From Program Equivalences to Program Metrics
Lecture Notes
OPLSS 2021

Ugo Dal Lago Francesco Gavazzo

Chapter 1

Introduction

These notes are designed to be a support for the students following the eponymous course held by Ugo Dal Lago within the Oregon Summer School in Programming Language Semantics. But they are freely downloadable and can circulate freely.

The aim of these notes is to introduce the fundamentals of relational reasoning between higher-order programs. The notes are divided in three chapters:

- In the first chapter, notions of equivalence for more and more sophisticated lambda-calculi are introduced, starting from the simply-typed lambda-calculus and gradually adding features, i.e., recursive types and algebraic effects. In particular, contextual, denotational, inductive, and coinductive equivalences will be introduced.
- In the second chapter, relational reasoning will take a more quantitative flavor: programs are not related by judging the latter equivalent (or not), but by evaluating the distance between them. This is possible through the notion of metrics and to its application to the programs described in the previous point. However, there is a price to pay, namely the fact that the underlying type system must become linear, thus allowing to tame the phenomena of distance amplification and trivialization.
- Finally, in the third chapter, we will show how program distances can be modified in such a way as to take the behavior of the environment into account, thus inducing a genuinely contextual notion of distance. We will make this idea concrete in the particular case of logical relations, applying the latter to the simply-typed lambda-calculus.

Readers are encouraged to report any typos or errors to the authors by email.[1]

[1]Please write to ugo.dallago@unibo.it and francesco.gavazzo2@unibo.it

Chapter 2

Program Equivalences

2.1 Mathematical Preliminaries

We recall some background notions on relations and their algebra. That will allow us to reduce the complexity of some proofs by means of simple, pointfree calculations. Contrary to standard, set-based presentations of relations, we use an 'algebraic' notation for relations and their algebra. This choice has the advantage of highlighting the connection between relations and abstract notions of distance (the central theme of the second part of these notes).

Definition 1. *A relation R between two sets X and Y, written as $R : X \rightarrow\!\!\!\!\!\rightarrow Y$, is a set $R \subseteq X \times Y$.*

Given elements $x \in X$, $y \in Y$ we say that x is R-related to y, and write $x \mathrel{R} y$, if $(x, y) \in R$. For any set X the identity relation $\mathsf{I}_X : X \rightarrow\!\!\!\!\!\rightarrow X$ is defined as equality on X. Moreover, for relations $R : X \rightarrow\!\!\!\!\!\rightarrow Y$ and $S : Y \rightarrow\!\!\!\!\!\rightarrow Z$, we define the composition $R; S : X \rightarrow\!\!\!\!\!\rightarrow Z$ (read 'R after S') as:

$$xR; Sz \overset{\triangle}{\iff} \exists y \in Y.\, x \mathrel{R} y \,\&\, y \mathrel{S} z.$$

Composition of relations is associative, and I is the unit of composition.

Moreover, for all sets X, Y, we denote by $\mathsf{Rel}(X, Y)$ the collection of relations between X and Y. Such a set has a complete lattice structure with order given by set-theoretic inclusion $\subseteq$ and meets and joins given by $\bigcap$ and $\bigcup$, respectively. The complete lattice structure of sets of the form $\mathsf{Rel}(X, Y)$ nicely interacts with the monoid structure of relation composition, meaning that Rel forms a *quantaloid* Hofmann, Seal, and Tholen (2014). In particular, for all relations $R : X \rightarrow\!\!\!\!\!\rightarrow Y$, $S_i : Y \rightarrow\!\!\!\!\!\rightarrow Z$ $(i \in I)$, and $Q : Z \rightarrow\!\!\!\!\!\rightarrow W$ the following distributivity laws hold:

$$(\bigcup_{i \in I} S_i); Q = \bigcup_{i \in I}(S_i; Q),$$
$$R; (\bigcup_{i \in I} S_i) = \bigcup_{i \in I}(R; S_i).$$

Exercise 1. Show that relation composition is monotone in both arguments.

For a relation $R : X \rightarrow\!\!\!\!\!\rightarrow Y$ we denote by $R^\top : Y \rightarrow\!\!\!\!\!\rightarrow X$ its transportation defined by $y \mathrel{R^\top} x \overset{\triangle}{\iff} x \mathrel{R} y$. It is a routine exercise to verify that $-^\top$ is monotone and satisfies the following identities (the third one stating that $-^\top$ is an involution):

$$(R; S)^\top = S^\top; R^\top$$
$$\mathsf{I}^\top = \mathsf{I}$$
$$(R^\top)^\top = R.$$

Additionally, we can regard each function $f : X \to Y$ as a relation $f_\circ : X \rightarrow\!\!\!\!\!\rightarrow Y$ defined by

$$x \mathrel{f_\circ} y \overset{\triangle}{\iff} f(x) = y.$$

For readability, we overload the notation and write f in place of $f_\circ$ keeping in mind that whenever we write expressions such that $R; f$ or $f; R$ we really mean f regarded as a relation.

Oftentimes, we will write relations of the form $f; S; g^\top$, for a relation $S : Z \twoheadrightarrow W$ and functions $f : X \to Z$, $g : Y \to W$. We have:

$$x \; (f; S; g^\top) \; y \iff f(x) \; S \; g(y).$$

Given $R : X \twoheadrightarrow Y$ we can thus express a generalised monotonicity condition in pointfree fashion as:

$$R \subseteq f; S; g^\top.$$

Indeed, taking $f = g$, we obtain standard monotonicity of f with respect to R. The pointfree notation is compact and allows us to reduce nontrivial proofs to simple algebraic calculations. In doing so, we will make extensively use of the following *adjunction rules* (also known as shunting) Hofmann et al. (2014), for $f : X \to Y$, $g : Y \to Z$, $R : X \twoheadrightarrow Y$, $S : Y \twoheadrightarrow Z$, and $Q : X \twoheadrightarrow Z$:

$$\begin{aligned} R; g \subseteq Q &\iff R \subseteq Q; g^\top \\ f^\top; Q \subseteq S &\iff Q \subseteq f; S. \end{aligned}$$

Finally, since we are interested in preorder and equivalence relations, we recall that we can define the notions of a reflexive, a transitive, and a symmetric relation pointfree. In fact, a relation $R : X \twoheadrightarrow X$ is reflexive if $\mathsf{I}_X \subseteq R$, transitive if $R; R \subseteq R$, and symmetric if $R \subseteq R^\top$.

2.1.1 Set-theoretic Constructions

We summarise the main constructions on sets we use in these notes. For a set X, we define $X_\bot \triangleq \{\mathit{just}\ x \mid x \in X\} \cup \{\bot\}$. The set $X_\bot$ thus contains all elements of X plus a special element $\bot$ usually denoting the 'undefined'. We use sets of the form $X_\bot$ to define partial functions. More specifically, we define a partial function $\varphi : A \rightharpoonup B$ as a function $\varphi : A \to B_\bot$. Given sets A, B, we write B^A, $A \times B$, and $A + B$ for the set of functions from A to B, for the cartesian product of A and B, and for the disjoint union of A and B, respectively. Elements in $A \times B$ (i.e. pairs) are denoted by (a, b), whereas elements in $A + B$ are either of the form $\mathbf{inl}\ a$ with $a \in A$, or $\mathbf{inr}\ b$ with $b \in B$. In particular, we recall that $A + B$ is the coproduct of A and B; $A \times B$ is the (categorical) product of A and B (we denote by $\pi_i : X_1 \times X_2 \to X_i$ the i-th projection function); and B^A is the exponential object of A and B.

2.2 A Simply-Typed λ-Calculus

We consider a simply-typed λ-calculus with Booleans, arrows, and finitary product types, which we call Λ^{ST}. The grammar of types, values, and terms of Λ^{ST} is given in Figure 2.1.

$$\begin{array}{lll} \text{Types } \sigma, \tau ::= \mathbf{bool} & \text{Values } v, w ::= x & \text{Computations } e, f ::= \mathbf{return}\ v \\ \quad\mid \mathbf{unit} & \quad\mid \langle\rangle & \quad\mid \mathbf{if}\ v\ \mathbf{then}\ e\ \mathbf{else}\ f \\ \quad\mid \sigma \times \tau & \quad\mid \mathbf{true} & \quad\mid \mathbf{proj}_l\ v \\ \quad\mid \sigma \to \tau & \quad\mid \mathbf{false} & \quad\mid \mathbf{proj}_r\ v \\ & \quad\mid \langle v, w\rangle & \quad\mid vw \\ & \quad\mid \lambda x.e & \quad\mid \mathbf{let}\ x = e\ \mathbf{in}\ f \end{array}$$

Figure 2.1: Types, values, and computations of Λ^{ST}.

Following the methodology of fine-grain call-by-value (Levy, Power, & Thielecke, 2003), we divide expressions of Λ^{ST} in two disjoint classes: *values* and *computations* (also called *terms*). Intuitively, a value is the result of a computation, whereas a computation is a value producer, i.e. an expression that once evaluated may produce a value (the evaluation process might not terminate). Accordingly, computations must be explicitly sequenced by means of the sequencing constructor $\mathbf{let}\ x = - \ \mathbf{in}\ -$.

We adopt standard syntactical conventions from (Barendregt, 1984) (notably the so-called *variable convention*). The notion of a free (resp. bound) variable is defined as usual. In particular, we denote by $FV(e)$ (resp. $FV(v)$) the set of free variables of a computation e (resp. value v). As it is customary, we identify terms up to renaming of bound variables and say that a term is closed if it has no free variables. Oftentimes we refer to closed computations as *programs*. Moreover, for finite lists of syntactic expressions (such as variables, computations, or values) we sometimes employ the vector, thus writing $\boldsymbol{\phi}$ for a finite list $\phi_1, \dots, \phi_n$ of ϕs. As a notational convention, we agree that whenever we have a vector $\boldsymbol{\phi}$, the symbol ϕ_i stands for the i-th element of $\boldsymbol{\phi}$. For instance, we write $\boldsymbol{x}$ and $\boldsymbol{v}$ for a finite list of variables and values, respectively, and x_i,v_i for the their i-th element in the list.

Finally, we write $e[v/x]$ (resp. $w[v/x]$) for the capture-free substitution of the value v for all free occurrences of x in e (resp. w). Formally, we define $e[v/x]$ and $w[v/x]$ by mutual recursion on e and w as follows:

$$
\begin{aligned}
x[v/x] &\triangleq v \\
y[v/x] &\triangleq y \\
\mathbf{true}[v/x] &\triangleq \mathbf{true} \\
\mathbf{false}[v/x] &\triangleq \mathbf{false} \\
\langle\rangle[v/x] &\triangleq \langle\rangle \\
\langle w, u\rangle[v/x] &\triangleq \langle w[v/x], u[v/x]\rangle \\
(\lambda y.e)[v/x] &\triangleq \lambda y.e[v/x]
\end{aligned}
$$

$$
\begin{aligned}
(\mathbf{return}\ w)[v/x] &\triangleq \mathbf{return}\ w[v/x] \\
(\mathbf{if}\ w\ \mathbf{then}\ e\ \mathbf{else}\ f)[v/x] &\triangleq \mathbf{if}\ w[v/x]\ \mathbf{then}\ e[v/x]\ \mathbf{else}\ f[v/x] \\
(\mathbf{proj}_l\ w)[v/x] &\triangleq \mathbf{proj}_l\ w[v/x] \\
(\mathbf{proj}_r\ w)[v/x] &\triangleq \mathbf{proj}_r\ w[v/x] \\
(wu)[v/x] &\triangleq w[v/x]u[v/x] \\
(\mathbf{let}\ y = e\ \mathbf{in}\ f)[v/x] &\triangleq \mathbf{let}\ y = e[v/x]\ \mathbf{in}\ f[v/x]
\end{aligned}
$$

The notion of a substitution is extended to the one of *simultaneous substitution* in the natural way. Given vectors $\boldsymbol{v}$, $\boldsymbol{x}$ of values and variables (which we tacitly assume to have the same length), respectively, we write $e[\boldsymbol{v}/\boldsymbol{x}]$ for the simultaneous substitution of all values v_i for variables x_i in e. We define $w[\boldsymbol{v}/\boldsymbol{x}]$ similarly. Moreover, oftentimes we will use metavariables $\gamma, \delta, \dots$ for substitution maps $-[\boldsymbol{v}/\boldsymbol{x}]$.

The fine-grain style has several advantages when studying meta-theoretical properties of calculi: there is a neat distinction between computations and values (which gives, for instance, a smooth definition of the notion of an applicative (bi)simulation relation), and proofs are more streamlined. However, calculi in fine-grain style has the drawback of being cumbersome for writing examples. For instance, the identity combinator $\lambda x.x$ in fine-grain style is written as $\mathbf{return}\ (\lambda x.(\mathbf{return}\ x))$. For this reason, it is sometimes convenient to write examples using the standard, coarse-grain λ-calculus syntax:

$$e ::= x \mid \mathbf{true} \mid \mathbf{false} \mid \mathbf{if}\ e\ \mathbf{then}\ e\ \mathbf{else}\ e \mid \langle e, f\rangle \mid \pi_1 e \mid \pi_2 e \mid \lambda x.e \mid ee.$$

It is not hard to convince ourselves that the two presentations are equivalent, as summarised by Table 2.1.

2.2.1 Static Semantics

We endow Λ^{ST} with a static semantics (also called a type system), whose defining rules are in given in Figure 2.2. We use judgments of the form $\Gamma \vdash^{\Lambda} e : \sigma$ for computations, and $\Gamma \vdash^{\mathcal{V}} v : \sigma$ for values. Formally, judgments are given by two ternary relations: the first one, denoted by $\vdash^{\Lambda}$, relating typing environments,

coarse-grain syntax	fine-grain syntax
x	**return** x
true	**return true**
false	**return false**
if e **then** f **else** h	**let** $x = e$ **in if** x **then** f **else** h
$\langle e, f\rangle$	**let** $x = e$ **in** (**let** $y = f$ **in** (**return** $\langle x, y\rangle$))
$\pi_1 e$	**let** $x = e$ **in** (**return proj**$_l$ x)
$\pi_2 e$	**let** $x = e$ **in** (**return proj**$_r$ x)
$\lambda x.e$	**return** $\lambda x.e$
fe	**let** $x = f$ **in** (**let** $y = e$ **in** xy)

Table 2.1: Correspondence between fine-grain and coarse-grain calculi

terms, and types; and the second one, denoted by $\vdash^{\mathcal{V}}$, relating typing environments, values, and types. As it is customary, we write $\Gamma \vdash^{\Lambda} e : \sigma$ in place of $(\Gamma, e, \sigma) \in \vdash^{\Lambda}$, and similarity for values. A typing environment is a finite set $\{x_1 : \sigma_1, \dots, x_n : \sigma_n\}$ of pairs of variables and types assigning to each variable a unique type: that is, if $i \neq j$, then $x_i \neq x_j$. A more convenient definition of typing environments is given in terms of partial functions.

Definition 2. *A typing environment is a a partial function* Γ *from variables to types such that the domain* $\mathsf{dom}(\Gamma) \triangleq \{x \mid \Gamma(x) \neq \bot\}$ *of* Γ *is finite.*

Given a typing environment Γ, we use the notation $x_1 : \sigma_1, \dots, x_n : \sigma_n$ to denote it, where $\{x_1, \dots, x_n\} = \mathsf{dom}(\Gamma)$ and $\Gamma(x_i) = \mathit{just}\ \sigma_i$. Moreover, we write $(x : \sigma) \in \Gamma$ to mean that $\Gamma(x) = \mathit{just}\ \sigma$. We denote by $\cdot$ the totally undefined environment. Notice that whenever $\Gamma \vdash^{\Lambda} e : \sigma$ is derivable using the rules in Figure 2.2, then $FV(e) \subseteq \mathsf{dom}(\Gamma)$ (and similarity for values). In particular, if $\Gamma = \cdot$, then e is a program. As it is customary, we often write $e : \sigma$ (resp. $v : \sigma$) in place of $\cdot \vdash^{\Lambda} e : \sigma$ (resp. $\cdot \vdash^{\mathcal{V}} v : \sigma$). We also need to join typing environments together. To do that, we need environments to be disjoint. The typing environment disjointness relation $\perp$ is defined thus:

$$\Gamma \perp \Delta \overset{\triangle}{\iff} \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$$

The union Γ, Δ of two disjoint typing environments Γ and Δ is defined in the usual way:

$$(\Gamma, \Delta)(x) \triangleq \begin{cases} \Gamma(x) & \text{if } x \in \mathsf{dom}(\Gamma) \\ \Delta(x) & \text{otherwise.} \end{cases}$$

From now, whenever we write Γ, Δ, we assume Γ and Δ to be disjoint. We extend the vector notation to typing environments, so that we write $\boldsymbol{x} : \boldsymbol{\sigma}$ for $x_1 : \sigma_1, \dots, x_n : \sigma_n$. Finally, we introduce the following notation:

$$\begin{array}{lll} \Lambda_\sigma \triangleq \{e \mid \exists \Gamma.\, \Gamma \vdash^{\Lambda} e : \sigma\} & \Lambda_{\Gamma \vdash \sigma} \triangleq \{e \mid \Gamma \vdash^{\Lambda} e : \sigma\} & \Lambda^{\bullet}_\sigma \triangleq \{e \mid \cdot \vdash^{\Lambda} e : \sigma\} \\ \mathcal{V}_\sigma \triangleq \{v \mid \exists \Gamma.\, \Gamma \vdash^{\mathcal{V}} v : \sigma\} & \mathcal{V}_{\Gamma \vdash \sigma} \triangleq \{v \mid \Gamma \vdash^{\mathcal{V}} v : \sigma\} & \mathcal{V}^{\bullet}_\sigma \triangleq \{v \mid \cdot \vdash^{\mathcal{V}} v : \sigma\} \end{array}$$

Moreover, for $\mathcal{X} \in \{\Lambda, \mathcal{V}, \Lambda^\bullet, \mathcal{V}^\bullet\}$, we define $\mathcal{X} \triangleq \bigcup_\sigma \mathcal{X}_\sigma$.

