CVar\ x : \Gamma\ x$
| *cet-pair*: $\Gamma \vdash_c e1 : t1 \implies \Gamma \vdash_c e2 : t2 \implies \Gamma \vdash_c \langle e1, e2 \rangle_c : PRODUCT\ t1\ t2$
| *cet-op*: $\Gamma \vdash_c e : t \implies \text{op-type } oper\ t = Some\ t' \implies \Gamma \vdash_c oper\ \$\$_c\ e : t'$
| *cet-if*: $\Gamma \vdash_c b : BOOL \implies \Gamma \vdash_c e1 : t \implies \Gamma \vdash_c e2 : t$
 $\implies \Gamma \vdash_c IF_c\ b\ THEN\ e1\ ELSE\ e2 : t$
| *cet-int*: $t \cdot \Gamma \vdash_c e : REAL \implies \Gamma \vdash_c \int_c\ e\ \partial t : REAL$

lemma *cet-val'*: $t = \text{val-type } v \implies \Gamma \vdash_c CVal\ v : t$
<proof>

lemma *cet-var'*: $t = \Gamma\ x \implies \Gamma \vdash_c CVar\ x : t$
<proof>

lemma *cet-not*: $\Gamma \vdash_c e : BOOL \implies \Gamma \vdash_c \neg_c\ e : BOOL$
<proof>

lemma *cet-and*: $\Gamma \vdash_c e1 : BOOL \implies \Gamma \vdash_c e2 : BOOL \implies \Gamma \vdash_c e1 \wedge_c\ e2 : BOOL$
and
cet-or: $\Gamma \vdash_c e1 : BOOL \implies \Gamma \vdash_c e2 : BOOL \implies \Gamma \vdash_c e1 \vee_c\ e2 : BOOL$
<proof>

lemma *cet-minus-real*: $\Gamma \vdash_c e : REAL \implies \Gamma \vdash_c -_c\ e : REAL$ **and**
cet-inverse: $\Gamma \vdash_c e : REAL \implies \Gamma \vdash_c \text{inverse}_c\ e : REAL$ **and**
cet-sqrt: $\Gamma \vdash_c e : REAL \implies \Gamma \vdash_c \text{sqrt}_c\ e : REAL$ **and**
cet-exp: $\Gamma \vdash_c e : REAL \implies \Gamma \vdash_c \text{exp}_c\ e : REAL$ **and**
cet-ln: $\Gamma \vdash_c e : REAL \implies \Gamma \vdash_c \text{ln}_c\ e : REAL$
<proof>

lemma *cet-pow-real*: $\Gamma \vdash_c e1 : REAL \implies \Gamma \vdash_c e2 : INTEG \implies \Gamma \vdash_c e1 \hat{\wedge}_c\ e2 : REAL$
<proof>

lemma *cet-add-real*: $\Gamma \vdash_c e1 : REAL \implies \Gamma \vdash_c e2 : REAL \implies \Gamma \vdash_c e1 +_c\ e2 : REAL$ **and**
cet-mult-real: $\Gamma \vdash_c e1 : REAL \implies \Gamma \vdash_c e2 : REAL \implies \Gamma \vdash_c e1 *_c\ e2 : REAL$ **and**
cet-less-real: $\Gamma \vdash_c e1 : REAL \implies \Gamma \vdash_c e2 : REAL \implies \Gamma \vdash_c e1 <_c\ e2 : BOOL$
<proof>

lemma *cet-eq*: $\Gamma \vdash_c e1 : t \implies \Gamma \vdash_c e2 : t \implies \Gamma \vdash_c e1 =_c\ e2 : BOOL$
<proof>

lemma *cet-less-eq-real*: $\Gamma \vdash_c e1 : REAL \implies \Gamma \vdash_c e2 : REAL \implies \Gamma \vdash_c e1 \leq_c\ e2 : BOOL$
<proof>

lemma *cet-minus-int*: $\Gamma \vdash_c e : INTEG \implies \Gamma \vdash_c -_c\ e : INTEG$

<proof>

lemma *cet-add-int*: $\Gamma \vdash_c e1 : \text{INTEG} \implies \Gamma \vdash_c e2 : \text{INTEG} \implies \Gamma \vdash_c e1 +_c e2 : \text{INTEG}$ **and**

cet-mult-int: $\Gamma \vdash_c e1 : \text{INTEG} \implies \Gamma \vdash_c e2 : \text{INTEG} \implies \Gamma \vdash_c e1 *_c e2 : \text{INTEG}$ **and**

cet-less-int: $\Gamma \vdash_c e1 : \text{INTEG} \implies \Gamma \vdash_c e2 : \text{INTEG} \implies \Gamma \vdash_c e1 <_c e2 : \text{BOOL}$

<proof>

lemma *cet-less-eq-int*: $\Gamma \vdash_c e1 : \text{INTEG} \implies \Gamma \vdash_c e2 : \text{INTEG} \implies \Gamma \vdash_c e1 \leq_c e2 : \text{BOOL}$

<proof>

lemma *cet-sub-int*: $\Gamma \vdash_c e1 : \text{INTEG} \implies \Gamma \vdash_c e2 : \text{INTEG} \implies \Gamma \vdash_c e1 -_c e2 : \text{INTEG}$

<proof>

lemma *cet-fst*: $\Gamma \vdash_c e : \text{PRODUCT } t \ t' \implies \Gamma \vdash_c \text{fst}_c e : t$ **and**

cet-snd: $\Gamma \vdash_c e : \text{PRODUCT } t \ t' \implies \Gamma \vdash_c \text{snd}_c e : t'$

<proof>

lemma *cet-cast-real*: $\Gamma \vdash_c e : \text{BOOL} \implies \Gamma \vdash_c \langle e \rangle_c : \text{REAL}$

<proof>

lemma *cet-cast-real-int*: $\Gamma \vdash_c e : \text{INTEG} \implies \Gamma \vdash_c \langle e \rangle_c : \text{REAL}$

<proof>

lemma *cet-sub-real*: $\Gamma \vdash_c e1 : \text{REAL} \implies \Gamma \vdash_c e2 : \text{REAL} \implies \Gamma \vdash_c e1 -_c e2 : \text{REAL}$

<proof>

lemma *cet-pi*: $\Gamma \vdash_c \pi_c : \text{REAL}$

<proof>

lemmas *cet-op-intros* =

cet-minus-real cet-exp cet-sqrt cet-ln cet-inverse cet-pow-real cet-pi

cet-cast-real cet-add-real cet-mult-real cet-less-real

cet-not cet-and cet-or

inductive-cases *cexpr-typing-valE*[*elim*]: $\Gamma \vdash_c \text{CVal } v : t$

inductive-cases *cexpr-typing-varE*[*elim*]: $\Gamma \vdash_c \text{CVar } x : t$

inductive-cases *cexpr-typing-pairE*[*elim*]: $\Gamma \vdash_c \langle e1, e2 \rangle_c : t$

inductive-cases *cexpr-typing-opE*[*elim*]: $\Gamma \vdash_c \text{oper } \$\$_c e : t$

inductive-cases *cexpr-typing-ifE*[*elim*]: $\Gamma \vdash_c \text{IF}_c b \text{ THEN } e1 \text{ ELSE } e2 : t$

inductive-cases *cexpr-typing-intE*[*elim*]: $\Gamma \vdash_c \int_c e \ \partial t : t'$

primrec *cexpr-type* :: *tyenv* \Rightarrow *cexpr* \Rightarrow *pdf-type option* **where**

cexpr-type - (*CVal* *v*) = *Some (val-type v)*

$| \text{cexpr-type } \Gamma \text{ (CVar } x) = \text{Some } (\Gamma \ x)$
 $| \text{cexpr-type } \Gamma \text{ } \langle e1, e2 \rangle_c = \text{(case (cexpr-type } \Gamma \ e1, \text{ cexpr-type } \Gamma \ e2) \text{ of}$
 $\quad \text{(Some } t1, \text{ Some } t2) \Rightarrow \text{Some (PRODUCT } t1 \ t2)$
 $\quad | - \Rightarrow \text{None})$
 $| \text{cexpr-type } \Gamma \text{ (oper } \$\$_c \ e) = \text{(case cexpr-type } \Gamma \ e \text{ of}$
 $\quad \text{Some } t \Rightarrow \text{op-type oper } t$
 $\quad | - \Rightarrow \text{None})$
 $| \text{cexpr-type } \Gamma \text{ (IF}_c \ b \ \text{THEN } e1 \ \text{ELSE } e2) =$
 $\quad \text{(if cexpr-type } \Gamma \ b = \text{Some } \text{BOOL then}$
 $\quad \text{case (cexpr-type } \Gamma \ e1, \text{ cexpr-type } \Gamma \ e2) \text{ of}$
 $\quad \text{(Some } t, \text{ Some } t') \Rightarrow \text{if } t = t' \text{ then Some } t \text{ else None}$
 $\quad | - \Rightarrow \text{None}$
 $\quad \text{else None)}$
 $| \text{cexpr-type } \Gamma \text{ (}\int_c \ e \ \partial t) =$
 $\quad \text{(if cexpr-type (case-nat } t \ \Gamma) \ e = \text{Some } \text{REAL then Some } \text{REAL else None)}$

lemma *cexpr-type-Some-iff*: $\text{cexpr-type } \Gamma \ e = \text{Some } t \iff \Gamma \vdash_c \ e : t$
<proof>

lemmas *cexpr-typing-code*[code-unfold] = *cexpr-type-Some-iff*[symmetric]

lemma *cexpr-typing-cong'*:
assumes $\Gamma \vdash_c \ e : t \wedge x. x \in \text{free-vars } e \implies \Gamma \ x = \Gamma' \ x$
shows $\Gamma' \vdash_c \ e : t$
<proof>

lemma *cexpr-typing-cong*:
assumes $\wedge x. x \in \text{free-vars } e \implies \Gamma \ x = \Gamma' \ x$
shows $\Gamma \vdash_c \ e : t \iff \Gamma' \vdash_c \ e : t$
<proof>

primrec *cexpr-sem* :: $\text{state} \Rightarrow \text{cexpr} \Rightarrow \text{val}$ **where**
 $\text{cexpr-sem } \sigma \text{ (CVal } v) = v$
 $| \text{cexpr-sem } \sigma \text{ (CVar } x) = \sigma \ x$
 $| \text{cexpr-sem } \sigma \langle e1, e2 \rangle_c = \langle | \text{cexpr-sem } \sigma \ e1, \text{ cexpr-sem } \sigma \ e2 | \rangle$
 $| \text{cexpr-sem } \sigma \text{ (oper } \$\$_c \ e) = \text{op-sem oper (cexpr-sem } \sigma \ e)$
 $| \text{cexpr-sem } \sigma \text{ (IF}_c \ b \ \text{THEN } e1 \ \text{ELSE } e2) = \text{(if cexpr-sem } \sigma \ b = \text{TRUE then}$
 $\text{cexpr-sem } \sigma \ e1 \ \text{else cexpr-sem } \sigma \ e2)$
 $| \text{cexpr-sem } \sigma \text{ (}\int_c \ e \ \partial t) = \text{RealVal (}\int \ x. \text{extract-real (cexpr-sem (x } \cdot \ \sigma) \ e) \ \partial(\text{stock-measure } t))$

