2-or-more approximation for intuitionistic logic

Gabriel Scherer

2-or-more approximation for intuitionistic logic

Gabriel Scherer

November 25, 2014

1 Introduction

The correspondence between natural-deduction proofs of propositional intuitionistic logic, usually written as (logic) derivations for judgments of the form $\Gamma \vdash A$, and well-typed terms in the simply-typed lambda-calculus, with (typing) derivations for the judgment $\Gamma \vdash t : A$, is not one-to-one. In typing judgments $\Gamma \vdash t : A$, the context Γ is a mapping from free variables to their type. In logic derivations, the context Γ is a set of hypotheses; there is no notion of variable, and at most one hypothesis of each type in the set. This means, for example, that the following logic derivation

$$\dfrac{\dfrac{\dfrac{}{A \vdash A}}{A \vdash A \to A}}{\emptyset \vdash A \to A \to A}$$

corresponds to two *distinct* programs, namely $\lambda(x)\,\lambda(y)\,x$ and $\lambda(x)\,\lambda(y)\,y$. We say that those programs have the same *shape*, in the sense that the erasure of their typing derivation gives the same logic derivation – and they are the only programs of this shape.

Despite, or because, not being one-to-one, this correspondence is very helpful to answer questions about type systems. For example, the question of whether, in a given typing environment Γ, the type A is inhabited, can be answered by looking instead for a valid logic derivation of $\lfloor\Gamma\rfloor \vdash A$, where $\lfloor\Gamma\rfloor$ denotes the erasure of the mapping Γ into a set of hypotheses. If we can independently prove that only a finite number of different types need to be considered to find a valid proof (this is the case for propositional logic because of the *subformula property*), then there are finitely many set-of-hypothesis Δ, and the search space of sequents $\Delta \vdash B$ to consider during proof search is finite. This property is key to the termination of most search algorithms for the simply-typed lambda-calculus. Note that it would not work if we searched typing derivations $\Gamma \vdash t : A$ directly: even if there are finitely many types of interest, the set of mappings from variables to such types is infinite.

In our current brand of work, we are interested in a different problem. Instead of knowing whether there *exists* a term t such that $\Gamma \vdash t : A$, we want to know whether this term is *unique* – modulo a given notion of program equivalence. Intuitively, this can be formulated as a search problem where search does not stop to the first candidate, but tries to find whether a second one (that is nonequivalent as a program) exists. In this setting, the technique of searching for logic derivations $\lfloor \Gamma \rfloor \vdash A$ instead is not enough, because a unique logic derivation may correspond to several distinct programs of this shape: summarizing typing environments as set-of-hypotheses loses information about (non)-unicity.

To better preserve this information, one could keep track of the number of times an hypothesis has been added to the context, representing contexts as *multisets* of hypotheses; given a logic derivation annotated with such counts in the context, we can precisely compute the number of programs of this shape. However, even for a finite number of types/formulas, the space of such multisets is infinite; this breaks termination arguments. A natural idea is then to *approximate* multisets by labeling hypotheses with 0 (not available in the context), 1 (added exactly once), or $\bar{2}$ (available two times *or more*); this two-or-more approximation has three possible states, and there are thus finitely many contexts annotated in this way.

The question we answer in this note is the following: is the two-or-more approximation correct? By correct, we mean that if the *precise* number of times a given hypothesis is available varies, but remains in the same approximation class, then the total number of programs of this shape may vary, but will itself remain in the same annotation class. A possible counter-example would be a logic derivation $\Delta \vdash B$ such that, if a given hypothesis $A \in \Delta$ is present exactly twice in the context (or has two free variables of this type), there is one possible program of this shape, but having three copies of this hypothesis would lead to several distinct programs.

Is this approximation correct? We found it surprisingly difficult to have an intuition on this question (guessing what the answer should be), and discussions with colleagues indicate that there is no obvious guess – people have contradictory intuitions on this. In this note, we show (Corollary 2) that this approximation is in fact correct.

2 Terms, types and derivations

We will manipulate several different systems of inference rules and discuss the relations between them: the type system, the logic, and inference systems annotated with counts (precise and approximated). To work uniformly over those various judgments, we will define their context structure as a mapping from types to some set. A set of hypothesis is a mapping from types to booleans, a multiset is a mapping to natural number, and and typing judgment is a mapping from types to sets of free variables (we inverse the usual association order).

Let us first define the set of types (or formulas) $\mathbb{T}$, and the set of λ-terms, by the following grammars:

$$\begin{array}{rclcr}
\mathbb{T} & \ni & A, B, C & := & & \text{types} \\
& & & \mid & X, Y, Z & \text{atoms} \\
& & & \mid & A \to B & \text{arrows} \\
& & & \mid & A * B & \text{products} \\
& & & \mid & A + B & \text{sums}
\end{array}$$

$$\begin{array}{rclr}
t, u & := & & \text{untyped terms} \\
& \mid & x, y, z & \text{variables} \\
& \mid & \lambda(x)\, t & \lambda\text{-abstraction} \\
& \mid & t\ u & \text{application} \\
& \mid & (t, u) & \text{pair} \\
& \mid & \pi_i\, t & \text{projection} \\
& \mid & \sigma_i\, t & \text{sum injection} \\
& \mid & \mathtt{case}(t, x.u_1, y.u_2) & \text{sum destruction}
\end{array}$$

Besides the set of types $\mathbb{T}$, we will write $\mathbb{V}$ for the set of term variables, $\mathbb{B}$ for the set of booleans $\{1_{\mathbb{B}}, 0_{\mathbb{B}}\}$, $\mathbb{N}$ for the (non-negative) natural numbers, and $\bar{\mathbb{2}}$ for the set $\{0, 1, \bar{2}\}$ used by the two-or-more approximation – note the bar on $\bar{\mathbb{2}}$ to indicate the extra element $\bar{2}$ and avoid confusion with other notations for the booleans.