Exercise 2. Show that the weakening rules are admissible.[1]

$$\frac{\Gamma \vdash^{\Lambda} e : \sigma}{\Gamma, \Delta \vdash^{\Lambda} e : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{\Gamma, \Delta \vdash^{\mathcal{V}} v : \sigma}$$

To deal smoothly with substitutions, it is convenient to extend the vector notation to tying judgments as follows: we write $\Gamma \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}$ for $\forall i.\ \Gamma \vdash^{\mathcal{V}} v_i : \sigma_i$; and $\boldsymbol{\Gamma} \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}$ for $\forall i.\ \Gamma_i \vdash^{\mathcal{V}} v_i : \sigma_i$ (we will use the latter notation mostly when dealing with program distances). Notice, in particular, that the following rules are admissible:

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}}{\Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}}{\Delta \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

[1]Recall that whenever we write Γ, Δ, for two environments Γ and Δ we tacitly assume $\Gamma \perp \Delta$.

$$\frac{}{\Gamma, x : \sigma \vdash^{\mathcal{V}} x : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{\Gamma \vdash^{\Lambda} \textbf{return}\ v : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} e_1 : \sigma_1 \quad \Gamma, x : \sigma_1 \vdash^{\Lambda} e_2 : \sigma_2}{\Gamma \vdash^{\Lambda} \textbf{let}\ x = e_1\ \textbf{in}\ e_2 : \sigma_2}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} \textbf{true} : \textbf{bool}} \qquad \frac{}{\Gamma \vdash^{\mathcal{V}} \textbf{false} : \textbf{bool}} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \textbf{bool} \quad \Gamma \vdash^{\Lambda} e : \sigma \quad \Gamma \vdash^{\Lambda} f : \sigma}{\Gamma \vdash^{\Lambda} \textbf{if}\ v\ \textbf{then}\ e\ \textbf{else}\ f : \sigma}$$

$$\frac{\Gamma, x : \sigma_1 \vdash^{\Lambda} e : \sigma_2}{\Gamma \vdash^{\mathcal{V}} \lambda x.e : \sigma_1 \to \sigma_2} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v_1 : \sigma_1 \to \sigma_2 \quad \Gamma \vdash^{\mathcal{V}} v_2 : \sigma_2}{\Gamma \vdash^{\Lambda} v_1 v_2 : \sigma_2}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} \langle\rangle : \textbf{unit}} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \vdash^{\mathcal{V}} w : \tau}{\Gamma \vdash^{\mathcal{V}} \langle v, w\rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \textbf{proj}_l\ v : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \textbf{proj}_r\ v : \tau}$$

Figure 2.2: Static semantics of Λ^{ST}.

Exercise 3 (Substitution Lemma)**.** Prove that the following rules are admissible (*Hint.* Use the weakening lemma)

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \perp \Delta}{\Gamma, \Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \perp \Delta}{\Gamma, \Delta \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

Notice, however, that these rules are not equivalent to the one for substitution given above. In fact, instantiating, e.g., Γ as Δ we got stuck because $\Delta \perp \Delta$ never holds. Using a proof-theoretical vocabulary, we have problems with contraction. To overcome this issue, we introduce another operation between typing environment, which we will extensively use in the next chapter in the context of substructural type systems.

Definition 3. *1. Define the consistency relation between typing environments thus:*

$$\Gamma \sim \Delta \overset{\Delta}{\iff} x \in \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) \implies \Gamma(x) = \Delta(x).$$

2. The sum operation + is defined between consistent typing environments as follows:

$$(\Gamma + \Delta)(x) \triangleq \begin{cases} \Gamma(x) & \textit{if } x \notin \mathsf{dom}(\Delta) \\ \Delta(x) & \textit{otherwise.} \end{cases}$$

Show that:

1. The following laws hold:

$$\Gamma \sim \Gamma$$
$$\Gamma + \Gamma = \Gamma$$
$$\Gamma \perp \Delta \implies \Gamma \sim \Delta$$
$$\Gamma \perp \Delta \implies \Gamma + \Delta = \Gamma, \Delta$$

Can you spot the link between the identity $\Gamma + \Gamma = \Gamma$ and the structural rule of contraction?

2. Show that the following weakening rules are admissible:

$$\frac{\Gamma \vdash^{\Lambda} e : \sigma \quad \Gamma \sim \Delta}{\Gamma + \Delta \vdash^{\Lambda} e : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \sim \Delta}{\Gamma + \Delta \vdash^{\mathcal{V}} v : \sigma}$$

3. The following substitution rules are admissible

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \sim \Delta}{\Gamma + \Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \sim \Delta}{\Gamma + \Delta \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

4. Concludes that for the rules in previous point are equivalent with:

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}}{\Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}}{\Delta \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

5. Give a type system for Λ^{ST} the replaces the disjoint union of typing environments with the operation $+$. For instance, in such a system the typing for sequencing is (where we tacitly assume $\Gamma \sim \Delta$):

$$\frac{\Gamma \vdash^{\Lambda} e : \sigma \quad \Delta, x : \sigma \vdash^{\Lambda} f : \tau}{\Gamma + \Delta \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f}$$

   Type systems based on the operation $+$ are called *multiplicative*, whereas type systems based on disjoint union are called *additive*. Is there any difference (from the point of view of typability) between the additive and multiplicative type systems of Λ^{ST}?

2.2.2 Dynamic Semantics

The dynamic semantics of Λ^{ST} is given by an evaluation procedure that executes (well-typed) closed computations. Since the execution of a computation may not terminate, thus returning no value, we will define the evaluation procedure as a *partial* function.[2]

Recall that given two sets A, B, a partial function $\varphi : A \rightharpoonup B$ from A to B is a function $\varphi : A \to B_{\perp}$, where $B_{\perp} \triangleq \{just\ b \mid b \in B\} \cup \{\perp\}$, with $\perp$ denoting the 'undefined'. In particular, given $a \in A$, if $\varphi(a) = just\ b$, for some $b \in B$, then φ is said to be defined on a. Dually, if $\varphi(a) = \perp$, then we say that φ is undefined on a. Partial function can be ordered relying on the relation $\sqsubseteq$ defined by:

$$\varphi \sqsubseteq \psi \overset{\triangle}{\iff} \forall a \in A.\ (\varphi(a) = \perp \vee \varphi(a) = \psi(a).$$

Once endowed with the order $\sqsubseteq$, the collection of partial functions from a set A to a set B forms a ω-complete pointed partial order (ω**Cppo**, for short), that is, an ordered set with a bottom element and having all least upper bounds of ω-chains. Here the bottom element is given by the constant function $bot : a \mapsto \perp$ and, given a $\sqsubseteq$-ordered sequence $(\varphi_n)_{n \geq 0}$, the sequence has a (necessary unique) least upper bound $\bigsqcup_{n \geq 0} \varphi_n$ defined by:

$$(\bigsqcup_{n \geq 0} \varphi_n)(a) \triangleq \begin{cases} just\ b & \text{if } \exists i \geq 0.\ \varphi_i(a) = just\ b \\ \perp & \text{otherwise.} \end{cases}$$

Notice that if there there exists $i \geq 0$ such that $\varphi_i(a) = just\ b$, then for all $j \geq 0$ such that $\varphi_j(a) \neq \perp$ we have $\varphi_j(a) = just\ b$. In particular, $\varphi_{i+c}(a) = just\ b$, for any $c \geq 0$.

We can now come back to the definition of our evaluation procedure. Formally, we would like to define such a procedure as the *least* type-indexed family of partial functions $\varphi : \prod_{\sigma} \Lambda^{\bullet}_{\sigma} \rightharpoonup \mathcal{V}^{\bullet}_{\sigma}$ satisfying the following identities:[3]

$$\begin{aligned}
\varphi(\mathbf{return}\ v) &= just\ v \\
\varphi(\mathbf{if\ true\ then}\ e\ \mathbf{else}\ f) &= \varphi(e) \\
\varphi(\mathbf{if\ false\ then}\ e\ \mathbf{else}\ f) &= \varphi(f) \\
\varphi(\mathbf{proj}_l\ \langle v, w\rangle) &= just\ v \\
\varphi(\mathbf{proj}_r\ \langle v, w\rangle) &= just\ w \\
\varphi((\lambda x.e)v) &= \varphi(e[v/x]) \\
\varphi(\mathbf{let}\ x = e\ \mathbf{in}\ f) &= \begin{cases} \varphi(f[v/x]) & \text{if } \varphi(e) = just\ v \\ \perp & \text{otherwise.} \end{cases}
\end{aligned}$$

[2] Actually, since Λ^{ST} is a simply-typed calculus, program evaluation always terminate. Nonetheless, it is convenient to define the evaluation procedure in full generality, and only then observe that such a procedure indeed gives a total function.

[3]To improve readability, we omit type subscripts: that is, we write φ in place of φ_{σ}.

Obviously, there is no guarantee that such a least function does exist. To prove its existence, we define a family of approximate evaluation functions that evaluate programs for a given number of steps only, and then form an ω-chain out of such approximate evaluation functions. The least upper bound of such a chain indeed gives our desired evaluation function.

Definition 4. *The $\mathbb{N}$-indexed family of type-indexed family of functions $[\![-]\!]^n_{\mathcal{E}} : \prod_\sigma \Lambda^\bullet_\sigma \rightharpoonup \mathcal{V}^\bullet_\sigma$ is inductively defined as follows:*[4]

$$
\begin{aligned}
[\![e]\!]^0_{\mathcal{E}} &\triangleq \bot \\
[\![\mathbf{return}\ v]\!]^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
[\![\mathbf{if\ true\ then}\ e\ \mathbf{else}\ f]\!]^{n+1}_{\mathcal{E}} &= [\![e]\!]^n_{\mathcal{E}} \\
[\![\mathbf{if\ false\ then}\ e\ \mathbf{else}\ f]\!]^{n+1}_{\mathcal{E}} &= [\![f]\!]^n_{\mathcal{E}} \\
[\![\mathbf{proj}_l\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &= \mathit{just}\ v \\
[\![\mathbf{proj}_r\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &= \mathit{just}\ w \\
[\![(\lambda x.e)v]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]^{n+1}_{\mathcal{E}} &\triangleq \begin{cases} [\![f[v/x]]\!]^n_{\mathcal{E}} & \textit{if } [\![e]\!]^n_{\mathcal{E}} = \mathit{just}\ v \\ \bot & \textit{otherwise.} \end{cases}
\end{aligned}
$$

Notice that the index n in $[\![-]\!]^n_{\mathcal{E}}$ does not actually refer to the number of β-reductions performed.

Lemma 1. *For each type σ, the sequence of maps $([\![-]\!]^n_{\mathcal{E}})_{n\geq 0}$ forms an ω-chain in $\Lambda^\bullet_\sigma \rightharpoonup \mathcal{V}^\bullet_\sigma$.*

Exercise 4. Prove Lemma 1.

By Lemma 1, we define the evaluation map $[\![-]\!]_{\mathcal{E}} : \prod_\sigma \Lambda^\bullet_\sigma \rightharpoonup \mathcal{V}^\bullet_\sigma$ as $\bigsqcup_{n\geq 0} [\![-]\!]^n_{\mathcal{E}}$.

Exercise 5. Show that the following identities hold, and that $[\![-]\!]_{\mathcal{E}}$ is the least map satisfying them.

$$
\begin{aligned}
[\![\mathbf{return}\ v]\!]_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
[\![\mathbf{if\ true\ then}\ e\ \mathbf{else}\ f]\!]_{\mathcal{E}} &= [\![e]\!]_{\mathcal{E}} \\
[\![\mathbf{if\ false\ then}\ e\ \mathbf{else}\ f]\!]_{\mathcal{E}} &= [\![f]\!]_{\mathcal{E}} \\
[\![\mathbf{proj}_l\ \langle v, w\rangle]\!]_{\mathcal{E}} &= \mathit{just}\ v \\
[\![\mathbf{proj}_r\ \langle v, w\rangle]\!]_{\mathcal{E}} &= \mathit{just}\ w \\
[\![(\lambda x.e)v]\!]_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]_{\mathcal{E}} \\
[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]_{\mathcal{E}} &\triangleq \begin{cases} [\![f[v/x]]\!]_{\mathcal{E}} & \text{if } [\![e]\!]_{\mathcal{E}} = \mathit{just}\ v \\ \bot & \text{otherwise.} \end{cases}
\end{aligned}
$$

Notice by its very functional nature, $[\![-]\!]_{\mathcal{E}}$ gives a *deterministic* notion of evaluation. Additionally, since Λ^{ST} is a simply-typed calculus, a standard reducibility argument shows that Λ^{ST} is strongly normalising.

Proposition 1. *For any program $e \in \Lambda^\bullet_\sigma$, there exists a number $n \geq 0$ such that $[\![e]\!]_{\mathcal{E}} = [\![e]\!]^n_{\mathcal{E}} = \mathit{just}\ v$, for some closed value $v \in \mathcal{V}^\bullet_\sigma$.*

Thanks to previous proposition, $[\![-]\!]_{\mathcal{E}}$ defines total map, so that we write v in place of *just* v whenever we have $[\![e]\!]_{\mathcal{E}} = v$, for a program e and a closed value v.

2.3 Relational Calculus

Studying notions of program equivalence and refinement, it is useful to fix a proper notation and vocabulary for program relations. We do so following Gordon (1994); Lassen (1998). Relational reasoning about programs oftentimes requires to reason about open terms. It is thus useful to work with families of relations relating expressions typable within the same sequent. We refer to such (families of) relations as *term relations*.

[4]As usual, we omit type subscripts.

Definition 5. 1. *A* closed *term relation is a pair $R = (R^\Lambda, R^\mathcal{V})$ of maps associating to each type σ relations R^Λ_σ and $R^\mathcal{V}_\sigma$ on closed computations and values of type σ, respectively. We refer to R^Λ as the computation component of R, and to $R^\mathcal{V}$ as the value component of R.*

2. *An (*open*) term relation R associates to each sequent $\Gamma \vdash \sigma$ a relation $\Gamma \vdash^\Lambda - R - : \sigma$ on $\Lambda_{\Gamma \vdash^\Lambda \sigma}$ and a relation $\Gamma \vdash^\mathcal{V} - R -$ on $\mathcal{V}_{\Gamma \vdash^\mathcal{V} \sigma}$. We require term relations to be closed under weakening:*

$$\frac{\Gamma \vdash^\Lambda e \, R \, f : \tau}{\Gamma, \Delta \vdash^\Lambda e \, R \, f : \tau} \qquad \frac{\Gamma \vdash^\mathcal{V} e \, R \, w : \tau}{\Gamma, \Delta \vdash^\mathcal{V} v \, R \, w : \tau}$$

Example 1. Both the discrete/identity relation I and the indiscrete relation ∇ defined by the rules below are open term relations. The empty relation is an open term relation too.

$$\frac{\Gamma \vdash^\Lambda e : \sigma}{\Gamma \vdash^\Lambda e \, \mathsf{I} \, e : \sigma} \qquad \frac{\Gamma \vdash^\mathcal{V} v : \sigma}{\Gamma \vdash^\mathcal{V} v \, \mathsf{I} \, v : \sigma} \qquad \frac{\Gamma \vdash^\Lambda e, f : \sigma}{\Gamma \vdash^\Lambda e \, \nabla \, f : \sigma} \qquad \frac{\Gamma \vdash^\mathcal{V} v, w : \sigma}{\Gamma \vdash^\mathcal{V} v \, \nabla \, w : \sigma}$$

Since $\Gamma \vdash^\Lambda e : \sigma$ (resp. $\Gamma \vdash^\Lambda v : \sigma$) implies $\Gamma, \Delta \vdash^\Lambda e : \sigma$ (resp. $\Gamma, \Delta \vdash^\Lambda v : \sigma$), for any finite set of variables Δ, both ∇ and I are closed under weakening. ⊠

We extend the vector notation to term relations. We write:

- $\boldsymbol{e} \, R^\Lambda_{\boldsymbol{\sigma}} \, \boldsymbol{f}$ for $\forall i. \, e_i \, R^\Lambda_{\sigma_i} \, f_i$ (and similarity for values).
- $\Gamma \vdash^\Lambda \boldsymbol{e} \, R \, \boldsymbol{f} : \boldsymbol{\sigma}$ for $\forall i. \, \Gamma \vdash^\Lambda e_i \, R \, f_i : \sigma_i$ (and similarity for values).
- $\boldsymbol{\Gamma} \vdash^\Lambda \boldsymbol{e} \, R \, \boldsymbol{f} : \boldsymbol{\sigma}$ for $\forall i. \, \Gamma_i \vdash^\Lambda e_i \, R \, f_i : \sigma_i$ (and similarity for values).