definition *cexpr-equiv* :: $\text{cexpr} \Rightarrow \text{cexpr} \Rightarrow \text{bool}$ **where**
 $\text{cexpr-equiv } e1 \ e2 \equiv \forall \sigma. \text{cexpr-sem } \sigma \ e1 = \text{cexpr-sem } \sigma \ e2$

lemma *cexpr-equiv-commute*: $\text{cexpr-equiv } e1 \ e2 \iff \text{cexpr-equiv } e2 \ e1$
<proof>

lemma *val-type-cexpr-sem[simp]*:

assumes $\Gamma \vdash_c e : t$ *free-vars* $e \subseteq V$ $\sigma \in \text{space}$ (*state-measure* $V \Gamma$)

shows *val-type* (*cexpr-sem* σe) = t

<proof>

lemma *cexpr-sem-eq-on-vars*:

assumes $\bigwedge x. x \in \text{free-vars } e \implies \sigma x = \sigma' x$

shows *cexpr-sem* $\sigma e = \text{iexpr-sem } \sigma' e$

<proof>

definition *eval-cexpr* :: *cexpr* \Rightarrow *state* \Rightarrow *val* \Rightarrow *real* **where**

eval-cexpr $e \sigma v = \text{extract-real} (\text{iexpr-sem} (\text{case-nat } v \sigma) e)$

lemma *measurable-cexpr-sem[measurable]*:

$\Gamma \vdash_c e : t \implies \text{free-vars } e \subseteq V \implies$

$(\lambda \sigma. \text{iexpr-sem } \sigma e) \in \text{measurable} (\text{state-measure } V \Gamma) (\text{stock-measure } t)$

<proof>

lemma *measurable-eval-cexpr[measurable]*:

assumes *case-nat* $t \Gamma \vdash_c e : \text{REAL}$

assumes *free-vars* $e \subseteq \text{shift-var-set } V$

shows *case-prod* (*eval-cexpr* e) $\in \text{borel-measurable} (\text{state-measure } V \Gamma \otimes_M \text{stock-measure } t)$

<proof>

lemma *cexpr-sem-Add*:

assumes $\Gamma \vdash_c e1 : \text{REAL}$ $\Gamma \vdash_c e2 : \text{REAL}$

assumes $\sigma \in \text{space} (\text{state-measure } V \Gamma)$ *free-vars* $e1 \subseteq V$ *free-vars* $e2 \subseteq V$

shows $\text{extract-real} (\text{iexpr-sem } \sigma (e1 +_c e2)) = \text{extract-real} (\text{iexpr-sem } \sigma e1) + \text{extract-real} (\text{iexpr-sem } \sigma e2)$

<proof>

lemma *cexpr-sem-Mult*:

assumes $\Gamma \vdash_c e1 : \text{REAL}$ $\Gamma \vdash_c e2 : \text{REAL}$

assumes $\sigma \in \text{space} (\text{state-measure } V \Gamma)$ *free-vars* $e1 \subseteq V$ *free-vars* $e2 \subseteq V$

shows $\text{extract-real} (\text{iexpr-sem } \sigma (e1 *_c e2)) = \text{extract-real} (\text{iexpr-sem } \sigma e1) * \text{extract-real} (\text{iexpr-sem } \sigma e2)$

<proof>

8.1 General functions on Expressions

Transform variable names in an expression.

primrec *map-vars* :: (*vname* \Rightarrow *vname*) \Rightarrow *cexpr* \Rightarrow *cexpr* **where**

map-vars f (*CVal* v) = *CVal* v

| *map-vars* f (*CVar* x) = *CVar* ($f x$)

| *map-vars* f ($\langle e1, e2 \rangle_c$) = $\langle \text{map-vars } f e1, \text{map-vars } f e2 \rangle_c$

| *map-vars* f (*oper* $\$_c e$) = *oper* $\$_c (\text{map-vars } f e)$

| *map-vars* f (*IF* b *THEN* $e1$ *ELSE* $e2$) = (*IF* $\text{map-vars } f b$ *THEN* $\text{map-vars } f e1$ *ELSE* $\text{map-vars } f e2$)

$e1$ *ELSE* $\text{map-vars } f \ e2$)
 $\mid \text{map-vars } f \ (\int_c e \ \partial t) = \int_c \text{map-vars } (\text{case-nat } 0 \ (\lambda x. \text{Suc } (f \ x))) \ e \ \partial t$

lemma *free-vars-map-vars[simp]*:
 $\text{free-vars } (\text{map-vars } f \ e) = f \ ' \ \text{free-vars } e$
 $\langle \text{proof} \rangle$

lemma *cepr-typing-map-vars*:
 $\Gamma \circ f \vdash_c e : t \implies \Gamma \vdash_c \text{map-vars } f \ e : t$
 $\langle \text{proof} \rangle$

lemma *cepr-sem-map-vars*:
 $\text{cepr-sem } \sigma \ (\text{map-vars } f \ e) = \text{cepr-sem } (\sigma \circ f) \ e$
 $\langle \text{proof} \rangle$

definition *insert-var* :: $vname \Rightarrow (vname \Rightarrow 'a) \Rightarrow 'a \Rightarrow vname \Rightarrow 'a$ **where**
 $\text{insert-var } v \ f \ x \ w \equiv \text{if } w = v \ \text{then } x \ \text{else if } w > v \ \text{then } f \ (w - 1) \ \text{else } f \ w$

lemma *insert-var-0[simp]*: $\text{insert-var } 0 \ f \ x = \text{case-nat } x \ f$
 $\langle \text{proof} \rangle$

Substitutes expression e for variable x in e' .

primrec *cepr-subst* :: $vname \Rightarrow \text{cepr} \Rightarrow \text{cepr} \Rightarrow \text{cepr}$ **where**
 $\text{cepr-subst } - \ (CVal \ v) = CVal \ v$
 $\mid \text{cepr-subst } x \ e \ (CVar \ y) = \text{insert-var } x \ CVar \ e \ y$
 $\mid \text{cepr-subst } x \ e \ \langle e1, e2 \rangle_c = \langle \text{cepr-subst } x \ e \ e1, \text{cepr-subst } x \ e \ e2 \rangle_c$
 $\mid \text{cepr-subst } x \ e \ (\text{oper } \$_\$_c \ e') = \text{oper } \$_\$_c \ (\text{cepr-subst } x \ e \ e')$
 $\mid \text{cepr-subst } x \ e \ (\text{IF}_c \ b \ \text{THEN } e1 \ \text{ELSE } e2) =$
 $\quad (\text{IF}_c \ \text{cepr-subst } x \ e \ b \ \text{THEN } \text{cepr-subst } x \ e \ e1 \ \text{ELSE } \text{cepr-subst } x \ e \ e2)$
 $\mid \text{cepr-subst } x \ e \ (\int_c e' \ \partial t) = (\int_c \text{cepr-subst } (\text{Suc } x) \ (\text{map-vars } \text{Suc } e) \ e' \ \partial t)$

lemma *cepr-sem-cepr-subst-aux*:
 $\text{cepr-sem } \sigma \ (\text{cepr-subst } x \ e \ e') = \text{cepr-sem } (\text{insert-var } x \ \sigma \ (\text{cepr-sem } \sigma \ e))$
 e'
 $\langle \text{proof} \rangle$

This corresponds to a Let-binding; the variable with index 0 is substituted with the given expression.

lemma *cepr-sem-cepr-subst*:
 $\text{cepr-sem } \sigma \ (\text{cepr-subst } 0 \ e \ e') = \text{cepr-sem } (\text{case-nat } (\text{cepr-sem } \sigma \ e) \ \sigma) \ e'$
 $\langle \text{proof} \rangle$

lemma *cepr-typing-subst-aux*:
assumes $\text{insert-var } x \ \Gamma \ t \vdash_c e' : t' \ \Gamma \vdash_c e : t$
shows $\Gamma \vdash_c \text{cepr-subst } x \ e \ e' : t'$
 $\langle \text{proof} \rangle$

lemma *cepr-typing-subst[intro]*:
assumes $\Gamma \vdash_c e : t \ \text{case-nat } t \ \Gamma \vdash_c e' : t'$

shows $\Gamma \vdash_c \text{cexpr-subst } 0 \ e' : t'$
 ⟨proof⟩

lemma *free-vars-cexpr-subst-aux*:

$\text{free-vars } (\text{cexpr-subst } x \ e') \subseteq (\lambda y. \text{if } y \geq x \text{ then } y + 1 \text{ else } y) - ' \text{free-vars } e' \cup \text{free-vars } e$
 (is $\text{free-vars } - \subseteq ?f \ x \ - ' - \cup -$)
 ⟨proof⟩

lemma *free-vars-cexpr-subst*:

$\text{free-vars } (\text{cexpr-subst } 0 \ e') \subseteq \text{Suc } - ' \text{free-vars } e' \cup \text{free-vars } e$
 ⟨proof⟩

primrec *cexpr-comp-aux* :: $vname \Rightarrow \text{cexpr} \Rightarrow \text{cexpr} \Rightarrow \text{cexpr}$ **where**

$\text{cexpr-comp-aux } - \ (CVal \ v) = CVal \ v$
 | $\text{cexpr-comp-aux } x \ e \ (CVar \ y) = (\text{if } x = y \ \text{then } e \ \text{else } CVar \ y)$
 | $\text{cexpr-comp-aux } x \ e \ \langle e1, \ e2 \rangle_c = \langle \text{cexpr-comp-aux } x \ e \ e1, \ \text{cexpr-comp-aux } x \ e \ e2 \rangle_c$
 | $\text{cexpr-comp-aux } x \ e \ (\text{oper } \$\$_c \ e') = \text{oper } \$\$_c \ (\text{cexpr-comp-aux } x \ e \ e')$
 | $\text{cexpr-comp-aux } x \ e \ (\text{IF}_c \ b \ \text{THEN } e1 \ \text{ELSE } e2) =$
 $(\text{IF}_c \ \text{cexpr-comp-aux } x \ e \ b \ \text{THEN } \text{cexpr-comp-aux } x \ e \ e1 \ \text{ELSE } \text{cexpr-comp-aux } x \ e \ e2)$
 | $\text{cexpr-comp-aux } x \ e \ (\int_c \ e' \ \partial t) = (\int_c \ \text{cexpr-comp-aux } (\text{Suc } x) \ (\text{map-vars } \text{Suc } e) \ e' \ \partial t)$

lemma *cexpr-sem-cexpr-comp-aux*:

$\text{cexpr-sem } \sigma \ (\text{cexpr-comp-aux } x \ e \ e') = \text{cexpr-sem } (\sigma(x := \text{cexpr-sem } \sigma \ e)) \ e'$
 ⟨proof⟩

definition *cexpr-comp* (infixl \circ_c 55) **where**

$\text{cexpr-comp } b \ a \equiv \text{cexpr-comp-aux } 0 \ a \ b$

lemma *cexpr-typing-cexpr-comp-aux*:

assumes $\Gamma(x := t1) \vdash_c \ e' : t2 \ \Gamma \vdash_c \ e : t1$
shows $\Gamma \vdash_c \ \text{cexpr-comp-aux } x \ e \ e' : t2$
 ⟨proof⟩

lemma *cexpr-typing-cexpr-comp[intro]*:

assumes $\text{case-nat } t1 \ \Gamma \vdash_c \ g : t2$
assumes $\text{case-nat } t2 \ \Gamma \vdash_c \ f : t3$
shows $\text{case-nat } t1 \ \Gamma \vdash_c \ f \circ_c \ g : t3$
 ⟨proof⟩

lemma *free-vars-cexpr-comp-aux*:

$\text{free-vars } (\text{cexpr-comp-aux } x \ e \ e') \subseteq (\text{free-vars } e' - \{x\}) \cup \text{free-vars } e$

$\langle \text{proof} \rangle$

lemma *free-vars-cexpr-comp*:

$$\text{free-vars } (\text{cexpr-comp } e \ e') \subseteq (\text{free-vars } e - \{0\}) \cup \text{free-vars } e'$$

$\langle \text{proof} \rangle$

lemma *free-vars-cexpr-comp'*:

$$\text{free-vars } (\text{cexpr-comp } e \ e') \subseteq \text{free-vars } e \cup \text{free-vars } e'$$

$\langle \text{proof} \rangle$

lemma *cexpr-sem-cexpr-comp*:

$$\text{cexpr-sem } \sigma \ (f \circ_c g) = \text{cexpr-sem } (\sigma(0 := \text{cexpr-sem } \sigma \ g)) \ f$$

$\langle \text{proof} \rangle$

lemma *eval-cexpr-comp*:

$$\text{eval-cexpr } (f \circ_c g) \ \sigma \ x = \text{eval-cexpr } f \ \sigma \ (\text{cexpr-sem } (\text{case-nat } x \ \sigma) \ g)$$

$\langle \text{proof} \rangle$

primrec *cexpr-subst-val-aux* :: $\text{nat} \Rightarrow \text{cexpr} \Rightarrow \text{val} \Rightarrow \text{cexpr}$ **where**

$$\begin{aligned} & \text{cexpr-subst-val-aux } - \ (CVal \ v) \ - = CVal \ v \\ & | \text{cexpr-subst-val-aux } x \ (CVar \ y) \ v = \text{insert-var } x \ CVar \ (CVal \ v) \ y \\ & | \text{cexpr-subst-val-aux } x \ (IF_c \ b \ THEN \ e1 \ ELSE \ e2) \ v = \\ & \quad (IF_c \ \text{cexpr-subst-val-aux } x \ b \ v \ THEN \ \text{cexpr-subst-val-aux } x \ e1 \ v \ ELSE \ \text{cexpr-subst-val-aux} \\ & \quad x \ e2 \ v) \\ & | \text{cexpr-subst-val-aux } x \ (\text{oper } \$_\$_c \ e) \ v = \text{oper } \$_\$_c \ (\text{cexpr-subst-val-aux } x \ e \ v) \\ & | \text{cexpr-subst-val-aux } x \ \langle e1, \ e2 \rangle_c \ v = \langle \text{cexpr-subst-val-aux } x \ e1 \ v, \ \text{cexpr-subst-val-aux} \\ & \quad x \ e2 \ v \rangle_c \\ & | \text{cexpr-subst-val-aux } x \ (\int_c \ e \ \partial t) \ v = \int_c \ \text{cexpr-subst-val-aux } (Suc \ x) \ e \ v \ \partial t \end{aligned}$$

lemma *cexpr-subst-val-aux-eq-cexpr-subst*:

$$\text{cexpr-subst-val-aux } x \ e \ v = \text{cexpr-subst } x \ (CVal \ v) \ e$$

$\langle \text{proof} \rangle$

definition *cexpr-subst-val* :: $\text{cexpr} \Rightarrow \text{val} \Rightarrow \text{cexpr}$ **where**

$$\text{cexpr-subst-val } e \ v \equiv \text{cexpr-subst-val-aux } 0 \ e \ v$$

lemma *cexpr-sem-cexpr-subst-val[simp]*:

$$\text{cexpr-sem } \sigma \ (\text{cexpr-subst-val } e \ v) = \text{cexpr-sem } (\text{case-nat } v \ \sigma) \ e$$

$\langle \text{proof} \rangle$

lemma *cexpr-typing-subst-val[intro]*:

$$\text{case-nat } t \ \Gamma \vdash_c \ e : t' \Longrightarrow \text{val-type } v = t \Longrightarrow \Gamma \vdash_c \ \text{cexpr-subst-val } e \ v : t'$$

$\langle \text{proof} \rangle$

lemma *free-vars-cexpr-subst-val-aux*:

$$\text{free-vars } (\text{cexpr-subst-val-aux } x \ e \ v) = (\lambda y. \text{if } y \geq x \ \text{then } Suc \ y \ \text{else } y) \ -' \\ \text{free-vars } e$$

$\langle \text{proof} \rangle$

lemma *free-vars-cexpr-subst-val[simp]*:

$free\text{-vars } (cexpr\text{-subst-}val\ e\ v) = Suc - ' free\text{-vars } e$
 ⟨proof⟩

8.2 Nonnegative expressions

definition *nonneg-cexpr* $V\ \Gamma\ e \equiv$

$\forall \sigma \in space\ (state\text{-measure } V\ \Gamma). extract\text{-real } (cexpr\text{-sem } \sigma\ e) \geq 0$

lemma *nonneg-cexprI*:

$(\bigwedge \sigma. \sigma \in space\ (state\text{-measure } V\ \Gamma) \implies extract\text{-real } (cexpr\text{-sem } \sigma\ e) \geq 0) \implies$
nonneg-cexpr $V\ \Gamma\ e$
 ⟨proof⟩

lemma *nonneg-cexprD*:

nonneg-cexpr $V\ \Gamma\ e \implies \sigma \in space\ (state\text{-measure } V\ \Gamma) \implies extract\text{-real}$
 $(cexpr\text{-sem } \sigma\ e) \geq 0$
 ⟨proof⟩

lemma *nonneg-cexpr-map-vars*:

assumes *nonneg-cexpr* $(f - ' V)\ (\Gamma \circ f)\ e$
shows *nonneg-cexpr* $V\ \Gamma\ (map\text{-vars } f\ e)$
 ⟨proof⟩

lemma *nonneg-cexpr-subset*:

assumes *nonneg-cexpr* $V\ \Gamma\ e\ V \subseteq V'\ free\text{-vars } e \subseteq V$
shows *nonneg-cexpr* $V'\ \Gamma\ e$
 ⟨proof⟩

lemma *nonneg-cexpr-Mult*:

assumes $\Gamma \vdash_c e1 : REAL\ \Gamma \vdash_c e2 : REAL$
assumes $free\text{-vars } e1 \subseteq V\ free\text{-vars } e2 \subseteq V$
assumes $N1: nonneg\text{-cexpr } V\ \Gamma\ e1$ **and** $N2: nonneg\text{-cexpr } V\ \Gamma\ e2$
shows *nonneg-cexpr* $V\ \Gamma\ (e1 *_c e2)$
 ⟨proof⟩

lemma *nonneg-indicator*:

assumes $\Gamma \vdash_c e : BOOL\ free\text{-vars } e \subseteq V$
shows *nonneg-cexpr* $V\ \Gamma\ (\langle e \rangle_c)$
 ⟨proof⟩

lemma *nonneg-cexpr-comp-aux*:

assumes *nonneg: nonneg-cexpr* $V\ (\Gamma(x := t1))\ e$ **and** $x:x \in V$
assumes $t2: \Gamma(x:=t1) \vdash_c e : t2$ **and** $t1: \Gamma \vdash_c f : t1$ **and** $vars: free\text{-vars } f \subseteq V$
shows *nonneg-cexpr* $V\ \Gamma\ (cexpr\text{-comp-aux } x\ f\ e)$
 ⟨proof⟩

lemma *nonneg-cexpr-comp*:

assumes *nonneg-cexpr* $(shift\text{-var-set } V)\ (case\text{-nat } t2\ \Gamma)\ e$

assumes $\text{case-nat } t1 \ \Gamma \vdash_c f : t2 \ \text{free-vars } f \subseteq \text{shift-var-set } V$
shows $\text{nonneg-cexpr } (\text{shift-var-set } V) \ (\text{case-nat } t1 \ \Gamma) \ (e \circ_c f)$
 $\langle \text{proof} \rangle$

lemma $\text{nonneg-cexpr-subst-val}$:

assumes $\text{nonneg-cexpr } (\text{shift-var-set } V) \ (\text{case-nat } t \ \Gamma) \ e \ \text{val-type } v = t$
shows $\text{nonneg-cexpr } V \ \Gamma \ (\text{cexpr-subst-val } e \ v)$
 $\langle \text{proof} \rangle$

lemma nonneg-cexpr-int :

assumes $\text{nonneg-cexpr } (\text{shift-var-set } V) \ (\text{case-nat } t \ \Gamma) \ e$
shows $\text{nonneg-cexpr } V \ \Gamma \ (\int_c e \ \partial t)$
 $\langle \text{proof} \rangle$