We write F^E for the set of functions from the set E to the set F, and $\#E$ for the cardinal of the set E.

To make our discussion of *shapes* (of propositional judgments) precise and notationally convenient, we will also give a syntax for them, instead of manipulating derivation trees directly. A shape is a variable-less proof-term; we will manipulate *explicitly typed* shapes, where variables have been replaced with their typing information.

$$\begin{array}{rclr}
S, T & := & & \text{typed shapes} \\
& \mid & A, B, C & \text{axioms} \\
& \mid & \lambda(A)\, S & \lambda\text{-abstraction} \\
& \mid & S\ T & \text{application} \\
& \mid & (S, T) & \text{pair} \\
& \mid & \pi_i\, S & \text{projection} \\
& \mid & \sigma_i\, S & \text{sum injection} \\
& \mid & \mathtt{case}(S, A.T_1, B.T_2) & \text{sum destruction}
\end{array}$$

There is a immediate mapping from valid derivations of the usual logic judgment $\Gamma \vdash A$ into shapes, which suggests reformulating the judgment as $S :: \Gamma \vdash A$. Valid judgments are then in direct one-to-one mapping with their valid derivations – a principle all our different judgments will satisfy.

A shape may be invalid, that is not correspond to any valid logic derivation (for example $\pi_1\,(\lambda(X)\,Y)$ is an invalid shape), but a shape-indexed judgment only classifies valid shapes, and we will only consider valid shapes in the rest of this document.

We will manipulate the following judgments, each annotated with a propositional shape S:

- the provability judgment $S :: \Gamma \vdash A$, where the context Γ is in $\mathbb{B}^{\mathbb{T}}$ – isomorphic to sets of types;
- the typing judgment $S :: E \vdash t : A$, where the context E is in $\mathcal{P}(\mathbb{V})^{\mathbb{T}}$ – isomorphic to mappings from term variables to types;
- various counting judgments of the form $S :: \Phi \vdash_K A : a$ for a set K, where Φ is in $K^{\mathbb{T}}$ – mapping from types to a multiplicity in K – and a, in K, represents the output count of the derivation.

The context annotations of all those judgments each have a (commutative) monoid structure $((+_M), 0_M)$ of a binary operation and its unit/neutral element: $((\vee), 0_{\mathbb{B}})$ for $\mathbb{B}$ and $((\cup), \emptyset)$ for $\mathcal{P}(\mathbb{V})$. Our counting sets K will even have the stronger algebraic structure of a *semiring*, we detail this in the next section (Section 3). This is used to define common notations as follows.

The binary operation of the monoid can be lifted to whole context, and we will write Γ, Δ for the addition of contexts: $(\Gamma, \Delta)(A) = \Gamma(A) +_M \Delta(A)$. We will also routinely specify a context as a *partial* mapping from types to annotations, for example the singleton mapping $[A \mapsto a]$ (for some a in the codomain of the mapping); by this we mean that the value for any other element of the domain is the neutral element 0_M. In particular, the notation Γ, A on sets of hypotheses corresponds to the addition $\Gamma, [A \mapsto 1_{\mathbb{B}}]$ in $\mathbb{B}^{\mathbb{T}}$, and the notation $\Gamma, x : A$ on mapping from variables to types corresponds to the addition $\Gamma, [A \mapsto \{x\}]$ in $\mathcal{P}(\mathbb{V})^{\mathbb{T}}$.

Finally, for any function $f : E \to F$, we will write $\lfloor \_ \rfloor_f : E^{\mathbb{T}} \to F^{\mathbb{T}}$ the pointwise lifting of f on contexts: $\lfloor \Phi \rfloor_f(A) \triangleq f(\Phi(A))$. In particular, $\lfloor \_ \rfloor_{\neq \emptyset}$ erases typing environments $\mathcal{P}(\mathbb{V})^{\mathbb{T}}$ into logic contexts $\mathbb{B}^{\mathbb{T}}$, $\lfloor \_ \rfloor_{\neq 0}$ erases multiplicity-annotated contexts $\mathbb{N}^{\mathbb{T}}$ into logic context $\mathbb{B}^{\mathbb{T}}$, and $\lfloor \_ \rfloor_{\#}$ erases typing environments $\mathcal{P}(\mathbb{V})^{\mathbb{T}}$ into multiplicity-annotated contexts $\mathbb{N}^{\mathbb{T}}$.