We denote by Rel and Rel^c the collections of open and closed term relations, respectively. Formally, we define Rel as $\prod_{\Gamma \vdash \sigma} \mathsf{Rel}(\Lambda_{\Gamma \vdash \sigma}, \Lambda_{\Gamma \vdash \sigma}) \times \mathsf{Rel}(\mathcal{V}_{\Gamma \vdash \sigma}, \mathcal{V}_{\Gamma \vdash \sigma})$. Rel inherits a rich structure from $\mathsf{Rel}(\Lambda_{\Gamma \vdash \sigma}, \Lambda_{\Gamma \vdash \sigma})$ and $\mathsf{Rel}(\mathcal{V}_{\Gamma \vdash \sigma}, \mathcal{V}_{\Gamma \vdash \sigma})$, as witnessed by the following results.

Lemma 2. Rel *is a complete lattice.*

Proof. Since $\mathsf{Rel} = \prod_{\Gamma \vdash \sigma} \mathsf{Rel}(\Lambda_{\Gamma \vdash \sigma}, \Lambda_{\Gamma \vdash \sigma}) \times \mathsf{Rel}(\mathcal{V}_{\Gamma \vdash \sigma}, \mathcal{V}_{\Gamma \vdash \sigma})$, the thesis follows because $\mathsf{Rel}(\Lambda_{\Gamma \vdash \sigma}, \Lambda_{\Gamma \vdash \sigma})$ and $\mathsf{Rel}(\mathcal{V}_{\Gamma \vdash \sigma}, \mathcal{V}_{\Gamma \vdash \sigma})$ are complete lattices, for any sequent $\Gamma \vdash \sigma$ (recall that the countable product of complete lattices is a complete lattice). □

Notice that Lemma 2 allows us to define term relations both inductively and coinductively.

Exercise 6. Show that the countable product of complete lattices is a complete lattice. Use your proof to write explicitly the complete lattice structure of Rel. In particular show that $R \subseteq S$ if:

$$\forall \Gamma, e, f, \sigma. \, \Gamma \vdash^\Lambda e \, R \, f : \sigma \implies \Gamma \vdash^\Lambda e \, S \, e : \sigma, \qquad \forall \Gamma, v, w, \sigma \; \Gamma \vdash^\mathcal{V} v \, R \, w : \sigma \implies \Gamma \vdash^\mathcal{V} v \, S \, w : \sigma.$$

and that the bottom element of Rel is the empty relation, whereas its top element is the indiscrete relation.

Given term relations R and S, we define the composition of R with S, denoted by $R; S$, as:

$$\frac{\Gamma \vdash^\Lambda e \, R \, h : \sigma \quad \Gamma \vdash^\Lambda h \, S \, f : \sigma}{\Gamma \vdash^\Lambda e \, (R; S) \, f : \sigma} \qquad \frac{\Gamma \vdash^\mathcal{V} v \, R \, u : \sigma \quad \Gamma \vdash^\mathcal{V} u \, S \, w : \sigma}{\Gamma \vdash^\mathcal{V} v \, (R; S) \, w : \sigma}$$

Since both R and S are closed under weakening, then so is $R; S$. It is straightforward to see that composition is associative and that the unit of composition is given by the discrete term relation I, meaning that Rel is a monoid. Actually, easy calculations show that term relation composition is monotone and continuous in both arguments:

$$R; \bigcup_{i \in I} S_i = \bigcup_{i \in I} (R; S_i) \qquad \bigcup_{i \in I} R_i; S = \bigcup_{i \in I} (R_i; S)$$

This makes Rel a quantale. Finally, we observe that Rel also has an involution, mapping a term relation R to its converse $R^{\top}$. The latter is defined as follow:

$$\frac{\Gamma \vdash^{\Lambda} f \mathrel{R} e : \sigma}{\Gamma \vdash^{\Lambda} e \mathrel{R^{\top}} f : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} w \mathrel{R} v : \sigma}{\Gamma \vdash^{\mathcal{V}} v \mathrel{R^{\top}} w : \sigma}$$

Summing up, we see that the algebra of term relations is the same as the one of ordinary relations, meaning that we can transfer notions and results from the realm of relations to the one of open term relations. For instance, we can define the notion of a *preorder* and *equivalence* term relation in the usual way. In fact, we say that a term relation R is reflexive if $\mathsf{I} \subseteq R$, transitive if $R;R \subseteq R$, and symmetric if $R^{\top} \subseteq R$.

Having analysed the algebra of term relations, we now introduce some useful, syntax-oriented constructions on term relations that allow us to build a calculus of term relations, which we will rely on to define notions of program equivalence. The first constructions we define allows us to move back and forth between open and closed term relations.

Definition 6. 1. *Given a term relation $R \in$ Rel, define the* closed restriction $R^{c} = (R^{\Lambda}, R^{\mathcal{V}})$ *of R by:*

$$\frac{\cdot \vdash^{\Lambda} e \mathrel{R} f : \sigma}{e \mathrel{(R^{c})^{\Lambda}_{\sigma}} f} \qquad \frac{\cdot \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma}{v \mathrel{(R^{c})^{\mathcal{V}}_{\sigma}} w}$$

2. *Given a closed term relation R we define the* open extension *of R as the (open) term relation R^{o} defined thus:*

$$\frac{\forall \boldsymbol{v} : \boldsymbol{\sigma}.\ e[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} f[\boldsymbol{v}/\boldsymbol{x}]}{\boldsymbol{x} : \boldsymbol{\sigma} \vdash^{\Lambda} e \mathrel{R^{o}} f : \sigma} \qquad \frac{\forall \boldsymbol{v} : \boldsymbol{\sigma}.\ v[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} w[\boldsymbol{v}/\boldsymbol{x}]}{\boldsymbol{x} : \boldsymbol{\sigma} \vdash^{\mathcal{V}} v \mathrel{R^{o}} w : \sigma}$$

3. *Given a closed term relation R we define the* substitutive extension *of R as the (open) term relation R^{s} defined thus:*

$$\frac{\forall \boldsymbol{v}, \boldsymbol{w} : \boldsymbol{\sigma}.\ (\boldsymbol{v} \mathrel{R^{\mathcal{V}}_{\boldsymbol{\sigma}}} \boldsymbol{w} \implies e[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} f[\boldsymbol{w}/\boldsymbol{x}])}{\boldsymbol{x} : \boldsymbol{\sigma} \vdash^{\Lambda} e \mathrel{R^{s}} f : \sigma} \qquad \frac{\forall \boldsymbol{v}, \boldsymbol{w} : \boldsymbol{\sigma}.\ (\boldsymbol{v} \mathrel{R^{\mathcal{V}}_{\boldsymbol{\sigma}}} \boldsymbol{w} \implies v[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} w[\boldsymbol{w}/\boldsymbol{x}])}{\boldsymbol{x} : \boldsymbol{\sigma} \vdash^{\mathcal{V}} v \mathrel{R^{s}} w : \sigma}$$

Taking advantage of Definition 6, for a *closed* term relation $R = (R^{\Lambda}, R^{\mathcal{V}})$ we will often write $\cdot \vdash^{\Lambda} e R f : \sigma$ in place of $e \mathrel{R^{\Lambda}_{\sigma}} f$ (and similarity for values). In light of that, for an open term relation R we will use the notations $\cdot \vdash^{\Lambda} e \mathrel{R} f : \sigma$ and $e \mathrel{(R^{c})^{\Lambda}_{\sigma}} f : \sigma$ interchangeably (and similarity for values).

Lemma 3. *Let R, S be open term relations, and Q, P be closed term relations. The following monotonicity laws hold.*

$$R \subseteq S \implies R^{c} \subseteq S^{c}$$
$$Q \subseteq P \implies Q^{o} \subseteq P^{o}$$

Moreover, we have the inclusions:

$$R^{c}; S^{c} \subseteq (R;S)^{c}$$
$$Q^{o}; P^{o} \subseteq (Q;P)^{o}$$
$$Q^{s}; P^{s} \subseteq (Q;P)^{s}$$

Exercise 7. Why $-^{s}$ is not monotone? Does $(Q;P)^{o} \subseteq Q^{o};P^{o}$ hold?

Exercise 8. Show that the open extension of a reflexive closed term relation is reflexive. Show also that the open extension of a transitive closed term relation is transitive. What goes wrong if you try to prove the same result for substitutive extensions?

Next we define the useful operation of term relation substitution.

Definition 7. *Given term relations R, S, define the* substitution of S into R *as the term relation $R[S]$ thus defined:*

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Gamma \vdash^{\Lambda} e \mathrel{R} f : \sigma \qquad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} \mathrel{S} \boldsymbol{w} : \boldsymbol{\sigma}}{\Gamma \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R[S]} f[\boldsymbol{w}/\boldsymbol{x}] : \sigma}$$

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \qquad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} \mathrel{S} \boldsymbol{w} : \boldsymbol{\sigma}}{\Gamma \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R[S]} w[\boldsymbol{w}/\boldsymbol{x}] : \sigma}$$

Exercise 9. Show that the defining rules of $R[S]$ in Definition 42 are equivalent to the following ones.

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Gamma \vdash^{\Lambda} e \mathrel{R} f : \sigma \qquad \Gamma \vdash^{\mathcal{V}} \boldsymbol{v} \mathrel{S} \boldsymbol{w} : \boldsymbol{\sigma}}{\Delta, \Gamma \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R[S]} f[\boldsymbol{w}/\boldsymbol{x}] : \sigma}$$

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \qquad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} \mathrel{S} \boldsymbol{w} : \boldsymbol{\sigma}}{\Delta, \Gamma \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R[S]} w[\boldsymbol{w}/\boldsymbol{x}] : \sigma}$$

Exercise 10. Show that

$$R \subseteq R[S]$$

for all term relations R, S (*Hint*: consider the empty substitution).

Lemma 4. *Let R, S, Q, P be term relations. Then, the following monotonicity law holds.*

$$R \subseteq Q, S \subseteq P \implies R[S] \subseteq Q[P]$$

Moreover, we have the following identities.

$$\mathsf{I}[\mathsf{I}] = \mathsf{I}$$
$$R^{\top}[S^{\top}] = (R[S])^{\top}$$
$$(R; Q)[S; P] = R[S]; Q[P]$$

Using term relation substitution, we can define the notions of a substitutive and value substitutive term relation.

Definition 8. 1. *A term relation R is* value-substitutive *if $R[\mathsf{I}] \subseteq R$. Pointwise, a term relation is value substitutive if:*

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e \mathrel{R} f : \sigma \qquad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}}{\Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} f[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e \mathrel{R} f : \sigma \qquad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma}}{\Delta \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} w[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

2. *A term relation R is* substitutive *if $R[R] \subseteq R$. Pointwise:*

$$\frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e \mathrel{R} f : \sigma \qquad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} \mathrel{R} \boldsymbol{w} : \boldsymbol{\sigma}}{\Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} f[\boldsymbol{w}/\boldsymbol{x}] : \sigma} \qquad \frac{\boldsymbol{x} : \boldsymbol{\sigma}, \Delta \vdash^{\Lambda} e \mathrel{R} f : \sigma \qquad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} \mathrel{R} \boldsymbol{w} : \boldsymbol{\sigma}}{\Delta \vdash^{\mathcal{V}} v[\boldsymbol{v}/\boldsymbol{x}] \mathrel{R} w[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

A closed λ-relation is substitutive if its open extension is. (Notice that for value-substitution the open extension of a closed λ-term relation is trivially value-substitutive.)

Exercise 11. Let R be a closed term relation. Show that the open extension of R is value substitutive and that the substitutive extension of R is substitutive:

$$R^{\circ}[\mathsf{I}] \subseteq R^{\circ}$$
$$R^{\mathsf{s}}[R^{\mathsf{s}}] \subseteq R^{\mathsf{s}}$$

In order to define the notion of a (pre)congruence term relation, we introduce the notion of *compatibility*. Roughly speaking, a term relation is compatible if it is preserved by all Λ^{ST} syntactic constructors, i.e. if it closed under Λ^{ST}-context. Giving a formally precise definition of the notion of a context, however, is painful. We overcome that issue relying on the (relational) calculus of term relations and introducing the notion of a compatible refinement (Gordon, 1994; Lassen, 1998).

$$\frac{}{\Gamma, x : \sigma \vdash^{\mathcal{V}} x \mathrel{\widehat{R}} x : \sigma}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} \mathbf{true} \mathrel{\widehat{R}} \mathbf{true} : \mathbf{bool}} \qquad \frac{}{\Gamma \vdash^{\mathcal{V}} \mathbf{false} \mathrel{\widehat{R}} \mathbf{false} : \mathbf{bool}}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \mathbf{bool} \quad \Gamma \vdash^{\Lambda} e \mathrel{R} f : \sigma \quad \Gamma \vdash^{\Lambda} g \mathrel{R} h : \sigma}{\Gamma \vdash^{\Lambda} \mathbf{if}\ v\ \mathbf{then}\ e\ \mathbf{else}\ g \mathrel{\widehat{R}} \mathbf{if}\ w\ \mathbf{then}\ f\ \mathbf{else}\ h : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} u : \sigma \quad \Gamma \vdash^{\mathcal{V}} w \mathrel{R} z : \tau}{\Gamma \vdash^{\mathcal{V}} \langle v, w\rangle \mathrel{\widehat{R}} \langle u, z\rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \mathbf{proj}_l\ v \mathrel{\widehat{R}} \mathbf{proj}_l\ w : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \mathbf{proj}_r\ v \mathrel{\widehat{R}} \mathbf{proj}_r\ w : \tau}$$

$$\frac{\Gamma, x : \sigma \vdash^{\Lambda} e \mathrel{R} f : \tau}{\Gamma \vdash^{\mathcal{V}} \lambda x.e \mathrel{\widehat{R}} \lambda x.f : \sigma \to \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} v' : \sigma \to \tau \quad \Gamma \vdash^{\mathcal{V}} w \mathrel{R} w' : \sigma}{\Gamma \vdash^{\Lambda} vw \mathrel{\widehat{R}} v'w' : \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma}{\Gamma \vdash^{\Lambda} \mathbf{return}\ v \mathrel{\widehat{R}} \mathbf{return}\ w : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} e \mathrel{R} f : \sigma \quad \Gamma, x : \sigma \vdash^{\Lambda} g \mathrel{R} h : \tau}{\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ g \mathrel{\widehat{R}} \mathbf{let}\ x = f\ \mathbf{in}\ h : \tau}$$

Figure 2.3: Compatible refinement Λ^{ST}.

Definition 9. *The* compatible refinement $\widehat{R}$ *of an open term relation R is defined by the rules in Figure 2.3. We say R is compatible if $\widehat{R} \subseteq R$, and that a closed term relation is compatible if its open extension is.*

Notice that $\widehat{R}$ is indeed a term relation (notably, $\widehat{R}$ is closed under weakening).

Lemma 5. *The following hold:*

$$\widehat{R;S} = \widehat{R};\widehat{S}$$
$$\widehat{R^{\top}} = (\widehat{R})^{\top}$$
$$R \subseteq S \implies \widehat{R} \subseteq \widehat{S}.$$

Definition 45 induces a map $R \mapsto \widehat{R}$ on the collection of open term relations which is monotone, by Lemma 5. In particular, a term relation is compatible if and only if it is a pre-fixed point of $R \mapsto \widehat{R}$.

Lemma 6. *The discrete term relation I is the least pre-fixed point of $R \mapsto \widehat{R}$. I.e. $\mathsf{I} \subseteq \widehat{\mathsf{I}}$ and $R \subseteq \widehat{R} \implies \mathsf{I} \subseteq R$. As a consequence, any compatible relation is reflexive.*

Exercise 12. Show that the indiscrete term relation ∇ is the greatest fixed point of $R \mapsto \widehat{R}$.