Subprobability density expressions

definition $\text{subprob-cexpr } V \ V' \ \Gamma \ e \equiv$

$\forall \varrho \in \text{space } (\text{state-measure } V' \ \Gamma).$
 $(\int^+ \sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } V \ V' \ (\sigma, \varrho)) \ e) \ \partial \text{state-measure } V \ \Gamma) \leq$
 1

lemma subprob-cexprI :

assumes $\bigwedge \varrho. \varrho \in \text{space } (\text{state-measure } V' \ \Gamma) \implies$
 $(\int^+ \sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } V \ V' \ (\sigma, \varrho)) \ e) \ \partial \text{state-measure}$
 $V \ \Gamma) \leq 1$
shows $\text{subprob-cexpr } V \ V' \ \Gamma \ e$
 $\langle \text{proof} \rangle$

lemma subprob-cexprD :

assumes $\text{subprob-cexpr } V \ V' \ \Gamma \ e$
shows $\bigwedge \varrho. \varrho \in \text{space } (\text{state-measure } V' \ \Gamma) \implies$
 $(\int^+ \sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } V \ V' \ (\sigma, \varrho)) \ e) \ \partial \text{state-measure}$
 $V \ \Gamma) \leq 1$
 $\langle \text{proof} \rangle$

lemma subprob-indicator :

assumes $\text{subprob: subprob-cexpr } V \ V' \ \Gamma \ e1$ **and** $\text{nonneg: nonneg-cexpr } (V \cup V')$
 $\Gamma \ e1$
assumes $t1: \Gamma \vdash_c e1 : \text{REAL}$ **and** $t2: \Gamma \vdash_c e2 : \text{BOOL}$
assumes $\text{vars1: free-vars } e1 \subseteq V \cup V'$ **and** $\text{vars2: free-vars } e2 \subseteq V \cup V'$
shows $\text{subprob-cexpr } V \ V' \ \Gamma \ (e1 *_c \langle e2 \rangle_c)$
 $\langle \text{proof} \rangle$

lemma $\text{measurable-cexpr-sem}'$:

assumes $\varrho: \varrho \in \text{space } (\text{state-measure } V' \ \Gamma)$
assumes $e: \Gamma \vdash_c e : \text{REAL}$ $\text{free-vars } e \subseteq V \cup V'$
shows $(\lambda \sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } V \ V' \ (\sigma, \varrho)) \ e))$
 $\in \text{borel-measurable } (\text{state-measure } V \ \Gamma)$
 $\langle \text{proof} \rangle$

lemma *measurable-fun-upd-state-measure*[*measurable*]:
assumes $v \notin V$
shows $(\lambda(x,y). y(v := x)) \in \text{measurable} (\text{stock-measure } (\Gamma v) \otimes_M \text{state-measure } V \Gamma)$
(*state-measure (insert v V) Γ*)
<proof>

lemma *integrable-cexpr-projection*:
assumes *fin*: *finite V*
assumes *disjoint*: $V \cap V' = \{\}$ $v \notin V$ $v \notin V'$
assumes $\varrho: \varrho \in \text{space} (\text{state-measure } V' \Gamma)$
assumes $e: \Gamma \vdash_c e : \text{REAL}$ *free-vars* $e \subseteq \text{insert } v V \cup V'$
assumes *int*: *integrable (state-measure (insert v V) Γ)*
($\lambda\sigma. \text{extract-real} (\text{cexpr-sem} (\text{merge} (\text{insert } v V) V' (\sigma, \varrho)) e)$)
(*is integrable - ?f'*)
shows *AE x in stock-measure (Γ v).*
(*integrable (state-measure V Γ)*)
($\lambda\sigma. \text{extract-real} (\text{cexpr-sem} (\text{merge } V (\text{insert } v V') (\sigma, \varrho(v := x))) e)$)
(*is AE x in ?N. integrable ?M (?f x)*)
<proof>

definition *cdens-ctxt-invar* :: *vname list* \Rightarrow *vname list* \Rightarrow *tyenv* \Rightarrow *cexpr* \Rightarrow *bool*
where
 $\text{cdens-ctxt-invar } vs \text{ vs}' \Gamma \delta \equiv$
distinct (vs @ vs') \wedge
free-vars $\delta \subseteq \text{set } (vs @ vs')$ \wedge
 $\Gamma \vdash_c \delta : \text{REAL}$ \wedge
nonneg-cexpr (set vs \cup set vs') $\Gamma \delta$ \wedge
subprob-cexpr (set vs) (set vs') $\Gamma \delta$

lemma *cdens-ctxt-invarI*:
 $\llbracket \text{distinct } (vs @ vs'); \text{free-vars } \delta \subseteq \text{set } (vs @ vs'); \Gamma \vdash_c \delta : \text{REAL};$
nonneg-cexpr (set vs \cup set vs') $\Gamma \delta;$
subprob-cexpr (set vs) (set vs') $\Gamma \delta \rrbracket \implies$
cdens-ctxt-invar vs vs' Γ δ
<proof>

lemma *cdens-ctxt-invarD*:
assumes *cdens-ctxt-invar vs vs' Γ δ*
shows *distinct (vs @ vs')* *free-vars* $\delta \subseteq \text{set } (vs @ vs')$ $\Gamma \vdash_c \delta : \text{REAL}$
nonneg-cexpr (set vs \cup set vs') $\Gamma \delta$ *subprob-cexpr (set vs) (set vs')* $\Gamma \delta$
<proof>

lemma *cdens-ctxt-invar-empty*:
assumes *cdens-ctxt-invar vs vs' Γ δ*
shows *cdens-ctxt-invar* $\llbracket (vs @ vs') \Gamma (\text{CReal } 1)$
<proof>

lemma *cdens-ctxt-invar-imp-integrable*:

assumes *cdens-ctxt-invar vs vs' Γ δ* **and** ϱ : $\varrho \in \text{space } (\text{state-measure } (\text{set } vs') \Gamma)$
shows *integrable (state-measure (set vs) Γ)*
 $(\lambda\sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } (\text{set } vs) (\text{set } vs') (\sigma, \varrho)) \delta))$ **(is**
integrable ?M ?f)
 $\langle \text{proof} \rangle$

8.3 Randomfree expressions

Translates an expression with no occurrences of Random or Fail into an equivalent target language expression.

primrec *expr-rf-to-cexpr* :: *expr* \Rightarrow *cexpr* **where**

expr-rf-to-cexpr (Val v) = CVal v
 $| \text{expr-rf-to-cexpr (Var } x) = \text{CVar } x$
 $| \text{expr-rf-to-cexpr } \langle e1, e2 \rangle = \langle \text{expr-rf-to-cexpr } e1, \text{expr-rf-to-cexpr } e2 \rangle_c$
 $| \text{expr-rf-to-cexpr (oper } \$\$ e) = \text{oper } \$\$_c (\text{expr-rf-to-cexpr } e)$
 $| \text{expr-rf-to-cexpr (IF } b \text{ THEN } e1 \text{ ELSE } e2) =$
 $(\text{IF}_c \text{expr-rf-to-cexpr } b \text{ THEN expr-rf-to-cexpr } e1 \text{ ELSE expr-rf-to-cexpr } e2)$
 $| \text{expr-rf-to-cexpr (LET } e1 \text{ IN } e2) =$
 $\text{cexpr-subst } 0 (\text{expr-rf-to-cexpr } e1) (\text{expr-rf-to-cexpr } e2)$
 $| \text{expr-rf-to-cexpr (Random -)} = \text{undefined}$
 $| \text{expr-rf-to-cexpr (Fail -)} = \text{undefined}$

lemma *cexpr-sem-expr-rf-to-cexpr*:

randomfree e \Longrightarrow *cexpr-sem σ (expr-rf-to-cexpr e) = expr-sem-rf σ e*
 $\langle \text{proof} \rangle$

lemma *cexpr-typing-expr-rf-to-cexpr[intro]*:

assumes $\Gamma \vdash e : t$ *randomfree e*
shows $\Gamma \vdash_c \text{expr-rf-to-cexpr } e : t$
 $\langle \text{proof} \rangle$

lemma *free-vars-expr-rf-to-cexpr*:

randomfree e \Longrightarrow *free-vars (expr-rf-to-cexpr e) \subseteq free-vars e*
 $\langle \text{proof} \rangle$

8.4 Builtin density expressions

primrec *dist-dens-cexpr* :: *pdf-dist* \Rightarrow *cexpr* \Rightarrow *cexpr* \Rightarrow *cexpr* **where**

dist-dens-cexpr Bernoulli p x = (IF_c CReal 0 \leq_c p \wedge_c p \leq_c CReal 1 THEN
 $\text{IF}_c x \text{ THEN } p \text{ ELSE CReal } 1 -_c p$
 $\text{ELSE CReal } 0)$
 $| \text{dist-dens-cexpr UniformInt } p x = (\text{IF}_c \text{fst}_c p \leq_c \text{snd}_c p \wedge_c \text{fst}_c p \leq_c x \wedge_c x \leq_c$
 $\text{snd}_c p \text{ THEN}$
 $\text{inverse}_c (\langle \text{snd}_c p -_c \text{fst}_c p +_c \text{CInt } 1 \rangle_c) \text{ ELSE}$
 $\text{CReal } 0)$
 $| \text{dist-dens-cexpr UniformReal } p x = (\text{IF}_c \text{fst}_c p <_c \text{snd}_c p \wedge_c \text{fst}_c p \leq_c x \wedge_c x \leq_c$
 $\text{snd}_c p \text{ THEN}$

$$\text{inverse}_c (\text{snd}_c p -_c \text{fst}_c p) \text{ ELSE } \text{CReal } 0)$$

$$| \text{dist-dens-cexpr Gaussian } p \ x = (\text{IF}_c \ \text{CReal } 0 <_c \text{snd}_c p \ \text{THEN}$$

$$\text{exp}_c (-_c((x -_c \text{fst}_c p) \widehat{c} \text{CInt } 2 /_c (\text{CReal } 2 *_c \text{snd}_c$$

$$p \widehat{c} \text{CInt } 2))) /_c$$

$$\text{sqr}_c (\text{CReal } 2 *_c \pi_c *_c \text{snd}_c p \widehat{c} \text{CInt } 2) \ \text{ELSE}$$

$$\text{CReal } 0)$$

$$| \text{dist-dens-cexpr Poisson } p \ x = (\text{IF}_c \ \text{CReal } 0 <_c p \wedge_c \text{CInt } 0 \leq_c x \ \text{THEN}$$

$$p \widehat{c} x /_c \langle \text{fact}_c x \rangle_c *_c \text{exp}_c (-_c p) \ \text{ELSE } \text{CReal } 0)$$

lemma *free-vars-dist-dens-cexpr*:

$$\text{free-vars } (\text{dist-dens-cexpr } \text{dst } e1 \ e2) \subseteq \text{free-vars } e1 \cup \text{free-vars } e2$$

$$\langle \text{proof} \rangle$$

lemma *cexpr-typing-dist-dens-cexpr*:

assumes $\Gamma \vdash_c e1 : \text{dist-param-type } \text{dst}$ $\Gamma \vdash_c e2 : \text{dist-result-type } \text{dst}$
shows $\Gamma \vdash_c \text{dist-dens-cexpr } \text{dst } e1 \ e2 : \text{REAL}$

$$\langle \text{proof} \rangle$$

lemma *val-type-eq-BOOL*: $\text{val-type } x = \text{BOOL} \longleftrightarrow x \in \text{BoolVal}'\text{UNIV}$

$$\langle \text{proof} \rangle$$

lemma *val-type-eq-INTEG*: $\text{val-type } x = \text{INTEG} \longleftrightarrow x \in \text{IntVal}'\text{UNIV}$