The logic and typing judgments are defined below. In logic derivations we will simply write A for the singleton mapping $[A \mapsto 1_{\mathbb{B}}]$.

$$\frac{\Gamma(A) = 1_{\mathbb{B}}}{A :: \Gamma \vdash A}$$

$$\frac{S :: \Gamma, A \vdash B}{\lambda(A)\, S :: \Gamma \vdash A \to B} \qquad \frac{S :: \Gamma \vdash A \to B \qquad T :: \Gamma \vdash A}{S\, T :: \Gamma \vdash B}$$

$$\frac{S :: \Gamma \vdash A \qquad T :: \Gamma \vdash B}{(S, T) :: \Gamma \vdash A * B} \qquad \frac{S :: \Gamma \vdash A_1 * A_2}{\pi_i\, S :: \Gamma \vdash A_i}$$

$$\frac{S :: \Gamma \vdash A_i}{\sigma_i\, S :: \Gamma \vdash A_1 + A_2} \qquad \frac{S :: \Gamma \vdash A + B \qquad T_1 :: \Gamma, A \vdash C \qquad T_2 :: \Gamma, B \vdash C}{\texttt{case}(S, A.T_1, B.T_2) :: \Gamma \vdash C}$$

In typing derivations, we write $x : A$ for the singleton mapping $[A \mapsto \{x\}]$. Similarly, the variable freshness condition $x \notin E$ means $(\forall A \in \mathbb{T}, x \notin E(A))$.

$$\frac{x \in E(A)}{A :: E \vdash x : A}$$

$$\frac{x \notin E \qquad S :: E, x : A \vdash t : B}{\lambda(A)\, S :: E \vdash \lambda(x)\, t : A \to B} \qquad \frac{S :: E \vdash t : A \to B \qquad T :: E \vdash u : A}{S\, T :: E \vdash t\, u : B}$$

$$\frac{S :: E \vdash t : A \qquad T :: E \vdash u : B}{(S, T) :: E \vdash (t, u) : A * B} \qquad \frac{S :: E \vdash t : A_1 * A_2}{\pi_i\, S :: E \vdash \pi_i\, t : A_i}$$

$$\frac{S :: E \vdash t : A_i}{\sigma_i\, S :: E \vdash \sigma_i\, t : A_1 + A_2}$$

$$\frac{S :: E \vdash t : A + B \quad x \notin E, y \notin E \quad T_1 :: E, x : A \vdash u_1 : C \quad T_2 :: E, y : B \vdash u_2 : C}{\mathtt{case}(S, A.T_1, B.T_2) :: E \vdash \mathtt{case}(t, x.u_1, y.u_2) : C}$$

Note that while changing the logic judgment from $\Gamma \vdash A$ to $S :: \Gamma \vdash A$ has the clear notational benefit of making valid judgments equivalent to derivations, this argument does not apply to changing the typing judgment from $E \vdash t : A$ to $S :: E \vdash t : A$, as the valid judgments $E \vdash t : A$ are already in one-to-one correspondence with their derivations; S adds some extra redundancy and could be computed from the triple (E, t, A) (or directly from t if we had used *explicitly typed* λ-terms). The benefit of $S :: E \vdash t : A$ is to let us talk very simply of the logical shape of a program, without having to define an additional erasure function from typing derivation to logical derivations: the set of programs of shape S and type A in the environment E is simply defined as:

$$\{t \mid S :: E \vdash t : A\}$$

3 Counting terms in semirings

We are trying to connect two distinct ways of "counting" things about a logic derivation $S :: \Gamma \vdash A$. One is precise, it counts the number of distinct programs of shape S, and the other is the two-or-more approximation.

We generalize those two ways of counting as instances of a generic counting scheme that works for any *semiring* $(K, 0_K, 1_K, +_K, \times_K)$. A semiring is defined as a two-operation structure where $(0_K, +_K)$ and $(1_K, \times_K)$ are monoids, $(+_K)$ commutes and distributes over $(\times_K)$ (which may or may not commute), 0_K is a zero/absorbing element for $(\times_K)$, but $(+_K)$ and $(\times_K)$ need not have inverses[1]

[1]For a *ring* $(K, 0_K, 1_K, +_K, \times_K)$, $(+_K)$ must be invertible, so $\mathbb{Z}$ is a ring while $\mathbb{N}$ is only a semiring.

The usual semiring is $(\mathbb{N}, 0, 1, +, *)$, and it will give the precise counting scheme. The 2-or-more semiring, which we will call $\bar{\mathbb{2}}$, will correspond to the approximated scheme:

- its support is $\bar{\mathbb{2}} = \{0, 1, \bar{2}\}$; 0_K is 0, 1_K is 1
- we define the addition by $1 +_K 1 = \bar{2}$ and $\bar{2} +_K 1 = \bar{2} +_K \bar{2} = \bar{2}$.
- we define the (commutative) multiplication by $\bar{2} \times_K \bar{2} = \bar{2}$.