Exercise 13. Show that for any closed term relation R we have:[5]

$$\widehat{R^{\mathrm{s}}} \subseteq \mathsf{I}[R^{\mathrm{s}}]$$

Do we really need to restrict to R^{s}? That is, does

$$\widehat{S} \subseteq \mathsf{I}[S]$$

hold, for arbitrary term relations S?

In the next section, we will extensively work with compatible relations. For that reason, it is useful to notice that the collection of compatible term relations form a complete lattice, meaning that we can define compatible term relations both inductively and coinductively.

[5]We will use this result to prove that logical equivalence.

Lemma 7. *The collection of compatible λ-term relations forms a complete lattice ordered by $\subseteq$.*

Proof. Given a set ρ of compatible relations we define the meet of ρ as $\bigcap \rho$. We cannot define the join of ρ as $\bigcup \rho$, since the union of compatible λ-relations is not necessarily compatible. We get round the problem by defining the join of ρ as

$$\bigcap \{S \mid \widehat{S} \subseteq S, \bigcup \rho \subseteq S\}.$$

It is easy to see that the latter indeed satisfies the universal property of the join. Finally, we observe that the bottom element is given by the discrete λ-term relation I, whereas the top element is the indiscrete λ-term relation 0. $\square$

Finally, we rely on Definition 45 to define a closure operator mapping a term relation to the least compatible term relation extending it.

Definition 10. *The* compatible closure R^{cc} *of an term relation is inductively defined by the following rules.*

$$\frac{\Gamma \vdash^{\Lambda} e \mathrel{R} f : \sigma}{\Gamma \vdash^{\Lambda} e \mathrel{R^{\mathrm{cc}}} f : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma}{\Gamma \vdash^{\mathcal{V}} v \mathrel{R^{\mathrm{cc}}} w : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} e \mathrel{\widehat{R^{\mathrm{cc}}}} f : \sigma}{\Gamma \vdash^{\Lambda} e \mathrel{R^{\mathrm{cc}}} f : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{\widehat{R^{\mathrm{cc}}}} w : \sigma}{\Gamma \vdash^{\mathcal{V}} v \mathrel{R^{\mathrm{cc}}} w : \sigma}$$

It is easy to see that $-^{\mathrm{cc}}$ is a closure operator, i.e. it is a monotone and idempotent map extending the identity function. Moreover, R^{cc} is the least compatible term relation containing R.

Lemma 8. *The following hold:*

$$\begin{aligned} R \subseteq S &\implies R^{\mathrm{cc}} \subseteq S^{\mathrm{cc}} \\ R &\subseteq R^{\mathrm{cc}} \\ (R^{\mathrm{cc}})^{\mathrm{cc}} &= R^{\mathrm{cc}} \\ \widehat{S} \subseteq S \,\&\, R \subseteq S &\implies R^{\mathrm{cc}} \subseteq S \end{aligned}$$

We also have the inclusion $\mathsf{I}[R] \subseteq R^{\mathrm{cc}}$.

2.4 Denotational and Contextual Equivalences and Refinements

The universally accepted notion of operational equivalence (resp. preorder) for sequential, higher-order language is Morris' style *contextual equivalence* (reps. preorder) (Morris, 1969). The latter is a syntax directed notion of equivalence (resp. preorder) equating (resp. ordering) programs according to a given notion of observation (mostly based on notions of convergence): two programs are deemed as contextually equivalent if there is no contexts of the language (the latter being a kind of program with a hole to be filled in with the tested program) capable of detecting differences in the operational behaviour of the two programs, according to the notion of observation given. For our purposes, it is convenient to stipulate that we can observe the value to which a program of type **bool** evaluates to. We can thus propose the following definition.

Definition 11. *A context is a term C with a hole $[-]$. We write $C[e]$ (resp. $C[v]$) for the expression obtained by replacing the hole $[-]$ with e (resp. v) in C. We write $C : (\Gamma \vdash^{\mathsf{x}} \sigma) \implies (\Delta \vdash^{\mathsf{y}} \tau)$, with $\mathsf{x}, \mathsf{y} \in \{\mathcal{V}, \Lambda\}$, to state that one we fill-in C with an expression ϕ such that $\Gamma \vdash^{\mathsf{x}} \phi : \sigma$, we have $\Delta \vdash^{\mathsf{y}} C[\phi] : \tau$.*

1. Contextual equivalence *is the term relation $\simeq^{\mathrm{ctx}}$ defined thus:*

$$\Gamma \vdash^{\Lambda} e \simeq^{\mathrm{ctx}} f : \sigma \overset{\triangle}{\iff} \forall C : (\Gamma \vdash^{\Lambda}: \sigma) \implies (\cdot \vdash^{\Lambda} \textbf{bool}).\ [\![C[e]]\!]_{\mathcal{E}} = [\![C[f]]\!]_{\mathcal{E}}$$
$$\Gamma \vdash^{\mathcal{V}} v \simeq^{\mathrm{ctx}} w : \sigma \overset{\triangle}{\iff} \forall C : (\Gamma \vdash^{\mathcal{V}}: \sigma) \implies (\cdot \vdash^{\Lambda} \textbf{bool}).\ [\![C[v]]\!]_{\mathcal{E}} = [\![C[w]]\!]_{\mathcal{E}}.$$

Contextual approximation/preorder *is the term relation $\preceq^{\mathrm{ctx}}$ defined thus:*

$$\Gamma \vdash^{\Lambda} e \preceq^{\mathrm{ctx}} f : \sigma \overset{\triangle}{\iff} \forall C : (\Gamma \vdash^{\Lambda}: \sigma) \implies (\cdot \vdash^{\Lambda} \textbf{bool}).\ [\![C[e]]\!]_{\mathcal{E}} = [\![C[f]]\!]_{\mathcal{E}}$$
$$\Gamma \vdash^{\mathcal{V}} v \preceq^{\mathrm{ctx}} w : \sigma \overset{\triangle}{\iff} \forall C : (\Gamma \vdash^{\mathcal{V}}: \sigma) \implies (\cdot \vdash^{\Lambda} \textbf{bool}).\ [\![C[v]]\!]_{\mathcal{E}} = [\![C[w]]\!]_{\mathcal{E}}$$

The problem with the definition relies on the very notion of a context: first, contexts cannot be simply defined as expressions with a hole, as we would like a context C to bind variables of e in $C[e]$. Additionally, we should define when a judgment of the form $C : (\Gamma \vdash^{\mathsf{x}} \sigma) \implies (\Delta \vdash^{\mathsf{y}} \tau)$ is provable. All of this can be done, but at the price of introducing some syntactical bureaucracy. Here, instead, we give a syntax-free characterisation of contextual equivalence (resp. preorder) as the largest adequate (preadeqaute) compatible term relation.

Usually, a term relation is adequate (resp. preadequate) if it only relates (observable) programs exhibiting the same operational behaviour (resp. the operational behaviour of the second refines the operational behaviour of the first). In this setting, (pre)adequacy predicates can be used in place of notions of observation. That is, instead of dealing with an explicit notion of observation, we fix a predicate $\mathsf{ADEQ} \subseteq \mathsf{Rel}$ on term relations with the idea that if a term relation R belongs to ADEQ, then it does not relate programs which are observationally distinguishable. For instance, in the pure, untyped λ-calculus one is usually interested in observing convergence of a term (according to a given reduction strategy (Plotkin, 1975)). In that case, a relation is said to be adequate if whenever it relates two programs, then one of the programs converges if and only so does the other.

The notion of (pre)adeqaucy we focus on here relates programs that converge to the same Boolean value

Definition 12. *1. A term relation R is* adequate *if $\cdot \vdash^{\Lambda} e \mathrel{R} f : \mathbf{bool}$ implies $[\![e]\!]_{\mathcal{E}} = [\![f]\!]_{\mathcal{E}}$ (so that $[\![e]\!]_{\mathcal{E}} = \mathbf{true}$ if and only if $[\![f]\!]_{\mathcal{E}} = \mathbf{true}$).*

2. A term relation is preadequate *if $\cdot \vdash^{\Lambda} e \mathrel{R} f : \mathbf{bool}$ implies $[\![e]\!]_{\mathcal{E}} \leq [\![f]\!]_{\mathcal{E}}$ (i.e. $[\![e]\!]_{\mathcal{E}} = \mathbf{true}$ implies $[\![f]\!]_{\mathcal{E}} = \mathbf{true}$).*

Contextual equivalence and approximations are defined as the largest adequate and preadequate compatible term relations, respectively. We have already seen that the collection of compatible term relations forms a complete lattice. However, we soon realise that (pre)adequacy is not a 'monotone property', and thus we cannot appeal to the Knaster-Tarski Theorem to infer the existence of the largest adequate precongruence. We thus prove the existence of the desired relation explicitly.

Lemma 9. *Let $\alpha \subseteq \mathsf{Rel}$ be a predicate on term relations closed under non-empty union and relation composition (i.e. $\bigcup_{i \in I} R_i \in \alpha$, whenever $I \neq \emptyset$ and $R_i \in \alpha$ for any i; and $R; S \in \alpha$ whenever $R \in \alpha$ and $S \in \alpha$). If $\mathsf{I} \in \alpha$, then there exists a largest compatible term relation $S \in \alpha$. Additionally, S is transitive and it is symmetric if α is closed under relation transpose, (i.e. $R \in \alpha \implies R^{\top} \in \alpha$).*

Proof. Define the term relation S as

$$S \triangleq \bigcup \{R \in \alpha \mid \widehat{R} \subseteq R\}.$$

By hypothesis $\mathsf{I} \in \alpha$, so that $\{R \in \alpha \mid \widehat{R} \subseteq R\}$ is non-empty. As a consequence, since α is closed under non-empty union, $S \in \alpha$. To conclude the first part of the thesis it remains to prove $\widehat{S} \subseteq S$. We prove by simultaneous induction on the definition of $\widehat{S}$ the following statements:

1. If $\Gamma \vdash^{\Lambda} e \mathrel{\widehat{S}} f : \sigma$, then $\Gamma \vdash^{\Lambda} e \mathrel{S} f : \sigma$.
2. If $\Gamma \vdash^{\mathcal{V}} v \mathrel{\widehat{S}} w : \sigma$, then $\Gamma \vdash^{\mathcal{V}} v \mathrel{S} w : \sigma$.

We prove the case of rule (comp-let) as an illustrative example. Suppose that:

$$\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f \mathrel{\widehat{S}} \mathbf{let}\ x = g\ \mathbf{in}\ h : \sigma$$

then we have $\Gamma \vdash^{\Lambda} e \mathrel{S} g : \tau$ and $\Gamma, x : \tau \vdash^{\Lambda} f \mathrel{S} h : \sigma$. By very definition of S there exist compatible term relations $P, Q \in \alpha$ such that $\Gamma \vdash^{\Lambda} e P g : \tau$ and $\Gamma, x : \tau \vdash^{\Lambda} f Q h : \sigma$. Since both P and Q are compatible, then so is $P; Q$ (recall that $\widehat{A; B} = \widehat{A}; \widehat{B}$, for all term relations A, B: therefore, $\widehat{P; Q} = \widehat{P}; \widehat{Q} \subseteq P; Q$). Moreover, since α is closed under composition, then $P; Q \in \alpha$. We also notice that, since compatibility implies reflexivity, we have $\Gamma \vdash^{\Lambda} e\ (P; Q)\ g : \tau$ and $\Gamma, x : \tau \vdash^{\Lambda} f\ (P; Q)\ h : \sigma$. In fact, $P = P; \mathsf{I} \subseteq P; \widehat{Q} \subseteq P; Q$. Therefore

$$\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f\ \widehat{(P; Q)}\ \mathbf{let}\ x = g\ \mathbf{in}\ h : \sigma,$$

and thus by compatibility of $P;Q$

$$\Gamma \vdash^{\Lambda} \textbf{let } x = e \textbf{ in } f \ (P;Q) \ \textbf{let } x = g \textbf{ in } h : \tau.$$

Since $P;Q$ is compatible and belong to α, we conclude $\Gamma \vdash^{\Lambda} \textbf{let } x = e \textbf{ in } f \ S \ \textbf{let } x = g \textbf{ in } h : \tau$.
We now prove that S is transitive, i.e. $S;S \subseteq S$. Since S is the largest compatible relation that belongs to α, it is enough to show that $S;S$ is compatible and belongs to α. The latter follows since α is closed under composition, whereas the former follows by compatibility of S:

$$\widehat{S;S} = \widehat{S};\widehat{S} \subseteq S;S.$$

In a similar fashion one shows that if α is closed under transpose, then $S^{\top} \subseteq S$, meaning that S is symmetric. □

Lemma 10. *The adequacy and preadequacy properties*

$$\mathsf{ADEQ} \triangleq \{R \in \mathsf{Rel} \mid \cdot \vdash^{\Lambda} e \, R \, f : \textbf{bool} \implies [\![e]\!]_{\mathcal{E}} = [\![f]\!]_{\mathcal{E}}\}$$
$$\mathsf{PREADEQ} \triangleq \{R \in \mathsf{Rel} \mid \cdot \vdash^{\Lambda} e \, R \, f : \textbf{bool} \implies ([\![e]\!]_{\mathcal{E}} = \textbf{true} \implies [\![f]\!]_{\mathcal{E}} = \textbf{true})\}$$

contain the discrete term relation I*, and are closed under non-empty union and composition. Moreover,* ADEQ *is closed under transpose.*

Exercise 14. Prove Lemma 10.

We are now ready to define contextual approximation and equivalence.

Definition 13. 1. *Define the term relation* $\simeq^{\text{ctx}}$*, called* contextual equivalence*, as the largest compatible and adequate term relation.*

2. *Define the term relation* $\precsim^{\text{ctx}}$*, called* contextual approximation*, as the largest compatible and preadequate term relation.*

Notice that $\simeq^{\text{ctx}}$ is compatible, reflexive, transitive, and symmetric (and thus a congruence), whereas $\precsim^{\text{ctx}}$ is compatible, reflexive, and transitive (and thus a precongruence).

The reader might have noticed that the definition of ADEQ and PREADEQ involve the behaviour of term relations on programs only, and does not say anything about their behaviour on values. This is rather obvious, as adequacy is an operational notion based on the evaluation semantics of a program. Formally, this does *not* mean that on values $\simeq^{\text{ctx}}$ and $\precsim^{\text{ctx}}$ coincide with the indiscrete relation. In fact, compatibility forces $\simeq^{\text{ctx}}$ and $\precsim^{\text{ctx}}$ to relate only those values that behave appropriately when used inside computations. For instance, if $\lambda x.e \ (\simeq^{\text{ctx}})^{\mathcal{V}}_{\sigma \to \textbf{bool}} \ \lambda x.f$, then by compatibility (and reflexivity) of $\simeq^{\text{ctx}}$, we have

$$(\lambda x.e)v \ (\simeq^{\text{ctx}})^{\Lambda}_{\textbf{bool}} \ (\lambda x.f)v$$

for any value $v \in \mathcal{V}^{\bullet}_{\sigma}$, which gives $[\![e[v/x]]\!]_{\mathcal{E}} = [\![f[v/x]]\!]_{\mathcal{E}}$. Obviously, this cannot be the case for all e, f (take, e.g., $\lambda x.\textbf{true}$ and $\lambda x.\textbf{false}$).

Definition 13 comes with associated proof techniques resembling a coinduction proof principle. To prove that a program e is contextual equivalent to a program f, it is sufficient to exhibit a compatible and adequate term relation relating e and f.

$$\frac{\widehat{R} \subseteq R \quad R \in \mathsf{ADEQ}}{R \subseteq \simeq^{\text{ctx}}} \qquad \frac{\widehat{R} \subseteq R \quad R \in \mathsf{PREADEQ}}{R \subseteq \precsim^{\text{ctx}}}$$

We can use this proof technique to prove that $\precsim^{\text{ctx}}$ is actually a precongruence relation.

Exercise 15. Show that $\simeq^{\text{ctx}} = \precsim^{\text{ctx}} \cap (\precsim^{\text{ctx}})^{\top}$ (*Hint.* Use the proof technique associated with $\precsim^{\text{ctx}}$ and $\simeq^{\text{ctx}}$).

2.4.1 Denotational Equality

Contextual equivalence is defined relying on the operational semantics of Λ^{ST}. For this reason, one says that contextual equivalence is a *operationally-based equivalence* (or just *operational equivalence*, for short). Other examples of operationally-based equivalences are logical relations, applicative bisimilarity, Böhm tree-like equivalences, and CIU equivalences. Although in these notes we mostly deal with operational equivalences, interesting notions of program equivalence can be obtained via other forms of semantics, such as *axiomatic* or *denotational* semantics. In particular, any denotational semantics $[\![-]\!]_{\mathcal{D}}$ induces a notion of equivalence $\overset{\text{D}}{\simeq}$, called *denotational equivalence*, equating two expressions if they have the same denotation: $e \overset{\text{D}}{\simeq} f$ if and only if $[\![e]\!]_{\mathcal{D}} = [\![f]\!]_{\mathcal{D}}$.