$$\langle \text{proof} \rangle$$

lemma *val-type-eq-PRODUCT*: $\text{val-type } x = \text{PRODUCT } t1 \ t2 \longleftrightarrow$

$$(\exists a \ b. \ \text{val-type } a = t1 \wedge \ \text{val-type } b = t2 \wedge \ x = \langle | \ a, \ b \ | \rangle)$$

$$\langle \text{proof} \rangle$$

lemma *cexpr-sem-dist-dens-cexpr-nonneg*:

assumes $\Gamma \vdash_c e1 : \text{dist-param-type } \text{dst}$ $\Gamma \vdash_c e2 : \text{dist-result-type } \text{dst}$

assumes $\text{free-vars } e1 \subseteq V$ $\text{free-vars } e2 \subseteq V$

assumes $\sigma \in \text{space } (\text{state-measure } V \ \Gamma)$

shows $\text{ennreal } (\text{extract-real } (\text{cexpr-sem } \sigma (\text{dist-dens-cexpr } \text{dst } e1 \ e2))) =$

$$\text{dist-dens } \text{dst } (\text{cexpr-sem } \sigma \ e1) (\text{cexpr-sem } \sigma \ e2) \wedge$$

$$0 \leq \text{extract-real } (\text{cexpr-sem } \sigma (\text{dist-dens-cexpr } \text{dst } e1 \ e2))$$

$$\langle \text{proof} \rangle$$

lemma *cexpr-sem-dist-dens-cexpr*:

assumes $\Gamma \vdash_c e1 : \text{dist-param-type } \text{dst}$ $\Gamma \vdash_c e2 : \text{dist-result-type } \text{dst}$

assumes $\text{free-vars } e1 \subseteq V$ $\text{free-vars } e2 \subseteq V$

assumes $\sigma \in \text{space } (\text{state-measure } V \ \Gamma)$

shows $\text{ennreal } (\text{extract-real } (\text{cexpr-sem } \sigma (\text{dist-dens-cexpr } \text{dst } e1 \ e2))) =$

$$\text{dist-dens } \text{dst } (\text{cexpr-sem } \sigma \ e1) (\text{cexpr-sem } \sigma \ e2)$$

$$\langle \text{proof} \rangle$$

lemma *nonneg-dist-dens-cexpr*:

assumes $\Gamma \vdash_c e1 : \text{dist-param-type } \text{dst}$ $\Gamma \vdash_c e2 : \text{dist-result-type } \text{dst}$

assumes $\text{free-vars } e1 \subseteq V$ $\text{free-vars } e2 \subseteq V$

shows *nonneg-cexpr* $V \Gamma$ (*dist-dens-cexpr* *dst* $e1$ $e2$)
 ⟨*proof*⟩

8.5 Integral expressions

definition *integrate-var* :: *tyenv* \Rightarrow *vname* \Rightarrow *cexpr* \Rightarrow *cexpr* **where**
integrate-var Γ v $e = \int_c \text{map-vars } (\lambda w. \text{if } v = w \text{ then } 0 \text{ else } \text{Suc } w) e \partial(\Gamma v)$

definition *integrate-vars* :: *tyenv* \Rightarrow *vname list* \Rightarrow *cexpr* \Rightarrow *cexpr* **where**
integrate-vars $\Gamma = \text{foldr } (\text{integrate-var } \Gamma)$

lemma *cexpr-sem-integrate-var*:
cexpr-sem σ (*integrate-var* Γ v e) =
 $\text{RealVal } (\int x. \text{extract-real } (\text{cexpr-sem } (\sigma(v := x)) e) \partial \text{stock-measure } (\Gamma v))$
 ⟨*proof*⟩

lemma *cexpr-sem-integrate-var'*:
 $\text{extract-real } (\text{cexpr-sem } \sigma (\text{integrate-var } \Gamma v e)) =$
 $(\int x. \text{extract-real } (\text{cexpr-sem } (\sigma(v := x)) e) \partial \text{stock-measure } (\Gamma v))$
 ⟨*proof*⟩

lemma *cexpr-typing-integrate-var[simp]*:
 $\Gamma \vdash_c e : \text{REAL} \Longrightarrow \Gamma \vdash_c \text{integrate-var } \Gamma v e : \text{REAL}$
 ⟨*proof*⟩

lemma *cexpr-typing-integrate-vars[simp]*:
 $\Gamma \vdash_c e : \text{REAL} \Longrightarrow \Gamma \vdash_c \text{integrate-vars } \Gamma vs e : \text{REAL}$
 ⟨*proof*⟩

lemma *free-vars-integrate-var[simp]*:
 $\text{free-vars } (\text{integrate-var } \Gamma v e) = \text{free-vars } e - \{v\}$
 ⟨*proof*⟩

lemma *free-vars-integrate-vars[simp]*:
 $\text{free-vars } (\text{integrate-vars } \Gamma vs e) = \text{free-vars } e - \text{set } vs$
 ⟨*proof*⟩

lemma (*in product-sigma-finite*) *product-integral-insert'*:
fixes $f :: - \Rightarrow \text{real}$
assumes *finite* I $i \notin I$ *integrable* $(\text{Pi}_M (\text{insert } i I) M) f$
shows $\text{integral}^L (\text{Pi}_M (\text{insert } i I) M) f = \text{LINT } y | M i. \text{LINT } x | \text{Pi}_M I M. f (x(i := y))$
 ⟨*proof*⟩

lemma *cexpr-sem-integrate-vars*:
assumes $\varrho \in \text{space } (\text{state-measure } V' \Gamma)$
assumes *disjoint*: $\text{distinct } vs \text{ set } vs \cap V' = \{\}$
assumes *integrable* $(\text{state-measure } (\text{set } vs) \Gamma)$

$(\lambda\sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } (\text{set } vs) V' (\sigma, \varrho)) e))$
assumes $e: \Gamma \vdash_c e : \text{REAL free-vars } e \subseteq \text{set } vs \cup V'$
shows $\text{extract-real } (\text{cexpr-sem } \varrho (\text{integrate-vars } \Gamma vs e)) =$
 $\int \sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } (\text{set } vs) V' (\sigma, \varrho)) e) \partial \text{state-measure}$
 $(\text{set } vs) \Gamma$
 $\langle \text{proof} \rangle$

lemma *cexpr-sem-integrate-vars'*:

assumes $\varrho: \varrho \in \text{space } (\text{state-measure } V' \Gamma)$
assumes *disjoint*: $\text{distinct } vs \text{ set } vs \cap V' = \{\}$
assumes *nonneg*: $\text{nonneg-cexpr } (\text{set } vs \cup V') \Gamma e$
assumes *integrable* $(\text{state-measure } (\text{set } vs) \Gamma)$
 $(\lambda\sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } (\text{set } vs) V' (\sigma, \varrho)) e))$
assumes $e: \Gamma \vdash_c e : \text{REAL free-vars } e \subseteq \text{set } vs \cup V'$
shows $\text{ennreal } (\text{extract-real } (\text{cexpr-sem } \varrho (\text{integrate-vars } \Gamma vs e))) =$
 $\int^+ \sigma. \text{extract-real } (\text{cexpr-sem } (\text{merge } (\text{set } vs) V' (\sigma, \varrho)) e) \partial \text{state-measure}$
 $(\text{set } vs) \Gamma$
 $\langle \text{proof} \rangle$

lemma *nonneg-cexpr-sem-integrate-vars*:

assumes $\varrho: \varrho \in \text{space } (\text{state-measure } V' \Gamma)$
assumes *disjoint*: $\text{distinct } vs \text{ set } vs \cap V' = \{\}$
assumes *nonneg*: $\text{nonneg-cexpr } (\text{set } vs \cup V') \Gamma e$
assumes $e: \Gamma \vdash_c e : \text{REAL free-vars } e \subseteq \text{set } vs \cup V'$
shows $\text{extract-real } (\text{cexpr-sem } \varrho (\text{integrate-vars } \Gamma vs e)) \geq 0$
 $\langle \text{proof} \rangle$

lemma *nonneg-cexpr-sem-integrate-vars'*:

$\text{distinct } vs \implies \text{set } vs \cap V' = \{\} \implies \text{nonneg-cexpr } (\text{set } vs \cup V') \Gamma e \implies \Gamma \vdash_c e$
 $: \text{REAL} \implies$
 $\text{free-vars } e \subseteq \text{set } vs \cup V' \implies \text{nonneg-cexpr } V' \Gamma (\text{integrate-vars } \Gamma vs e)$
 $\langle \text{proof} \rangle$

lemma *cexpr-sem-integral-nonneg*:

assumes *finite*: $(\int^+ x. \text{extract-real } (\text{cexpr-sem } (\text{case-nat } x \sigma) e) \partial \text{stock-measure } t) < \infty$
assumes *nonneg*: $\text{nonneg-cexpr } (\text{shift-var-set } V) (\text{case-nat } t \Gamma) e$
assumes $t: \text{case-nat } t \Gamma \vdash_c e : \text{REAL and vars: free-vars } e \subseteq \text{shift-var-set } V$
assumes $\varrho: \sigma \in \text{space } (\text{state-measure } V \Gamma)$
shows $\text{ennreal } (\text{extract-real } (\text{cexpr-sem } \sigma (\int_c e \partial t))) =$
 $\int^+ x. \text{extract-real } (\text{cexpr-sem } (\text{case-nat } x \sigma) e) \partial \text{stock-measure } t$
 $\langle \text{proof} \rangle$

lemma *has-parametrized-subprob-density-cexpr-sem-integral*:

assumes *dens*: $\text{has-parametrized-subprob-density } (\text{state-measure } V' \Gamma) M (\text{stock-measure } t)$
 $(\lambda\varrho x. \int^+ y. \text{eval-cexpr } f (\text{case-nat } x \varrho) y \partial \text{stock-measure } t')$
assumes *nonneg*: $\text{nonneg-cexpr } (\text{shift-var-set } (\text{shift-var-set } V')) (\text{case-nat } t' (\text{case-nat } t \Gamma)) f$

assumes tf : $\text{case-nat } t' (\text{case-nat } t \Gamma) \vdash_c f : \text{REAL}$
assumes $\text{vars}f$: $\text{free-vars } f \subseteq \text{shift-var-set } (\text{shift-var-set } V')$
assumes ϱ : $\varrho \in \text{space } (\text{state-measure } V' \Gamma)$
shows $AE\ x$ in $\text{stock-measure } t$.
 $(\int_c^{+y} \text{eval-cexpr } f (\text{case-nat } x \varrho) y \partial \text{stock-measure } t') = \text{ennreal } (\text{eval-cexpr } (\int_c f \partial t') \varrho x)$
 $\langle \text{proof} \rangle$

end

9 Concrete Density Contexts

theory *PDF-Target-Density-Contexts*
imports *PDF-Density-Contexts PDF-Target-Semantics*
begin