Definition 1 (Semiring notations) *Addition and multiplication can be lifted pointwise from K to $K^{\mathbb{T}}$: for any $A \in \mathbb{T}$ we define $(\Phi +_K \Psi)(A) \triangleq \Phi(A) +_K \Psi(A)$ and $(\Phi \times_K \Psi)(A) \triangleq \Phi(A) \times_K \Psi(A)$.*

Finally, we define a morphism from the semiring $\mathbb{N}$ to the semiring $\bar{\mathbb{2}}$. Recall that $\varphi : K \to K'$ is a semiring morphism if $\varphi(0_K) = 0_{K'}$, $\varphi(1_K) = 1_{K'}$, $\varphi(a +_K b) = \varphi(a) +_{K'} \varphi(b)$ and $\varphi(a \times_K b) = \varphi(a) \times_{K'} \varphi(b)$.

Definition 2 (The 2-or-more morphism $\varphi_{\bar{2}}$) *We define $\varphi_{\bar{2}} : \mathbb{N} \to \bar{\mathbb{2}}$ as follows:*

$$\begin{cases} \varphi_2(0) = 0 \\ \varphi_2(1) = 1 \\ \varphi_2(n) = \bar{2} & \text{if } n \geq 2 \end{cases}$$

$\varphi_{\bar{2}}$ is a semiring morphism.

Note that $(\mathbb{B}, 0_{\mathbb{B}}, 1_{\mathbb{B}}, \vee, \wedge)$ is also a semiring. For any semiring K, the function $(\_ \neq 0_K) : K \to \mathbb{B}$ (which we may also write $(\neq 0)$) is a semiring morphism.

3.1 Semiring-annotated derivations

Given a semiring K, we now define derivations $S :: \Phi \vdash_K A : a$ where Φ is a a set of types labeled with counts in K (that is, an element of the product $K^{\mathbb{T}}$ for some set Γ), and a is itself in K.

We construct those inference rules such that, when K is instantiated with the semiring of natural numbers $\mathbb{N}$, they really count the different programs of the same shape. For example, consider a logic derivation $S :: \Gamma \vdash B$ starting with a function elimination rule

$$\frac{S_1 :: \Gamma \vdash A \to B \qquad S_2 :: \Gamma \vdash A}{S_1\ S_2 :: \Gamma \vdash B}$$

A program of this shape is of the form $t\ u$, at type B; it can be obtained by pairing any possible program t (of shape S_1) at type $A \to B$ with any possible program u at type A (of shape S_2), so the number of possible applications is the product of the number of possible functions and possible arguments. Formally we have that, for any typing environment E, writing $\#S$ for the cardinal of the set S:

$$\{t_0 \mid S_1\ S_2 :: E \vdash B\} = \left\{(t\ u) \mid \begin{array}{l} S_1 :: E \vdash t : A \to B, \\ S_2 :: E \vdash u : A \end{array}\right\}$$

$$\#\{t_0 \mid S_1\ S_2 :: E \vdash B\} = \#\{t \mid S_1 :: E \vdash t : A \to B\} \times \#\{u \mid S_2 :: E \vdash u : A\}$$

This suggests the following semiring-annotated inference rule:

$$\frac{S_1 :: \Phi \vdash_K A \to B : a_1 \qquad S_2 :: \Phi \vdash_K B : a_2}{S_1\ S_2 :: \Phi \vdash_K B : a_1 \times_K a_2}$$

The other rules are constructed in the same way, and the full inference system is as follows. We write $A : a$ for the singleton mapping $[A \mapsto a]$.

$$A :: \Phi \vdash_K A : \Phi(A)$$

$$\frac{S :: \Phi, A : 1 \vdash_K B : a}{\lambda(A)\, S :: \Phi \vdash_K A \to B : a} \qquad \frac{S_1 :: \Phi \vdash_K A \to B : a_1 \qquad S_2 :: \Phi \vdash_K A : a_2}{S_1\ S_2 :: \Phi \vdash_K B : a_1 \times a_2}$$

$$\frac{S_1 :: \Phi \vdash_K A : a_1 \qquad S_2 :: \Phi \vdash_K B : a_2}{(S_1, S_2) :: \Phi \vdash_K A * B : a_1 \times a_2} \qquad \frac{S :: \Phi \vdash_K A_1 * A_2 : a}{\pi_i\, S :: \Phi \vdash_K A_i : a}$$

$$\frac{S :: \Phi \vdash_K A_i : a}{\sigma_i\, S :: \Phi \vdash_K A_1 + A_2 : a}$$

$$\frac{S :: \Phi \vdash_K A + B : a_1 \qquad T_1 :: \Phi, A : 1 \vdash_K C : a_2 \qquad T_2 :: \Phi, B : 1 \vdash_K C : a_3}{\texttt{case}(S, A.T_1, B.T_2) :: \Phi \vdash_K C : a_1 \times a_2 \times a_3}$$

The identity rule says that if we have a different program variables of type A in our context, then using the variable rule of our typing judgment we can form a different programs. In particular, if A is absent from the context Φ, we have $A :: \Phi \vdash A : 0$. In the function-introduction rule, the number of programs of the form $\lambda(x)\, t : A \to B$ is the number of programs $t : B$ in a context enriched with one extra variable of type A. The most complex rule is the sum elimination rule: the number of case-eliminations $\texttt{case}(t, x_1.u_1, x_2.u_2) : C$ is the product of the number of possible scrutinees $t : A + B$ and cases $u_1 : C$ and $u_2 : C$, with u_1 and u_2 built from one extra formal variable of type A or B accordingly.