Program equivalence is thus reduced to mathematical equality, this way giving a compatible notion of equivalence for free. Denotational equivalence, however, may not respect the operational properties of the calculus. The notion of *adequacy* precisely describes those denotational equivalences respecting the operational behaviour of the calculus.

Definition 14. *We say that* $\overset{\text{D}}{\simeq}$ *is* adequate *if* $\overset{\text{D}}{\simeq} \subseteq \simeq^{\text{ctx}}$. *If we also have* $\simeq^{\text{ctx}} \subseteq \overset{\text{D}}{\simeq}$, *we say that* $\overset{\text{D}}{\simeq}$ *is* fully abstract.

The main drawback of denotational equivalences is that giving a denotational semantics to rich calculi may be non-trivial. Additionally, even if a denotational semantics has been given, the latter may not be robust with respect to language extensions: the addition of new features to a language often leads to drastic changes in the associated denotational semantics. Nonetheless, denotational equivalences constitute a powerful tool to reason about program equivalence. For this reason, in this section we shortly outline how to give a simple, set-based denotational semantics to Λ^{ST}, that gives an adequate (and actually fully abstract) denotational equivalence.

Before proceeding any further, however, we remark that program equivalence and denotational semantics are two sides of the same coin. As any denotational semantics induces a notion of program equivalence, any notion of program equivalence induces denotational semantics by means of equivalence classes: accordingly, the denotational meaning of an expression is the equivalence class of that expression.[6]

We now define a denotational semantics for Λ^{ST}. In general, we should define a value semantics $[\![\sigma]\!]^{\mathcal{V}}_{\mathcal{D}}$ and term semantics $[\![\sigma]\!]^{\Lambda}_{\mathcal{D}}$, for any type σ, and stipulate the denotational semantics of value judgments $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\mathcal{V}} v : \sigma$ and of computation judgments $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\Lambda} e : \sigma$ to be arrows

$$[\![x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\mathcal{V}} v : \sigma]\!]_{\mathcal{D}} : [\![\sigma_1]\!]^{\mathcal{V}}_{\mathcal{D}} \times \cdots \times [\![\sigma_n]\!]^{\mathcal{V}}_{\mathcal{D}} \to [\![\sigma]\!]^{\mathcal{V}}_{\mathcal{D}}$$
$$[\![x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\Lambda} e : \sigma]\!]_{\mathcal{D}} : [\![\sigma_1]\!]^{\mathcal{V}}_{\mathcal{D}} \times \cdots \times [\![\sigma_n]\!]^{\mathcal{V}}_{\mathcal{D}} \to [\![\sigma]\!]^{\Lambda}_{\mathcal{D}}$$

in categories with enough structures, respectively. This idea leads to the abstract notion of a Freyd category. Since Λ^{ST} is strongly normalising, we find ourselves in a particularly simple case where we can stipulate the denotation of a type to be just a set and make $[\![\sigma]\!]^{\Lambda}_{\mathcal{D}}$ coincide with $[\![\sigma]\!]^{\mathcal{V}}_{\mathcal{D}}$.[7]

Definition 15. *For any type* σ, *define the set*[8] $[\![\sigma]\!]_{\mathcal{D}}$ *recursively as follows:*

$$\begin{aligned}
[\![\textbf{unit}]\!]_{\mathcal{D}} &\triangleq 1 = \{*\} \\
[\![\textbf{bool}]\!]_{\mathcal{D}} &\triangleq 2 = \{\mathit{false}, \mathit{true}\} \\
[\![\sigma \times \tau]\!]_{\mathcal{D}} &\triangleq [\![\sigma]\!]_{\mathcal{D}} \times [\![\tau]\!]_{\mathcal{D}} \\
[\![\sigma \to \tau]\!]_{\mathcal{D}} &\triangleq [\![\tau]\!]_{\mathcal{D}}^{[\![\sigma]\!]_{\mathcal{D}}}.
\end{aligned}$$

To define the denotational semantics of Λ^{ST} it is useful to recall the basic set-theoretic combinators making the category of sets and functions cartesian closed. Given maps $\alpha : C \to A$, $\beta : C \to B$, we

[6]That is like to say that the meaning of a word is the collection of its synonyms. Although that may appear circular at first, it is worth noticing that giving synonyms is a standard practice in natural languages to learn the meaning of unknown words.

[7] If, for instance, one considers a calculus with effects, then she may want to rely on a strong monad T and to define $[\![\sigma]\!]^{\Lambda}_{\mathcal{D}}$ as $T[\![\sigma]\!]^{\mathcal{V}}_{\mathcal{D}}$.

[8] Since $[\![\sigma]\!]^{\mathcal{V}}_{\mathcal{D}} = [\![\sigma]\!]^{\Lambda}_{\mathcal{D}}$, we omit superscripts and just write $[\![\sigma]\!]_{\mathcal{D}}$.

$$\begin{aligned}
\llbracket x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\mathcal{V}} x_i : \sigma_i \rrbracket_{\mathcal{D}} &= \pi_i : \llbracket \sigma_1 \rrbracket_{\mathcal{D}} \cdots \llbracket \sigma_n \rrbracket_{\mathcal{D}} \to \llbracket \sigma_i \rrbracket_{\mathcal{D}} \\
\llbracket \Gamma \vdash^{\mathcal{V}} \langle\rangle : \mathbf{unit} \rrbracket_{\mathcal{D}} &\triangleq !_{\llbracket \Gamma \rrbracket_{\mathcal{D}}} : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to 1 \\
\llbracket \Gamma \vdash^{\mathcal{V}} \langle v, w \rangle : \sigma \times \tau \rrbracket_{\mathcal{D}} &\triangleq \langle \llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \rrbracket_{\mathcal{D}}, \llbracket \Gamma \vdash^{\mathcal{V}} w : \tau \rrbracket_{\mathcal{D}} \rangle \\
\llbracket \Gamma \vdash^{\Lambda} \mathbf{proj}_l \; v : \sigma \rrbracket_{\mathcal{D}} &\triangleq \pi_1 \circ \llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau \rrbracket_{\mathcal{D}} \\
\llbracket \Gamma \vdash^{\Lambda} \mathbf{proj}_r \; v : \sigma \rrbracket_{\mathcal{D}} &\triangleq \pi_2 \circ \llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau \rrbracket_{\mathcal{D}} \\
\llbracket \Gamma \vdash^{\mathcal{V}} \mathbf{true} : \mathbf{bool} \rrbracket_{\mathcal{D}} &\triangleq \overline{\mathit{true}} \circ !_{\llbracket \Gamma \rrbracket_{\mathcal{D}}} : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to 2 \\
\llbracket \Gamma \vdash^{\mathcal{V}} \mathbf{false} : \mathbf{bool} \rrbracket_{\mathcal{D}} &\triangleq \overline{\mathit{false}} \circ !_{\llbracket \Gamma \rrbracket_{\mathcal{D}}} : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to 2 \\
\llbracket \Gamma \vdash^{\Lambda} \mathbf{if}\ v\ \mathbf{then}\ e\ \mathbf{else}\ f : \sigma \rrbracket_{\mathcal{D}} &\triangleq \mathit{ite} \circ \langle \llbracket \Gamma \vdash^{\mathcal{V}} v : \mathbf{bool} \rrbracket_{\mathcal{D}}, \llbracket \Gamma \vdash^{\Lambda} e : \sigma \rrbracket_{\mathcal{D}}, \llbracket \Gamma \vdash^{\Lambda} f : \sigma \rrbracket_{\mathcal{D}} \rangle : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to \llbracket \sigma \rrbracket_{\mathcal{D}} \\
\llbracket \Gamma \vdash^{\mathcal{V}} \lambda x.e : \sigma \to \tau \rrbracket_{\mathcal{D}} &\triangleq \Lambda \llbracket \Gamma, x : \sigma \vdash^{\Lambda} e : \rho \rrbracket_{\mathcal{D}} : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to \llbracket \tau \rrbracket_{\mathcal{D}}^{\llbracket \sigma \rrbracket_{\mathcal{D}}} \\
\llbracket \Gamma \vdash^{\Lambda} vw : \tau \rrbracket_{\mathcal{D}} &\triangleq \mathit{eval} \circ \langle \llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \to \tau \rrbracket_{\mathcal{D}}, \llbracket \Gamma \vdash^{\mathcal{V}} w : \sigma \rrbracket_{\mathcal{D}} \rangle : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to \llbracket \tau \rrbracket_{\mathcal{D}} \\
\llbracket \Gamma \vdash^{\Lambda} \mathbf{return}\ v : \sigma \rrbracket_{\mathcal{D}} &\triangleq \llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \rrbracket_{\mathcal{D}} : \llbracket \Gamma \rrbracket_{\mathcal{D}} \to \llbracket \sigma \rrbracket_{\mathcal{D}} \\
\llbracket \Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f : \sigma \rrbracket_{\mathcal{D}} &\triangleq \llbracket \Gamma, x : \tau \vdash^{\Lambda} f : \sigma \rrbracket_{\mathcal{D}} \circ \langle 1_{\llbracket \Gamma \rrbracket_{\mathcal{D}}, \llbracket \Gamma \vdash^{\Lambda} e : \tau \rrbracket_{\mathcal{D}}} \rangle
\end{aligned}$$

Figure 2.4: Denotational Semantics of Λ^{ST}

denote by $\langle \alpha, \beta \rangle : C \to A \times B$ the function defined by $\langle \alpha, \beta \rangle(c) = (\alpha(c), \beta(c))$, whereas we denote by $\pi_i : A_1 \times A_n \to A_i$ the i-th projection function. The currying of a function $\alpha : C \times A \to B$ is the function $\Lambda(\alpha) : C \to B^A$ defined by $\Lambda(\alpha)(c)(a) \triangleq \alpha(c, a)$, whereas we write $\mathit{eval} : A \times B^A \to B$ for the function mapping the pair (a, α) to $\alpha(a)$. For a set A, we write $!_A : A \to 1$ for the unique map sending each $a \in A$ to $*$. Using such a map, we associate to each element $a \in A$ the function $\bar{a} : 1 \to A$ mapping $*$ to a. Finally, we write $\mathit{ite} : 2 \times A \times A \to A$ for the if-then-else function mapping a triple (t, a, b) to a, if $x = \mathit{true}$, and to b, otherwise.

Definition 16. *The denotational semantics of value judgments $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\mathcal{V}} v : \sigma$ and of computation judgments $x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\Lambda} e : \sigma$ is given by the functions*

$$\llbracket x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\mathcal{V}} v : \sigma \rrbracket_{\mathcal{D}} : \llbracket \sigma_1 \rrbracket_{\mathcal{D}} \times \cdots \times \llbracket \sigma_n \rrbracket_{\mathcal{D}} \to \llbracket \sigma \rrbracket_{\mathcal{D}}$$
$$\llbracket x_1 : \sigma_1, \ldots, x_n : \sigma_n \vdash^{\Lambda} e : \sigma \rrbracket_{\mathcal{D}} : \llbracket \sigma_1 \rrbracket_{\mathcal{D}} \times \cdots \times \llbracket \sigma_n \rrbracket_{\mathcal{D}} \to \llbracket \sigma \rrbracket_{\mathcal{D}}$$

recursively defined by the rules in Figure 2.4

Finally, we define denotational equivalence as the term relation $\overset{\mathrm{D}}{\simeq}$ defined by:

$$\Gamma \vdash^{\mathcal{V}} v \overset{\mathrm{D}}{\simeq} w : \sigma \overset{\triangle}{\iff} \llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \rrbracket_{\mathcal{D}} = \llbracket \Gamma \vdash^{\mathcal{V}} w : \sigma \rrbracket_{\mathcal{D}}$$
$$\Gamma \vdash^{\Lambda} v \overset{\mathrm{D}}{\simeq} w : \sigma \overset{\triangle}{\iff} \llbracket \Gamma \vdash^{\Lambda} v : \sigma \rrbracket_{\mathcal{D}} = \llbracket \Gamma \vdash^{\Lambda} w : \sigma \rrbracket_{\mathcal{D}}.$$

Exercise 16. Show that $\overset{\mathrm{D}}{\simeq}$ is a congruence.

Exercise 17. Show that $\llbracket e \rrbracket_{\mathcal{E}} = \llbracket f \rrbracket_{\mathcal{E}}$ implies $\cdot \vdash^{\Lambda} e \overset{\mathrm{D}}{\simeq} f : \sigma$.

Theorem 1. $\overset{\mathrm{D}}{\simeq} = \simeq^{\mathrm{ctx}}$.

Even in the simple setting of Λ^{ST}, the proof of Theorem 1 requires some non-trivial techniques (viz. a denotational version of the notion of a logical relations we will see in the next section) which are outside the scope of these notes.

2.5 Logical Relations

In this section, we define logical relations. Logical relations are closed term relations satisfying certain properties that qualifies them as candidate notions of program equality (and approximation) for a given calculus. When defining contextual equivalence we focused on the *structural* properties — viz. being adequate, compatible, reflexive, etc. — and defined contextual equivalence as a canonical term relation for them. Here, we take a different perspective and associate to any type a relation between program of that type such that it reflects the logical structure of types and, in principle, it does not need to satisfy any structural property.

Historically, logical relations have been defined as a *relational semantics* for types. In that setting, one interprets each type σ as a set $[\![\sigma]\!]_{\mathcal{E}}$ together with a relation $R_\sigma : [\![\sigma]\!]_{\mathcal{E}} \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} [\![\sigma]\!]_{\mathcal{E}}$. To do so, one defines for any type constructor F (such as the arrow type constructor $\to$) a construction $\hat{F}$ on relations so as to define $R_{F(\sigma_1,\ldots,\sigma_n)} : [\![F(\sigma_1,\ldots,\sigma_n)]\!]_{\mathcal{E}} \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} [\![F(\sigma_1,\ldots,\sigma_n)]\!]_{\mathcal{E}}$ as $\hat{F}(R_{\sigma_1},\ldots,R_{\sigma_n})$. The so-called fundamental lemma of logical relations states that open programs act as relational homomorphisms (i.e. as maps preserving logical relations), which amounts to prove that logical relations are reflexive. From such a result it also follows that logical relations are compatible.

Logical relations can also be understood in purely operational terms by giving a term-like relational model (i.e. by taking the semantics of a type as a suitable relation over *expressions* of the type). This approach is the main one followed in the more recent literature on program semantics, and the one we follow too. Accordingly, let us begin by recalling the usual definition of logical relations one finds in the literature.

Definition 17. *The logical equivalence over Λ^{ST} is the* closed *term relation $\overset{\mathsf{L}}{\simeq}$ recursively defined on types by:*

$$\begin{aligned}
\cdot \vdash^{\mathcal{V}} v \overset{\mathsf{L}}{\simeq} w : \mathbf{bool} &\overset{\triangle}{\iff} v = w \\
\cdot \vdash^{\mathcal{V}} v \overset{\mathsf{L}}{\simeq} w : \mathbf{unit} &\overset{\triangle}{\iff} \textit{always} \\
\cdot \vdash^{\mathcal{V}} \langle v, w\rangle \overset{\mathsf{L}}{\simeq} \langle u, z\rangle : \sigma \times \tau &\overset{\triangle}{\iff} \cdot \vdash^{\mathcal{V}} v \overset{\mathsf{L}}{\simeq} u : \sigma \;\&\; \cdot \vdash^{\mathcal{V}} w \overset{\mathsf{L}}{\simeq} z : \tau \\
\cdot \vdash^{\mathcal{V}} \lambda x.e \overset{\mathsf{L}}{\simeq} \lambda x.f : \sigma \to \tau &\overset{\triangle}{\iff} \forall v, w.(\cdot \vdash^{\mathcal{V}} v \overset{\mathsf{L}}{\simeq} w : \sigma \implies \cdot \vdash^{\Lambda} e[v/x] \overset{\mathsf{L}}{\simeq} f[w/x] : \tau) \\
\cdot \vdash^{\Lambda} e \overset{\mathsf{L}}{\simeq} f : \sigma &\overset{\triangle}{\iff} \cdot \vdash^{\mathcal{V}} [\![e]\!]_{\mathcal{E}} \overset{\mathsf{L}}{\simeq} [\![f]\!]_{\mathcal{E}} : \sigma.
\end{aligned}$$

Notice that, contrary to contextual equivalence, Definition 17 defines a *closed* term relation. Its rationale is the following: two computations are logically equivalent if the they evaluate to logically equivalent values, this way taking advantage of strong normalisation of Λ^{ST}. The crucial point in Definition 17 is the definition of $\overset{\mathsf{L}}{\simeq}$ on (closed) values. At ground types, (the value component of) $\overset{\mathsf{L}}{\simeq}$ behaves as syntactic equality, whereas, at more complex types, $\overset{\mathsf{L}}{\simeq}$ hereditary preserves logical equality: in particular, two λ-abstractions are logically equivalent if they map logically equivalent inputs to logically equivalent outputs. Finally, we extend $\overset{\mathsf{L}}{\simeq}$ to an open term relation by taking its substitutive extension. Notice that the choice of using the substitutive extension operator — rather than the open extension one — reflects the definition of $\overset{\mathsf{L}}{\simeq}$ at function types.