9.1 Definition

type-synonym $\text{cdens-ctxt} = \text{vname list} \times \text{vname list} \times \text{tyenv} \times \text{cexpr}$

definition $\text{dens-ctxt-}\alpha :: \text{cdens-ctxt} \Rightarrow \text{dens-ctxt}$ **where**
 $\text{dens-ctxt-}\alpha \equiv \lambda(vs, vs', \Gamma, \delta). (\text{set } vs, \text{set } vs', \Gamma, \lambda\sigma. \text{extract-real } (\text{cexpr-sem } \sigma \delta))$

definition $\text{shift-vars} :: \text{nat list} \Rightarrow \text{nat list}$ **where**
 $\text{shift-vars } vs = 0 \# \text{map } \text{Suc } vs$

lemma $\text{set-shift-vars}[simp]$: $\text{set } (\text{shift-vars } vs) = \text{shift-var-set } (\text{set } vs)$
 $\langle \text{proof} \rangle$

definition $\text{is-density-cexpr} :: \text{cdens-ctxt} \Rightarrow \text{pdf-type} \Rightarrow \text{cexpr} \Rightarrow \text{bool}$ **where**
 $\text{is-density-cexpr} \equiv \lambda(vs, vs', \Gamma, \delta) t e.$
 $\text{case-nat } t \Gamma \vdash_c e : \text{REAL} \wedge$
 $\text{free-vars } e \subseteq \text{shift-var-set } (\text{set } vs') \wedge$
 $\text{nonneg-cexpr } (\text{shift-var-set } (\text{set } vs')) (\text{case-nat } t \Gamma) e$

lemma is-density-cexprI :
 $\text{case-nat } t \Gamma \vdash_c e : \text{REAL} \Longrightarrow$
 $\text{free-vars } e \subseteq \text{shift-var-set } (\text{set } vs') \Longrightarrow$
 $\text{nonneg-cexpr } (\text{shift-var-set } (\text{set } vs')) (\text{case-nat } t \Gamma) e \Longrightarrow$
 $\text{is-density-cexpr } (vs, vs', \Gamma, \delta) t e$
 $\langle \text{proof} \rangle$

lemma is-density-cexprD :
assumes $\text{is-density-cexpr } (vs, vs', \Gamma, \delta) t e$
shows $\text{case-nat } t \Gamma \vdash_c e : \text{REAL}$ $\text{free-vars } e \subseteq \text{shift-var-set } (\text{set } vs')$
and $\text{is-density-cexprD-nonneg}$: $\text{nonneg-cexpr } (\text{shift-var-set } (\text{set } vs')) (\text{case-nat } t \Gamma) e$
 $\langle \text{proof} \rangle$

lemma *density-context- α* :

assumes *cdens-ctxt-invar* $vs\ vs'\ \Gamma\ \delta$

shows *density-context* ($set\ vs$) ($set\ vs'$) $\Gamma\ (\lambda\sigma.\ extract\ real\ (cexpr\ sem\ \sigma\ \delta))$

<proof>

9.2 Expressions for density context operations

definition *marg-dens-cexpr* :: $tyenv \Rightarrow vname\ list \Rightarrow vname \Rightarrow cexpr \Rightarrow cexpr$

where

marg-dens-cexpr $\Gamma\ vs\ x\ e =$

$map\ vars\ (\lambda y.\ if\ y = x\ then\ 0\ else\ Suc\ y)\ (integrate\ vars\ \Gamma\ (filter\ (\lambda y.\ y \neq x)\ vs)\ e)$

lemma *free-vars-marg-dens-cexpr*:

assumes *cdens-ctxt-invar* $vs\ vs'\ \Gamma\ \delta$

shows *free-vars* (*marg-dens-cexpr* $\Gamma\ vs\ x\ \delta$) $\subseteq shift\ var\ set\ (set\ vs')$

<proof>

lemma *cexpr-typing-marg-dens-cexpr*[*intro*]:

$\Gamma \vdash_c \delta : REAL \implies case\ nat\ (\Gamma\ x)\ \Gamma \vdash_c\ marg\ dens\ cexpr\ \Gamma\ vs\ x\ \delta : REAL$

<proof>

lemma *cexpr-sem-marg-dens*:

assumes *cdens-ctxt-invar* $vs\ vs'\ \Gamma\ \delta$

assumes $x: x \in set\ vs$ **and** $\rho: \rho \in space\ (state\ measure\ (set\ vs')\ \Gamma)$

shows $AE\ v\ in\ stock\ measure\ (\Gamma\ x).$

$ennreal\ (extract\ real\ (cexpr\ sem\ (case\ nat\ v\ \rho)\ (marg\ dens\ cexpr\ \Gamma\ vs\ x\ \delta))) =$

$marg\ dens\ (dens\ ctxt\ \alpha\ (vs, vs', \Gamma, \delta))\ x\ \rho\ v$

<proof>

lemma *nonneg-cexpr-sem-marg-dens*:

assumes *cdens-ctxt-invar* $vs\ vs'\ \Gamma\ \delta$

assumes $x: x \in set\ vs$ **and** $\rho: \rho \in space\ (state\ measure\ (set\ vs')\ \Gamma)$

assumes $v: v \in type\ universe\ (\Gamma\ x)$

shows $extract\ real\ (cexpr\ sem\ (case\ nat\ v\ \rho)\ (marg\ dens\ cexpr\ \Gamma\ vs\ x\ \delta)) \geq 0$

<proof>

definition *marg-dens2-cexpr* :: $tyenv \Rightarrow vname\ list \Rightarrow vname \Rightarrow vname \Rightarrow cexpr \Rightarrow cexpr$

where

marg-dens2-cexpr $\Gamma\ vs\ x\ y\ e =$

$(cexpr\ comp\ aux\ (Suc\ x)\ (fst_c\ (CVar\ 0)))$

$(cexpr\ comp\ aux\ (Suc\ y)\ (snd_c\ (CVar\ 0)))$

$(map\ vars\ Suc\ (integrate\ vars\ \Gamma\ (filter\ (\lambda z.\ z \neq x \wedge z \neq y)\ vs)\ e))))$

lemma *free-vars-marg-dens2-cexpr*:

assumes $cdens\text{-}ctxt\text{-}invar\ vs\ vs'\ \Gamma\ \delta$
shows $free\text{-}vars\ (marg\text{-}dens2\text{-}cexpr\ \Gamma\ vs\ x\ y\ \delta) \subseteq shift\text{-}var\text{-}set\ (set\ vs')$
 $\langle proof \rangle$

lemma $cexpr\text{-}typing\text{-}marg\text{-}dens2\text{-}cexpr[intro]$:
assumes $\Gamma \vdash_c \delta : REAL$
shows $case\text{-}nat\ (PRODUCT\ (\Gamma\ x)\ (\Gamma\ y))\ \Gamma \vdash_c\ marg\text{-}dens2\text{-}cexpr\ \Gamma\ vs\ x\ y\ \delta :$
 $REAL$
 $\langle proof \rangle$

lemma $cexpr\text{-}sem\text{-}marg\text{-}dens2$:
assumes $cdens\text{-}ctxt\text{-}invar\ vs\ vs'\ \Gamma\ \delta$
assumes $x: x \in set\ vs$ **and** $y: y \in set\ vs$ **and** $x \neq y$
assumes $\varrho: \varrho \in space\ (state\text{-}measure\ (set\ vs')\ \Gamma)$
shows $AE\ z\ in\ stock\text{-}measure\ (PRODUCT\ (\Gamma\ x)\ (\Gamma\ y)).$
 $ennreal\ (extract\text{-}real\ (cexpr\text{-}sem\ (case\text{-}nat\ z\ \varrho)\ (marg\text{-}dens2\text{-}cexpr\ \Gamma\ vs\ x$
 $y\ \delta))) =$
 $marg\text{-}dens2\ (dens\text{-}ctxt\text{-}\alpha\ (vs,vs',\Gamma,\delta))\ x\ y\ \varrho\ z$
 $\langle proof \rangle$

lemma $nonneg\text{-}cexpr\text{-}sem\text{-}marg\text{-}dens2$:
assumes $cdens\text{-}ctxt\text{-}invar\ vs\ vs'\ \Gamma\ \delta$
assumes $x: x \in set\ vs$ **and** $y: y \in set\ vs$ **and** $\varrho: \varrho \in space\ (state\text{-}measure\ (set\ vs')\ \Gamma)$
assumes $v: v \in type\text{-}universe\ (PRODUCT\ (\Gamma\ x)\ (\Gamma\ y))$
shows $extract\text{-}real\ (cexpr\text{-}sem\ (case\text{-}nat\ v\ \varrho)\ (marg\text{-}dens2\text{-}cexpr\ \Gamma\ vs\ x\ y\ \delta)) \geq 0$
 $\langle proof \rangle$

definition $branch\text{-}prob\text{-}cexpr :: cdens\text{-}ctxt \Rightarrow cexpr$ **where**
 $branch\text{-}prob\text{-}cexpr \equiv \lambda(vs, vs', \Gamma, \delta). integrate\text{-}vars\ \Gamma\ vs\ \delta$

lemma $free\text{-}vars\text{-}branch\text{-}prob\text{-}cexpr[simp]$:
 $free\text{-}vars\ (branch\text{-}prob\text{-}cexpr\ (vs, vs', \Gamma, \delta)) = free\text{-}vars\ \delta - set\ vs$
 $\langle proof \rangle$

lemma $cexpr\text{-}typing\text{-}branch\text{-}prob\text{-}cexpr[intro]$:
 $\Gamma \vdash_c \delta : REAL \implies \Gamma \vdash_c\ branch\text{-}prob\text{-}cexpr\ (vs,vs',\Gamma,\delta) : REAL$
 $\langle proof \rangle$

lemma $cexpr\text{-}sem\text{-}branch\text{-}prob$:
assumes $cdens\text{-}ctxt\text{-}invar\ vs\ vs'\ \Gamma\ \delta$
assumes $\varrho: \varrho \in space\ (state\text{-}measure\ (set\ vs')\ \Gamma)$
shows $ennreal\ (extract\text{-}real\ (cexpr\text{-}sem\ \varrho\ (branch\text{-}prob\text{-}cexpr\ (vs, vs', \Gamma, \delta)))) =$
 $branch\text{-}prob\ (dens\text{-}ctxt\text{-}\alpha\ (vs, vs', \Gamma, \delta))\ \varrho$
 $\langle proof \rangle$

lemma $subprob\text{-}imp\text{-}subprob\text{-}cexpr$:

assumes *density-context* $V V' \Gamma (\lambda\sigma. \text{extract-real } (cexpr\text{-sem } \sigma \delta))$
shows *subprob-cexpr* $V V' \Gamma \delta$
 <proof>
end