We now precisely formulate the fact that the system $\vdash_{\mathbb{N}}$ really counts the number of programs of a given shape. Recall that $\lfloor \_ \rfloor_{\#} : \mathcal{P}(\mathbb{V})^{\mathbb{T}} \to \mathbb{N}^{\mathbb{T}}$ erases a typing environment into a multiplicity-annotated context.

Lemma 1 (Cardinality count) *For any typing environment $E \in \mathcal{P}(\mathbb{V})^{\mathbb{T}}$, shape S and type A, the following is derivable:*

$$S :: \lfloor E \rfloor_{\#} \vdash_{\mathbb{N}} A : \#\{t \mid S :: E \vdash t : A\}$$

Proof: By induction on the shape S, using the following equalities (obtained by inversion of the shape-directed typing judgment):

$$\{t_0 \mid A :: E \vdash t_0 : A\} = \{x \in E(A)\}$$

$$\{t_0 \mid \lambda(A)\, S :: E \vdash t_0 : A \to B\} = \{\lambda(x)\, t \mid S :: E, x : A \vdash t : A\}$$

$$\{t_0 \mid S\, T :: E \vdash t_0 : B\} = \left\{ t\, u \;\middle|\; \begin{array}{l} S :: E \vdash t : A \to B \\ T :: E \vdash u : A \end{array} \right\}$$

$$\{t_0 \mid (S, T) :: E \vdash t_0 : A\} = \left\{ (t, u) \;\middle|\; \begin{array}{l} S :: E \vdash t : A \\ T :: E \vdash u : B \end{array} \right\}$$

$$\{t_0 \mid \pi_i\, S :: E \vdash t_0 : A\} = \{\pi_i\, t \mid S :: E \vdash t : A\}$$

$$\{t_0 \mid \sigma_i\, S :: E \vdash t_0 : A\} = \{\sigma_i\, t \mid S :: E \vdash t : A\}$$

$$\{t_0 \mid \texttt{case}(S, A.T_1, B.T_2) :: E \vdash t_0 : C\}$$

$$= \left\{ \texttt{case}(t, x.u_1, y.u_2) \;\middle|\; \begin{array}{l} S :: E \vdash t : A + B \\ T_1 :: E, x : A \vdash u_1 : C \\ T_2 :: E, y : B \vdash u_2 : C \end{array} \right\}$$

□

While the inference system $\vdash_{\mathbb{N}}$ corresponds to counting programs of a given shape (we formally claim and prove it below), other semirings indeed correspond to counting schemes of interest. The system $\vdash_{\bar{2}}$ corresponds to the "two-or-more" approximation, as can be exemplified by the following derivations:

$$\dfrac{\dfrac{\dfrac{}{A :: A : \bar{2} \vdash_{\bar{2}} A : \bar{2}}}{\lambda(A)\, A :: A : 1 \vdash_{\bar{2}} A \to A : \bar{2}}}{\lambda(A)\, \lambda(A)\, A :: 0 \vdash_{\bar{2}} A \to A \to A : \bar{2}} \qquad \dfrac{\dfrac{\dfrac{\dfrac{}{A :: A : \bar{2} \vdash_{\bar{2}} A : \bar{2}}}{\lambda(A)\, A :: A : \bar{2} \vdash_{\bar{2}} A \to A : \bar{2}}}{\lambda(A)\, \lambda(A)\, A :: A : 1 \vdash_{\bar{2}} A \to A \to A : \bar{2}}}{\lambda(A)\, \lambda(A)\, \lambda(A)\, A :: 0 \vdash_{\bar{2}} A \to A \to A \to A : \bar{2}}$$

The $\vdash_{\mathbb{B}}$ system intuitively corresponds to a system where the two possible counts are "zero" and "one-or-more", that is, it only counts inhabitation. There is a precise correspondence between this system and the logic derivation we formulated: derivations of the form $S :: \Gamma \vdash 1_{\mathbb{B}} : A$ are in one-to-one correspondence with valid logic derivations $S :: \Gamma \vdash A$, and derivations $S :: \Gamma \vdash 0_{\mathbb{B}} : A$ correspond to *invalid* logic derivations, where the shape S is valid but the context Γ lacks some hypothesis used in S. In particular, $\emptyset \vdash 0_{\mathbb{B}} : A$ is always provable by immediate application of the variable rule.

Lemma 2 (Provability count) *There is a one-to-one correspondence between logic derivations of $S :: \Gamma \vdash A$ and $\mathbb{B}$-counting derivations of $S :: \Gamma \vdash_{\mathbb{B}} 1_{\mathbb{B}} : A$.*

Proof: Immediate by induction on the shape S. □

3.2 Semiring morphisms determine correct approximations

The key reason why the two-or-more approximation is correct is that the mapping from $\mathbb{N}$ to $\bar{\mathbb{2}}$ is a semiring morphism and, as such, preserves the annotation structure of counting derivations.