Definition 17 makes reasoning about logical equivalence quite bureaucratic and hides the relationship with other notions of equivalence — notably, applicative bisimilarity and contextual equivalence. For those reasons, we take a more relational perspective and defines logical relation relying on our relational calculus. We begin defining some (relational) constructions extending the (logical meaning of) type constructors of Λ^{ST} to relations. In particular, since Λ^{ST} has product and arrow types, we define 'product' and 'arrow' relations.

Definition 18. 1. *Given relations $R : X \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} A$, $S : Y \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} B$, we define the product relation $R \times S : X \times Y \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} A \times B$ by*

$$(x, y)\; R \times S\; (a, b) \iff x \mathrel{R} a \;\&\; y \mathrel{S} b.$$

2. *Given relations $R : X \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} Y$, $S : A \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} B$, we define the relation*[9] *$[R \to S] : A^X \mathrel{\mkern3mu\mapstochar\mkern-3mu\rightarrow} B^Y$ by:*

$$\begin{aligned}
f\,[R \to S]\,g &\overset{\triangle}{\iff} \forall x, y.(x \mathrel{R} y \implies f(x) \mathrel{S} g(y)) \\
&\iff R \subseteq f; S; g^{\top}
\end{aligned}$$

[9] The construction $[\cdot \to \cdot]$ is sometimes called Reynolds arrow.

Lemma 11. *Relational product is monotone in both arguments, whereas Reynolds arrow is monotone in the second argument and anti-monotone in the first argument.*

$$R \subseteq Q \ \&\ S \subseteq P \implies R \times S \subseteq Q \times P$$
$$Q \subseteq R \ \&\ S \subseteq P \implies [R \to S] \subseteq [Q \to P].$$

Moreover, the following laws hold.

$$\begin{aligned} \mathsf{I}_{A\times B} &= \mathsf{I}_A \times \mathsf{I}_B \\ (R \times S)^\top &= R^\top \times S^\top \\ (R \times S); (Q \times P) &= (R;Q) \times (S;P) \end{aligned}$$

$$\begin{aligned} \mathsf{I}_{A\to B} &= [\mathsf{I}_A \to \mathsf{I}_B] \\ [R \to S]^\top &= [R^\top \to S^\top] \\ [R \to S]; [Q \to P] &\subseteq [R;Q \to S;P] \end{aligned}$$

To define logical relations, we need to unpack the logical meaning of types. For instance, the logical meaning of a closed value $\lambda x.f$ of type $\sigma \to \tau$ is to behave as a function that maps values v of type σ to computations $f[v/x]$ of type τ. We can thus say that the logical meaning of $\lambda x.f$ is the function $(v \mapsto f[v/x]) \in \mathcal{V}^\bullet_\sigma \to \Lambda^\bullet_\tau$. The crucial point that defines a logical relation is then the following: assuming to have defined logical relations R_σ, R_τ for expressions of type σ and τ, we define[10] $R^{\mathcal{V}}_{\sigma\to\tau}$ relying on the logical meaning of values in $\mathcal{V}^\bullet_{\sigma\to\tau}$. Since such logical meanings belong to $\mathcal{V}^\bullet_\sigma \to \Lambda^\bullet_\tau$ it is enough to extend R_σ, R_τ to the function space $\mathcal{V}^\bullet_\sigma \to \Lambda^\bullet_\tau$. But Definition 18 precisely tells us how to extend relations to function spaces, so that we can define $R^{\mathcal{V}}_{\sigma\to\tau}$ essentially as $[R^{\mathcal{V}}_\sigma \to R^{\Lambda}_\tau]$. A similar idea applies, *mutatis mutandins*, to product types.

Definition 19. *For each type σ, we define the logical meaning function $[\![-]\!]_{\mathcal{L}} : \mathcal{V}^\bullet_\sigma \to \mathcal{L}(\mathcal{V}^\bullet_\sigma)$, where*

$$\begin{aligned} \mathcal{L}(\mathcal{V}^\bullet_{\mathbf{bool}}) &\triangleq \mathcal{V}_{\mathbf{bool}} \\ \mathcal{L}(\mathcal{V}^\bullet_{\mathbf{unit}}) &\triangleq \mathcal{V}_{\mathbf{unit}} \\ \mathcal{L}(\mathcal{V}^\bullet_{\sigma\times\tau}) &\triangleq \mathcal{V}^\bullet_\sigma \times \mathcal{V}^\bullet_\tau \\ \mathcal{L}(\mathcal{V}^\bullet_{\sigma\to\tau}) &\triangleq \mathcal{V}^\bullet_\sigma \to \Lambda^\bullet_\tau. \end{aligned}$$

by:

$$\begin{aligned} [\![\mathbf{true}]\!]_{\mathcal{L}} &\triangleq \mathbf{true} \\ [\![\mathbf{false}]\!]_{\mathcal{L}} &\triangleq \mathbf{false} \\ [\![\langle\rangle]\!]_{\mathcal{L}} &\triangleq \langle\rangle \\ [\![\langle v, w\rangle]\!]_{\mathcal{L}} &\triangleq (v, w) \\ [\![\lambda x.f]\!]_{\mathcal{L}} &\triangleq v \mapsto f[v/x] \end{aligned}$$

We now have all the ingredients to define logical relations.

Definition 20. *A closed term relation R is a logical relation if:*

$$\begin{aligned} R^{\mathcal{V}}_{\mathbf{bool}} &\subseteq \mathsf{I}^{\mathcal{V}}_{\mathbf{bool}} \\ R^{\mathcal{V}}_{\sigma\times\tau} &\subseteq [\![-]\!]_{\mathcal{L}}; (R^{\mathcal{V}}_\sigma \times R^{\mathcal{V}}_\tau); [\![-]\!]^\top_{\mathcal{L}} \\ R^{\mathcal{V}}_{\sigma\to\tau} &\subseteq [\![-]\!]_{\mathcal{L}}; [R^{\mathcal{V}}_\sigma \to R^{\Lambda}_\tau]; [\![-]\!]^\top_{\mathcal{L}} \\ R^{\Lambda}_\sigma &\subseteq [\![-]\!]_{\mathcal{E}}; R^{\mathcal{V}}_\sigma; [\![-]\!]^\top_{\mathcal{E}}. \end{aligned}$$

[10] Since Λ^{ST} is strongly normalising, it is natural to stipulate that two programs are equivalent if the results of their evaluation are, so that we focus on defining logical relations on values.

$$
\begin{aligned}
[R]^{\mathcal{V}}_{\mathbf{bool}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{bool}}\\
[R]^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (R^{\mathcal{V}}_{\sigma} \times R^{\mathcal{V}}_{\tau}); [\![-]\!]^{\top}_{\mathcal{L}}\\
[R]^{\mathcal{V}}_{\sigma\to\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [R^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}}\\
[R]^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}.
\end{aligned}
$$

Figure 2.5: Logical relation operator

$$
\begin{aligned}
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\mathbf{bool}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{bool}}\\
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}}\\
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma} \times \overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}}\\
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma\to\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma} \to \overset{\mathsf{L}}{\simeq}{}^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}}\\
\overset{\mathsf{L}}{\simeq}{}^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}.
\end{aligned}
$$

Figure 2.6: Logical Equivalence

Examples of logical relations are the empty relations as well as the (closed restriction of the) identity relation. Both these relations, however, are way too fine to qualify as acceptable notions of program equivalence. What we would like to do then, is to define logical equivalence as a *canonical* logical relation. It is thus natural to try to define logical equivalence either by induction or by coinduction. In fact, the right hand-side clauses of Definition 20 define an operator $[-]$ on the complete lattice of closed term relations (see Figure 2.5) so that R is a logical relation if and only if $R \subseteq [R]$. We may thus try to define logical equivalence as the least or greatest fixed point of $[-]$ relying on the Knaster-Tarski Theorem. However, the presence of the Reynolds arrow operation makes $[-]$ non-monotone since $[- \to -]$ is antitone/contravariant in its first argument. This point is crucial: as we will see, applicative bisimilarity considers a restricted version of Reynolds arrow that allows one to restore monotonicity and thus to define relations coinductively.

Definition 17 overcomes the aforementioned problem by giving an explicit definition of logical equivalence taking advantages of the simply typed nature of Λ^{ST}. Rephrased in the relational calculus we obtain the definition of logical equivalence given in Figure 2.6.

Lemma 12. $\overset{\mathsf{L}}{\simeq}$ *is a logical relation.*

We extend $\overset{\mathsf{L}}{\simeq}$ to an open term relation by taking its substitutive extension $\overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}}$. As $\overset{\mathsf{L}}{\simeq}$ satisfies all the desired logical properties of notion of equivalence by definition, then so does $\overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}}$. We now show that $\overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}}$ satisfies all the desired structural properties too. Notice that since the substitutive extensions are always substitutive ($R^{\mathsf{s}}[R^{\mathsf{s}}] \subseteq R^{\mathsf{s}}$ holds for any closed term relation R), $\overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}}$ is trivially substitutive. We now show that it is reflexive, symmetric, transitive, and compatible. Compatibility and transitivity follows from reflexivity. For that reason the result stating that logical equivalence is reflexive is usually called the fundamental lemma of logical relations.

Proposition 2 (Fundamental Lemma). $\mathsf{I} \subseteq \overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}}$.

Proof. We simultaneously prove

$$\forall(\Gamma \vdash^{\Lambda} e : \sigma).\Gamma \vdash^{\Lambda} e \overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}} e : \sigma \tag{2.1}$$

$$\forall(\Gamma \vdash^{\mathcal{V}} v : \sigma).\Gamma \vdash^{\mathcal{V}} v \overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}} v : \sigma \tag{2.2}$$

by induction on the derivation of $\Gamma \vdash^{\Lambda} e : \sigma$ and $\Gamma \vdash^{\mathcal{V}} e : \sigma$. □

Corollary 1. $\overset{\mathsf{L}}{\simeq}{}^{\mathsf{s}}$ *is compatible.*

Proof. We have to show $\widehat{\overset{\llcorner}{\simeq}^{s}} \subseteq \overset{\llcorner}{\simeq}^{s}$. Recall that for any closed term relation R, we have $R^{s}[R^{s}] \subseteq R^{s}$ and $\widehat{R^{s}} \subseteq \mathsf{I}[R^{s}]$. We have:

$$\widehat{\overset{\llcorner}{\simeq}^{s}} \subseteq \mathsf{I}[\overset{\llcorner}{\simeq}^{s}] \subseteq \overset{\llcorner}{\simeq}^{s}[\overset{\llcorner}{\simeq}^{s}] \subseteq \overset{\llcorner}{\simeq}^{s}$$

where the ante-penultimate passage uses Proposition 2. □

Lemma 13. $\overset{\llcorner}{\simeq}^{s}$ *is transitive.*

Proof. Since we have

$$R; R \subseteq R \implies R^{s}; R^{s} \subseteq R^{s}$$

it is sufficient to prove that $\overset{\llcorner}{\simeq}$ is transitive. We prove that by induction on types. First, notice that if $\overset{\llcorner}{\simeq}^{\mathcal{V}}$ is transitive, then so is $\overset{\llcorner}{\simeq}^{\Lambda}$. Indeed, we have:

$$\begin{aligned}
\overset{\llcorner}{\simeq}^{\Lambda}; \overset{\llcorner}{\simeq}^{\Lambda} &= ([\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top}); ([\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top}) \\
&= [\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top}; [\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top} \\
&\subseteq [\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; \mathsf{I}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top} \\
&= [\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top} \\
&\subseteq [\![-]\!]_{\mathcal{E}}; \overset{\llcorner}{\simeq}^{\mathcal{V}}; [\![-]\!]_{\mathcal{E}}^{\top} \\
&= \overset{\llcorner}{\simeq}^{\Lambda}.
\end{aligned}$$

Therefore, it is enough to show that for any type σ, we have $\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}$ assuming both $\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\tau}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\tau} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\tau}$ and $\overset{\llcorner}{\simeq}^{\Lambda}_{\tau}; \overset{\llcorner}{\simeq}^{\Lambda}_{\tau} \subseteq \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}$ to hold for any type τ strictly simpler than σ. The only interesting case is for arrow types. We prove $\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma\to\tau}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma\to\tau} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma\to\tau}$. We have:

$$\begin{aligned}
\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma\to\tau}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma\to\tau} &= [\![-]\!]_{\mathcal{L}}; [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]; [\![-]\!]_{\mathcal{L}}^{\top}; [\![-]\!]_{\mathcal{L}}; [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]; [\![-]\!]_{\mathcal{L}}^{\top} \\
&\subseteq [\![-]\!]_{\mathcal{L}}; [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]; [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]; [\![-]\!]_{\mathcal{L}}^{\top} \\
&\subseteq [\![-]\!]_{\mathcal{L}}; [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}; \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]; [\![-]\!]_{\mathcal{L}}^{\top}
\end{aligned}$$

At this point, we would like to use the induction hypothesis to prove

$$[\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}; \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}] \subseteq [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]$$

this way concluding the desired thesis. Indeed, since $[- \to -]$ is monotone in the second argument, the induction hypothesis gives $[\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}; \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}] \subseteq [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]$. However, since $[- \to -]$ is antitone in the first argument, we cannot rely on $\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}$ to finalise our argument. The problem is overcame using the fundamental lemma: in fact, from $\mathsf{I}^{\mathcal{V}}_{\sigma} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}$ we obtain

$$\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \mathsf{I}^{\mathcal{V}}_{\sigma} \subseteq \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}$$

so that we conclude $[\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma}; \overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}] \subseteq [\overset{\llcorner}{\simeq}^{\mathcal{V}}_{\sigma} \to \overset{\llcorner}{\simeq}^{\Lambda}_{\tau}]$ by antitonocity. □

Exercise 18. Show that $\overset{\llcorner}{\simeq}^{s}$ is symmetric.

Collecting all the results just proved, we see that $\overset{\llcorner}{\simeq}^{s}$ is a congruence. Additionally, it is straightforward to see that it is also adequate. As a consequence, $\overset{\llcorner}{\simeq}^{s}$ is included in $\simeq^{\mathrm{ctx}}$.

Theorem 2. $\overset{\llcorner}{\simeq}^{s}$ *is an adequate congruence. Therefore, we have* $\overset{\llcorner}{\simeq}^{s} \subseteq \simeq^{\mathrm{ctx}}$.

Exercise 19. How would you modify the definition of a logical equivalence to obtain a notion of logical approximation?

Exercise 20 (Logical equality = extensional equality)**.** Recall that the presence of the Reynolds arrow operation does not allow us to define a canonical notion of logical relation in terms of least or greatest fixed point of the map $[-]$. One way to overcome this problem is to replace $[R^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]$ with $[\mathsf{I}^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]$ in the defining clauses of a logical relation. Pointwise, we would have:

$$\cdot \vdash^{\mathcal{V}} \lambda x.e \mathrel{R} \lambda x.f : \sigma \to \tau \implies \forall v \in \mathcal{V}^{\bullet}_{\sigma}.\ \cdot \vdash^{\Lambda} e[v/x] \mathrel{R} f[v/x] : \tau$$

meaning that we compare functions relying on the function extensionality principle. This is one of the core ideas behind the notion of applicative bisimilarity.
1. Show that the map $\langle - \rangle$ obtained from $[-]$ replacing the clause for arrow types with

$$\langle R \rangle^{\mathcal{V}}_{\sigma \to \tau} \triangleq [\![-]\!]_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}}$$

is monotone on the complete lattice of closed term relations.
We define extensional equality $\overset{\text{ext}}{\simeq}$ as the greatest fixed point of $\langle - \rangle$ (what is the least fixed point of $\overset{\text{ext}}{\simeq}$?).

Theorem 3. $\overset{\text{L}}{\simeq} = \overset{\text{ext}}{\simeq}$.

Proof sketch. The difficult part is showing that if two expressions are extensionally equivalent, then they are logically equivalent too. This follows from the so called Reynolds-Abramsky Lemma:

$$\frac{\mathsf{I} \subseteq R \qquad R; R \subseteq R \qquad g[R \to R]g}{f[\mathsf{I} \to R]g \iff f[R \to R]g}$$

Notice that Proposition 2 and Lemma 13 gives exactly the premises of the Reynolds-Abramsky Lemma. □

2. Complete the proof of Theorem 3.

Exercise 21 (Logical equivalence = contextual equivalence)**.** Show that $\overset{\text{L}}{\simeq} = \simeq^{\text{ctx}}$. *Hint.* See Chapter 46 and Chapter 47 in (Harper, 2016).