10 Concrete PDF Compiler

theory *PDF-Compiler*
imports *PDF-Compiler-Pred PDF-Target-Density-Contexts*
begin

inductive *expr-has-density-cexpr* :: *cdens-ctxt* \Rightarrow *expr* \Rightarrow *cexpr* \Rightarrow *bool*
 ((1 -/ \vdash_c / ($- \Rightarrow$ / $-$)) [$50,0,50$] 50) **where**

edc-val: *countable-type* (*val-type* v) \Longrightarrow
 $(vs, vs', \Gamma, \delta) \vdash_c \text{Val } v \Rightarrow$
 map-vars *Suc* (*branch-prob-cexpr* $(vs, vs', \Gamma, \delta)$) $*_c \langle CVar\ 0 =_c$
CVal $v \rangle_c$
 | *edc-var*: $x \in \text{set } vs \Longrightarrow (vs, vs', \Gamma, \delta) \vdash_c \text{Var } x \Rightarrow \text{marg-dens-cexpr } \Gamma\ vs\ x\ \delta$
 | *edc-pair*: $x \in \text{set } vs \Longrightarrow y \in \text{set } vs \Longrightarrow x \neq y \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c \langle \text{Var } x, \text{Var } y \rangle \Rightarrow \text{marg-dens2-cexpr } \Gamma\ vs\ x\ y\ \delta$
 | *edc-fail*: $(vs, vs', \Gamma, \delta) \vdash_c \text{Fail } t \Rightarrow CReal\ 0$
 | *edc-let*: $([], vs @ vs', \Gamma, CReal\ 1) \vdash_c e \Rightarrow f \Longrightarrow$
 (*shift-vars* vs , *map* *Suc* vs' , *the* (*expr-type* $\Gamma\ e$) $\cdot \Gamma$,
 map-vars *Suc* $\delta *_c f$) $\vdash_c e' \Rightarrow g \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c LET\ e\ IN\ e' \Rightarrow \text{map-vars } (\lambda x. x - 1)\ g$
 | *edc-rand*: $(vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c \text{Random } dst\ e \Rightarrow$
 $\int_c \text{map-vars } (\text{case-nat } 0\ (\lambda x. x + 2))\ f *_c$
 dist-dens-cexpr $dst\ (CVar\ 0)\ (CVar\ 1)\ \partial \text{dist-param-type } dst$
 | *edc-rand-det*: *randomfree* $e \Longrightarrow \text{free-vars } e \subseteq \text{set } vs' \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c \text{Random } dst\ e \Rightarrow$
 map-vars *Suc* (*branch-prob-cexpr* $(vs, vs', \Gamma, \delta)$) $*_c$
 dist-dens-cexpr $dst\ (\text{map-vars } Suc\ (\text{expr-rt-to-cexpr } e))\ (CVar\ 0)$
 | *edc-if-det*: *randomfree* $b \Longrightarrow$
 $(vs, vs', \Gamma, \delta *_c \langle \text{expr-rt-to-cexpr } b \rangle_c) \vdash_c e1 \Rightarrow f1 \Longrightarrow$
 $(vs, vs', \Gamma, \delta *_c \langle \neg_c \text{expr-rt-to-cexpr } b \rangle_c) \vdash_c e2 \Rightarrow f2 \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c IF\ b\ THEN\ e1\ ELSE\ e2 \Rightarrow f1 +_c f2$
 | *edc-if*: $([], vs @ vs', \Gamma, CReal\ 1) \vdash_c b \Rightarrow f \Longrightarrow$
 $(vs, vs', \Gamma, \delta *_c \text{cexpr-subst-val } f\ TRUE) \vdash_c e1 \Rightarrow g1 \Longrightarrow$
 $(vs, vs', \Gamma, \delta *_c \text{cexpr-subst-val } f\ FALSE) \vdash_c e2 \Rightarrow g2 \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c IF\ b\ THEN\ e1\ ELSE\ e2 \Rightarrow g1 +_c g2$
 | *edc-op-discr*: $(vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \Gamma \vdash e : t \Longrightarrow$
 op-type *oper* $t = \text{Some } t' \Longrightarrow \text{countable-type } t' \Longrightarrow$
 $(vs, vs', \Gamma, \delta) \vdash_c \text{oper } \$\$ e \Rightarrow$
 $\int_c \langle (\text{oper } \$\$_c\ (CVar\ 0)) =_c\ CVar\ 1 \rangle_c *_c \text{map-vars } (\text{case-nat}$
 $0\ (\lambda x. x+2))\ f\ \partial t$
 | *edc-fst*: $(vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \Gamma \vdash e : PRODUCT\ t\ t' \Longrightarrow$

$$\begin{array}{l}
\begin{array}{l}
0 >_c \partial t' \\
| \text{edc-snd:}
\end{array}
\quad
\begin{array}{l}
(vs, vs', \Gamma, \delta) \vdash_c \text{Fst } \$\$ e \Rightarrow \\
\int_c (\text{map-vars } (\text{case-nat } 0 (\lambda x. x + 2))) f \circ_c < \text{CVar } 1, \text{CVar} \\
(vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \Gamma \vdash e : \text{PRODUCT } t t' \Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Snd } \$\$ e \Rightarrow \\
\int_c (\text{map-vars } (\text{case-nat } 0 (\lambda x. x + 2))) f \circ_c < \text{CVar } 0, \text{CVar}
\end{array} \\
\begin{array}{l}
1 >_c \partial t \\
| \text{edc-neg:} \\
| \text{edc-addc:} \\
\Longrightarrow
\end{array}
\quad
\begin{array}{l}
(vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Minus } \$\$ e \Rightarrow f \circ_c (\lambda_c x. -_c x) \\
(vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \text{randomfree } e' \Longrightarrow \text{free-vars } e' \subseteq \text{set } vs' \\
\Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Add } \$\$ < e, e' > \Rightarrow \\
f \circ_c (\lambda_c x. x -_c \text{map-vars } \text{Suc } (\text{expr-xf-to-cexpr } e')) \\
| \text{edc-multc:} \quad (vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow c \neq 0 \Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Mult } \$\$ < e, \text{Val } (\text{RealVal } c) > \Rightarrow \\
(f \circ_c (\lambda_c x. x *_c \text{CReal } (\text{inverse } c))) *_c \text{CReal } (\text{inverse } (\text{abs } c)) \\
| \text{edc-add:} \quad (vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \Gamma \vdash e : \text{PRODUCT } t t \Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Add } \$\$ e \Rightarrow \\
\int_c (\text{map-vars } (\text{case-nat } 0 (\lambda x. x + 2))) f \circ_c (\lambda_c x. < x, \text{CVar } 1 -_c \\
x >_c) \partial t \\
| \text{edc-inv:} \quad (vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Inverse } \$\$ e \Rightarrow \\
(f \circ_c (\lambda_c x. \text{inverse}_c x)) *_c (\lambda_c x. (\text{inverse}_c x) \hat{ }_c \text{CInt } 2) \\
| \text{edc-exp:} \quad (vs, vs', \Gamma, \delta) \vdash_c e \Rightarrow f \Longrightarrow \\
(vs, vs', \Gamma, \delta) \vdash_c \text{Exp } \$\$ e \Rightarrow \\
(\lambda_c x. \text{IF}_c \text{CReal } 0 <_c x \text{ THEN } (f \circ_c \text{ln}_c x) *_c \text{inverse}_c x \text{ ELSE} \\
\text{CReal } 0)
\end{array}
\end{array}$$

code-pred *expr-has-density-cexpr* $\langle \text{proof} \rangle$

Auxiliary lemmas

lemma *cdens-ctxt-invar-insert*:

assumes *inv*: *cdens-ctxt-invar* *vs vs' Γ δ*
assumes *t* : $\Gamma \vdash e : t'$
assumes *free-vars*: $\text{free-vars } e \subseteq \text{set } vs \cup \text{set } vs'$
assumes *hd*: *dens-ctxt-α* (\square , *vs @ vs', Γ, CReal 1*) $\vdash_d e \Rightarrow (\lambda x \text{ xa. ennreal } (\text{eval-cexpr } f x \text{ xa}))$
notes *invar* = *cdens-ctxt-invarD*[*OF inv*]
assumes *wf1*: *is-density-cexpr* (\square , *vs @ vs', Γ, CReal 1*) *t' f*
shows *cdens-ctxt-invar* (*shift-vars vs*) (*map Suc vs'*) (*t' · Γ*) (*map-vars Suc δ *_c f*)
 $\langle \text{proof} \rangle$

lemma *cdens-ctxt-invar-insert-bool*:

assumes *dens*: *dens-ctxt-α* (\square , *vs @ vs', Γ, CReal 1*) $\vdash_d b \Rightarrow (\lambda \rho x. \text{ennreal } (\text{eval-cexpr } f \rho x))$
assumes *wf*: *is-density-cexpr* (\square , *vs @ vs', Γ, CReal 1*) *BOOL f*
assumes *t*: $\Gamma \vdash b : \text{BOOL}$ **and** *vars*: $\text{free-vars } b \subseteq \text{set } vs \cup \text{set } vs'$
assumes *invar*: *cdens-ctxt-invar* *vs vs' Γ δ*

shows $cdens\text{-}ctxt\text{-}invar\ vs\ vs'\ \Gamma\ (\delta\ *_c\ cexpr\text{-}subst\text{-}val\ f\ (BoolVal\ v))$
 $\langle proof \rangle$

lemma $space\text{-}state\text{-}measureD\text{-}shift$:

$\sigma \in space\ (state\text{-}measure\ (shift\text{-}var\text{-}set\ V)\ (case\text{-}nat\ t\ \Gamma)) \implies$
 $\exists x\ \sigma'.\ x \in type\text{-}universe\ t \wedge \sigma' \in space\ (state\text{-}measure\ V\ \Gamma) \wedge \sigma = case\text{-}nat\ x\ \sigma'$
 $\langle proof \rangle$

lemma $space\text{-}state\text{-}measure\text{-}shift\text{-}iff$:

$\sigma \in space\ (state\text{-}measure\ (shift\text{-}var\text{-}set\ V)\ (case\text{-}nat\ t\ \Gamma)) \iff$
 $(\exists x\ \sigma'.\ x \in type\text{-}universe\ t \wedge \sigma' \in space\ (state\text{-}measure\ V\ \Gamma) \wedge \sigma = case\text{-}nat\ x\ \sigma')$
 $\langle proof \rangle$

lemma $nonneg\text{-}cexprI\text{-}shift$:

assumes $\bigwedge x\ \sigma.\ x \in type\text{-}universe\ t \implies \sigma \in space\ (state\text{-}measure\ V\ \Gamma) \implies$
 $0 \leq extract\text{-}real\ (cexpr\text{-}sem\ (case\text{-}nat\ x\ \sigma)\ e)$
shows $nonneg\text{-}cexpr\ (shift\text{-}var\text{-}set\ V)\ (case\text{-}nat\ t\ \Gamma)\ e$
 $\langle proof \rangle$

lemma $nonneg\text{-}cexpr\text{-}shift\text{-}iff$:

$nonneg\text{-}cexpr\ (shift\text{-}var\text{-}set\ V)\ (case\text{-}nat\ t\ \Gamma)\ (map\text{-}vars\ Suc\ e) \iff nonneg\text{-}cexpr$
 $V\ \Gamma\ e$
 $\langle proof \rangle$

lemma $case\text{-}nat\text{-}case\text{-}nat$: $case\text{-}nat\ x\ n\ (case\text{-}nat\ y\ m\ i) = case\text{-}nat\ (case\text{-}nat\ x\ n\ y)\ (\lambda i'.\ case\text{-}nat\ x\ n\ (m\ i'))\ i$
 $\langle proof \rangle$

lemma $nonneg\text{-}cexpr\text{-}shift\text{-}iff2$:

$nonneg\text{-}cexpr\ (shift\text{-}var\text{-}set\ (shift\text{-}var\text{-}set\ V))$
 $(case\text{-}nat\ t1\ (case\text{-}nat\ t2\ \Gamma))\ (map\text{-}vars\ (case\text{-}nat\ 0\ (\lambda x.\ Suc\ (Suc\ x)))\ e) \iff$
 $nonneg\text{-}cexpr\ (shift\text{-}var\text{-}set\ V)\ (case\text{-}nat\ t1\ \Gamma)\ e$
 $\langle proof \rangle$

lemma $nonneg\text{-}cexpr\text{-}Add$:

assumes $\Gamma \vdash_c\ e1 : REAL\ \Gamma \vdash_c\ e2 : REAL$
assumes $free\text{-}vars\ e1 \subseteq V\ free\text{-}vars\ e2 \subseteq V$
assumes $N1$: $nonneg\text{-}cexpr\ V\ \Gamma\ e1$ **and** $N2$: $nonneg\text{-}cexpr\ V\ \Gamma\ e2$
shows $nonneg\text{-}cexpr\ V\ \Gamma\ (e1\ +_c\ e2)$
 $\langle proof \rangle$

lemma $expr\text{-}has\text{-}density\text{-}cexpr\text{-}sound\text{-}aux$:

assumes $\Gamma \vdash e : t\ (vs,\ vs',\ \Gamma,\ \delta) \vdash_c\ e \implies f\ cdens\text{-}ctxt\text{-}invar\ vs\ vs'\ \Gamma\ \delta$
 $free\text{-}vars\ e \subseteq set\ vs \cup set\ vs'$
shows $dens\text{-}ctxt\text{-}\alpha\ (vs,\ vs',\ \Gamma,\ \delta) \vdash_d\ e \implies eval\text{-}cexpr\ f \wedge is\text{-}density\text{-}expr\ (vs,\ vs',\ \Gamma,\ \delta)$
 $t\ f$
 $\langle proof \rangle$

lemma *expr-has-density-cexpr-sound*:

assumes $([], [], \Gamma, CReal\ 1) \vdash_c e \Rightarrow f \ \Gamma \vdash e : t \ \text{free-vars } e = \{\}$

shows *has-subprob-density* (*expr-sem* σ e) (*stock-measure* t) $(\lambda x. \text{ennreal } (\text{eval-cexpr } f \ \sigma \ x))$

$\forall x \in \text{type-universe } t. 0 \leq \text{extract-real } (\text{cexpr-sem } (\text{case-nat } x \ \sigma) \ f)$

$\Gamma' \ 0 = t \Longrightarrow \Gamma' \vdash_c f : REAL$

$\text{free-vars } f \subseteq \{0\}$

<proof>

inductive *expr-compiles-to* :: *expr* \Rightarrow *pdf-type* \Rightarrow *cexpr* \Rightarrow *bool* $(- : - \Rightarrow_c - [10,0,10] \ 10)$

for $e \ t \ f$ **where**

$(\lambda-. \text{UNIT}) \vdash e : t \Longrightarrow \text{free-vars } e = \{\} \Longrightarrow$

$([], [], \lambda-. \text{UNIT}, CReal\ 1) \vdash_c e \Rightarrow f \Longrightarrow$

$e : t \Rightarrow_c f$

code-pred *expr-compiles-to* *<proof>*

lemma *expr-compiles-to-sound*:

assumes $e : t \Rightarrow_c f$

shows *expr-sem* $\sigma \ e = \text{density } (\text{stock-measure } t) \ (\lambda x. \text{ennreal } (\text{eval-cexpr } f \ \sigma' \ x))$

$\forall x \in \text{type-universe } t. \text{eval-cexpr } f \ \sigma' \ x \geq 0$

$\Gamma \vdash e : t$

$t \cdot \Gamma' \vdash_c f : REAL$

$\text{free-vars } f \subseteq \{0\}$

<proof>

11 Tests

values $\{(t, f) \mid t \ f. \text{Val } (\text{IntVal } 42) : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{Minus } \$\$ (\text{Val } (\text{IntVal } 42)) : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{Fst } \$\$ (\text{Val } <|\text{IntVal } 13, \text{IntVal } 37|>) : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.5)) : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{Add } \$\$ <\text{Val } (\text{IntVal } 37), \text{Minus } \$\$ (\text{Val } (\text{IntVal } 13))> : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{LET } \text{Val } (\text{IntVal } 13) \ \text{IN } \text{LET } \text{Minus } \$\$ (\text{Val } (\text{IntVal } 37)) \ \text{IN} \\ \text{Add } \$\$ <\text{Var } 0, \text{Var } 1> : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{IF } \text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.5)) \ \text{THEN} \\ \text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.25))$

ELSE

$\text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.75)) : t \Rightarrow_c f\}$

values $\{(t, f) \mid t \ f. \text{LET } \text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.5)) \ \text{IN} \\ \text{IF } \text{Var } 0 \ \text{THEN}$

$\text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.25))$

ELSE

$\text{Random Bernoulli } (\text{Val } (\text{RealVal } 0.75)) : t \Rightarrow_c f\}$

values $\{(t, f) \mid t f. \text{LET Random Gaussian } \langle \text{Val } (\text{RealVal } 0), \text{Val } (\text{RealVal } 1) \rangle$
IN

LET Random Gaussian $\langle \text{Val } (\text{RealVal } 0), \text{Val } (\text{RealVal } 1) \rangle$ *IN*
Add \$\$ $\langle \text{Var } 0, \text{Var } 1 \rangle : t \Rightarrow_c f\}$

values $\{(t, f) \mid t f. \text{LET Random UniformInt } \langle \text{Val } (\text{IntVal } 1), \text{Val } (\text{IntVal } 6) \rangle$
IN

LET Random UniformInt $\langle \text{Val } (\text{IntVal } 1), \text{Val } (\text{IntVal } 6) \rangle$ *IN*
Add \$\$ $\langle \text{Var } 0, \text{Var } 1 \rangle : t \Rightarrow_c f\}$

values $\{(t, f) \mid t f. \text{LET Random UniformReal } \langle \text{Val } (\text{RealVal } 0), \text{Val } (\text{RealVal } 1) \rangle$
IN

LET Random Bernoulli (*Var* 0) *IN*
IF *Var* 0 *THEN* *Add \$\$* $\langle \text{Var } 1, \text{Val } (\text{RealVal } 1) \rangle$ *ELSE* *Var*

1 : $t \Rightarrow_c f\}$

definition *cthulhu skill* \equiv

LET Random UniformInt (*Val* $\langle |\text{IntVal } 1, \text{IntVal } 100| \rangle$)

IN IF *Less \$\$* $\langle \text{Val } (\text{IntVal } \textit{skill}), \text{Var } 0 \rangle$ *THEN*

Val (*IntVal* *skill*)

ELSE IF *Or \$\$* $\langle \text{Less } \$\$ \langle \text{Var } 0, \text{Val } (\text{IntVal } 6) \rangle,$

Less \$\$ $\langle \text{Mult } \$\$ \langle \text{Var } 0, \text{Val } (\text{IntVal } 5) \rangle,$

Add \$\$ $\langle \text{Val } (\text{IntVal } \textit{skill}), \text{Val } (\text{IntVal } 1) \rangle \rangle \rangle$ *THEN*

Add \$\$ $\langle \text{IF } \text{Less } \$\$ \langle \text{Val } (\text{IntVal } \textit{skill}),$

Random UniformInt $\langle \text{Val } (\text{IntVal } 1), \text{Val } (\text{IntVal } 100) \rangle \rangle$

THEN

Random UniformInt $\langle \text{Val } (\text{IntVal } 1), \text{Val } (\text{IntVal } 10) \rangle$

ELSE

Val (*IntVal* 0),

Val (*IntVal* *skill*)>

ELSE *Val* (*IntVal* *skill*)

definition *cthulhu'* (*skill* :: *int*) =

LET Random UniformInt (*Val* $\langle |\text{IntVal } 1, \text{IntVal } 100| \rangle$)

IN IF *Less \$\$* $\langle \text{Val } (\text{IntVal } \textit{skill}), \text{Var } 0 \rangle$ *THEN*

Val (*IntVal* *skill*)

ELSE IF *Or \$\$* $\langle \text{Less } \$\$ \langle \text{Var } 0, \text{Val } (\text{IntVal } 6) \rangle,$

Less \$\$ $\langle \text{Mult } \$\$ \langle \text{Var } 0, \text{Val } (\text{IntVal } 5) \rangle,$

Add \$\$ $\langle \text{Val } (\text{IntVal } \textit{skill}), \text{Val } (\text{IntVal } 1) \rangle \rangle \rangle$ *THEN*

```

    LET Random UniformInt (Val <|IntVal 1, IntVal 100|>)
    IN Add $$ <IF Less $$ <Val (IntVal skill), Var 1 > THEN
        Random UniformInt (Val <|IntVal 1, IntVal 10|>)
        ELSE
            Val (IntVal 0),
            Val (IntVal skill)>
    ELSE Val (IntVal skill)

```

```

values {(t, f) | t f. cthulhu' 42 : t  $\Rightarrow_c$  f}

```

```

end

```

References

- [1] S. Bhat, J. Borgström, A. D. Gordon, and C. Russo. Deriving probability density functions from probabilistic functional programs. In *Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *Lecture Notes in Computer Science*, pages 508–522. Springer, 2013.
- [2] M. Eberl. A verified compiler for probability density functions. Master’s thesis, Institut für Informatik, TU München, 2014. <http://home.in.tum.de/~eberlm/pdfcompiler.pdf>.