Theorem 1 (Morphism of derivations) *If $\varphi : K \to K'$ is a semiring morphism and $S :: \Phi \vdash A : a$ holds, then $S :: \lfloor \Phi \rfloor_\varphi \vdash A : \varphi(a)$ also holds.*

<u>Proof</u>: By induction on S.

$$A :: \Phi \vdash_K A : \Phi(A) \qquad \Rightarrow \qquad A :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A : \varphi(\Phi(A))$$

$$\frac{S :: \Phi, A : 1_K \vdash_K B : a}{\lambda(A)\, S :: \Phi \vdash_K A \to B : a} \qquad \Rightarrow \qquad \frac{S :: \lfloor \Phi \rfloor_\varphi, A : 1_{K'} \vdash_{K'} B : \varphi(a)}{\lambda(A)\, S :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A \to B : \varphi(a)}$$

To use our induction hypothesis, we needed the fact that $\lfloor \Phi \rfloor_\varphi, A : 1_{K'}$ is equal to $\lfloor \Phi, A : 1_K \rfloor_\varphi$; this comes from the fact that φ is a semiring morphism: $\varphi(1_K) = \varphi(1_{K'})$ and $\varphi(a +_K b) = \varphi(a) +_{K'} \varphi(b)$, thus $\lfloor \Phi, \Psi \rfloor_\varphi = \lfloor \Phi \rfloor_\varphi, \lfloor \Psi \rfloor_\varphi$.

$$\frac{S_1 :: \Phi \vdash_K A \to B : a_1 \qquad S_2 :: \Phi \vdash_K A : a_2}{S_1\ S_2 :: \Phi \vdash_K B : a_1 \times a_2} \qquad \Rightarrow$$

$$\frac{S_1 :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A \to B : \varphi(a_1) \qquad S_2 :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A : \varphi(a_2)}{S_1\ S_2 :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} B : \varphi(a_1) \times \varphi(a_2)}$$

To conclude we then use the fact that $\varphi(a_1) \times \varphi(a_2) = \varphi(a_1 \times a_2)$.

$$\frac{S_1 :: \Phi \vdash_K A : a_1 \qquad S_2 :: \Phi \vdash_K B : a_2}{(S_1, S_2) :: \Phi \vdash_K A * B : a_1 \times a_2} \qquad \Rightarrow$$

$$\frac{S_1 :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A : \varphi(a_1) \qquad S_2 :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} B : \varphi(a_2)}{(S_1, S_2) :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A * B : \varphi(a_1) \times \varphi(a_2)}$$

$$\frac{S :: \Phi \vdash_K A_1 * A_2 : a}{\pi_i\, S :: \Phi \vdash_K A_i : a} \qquad \Rightarrow \qquad \frac{S :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A_1 * A_2 : \varphi(a)}{\pi_i\, S :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A_i : \varphi(a)}$$

$$\frac{S :: \Phi \vdash_K A_i : a}{\sigma_i\, S :: \Phi \vdash_K A_1 + A_2 : a} \qquad \Rightarrow \qquad \frac{S :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A_i : \varphi(a)}{\sigma_i\, S :: \lfloor \Phi \rfloor_\varphi \vdash_{K'} A_1 + A_2 : \varphi(a)}$$

$$\frac{\begin{array}{c}S :: \Phi \vdash_K A + B : a_1 \\ T_1 :: \Phi, A : 1_K \vdash_K C : a_2 \qquad T_2 :: \Phi, B : 1_K \vdash_K C : a_3\end{array}}{\texttt{case}(S, A.T_1, B.T_2) :: \Phi \vdash_K C : a_1 \times a_2 \times a_3} \qquad \Rightarrow$$

$$\frac{S :: \lfloor\Phi\rfloor_\varphi \vdash_{K'} A + B : \varphi(a_1) \quad T_1 :: \lfloor\Phi\rfloor_\varphi, A : 1_{K'} \vdash_{K'} C : \varphi(a_2) \quad T_2 :: \lfloor\Phi\rfloor_\varphi, B : 1_{K'} \vdash_{K'} C : \varphi(a_3)}{\texttt{case}(S, A.T_1, B.T_2) :: \lfloor\Phi\rfloor_\varphi \vdash_{K'} C : \varphi(a_1) \times \varphi(a_2) \times \varphi(a_3)}$$

□

From there, it remains to point out that the right-hand-side count is uniquely determined by the context multiplicity.

Lemma 3 (Determinism) *If $S :: \Phi \vdash_K A : a$ and $S :: \Phi \vdash_K A : b$ then $a = b$.*

Proof: Immediate by induction on derivations. Note that the fact that the judgments are indexed by the same shape S is essential here. □

Corollary 1 (Relation under morphism) *If $\varphi : K \to K'$ is a semiring morphism and $\lfloor\Phi_1\rfloor_\varphi = \lfloor\Phi_2\rfloor_\varphi$, then $S :: \Phi_1 \vdash_K A : a_1$ and $S :: \Phi_2 \vdash_K A : a_2$ imply $\varphi(a_1) = \varphi(a_2)$*

Proof: By the Morphism Theorem 1, we have $S :: \lfloor\Phi_1\rfloor_\varphi \vdash_{K'} A : \varphi(a_1)$ and $S :: \lfloor\Phi_2\rfloor_\varphi \vdash_{K'} A : \varphi(a_2)$. If $\lfloor\Phi_1\rfloor_\varphi = \lfloor\Phi_2\rfloor_\varphi$ we can conclude by determinism (Lemma 3) that $\varphi(a_1) = \varphi(a_2)$.