2.6 Recursive Types and Applicative Bisimilarity

Λ^{ST} is a simply-typed calculus. As such, it provides a foundational formalism for a limited class of programming languages. In this section, we extend Λ^{ST} with recursive and sum types. We denote the resulting calculus by Λ^{μ}. The syntax of Λ^{μ} is given in Figure 2.7, where t denotes a type variable.

Types σ, τ	$::=$	t	Values v, w	$::=$	x	Computations e, f	$::=$	$\mathbf{return}\ v$
		$\mid \mathbf{unit}$			$\mid \langle\rangle$			$\mid \mathbf{proj}_l\ v$
		$\mid \sigma \times \tau$			$\mid \langle v, w \rangle$			$\mid \mathbf{proj}_r\ v$
		$\mid \mathbf{void}$			$\mid \mathbf{inl}\ v$			$\mid \mathbf{abort}\ v$
		$\mid \sigma + \tau$			$\mid \mathbf{inr}\ v$			$\mid \mathbf{case}\ v\ \mathbf{of}\ \{\mathbf{inl}\ x \to e \mid \mathbf{inr}\ x \to f\}$
		$\mid \sigma \to \tau$			$\mid \lambda x.e$			$\mid vw$
		$\mid \mu t.\sigma$			$\mid \mathbf{fold}\ v$			$\mid \mathbf{unfold}\ v$
								$\mid \mathbf{let}\ x = e\ \mathbf{in}\ f$

Figure 2.7: Types, values, and computations of Λ^{μ}.

Free and bound variables in computations and values are defined as for Λ^{ST}. We also adopt the same notational conventions defined for Λ^{ST} and extend such conventions to types. In particular, we denote by $\sigma[\tau/t]$ the result of capture-avoiding substitution Notice also that contrary to Λ^{ST}, Λ^{μ} does not have a type for Boolean values. That is because we can now define **bool** as **unit** + **unit**.

The static semantics of Λ^{μ} is defined relying on judgments $\Gamma \vdash^{\Lambda} e : \sigma$ and $\Gamma \vdash^{\mathcal{V}} v : \sigma$, where Γ is a set $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ with variables pairwise distinct and each type σ_i *closed*; e is computation; v a value; and σ is a *closed* type. Rules for derivable typing judgments are given in Figure 2.8.

$$\frac{}{\Gamma, x : \sigma \vdash^{\mathcal{V}} x : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{\Gamma \vdash^{\Lambda} \textbf{return}\ v : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} e : \tau \quad \Gamma, x : \tau \vdash^{\Lambda} f : \sigma}{\Gamma \vdash^{\Lambda} \textbf{let}\ x = e\ \textbf{in}\ f : \sigma}$$

$$\frac{\Gamma, x : \sigma \vdash^{\Lambda} e : \tau}{\Gamma \vdash^{\mathcal{V}} \lambda x.e : \sigma \to \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \to \tau \quad \Gamma \vdash^{\mathcal{V}} w : \sigma}{\Gamma \vdash^{\Lambda} vw : \tau}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} \langle\rangle : \textbf{unit}} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \vdash^{\mathcal{V}} w : \tau}{\Gamma \vdash^{\mathcal{V}} \langle v, w\rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \textbf{proj}_l\ v : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \textbf{proj}_r\ v : \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \textbf{void}}{\Gamma \vdash^{\Lambda} \textbf{abort}\ v : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{\Gamma \vdash^{\mathcal{V}} \textbf{inl}\ v : \sigma + \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \tau}{\Gamma \vdash^{\mathcal{V}} \textbf{inr}\ v : \sigma + \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma + \tau \quad \Gamma, x : \sigma \vdash^{\Lambda} e : \rho \quad \Gamma, y : \tau \vdash^{\Lambda} f : \rho}{\Gamma \vdash^{\Lambda} \textbf{case}\ v\ \textbf{of}\ \{\textbf{inl}\ x \to e \mid \textbf{inr}\ x \to f\} : \rho}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \sigma[\mu t.\sigma/t]}{\Gamma \vdash^{\mathcal{V}} \textbf{fold}\ v : \mu t.\sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \mu t.\tau}{\Gamma \vdash^{\Lambda} \textbf{unfold}\ v : \sigma[\mu t.\sigma/t]}$$

Figure 2.8: Static semantics of Λ^{μ}.

2.6.1 Dynamic Semantics

We extend the evaluation semantics of Λ^{ST} to Λ^{μ} in the natural way.

Definition 21. *The $\mathbb{N}$-indexed family of functions $[\![-]\!]^n_{\mathcal{E}} : \prod_{\sigma} \Lambda^{\bullet}_{\sigma} \rightharpoonup \mathcal{V}^{\bullet}_{\sigma}$ is inductively defined as follows:*[11]

$$\begin{aligned}
[\![e]\!]^0_{\mathcal{E}} &\triangleq \bot \\
[\![\textbf{return}\ v]\!]^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
[\![\textbf{proj}_l\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
[\![\textbf{proj}_r\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ w \\
[\![\textbf{case inl}\ v\ \textbf{of}\ \{\textbf{inl}\ x \to e \mid \textbf{inr}\ x \to f\}]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\textbf{case inr}\ w\ \textbf{of}\ \{\textbf{inl}\ x \to e \mid \textbf{inr}\ x \to f\}]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![f[w/x]]\!]^n_{\mathcal{E}} \\
[\![\textbf{unfold}\ (\textbf{fold}\ v)]\!]^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
[\![(\lambda x.e)v]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\textbf{let}\ x = e\ \textbf{in}\ f]\!]^{n+1}_{\mathcal{E}} &\triangleq \begin{cases} [\![f[v/x]]\!]^n_{\mathcal{E}} & \textit{if } [\![e]\!]^n_{\mathcal{E}} = \mathit{just}\ v \\ \bot & \textit{otherwise.} \end{cases}
\end{aligned}$$

Lemma 14. *For any type σ, the sequence of maps $([\![-]\!]^n_{\mathcal{E}})_{n \geq 0}$ forms an ω-chain in $\Lambda^{\bullet}_{\sigma} \rightharpoonup \mathcal{V}^{\bullet}_{\sigma}$.*

By Lemma 40, we define the evaluation map $[\![-]\!]_{\mathcal{E}} : \prod_{\sigma} \Lambda^{\bullet}_{\sigma} \rightharpoonup \mathcal{V}^{\bullet}_{\sigma}$ as $\bigsqcup_{n \geq 0} [\![-]\!]^n_{\mathcal{E}}$.

[11]As usual, we omit type subscripts.

Lemma 15. *The following identities hold.*

$$
\begin{aligned}
[\![\textbf{return } v]\!]_{\mathcal{E}} &\triangleq \textit{just } v \\
[\![\textbf{proj}_l\ \langle v, w\rangle]\!]_{\mathcal{E}} &= \textit{just } v \\
[\![\textbf{proj}_r\ \langle v, w\rangle]\!]_{\mathcal{E}} &= \textit{just } w \\
[\![\textbf{case inl } v \textbf{ of } \{\textbf{inl } x \to e \mid \textbf{inr } x \to f\}]\!]_{\mathcal{E}} &= [\![e[v/x]]\!]_{\mathcal{E}} \\
[\![\textbf{case inr } w \textbf{ of } \{\textbf{inl } x \to e \mid \textbf{inr } x \to f\}]\!]_{\mathcal{E}} &= [\![f[w/x]]\!]_{\mathcal{E}} \\
[\![\textbf{unfold } (\textbf{fold } v)]\!]_{\mathcal{E}} &= \textit{just } v \\
[\![(\lambda x.e)v]\!]_{\mathcal{E}} &= [\![e[v/x]]\!]_{\mathcal{E}} \\
[\![\textbf{let } x = e \textbf{ in } f]\!]^{n+1}_{\mathcal{E}} &= \begin{cases} [\![f[v/x]]\!]_{\mathcal{E}} & \textit{if } [\![e]\!]_{\mathcal{E}} = \textit{just } v \\ \bot & \textit{otherwise.} \end{cases}
\end{aligned}
$$

Notice that, as for Λ^{ST}, $[\![-]\!]_{\mathcal{E}}$ gives a *deterministic* notion of evaluation. However, contrary to what happens in Λ^{ST}, now $[\![-]\!]_{\mathcal{E}}$ is a truly partial function. For instance, define

$$
\begin{aligned}
\Omega &\triangleq \omega(\textbf{fold } \omega) \\
\omega &\triangleq \lambda x.\textbf{let } y = \textbf{unfold } x \textbf{ in } y(\textbf{fold } y)
\end{aligned}
$$

Notice that $\cdot \vdash^{\mathcal{V}} \omega : (\mu t.(t \to \sigma)) \to \sigma$, so that $\cdot \vdash^{\Lambda} \Omega : \sigma$.

Exercise 22. Show that $[\![\Omega]\!]_{\mathcal{E}} = \textbf{inr } (\bot)$.

2.6.2 Relational Calculus

The relational calculus developed for Λ^{ST} extends to Λ^{μ} *mutatis mutandis*, the only visible difference appearing in the definition of the compatible refinement operator. The latter is now defined by the rules in Figure 2.9.

2.6.3 Applicative Bisimilarity

To handle the infinitary behaviours introduced by recursive types, we define applicative (bi)similarity. Contrary to contextual equivalence (but similarly to logical equivalence), applicative bisimilarity is defined as a *closed* term relation, and then extended to open term relations using the open extension operation (this way departing from logical equivalence which, instead, relies on the substitutive extension operation).

As usual, we first give a syntax-oriented definition of applicative bisimilarity and then give an equivalent definition based on the relational calculus. The crucial notion to define applicative bisimilarity is the notion of an applicative simulation.

Definition 22. *A closed term relation R is an applicative simulation if it satisfies the following clauses.*

$$
\begin{aligned}
\cdot \vdash^{\mathcal{V}} \langle v, w\rangle \mathrel{R} \langle u, z\rangle : \sigma \times \tau &\implies (\cdot \vdash^{\mathcal{V}} v \mathrel{R} u : \sigma) \,\&\, (\cdot \vdash^{\mathcal{V}} w \mathrel{R} z : \tau) \\
\cdot \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma + \tau &\implies \begin{cases} v = \textbf{inl } u \implies w = \textbf{inl } z \,\&\, \cdot \vdash^{\mathcal{V}} u \mathrel{R} z : \sigma \\ v = \textbf{inr } u \implies w = \textbf{inr } z \,\&\, \cdot \vdash^{\mathcal{V}} u \mathrel{R} z : \tau \end{cases} \\
\cdot \vdash^{\mathcal{V}} \textbf{fold } v \mathrel{R} \textbf{fold } w : \mu t.\sigma &\implies \cdot \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma[t/\mu t.\sigma] \\
\cdot \vdash^{\mathcal{V}} \lambda x.e \mathrel{R} \lambda x.f : \sigma \to \tau &\implies \forall v \in \mathcal{V}^{\bullet}_{\sigma}.\ \cdot \vdash^{\Lambda} e[v/x] \mathrel{R} e[v/x] : \tau \\
\cdot \vdash^{\Lambda} e \mathrel{R} f : \sigma &\implies ([\![e]\!]_{\mathcal{E}} = \textit{just } v \implies [\![f]\!]_{\mathcal{E}} = \textit{just } w \,\&\, \cdot \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma)
\end{aligned}
$$

According to Definition 22, the main differences between an applicative bisimulation and a logical relation concerns the treatment of λ-abstraction and computations. Concerning the latter case, a logical relation requires related computation to give related values when evaluated. This is not possible anymore, since now term evaluation may diverge. We thus require that if the first term converges, then so does the

$$\frac{}{\Gamma, x : \sigma \vdash^{\mathcal{V}} x \mathrel{\widehat{R}} x : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} u : \sigma \quad \Gamma \vdash^{\mathcal{V}} w \mathrel{R} z : \tau}{\Gamma \vdash^{\mathcal{V}} \langle v, w \rangle \mathrel{\widehat{R}} \langle u, z \rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \mathbf{proj}_l\ v \mathrel{\widehat{R}} \mathbf{proj}_l\ w : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \mathbf{proj}_r\ v \mathrel{\widehat{R}} \mathbf{proj}_r\ w : \tau}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} \langle\rangle \mathrel{\widehat{R}} \langle\rangle : \mathbf{unit}}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma}{\Gamma \vdash^{\mathcal{V}} \mathbf{inl}\ v \mathrel{\widehat{R}} \mathbf{inl}\ w : \sigma + \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \tau}{\Gamma \vdash^{\mathcal{V}} \mathbf{inr}\ v \mathrel{\widehat{R}} \mathbf{inr}\ w : \sigma + \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma + \tau \quad \Gamma, x : \sigma \vdash^{\Lambda} e \mathrel{R} g : \rho \quad \Gamma, y : \tau \vdash^{\Lambda} f \mathrel{R} h : \rho}{\Gamma \vdash^{\Lambda} \mathbf{case}\ v\ \mathbf{of}\ \{\mathbf{inl}\ x \to e \mid \mathbf{inr}\ x \to f\} \mathrel{\widehat{R}} \mathbf{case}\ w\ \mathbf{of}\ \{\mathbf{inl}\ x \to g \mid \mathbf{inr}\ x \to h\} : \rho}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \mathbf{void}}{\Gamma \vdash^{\Lambda} \mathbf{abort}\ v \mathrel{\widehat{R}} \mathbf{abort}\ w : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma[t/\mu t.\sigma]}{\Gamma \vdash^{\mathcal{V}} \mathbf{fold}\ v \mathrel{\widehat{R}} \mathbf{fold}\ w : \mu t.\sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \mu t.\sigma}{\Gamma \vdash^{\Lambda} \mathbf{unfold}\ v \mathrel{\widehat{R}} \mathbf{unfold}\ w : \sigma[t/\mu t.\sigma]}$$

$$\frac{\Gamma, x : \sigma \vdash^{\Lambda} e \mathrel{R} f : \tau}{\Gamma \vdash^{\mathcal{V}} \lambda x.e \mathrel{\widehat{R}} \lambda x.f : \sigma \to \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} v' : \sigma \to \tau \quad \Gamma \vdash^{\mathcal{V}} w \mathrel{R} w' : \sigma}{\Gamma \vdash^{\Lambda} vw \mathrel{\widehat{R}} v'w' : \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma}{\Gamma \vdash^{\Lambda} \mathbf{return}\ v \mathrel{\widehat{R}} \mathbf{return}\ w : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} e \mathrel{R} f : \sigma \quad \Gamma, x : \sigma \vdash^{\Lambda} g \mathrel{R} h : \tau}{\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ g \mathrel{\widehat{R}} \mathbf{let}\ x = f\ \mathbf{in}\ h : \tau}$$

Figure 2.9: Compatible refinement Λ^{μ}.

second one and the resulting values must be themselves related. Concerning λ-abstractions, applicative simulation relates two values $\lambda x.e, \lambda x.f$ if they behaves as *extensionally equivalent* functions: for any input value v, $e[v/x]$ and $f[v/x]$ must be related. The shift from logical to extensional equality for functions is crucial, as it allows us to define *applicative similarity*, the *largest* applicative simulation, as the largest fixed point of a suitable operator.

Exercise 23. Define applicative similarity $\overset{a}{\lesssim}$ as the union of all applicative simulation. Show that $\overset{a}{\lesssim}$ is itself an applicative simulation (and thus the largest such).

Applicative similarity is then extended to an open term relation by taking its open extension. Notice that the choice of using the open extension operation reflects the presence of function extensionality in the defining clauses of an applicative simulation. Finally, *applicative bisimilarity* $\overset{a}{\simeq}$ is defined as $\overset{a}{\lesssim}^{\circ} \cap (\overset{a}{\lesssim}^{\circ})^{\top}$.

Exercise 24. Show that for all closed term relations R, S, we have $(R \cap S)^{\circ} = R^{\circ} \cap S^{\circ}$. Conclude that we have $\overset{a}{\simeq} = \overset{a}{\lesssim}^{\circ} \cap (\overset{a}{\lesssim}^{\circ})^{\top} = (\overset{a}{\lesssim} \cap \overset{a}{\lesssim}^{\top})^{\circ}$.

Let us now restore the definition of an applicative simulation in our relational setting. As already observed, the crucial features of its defining clauses are the presence of function extensionality and the computation clause handling possible divergent behaviours. We already know how to model the former in a relational setting using the 'extensional Reynolds arrow' $[\mathsf{I} \to R]$. Notice that such an operation is unary and satisfies the properties given by the following result. In particular, it is *monotone*.