□

Corollary 2 *The 2-or-more approximation is correct to decide unicity of inhabitants of a given shape S. If $\lfloor E_1\rfloor_{\varphi_2\#} = \lfloor E_2\rfloor_{\varphi_2\#}$, then*

$$\varphi_{\bar{2}}\#\{t \mid S :: E_1 \vdash t : A\} = \varphi_{\bar{2}}\#\{t \mid S :: E_2 \vdash t : A\}$$

Proof: Counting the inhabitants corresponds to the system $\vdash_{\mathbb{N}}$ (Lemma 1), so we have

$$S :: \lfloor E_1\rfloor_\# \vdash_{\mathbb{N}} A : \#\{t \mid S :: E_1 \vdash t : A\}$$

$$S :: \lfloor E_2\rfloor_\# \vdash_{\mathbb{N}} A : \#\{t \mid S :: E_2 \vdash t : A\}$$

The result then directly comes from the previous corollary, given that $\varphi_{\bar{2}}$ is a semiring morphism.

□

A Appendix: n-or-more logics

The result can be extended to any "n-or-more" approximation scheme given by the semiring $\bar{\mathbb{n}}$ and semiring morphism $\varphi_{\bar{\mathbb{n}}} : \mathbb{N} \to \bar{\mathbb{n}}$ defined as follows (assuming $n \geqslant 1$):

$$\bar{\mathbb{n}} \triangleq \{0, 1, \ldots, n-1, n\} \qquad 0_{\bar{\mathbb{n}}} \triangleq 0 \qquad 1_{\bar{\mathbb{n}}} \triangleq 1$$

$$(a +_{\bar{\mathbb{n}}} b) \triangleq \min(a +_{\mathbb{N}} b, n) \qquad (a \times_{\bar{\mathbb{n}}} b) \triangleq \min(a \times_{\mathbb{N}} b, n)$$

To check that $\varphi_{\bar{\mathbb{n}}}$ is indeed a morphism, one needs to remark that having either $a \geqslant n$ or $b \geqslant n$ implies $(a +_{\mathbb{N}} b) \geqslant n$ and, if a and b are non-null, $(a *_{\mathbb{N}} b) \geqslant n$.

B Appendix: Counting logics as counting functions

The semiring-annotated systems do not really have good properties as logics; in particular they do not allow cut elimination: the counts are static properties of a proof derivation that cannot be nicely connected to the dynamics of cut-elimination or program execution.

For example, assuming that the context Φ counts two variables for some fixed atomic type X (for example a parametrization of booleans), from any proof $S :: \Phi \vdash A : a$ that does not use X we can build a proof $\Phi \vdash A : 2 \times a$ that morally corresponds to the same programs:

$$\dfrac{\dfrac{\dfrac{\Phi(X) = 2}{X :: \Phi \vdash X : 2} \qquad S :: \Phi \vdash A : a}{(X, S) :: \Phi \vdash X * A : 2 \times a}}{\pi_2\,(X, S) :: \Phi \vdash A : 2 \times a}$$

Instead of presenting counting as part of a semiring-annotated logic, it is possible to decompose each system $S :: \Phi \vdash_K A : a$ as given by the usual logic (with good dynamic properties) $S :: \lfloor \Phi \rfloor_{\neq 0} \vdash A$ and a family of *counting functions* $\langle S \rangle_K : K^{\mathbb{T}} \to K$ that represents the (deterministic) computation of the output annotation a as function of the annotated context Φ.

$$\begin{aligned}
\langle A \rangle_K &\triangleq (\Phi \mapsto \Phi(A)) \\
\langle \lambda(A)\, S \rangle_K &\triangleq (\Phi \mapsto \langle S \rangle_K(\Phi, A : 1_K)) \\
\langle S\ T \rangle_K &\triangleq \langle S \rangle_K \times_K \langle T \rangle_K \\
\langle (S, T) \rangle_K &\triangleq \langle S \rangle_K \times_K \langle T \rangle_K \\
\langle \pi_i\, S \rangle_K &\triangleq \langle S \rangle_K \\
\langle \sigma_i\, S \rangle_K &\triangleq \langle S \rangle_K \\
\langle \mathtt{case}(S, A.T_1, B.T_2) \rangle_K &\triangleq \left(\Phi \mapsto \begin{array}{r} \langle S \rangle_K(\Phi) \times_K \langle T_1 \rangle_K(\Phi, A : 1_K) \\ \times_K \langle T_2 \rangle_K(\Phi, B : 1_K) \end{array} \right)
\end{aligned}$$

B.1 Correct approximation through counting functions

Counting functions provide an alternate way to formulate the correctness of approximations (the equivalent of Corollary 1 in our proof). Consider a function $\varphi : K \to K'$ between the support sets of two semirings K and K'; we will say that φ is a correct approximation if a certain relation $(\mathcal{R}_\varphi)$ holds between counting functions over K and K'.

Given two relations $\mathcal{R} \subseteq E \times F$ and $\mathcal{R}' \subseteq E' \times F'$, we define a relation $(\mathcal{R} \to \mathcal{R}')$ between the function spaces $E \to E'$ and $F \to F'$ in the standard way:

$$f\ (\mathcal{R} \to \mathcal{R}')\ g \quad \iff \quad \forall a, b,\ (a\ \mathcal{R}\ b) \implies f(a)\ \mathcal{R}'\ g(b)$$

Similarly, a relation $\mathcal{R} \subseteq E \times F$ can be lifted pointwise to products over some set G into a relation $\mathcal{R}^G \subseteq E^G \times F^G$ defined by

$$s \mathcal{R}^G t \quad \iff \quad \forall x \in G,\ s(x)\ \mathcal{R}\ t(x)$$

When manipulating counting functions in $K^{\mathbb{T}} \to K$ and $K'^{\mathbb{T}} \to K$, we will simply write $(\mathcal{R})$ instead of $(\mathcal{R}^{\mathbb{T}} \to \mathcal{R})$ to lift a relation $\mathcal{R} \subseteq K \times K'$ to a relation between functions in $K^{\mathbb{T}} \to K$ and $K'^{\mathbb{T}} \to K'$. Finally, given any function $\varphi : K \to K'$, we write $\mathcal{R}_\varphi$ the relation determined by the graph of φ:

$$k\ \mathcal{R}_\varphi\ k' \quad \iff \quad \varphi(k) = k'$$

We can then state the correctness of an approximation $\varphi : K \to K'$ as follows:

Definition 3 *A function $\varphi : K \to K'$ between the support sets of two semirings is a* correct counting approximation *if, for any logic derivation $P : \Gamma \vdash A$, we have*

$$\langle P \rangle_K\ \mathcal{R}_\varphi\ \langle P \rangle_{K'}$$

We can then prove that any semiring morphism φ is a *correct counting approximation*; this comes from the stability properties of $\mathcal{R}_\varphi$ exhibited by the following admissible reasoning rules:

$$0_K\ \mathcal{R}_\varphi\ 0_{K'} \qquad\qquad 1_K\ \mathcal{R}_\varphi\ 1_{K'}$$

$$\frac{k_1\ \mathcal{R}_\varphi\ k_1' \qquad k_2\ \mathcal{R}_\varphi\ k_2'}{(k_1 + k_2)\ \mathcal{R}_\varphi\ (k_1' + k_2')} \qquad\qquad \frac{k_1\ \mathcal{R}_\varphi\ k_1' \qquad k_2\ \mathcal{R}_\varphi\ k_2'}{(k_1 * k_2)\ \mathcal{R}_\varphi\ (k_1' + k_2')}$$

$$[A \mapsto 1_K]\ \mathcal{R}_\varphi{}^{\mathbb{T}}\ [A \mapsto 1_{K'}]$$

$$\frac{f\ (\mathcal{R}_\varphi{}^{\mathbb{T}} \to \mathcal{R}_\varphi)\ f' \qquad g\ (\mathcal{R}_\varphi{}^{\mathbb{T}} \to \mathcal{R}_\varphi)\ g'}{(f + g)\ (\mathcal{R}_\varphi{}^{\mathbb{T}} \to \mathcal{R}_\varphi)\ (f' + g')} \qquad \frac{f\ (\mathcal{R}_\varphi{}^{\mathbb{T}} \to \mathcal{R}_\varphi)\ f' \qquad g\ (\mathcal{R}_\varphi{}^{\mathbb{T}} \to \mathcal{R}_\varphi)\ g'}{(f * g)\ (\mathcal{R}_\varphi{}^{\mathbb{T}} \to \mathcal{R}_\varphi)\ (f' * g')}$$

Theorem 2 *Any semiring morphism φ is a* correct counting approximation*.*

B.2 Other uses of counting functions

In our main proof, we were able to side-step the definition and use of counting functions (which add some notational overhead) by directly formulating the Morphism Theorem 1. They nonetheless seem to say interesting things about logic derivations. We can for example define the following notions:

Definition 4 *Given a proof of $S :: \Phi \vdash A$, a type B is said to be* necessary *if $\Phi(B) = 0$ implies $\langle S\rangle_{\mathbb{N}} = 0$.*

Given a proof of $S :: \Phi \vdash A$, a type B is said to be saturating *if $\Phi(B) = \bar{2}$ implies $\langle S\rangle_{\bar{\mathbb{2}}} = \bar{2}$.*

In the intuitionistic logic used so far, it seems that all necessary hypotheses are saturating, but the converse is not true: A is saturating but not necessary for $\lambda(A)\, A$.

If we extended our logic with a notion of "squashed types" $[T]$ (which is empty if T is, and uniquely inhabited otherwise), with the counting rule

$$\frac{S :: \Phi \vdash_K T : a \qquad a \neq 0_K}{[S] :: \Phi \vdash_K [T] : 1_K}$$

then A would be necessary but not saturating for the shape $[A]$.

In presence of squashed types, the approximation result remains true for all morphisms $\varphi : K \to K'$ such that $a \neq 0_K$ implies $\varphi(a) \neq 0_{K'}$, which is in particular the case of the morphisms from $\mathbb{N}$ to any $\bar{\mathbb{n}}$.