Lemma 16. *The following laws hold.*

$$
\begin{aligned}
\mathsf{I}_{A\to B} &= [\mathsf{I}_A \to \mathsf{I}_B] \\
[\mathsf{I} \to R]^{\top} &= [\mathsf{I} \to R^{\top}] \\
[\mathsf{I} \to R];[\mathsf{I} \to S] &\subseteq [\mathsf{I} \to R;S] \\
R \subseteq S \implies [\mathsf{I} \to R] &\subseteq [\mathsf{I} \to S].
\end{aligned}
$$

Concerning diverge, the situation can be abstractly summarised as follows. We have a set X together with a relation R on it (in our case, the set of values and the value component of an applicative simulation), and we want to extend R to a relation[12] $\widetilde{\mathcal{M}}R$ over $X_{\perp} \times X_{\perp}$ (that is, we want to extend a relation on values to a relation on the results of a computation). Notice that once we have such an extension, we can then rephrase the term component of an applicative simulation as something of the form:

$$\cdot \vdash^{\Lambda} e\, R\, f : \sigma \implies [\![e]\!]_{\mathcal{E}}\, \widetilde{\mathcal{M}}R^{\mathcal{V}}_{\sigma}\, [\![f]\!]_{\mathcal{E}}.$$

Definition 22 precisely gives one possible definition of the operator $\widetilde{\mathcal{M}}$.

Definition 23. *Given a relation $R : X \nrightarrow Y$, the relation $\widetilde{\mathcal{M}}R : X_{\perp} \nrightarrow Y_{\perp}$ is defined by:*

$$t\, \widetilde{\mathcal{M}}R\, s \overset{\triangle}{\iff} (t = \mathit{just}\ x \implies s = \mathit{just}\ y\ \&\ x\, R\, y)$$

Lemma 17. *The map $\widetilde{\mathcal{M}}$ satisfies the following laws.*

$$
\begin{aligned}
\mathsf{I}_{A_{\perp}} &\subseteq \widetilde{\mathcal{M}}\mathsf{I}_A \\
\widetilde{\mathcal{M}}R;\widetilde{\mathcal{M}}S &\subseteq \widetilde{\mathcal{M}}(R;S) \\
R \subseteq S \implies \widetilde{\mathcal{M}}R &\subseteq \widetilde{\mathcal{M}}S.
\end{aligned}
$$

Exercise 25. Rephrase Definition 22 using the construction $[\mathsf{I} \to -]$ and $\widetilde{\mathcal{M}}$.

To give a completely relational definition of an applicative simulation it remains to extend the logical meaning function $[\![-]\!]_{\mathcal{L}}$ to sum and recursive types. That is straightforward.

[12] The construction mapping a set X to the set $X_{\perp}$ is part of a monad sometimes called the *Maybe monad* and denoted by $\mathcal{M}$. That is the reason why we choose the notation $\widetilde{\mathcal{M}}$ for our relational construction.

Definition 24. *Extend the logical meaning function $[\![-]\!]_{\mathcal{L}}$ with the maps*

$$\begin{aligned}
[\![-]\!]_{\mathcal{L}} &: \mathcal{V}^{\bullet}_{\sigma+\tau} \to \mathcal{V}^{\bullet}_{\sigma} + \mathcal{V}^{\bullet}_{\tau} \\
[\![-]\!]_{\mathcal{L}} &: \mathcal{V}^{\bullet}_{\mu t.\sigma} \to \mathcal{V}^{\bullet}_{\sigma[t/\mu t.\sigma]}
\end{aligned}$$

thus defined:

$$\begin{aligned}
[\![\mathbf{inl}\ v]\!]_{\mathcal{L}} &\triangleq in_l(v) \\
[\![\mathbf{inr}\ v]\!]_{\mathcal{L}} &\triangleq in_r(v) \\
[\![\mathbf{fold}\ v]\!]_{\mathcal{L}} &\triangleq v
\end{aligned}$$

Definition 25. *Given relations $R : X \nrightarrow Y$, $S : A \nrightarrow B$, define $R + S : X + A \nrightarrow Y + B$ as follows:*

$$t\ (R+S)\ s \overset{\triangle}{\iff} \begin{cases} t = in_l(x) \implies s = in_l(y)\ \&\ x\ R\ y \\ t = in_r(a) \implies s = in_r(b)\ \&\ a\ S\ b \end{cases}$$

Exercise 26. Show that the following laws hold.

$$\begin{gathered}
\mathsf{I}_{A+B} = \mathsf{I}_A + \mathsf{I}_B \\
(R+S);(Q+P) \subseteq (R;Q) + (S;P) \\
R \subseteq Q\ \&\ S \subseteq P \implies R + S \subseteq Q + P
\end{gathered}$$

We now have all the ingredients to define applicative similarity as the greatest fixed point of a monotone operator.

Definition 26. *Given a closed term relation R, we define the closed term relation $[R]$ as follows:*

$$\begin{aligned}
[R]^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}} \\
[R]^{\mathcal{V}}_{\mathbf{void}} &\triangleq \emptyset \\
[R]^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; R^{\mathcal{V}}_{\sigma} \times R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\
[R]^{\mathcal{V}}_{\sigma+\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; R^{\mathcal{V}}_{\sigma} + R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\
[R]^{\mathcal{V}}_{\mu\mathrm{t}.\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; R^{\mathcal{V}}_{\sigma[\mathrm{t}/\mu\mathrm{t}.\sigma]}; [\![-]\!]^{\top}_{\mathcal{L}} \\
[R]^{\mathcal{V}}_{\sigma\to\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}} \\
R^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \widetilde{\mathcal{M}}R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}
\end{aligned}$$

A term relation R is an applicative simulation *if $R \subseteq [R]$.*

Lemma 18. *The map $[-]$ is a monotone function on the complete lattice of closed relation.*

Proof. All the relational constructions used in Definition 26 are monotone. ☐

As a consequence, we can define applicative similarity as the greatest fixed point of $[-]$.

Proposition 3. $\overset{a}{\lesssim} = \nu R.[R]$.

Exercise 27. Prove Proposition 3. In particular, show that R is a post-fixed point of $[-]$ (i.e. $R \subseteq [R]$) if and only if R satisfies the clauses in Definition 26.

Exercise 28. What is the least fixed point of $[-]$?

Since applicative similarity is defined as a greatest fixed point, it comes with an associated *coinduction proof principle*:

$$R \subseteq [R] \implies R \subseteq \overset{a}{\lesssim}.$$

Such a proof principle states that to prove $\cdot \vdash^{\Lambda} e \overset{\mathrm{a}}{\preceq} f : \sigma$, it is sufficient to exhibit an applicative simulation R such that $\cdot \vdash^{\Lambda} e \mathrel{R} f : \sigma$ (and similarity for values). Symbolically:

$$\frac{\exists\, R\, .\, \cdot \vdash^{\Lambda} e \mathrel{R} f : \sigma \quad R \text{ applicative simulation}}{\cdot \vdash^{\Lambda} e \overset{\mathrm{a}}{\preceq} f : \sigma}$$

$$\frac{\exists\, R\, .\, \cdot \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma \quad R \text{ applicative simulation}}{\cdot \vdash^{\mathcal{V}} v \overset{\mathrm{a}}{\preceq} w : \sigma}$$

Example 2. Use the coinduction proof principle to show that $(\lambda x.e)v \overset{\mathrm{a}}{\preceq} e[v/x]$. More generally, we show that $\llbracket e \rrbracket_{\mathcal{E}} = \llbracket f \rrbracket_{\mathcal{E}}$ implies $e \preceq f$. By the coinduction proof principle, it is enough to exhibit an applicative simulation R relating terms e and f with the same evaluation semantics. Define:

$$R^{\Lambda}_{\sigma} \triangleq \llbracket - \rrbracket_{\mathcal{E}}; \mathsf{I}_{(\mathcal{V}^{\bullet}_{\sigma})_{\perp}}; \llbracket - \rrbracket^{\top}_{\mathcal{E}} \qquad\qquad R^{\mathcal{V}}_{\sigma} \triangleq \mathsf{I}^{\mathcal{V}}_{\sigma}$$

or, in pointwise notation:

$$\begin{aligned} \vdash^{\Lambda} e \mathrel{R} f : \sigma &\overset{\triangle}{\iff} \llbracket e \rrbracket_{\mathcal{E}} \mathsf{I}_{(\mathcal{V}^{\bullet}_{\sigma})_{\perp}} \llbracket f \rrbracket_{\mathcal{E}} \\ \vdash^{\mathcal{V}} v \mathrel{R} w : \sigma &\overset{\triangle}{\iff} \vdash^{\mathcal{V}} v \mathsf{I} w : \sigma. \end{aligned}$$

Showing $R \subseteq [R]$ essentially amounts to prove $R^{\Lambda}_{\sigma} \subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}\mathsf{I}^{\mathcal{V}}_{\sigma}; \llbracket - \rrbracket^{\top}_{\mathcal{E}}$. Since $\mathsf{I}_{A_{\perp}} \subseteq \widetilde{\mathcal{M}}\mathsf{I}_{A}$, we have:

$$R^{\Lambda}_{\sigma} = \llbracket - \rrbracket_{\mathcal{E}}; \mathsf{I}_{(\mathcal{V}^{\bullet}_{\sigma})_{\perp}}; \llbracket - \rrbracket^{\top}_{\mathcal{E}} \subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}\mathsf{I}^{\mathcal{V}}_{\sigma}; \llbracket - \rrbracket^{\top}_{\mathcal{E}}$$

⊠

The coinduction proof principle allows us to easily prove some useful structural properties of applicative bisimilarity, as shown by the following result.

Proposition 4. *Applicative similarity $\preceq$ is reflexive and transitive.*

Proof. The proof is by coinduction. To prove reflexivity, we show that I^{c} is an applicative simulation, whereas to prove transitivity we show that $\overset{\mathrm{a}}{\preceq}; \overset{\mathrm{a}}{\preceq}$ is an applicative simulation. Both these results follow from the structural properties of the relational constructions involved. We show a couple of cases as illustrative examples. We prove that $\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma\to\tau}; \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma\to\tau} \subseteq \llbracket - \rrbracket_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\tau}; \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}}$. We have:

$$\begin{aligned} \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma\to\tau}; \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma\to\tau} &\subseteq \llbracket - \rrbracket_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}}; \llbracket - \rrbracket_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}} \\ &\subseteq \llbracket - \rrbracket_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\tau}]; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}} \\ &\subseteq \llbracket - \rrbracket_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\tau}; \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}}. \end{aligned}$$

In a similar fashion, we show $\overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\sigma}; \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\sigma} \subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}(\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}; \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}); \llbracket - \rrbracket^{\top}_{\mathcal{E}}$.

$$\begin{aligned} \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\sigma}; \overset{\mathrm{a}}{\preceq}{}^{\Lambda}_{\sigma} &\subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}; \llbracket - \rrbracket^{\top}_{\mathcal{E}}; \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}; \llbracket - \rrbracket^{\top}_{\mathcal{E}} \\ &\subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}; \widetilde{\mathcal{M}}\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}; \llbracket - \rrbracket^{\top}_{\mathcal{E}} \\ &\subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{M}}(\overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}; \overset{\mathrm{a}}{\preceq}{}^{\mathcal{V}}_{\sigma}); \llbracket - \rrbracket^{\top}_{\mathcal{E}}. \end{aligned}$$

□

Corollary 2. *The open extension of applicative similarity is reflexive and transitive.*

Concerning applicative bisimilarity, since we defined $\overset{\mathrm{a}}{\simeq}$ as $\overset{\mathrm{a}}{\preceq} \cap \overset{\mathrm{a}}{\preceq}{}^{\top}$, it follows that $\overset{\mathrm{a}}{\simeq}$ is an equivalence, and that its open extension is an equivalence too. Moreover, we can characterise $\overset{\mathrm{a}}{\preceq}$ canonically.

Proposition 5. *Applicative bisimilarity is the largest symmetric applicative simulation.*

Proof. Define the closed term relation S as

$$S \triangleq \bigcup\{R \mid R = R^{\top}, R \subseteq [R]\}.$$

Obviously S is a symmetric applicative simulation and actually the largest such. We prove $\overset{a}{\simeq} \subseteq S$. To do so, it is sufficient to show that $\overset{a}{\simeq}$ is a symmetric applicative simulation, which is indeed the case. Conversely, one proves $S \subseteq \overset{a}{\simeq}$ using the coinduction proof principle associated with $\overset{a}{\preceq}$ and showing that $\square$

Exercise 29. Complete the proof of Proposition 5.

2.6.4 Compatibility

We have seen that applicative similarity is reflexive and transitive, and that applicative bisimilarity is also symmetric. The last, yet main, structural property that of applicative (bi)similarity that we would like to have is compatibility. Proving such a result, however, turns out to be non-trivial. Here, we only sketch the main passages of the proof of compatibility which uses a powerful technique known as *Howe's method* (Howe, 1996). At the heart of Howe's method is a relational construction (called precongruence candidate or Howe extension) extending $\overset{a}{\preceq}$ to a substitutive and compatible relation $\overset{a}{\preceq}^{\mathsf{H}}$. The key ingredient to make such a method work is the so-called *Key Lemma*. The latter essentially states that $\overset{a}{\preceq}^{\mathsf{H}}$ is an applicative simulation, and thus coincide with $\overset{a}{\preceq}$.

Definition 27. *The* Howe extension *of a closed term relation R is the open term relation R^{H} defined as the least fixed point of the map $X \mapsto \widehat{X}; R^{\circ}$.*

Notice that since both $-^{\circ}$ and $\widehat{-}$ are monotone, then so is $X \mapsto \widehat{X}; R^{\circ}$ meaning that it indeed has a least fixed point. In particular, R^{H} is the least term relation satisfying $\widehat{R^{\mathsf{H}}}; R^{\circ} \subseteq R^{\mathsf{H}}$. Pointwise, R^{H} is inductively described as follows:

$$\frac{\Gamma \vdash^{\Lambda} e\, \widehat{R^{\mathsf{H}}}\, h : \sigma \quad \Gamma \vdash^{\Lambda} h\, R\, f : \sigma}{\Gamma \vdash^{\Lambda} e\, R^{\mathsf{H}}\, f : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v\, \widehat{R^{\mathsf{H}}}\, u : \sigma \quad \Gamma \vdash^{\mathcal{V}} u\, R\, w : \sigma}{\Gamma \vdash^{\mathcal{V}} v\, R^{\mathsf{H}}\, w : \sigma}$$

The Howe extension of a reflexive and transitive term relation R satisfies many nice properties. In particular, it is compatible (and thus reflexive) and substitutive. Additionally, it indeed extends the open extension of R.

Lemma 19. *Let R be reflexive and transitive closed term relation. Then the following hold:*

$$\begin{aligned} R^{\circ} &\subseteq R^{\mathsf{H}} \\ \widehat{R^{\mathsf{H}}} &\subseteq R^{\mathsf{H}} \\ \mathsf{I} &\subseteq R^{\mathsf{H}} \\ R^{\mathsf{H}}[R^{\mathsf{H}}] &\subseteq R^{\mathsf{H}} \\ R^{\mathsf{H}}; R^{\circ} &\subseteq R^{\mathsf{H}}. \end{aligned}$$

Since $\overset{a}{\preceq}$ is reflexive and transitive, we can apply Lemma 19 on $\overset{a}{\preceq}^{\mathsf{H}}$. This way, we obtain a compatible and substitutive relation extending $\overset{a}{\preceq}^{\mathsf{H}}$. Relying on Lemma 19, we prove the so-called *Key Lemma* which states that the closed restriction of $\overset{a}{\preceq}^{\mathsf{H}}$ is an applicative simulation.

Lemma 20 (Key Lemma)**.** *Let R be a reflexive and transitive applicative simulation. Then $(R^{\mathsf{H}})^{\mathsf{c}} \subseteq [(R^{\mathsf{H}})^{\mathsf{c}}]$. In particular, $(\overset{a}{\preceq}^{\mathsf{H}})^{\mathsf{c}} \subseteq [(\overset{a}{\preceq}^{\mathsf{H}})^{\mathsf{c}}]$.*

Proof Sketch. Let R be a reflexive and transitive applicative simulation. For readability, oftentimes we will write R^{H} in place of $(R^{\mathsf{H}})^{\mathsf{c}}$: the context will clarify whether we are working with closed or open expressions. By Lemma 19, we see that $(R^{\mathsf{H}})^{\mathsf{c}}$ behaves as an applicative simulation on values. It remains to prove that it does so on computations as well. Assuming $\cdot \vdash^{\Lambda} e\, R^{\mathsf{H}}\, f : \sigma$, we have thus to show $[\![e]\!]_{\mathcal{E}}\, \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma}\, [\![f]\!]_{\mathcal{E}}$. Since $[\![e]\!]_{\mathcal{E}} = \bigsqcup_n [\![e]\!]^n_{\mathcal{E}}$, we would like to prove $[\![e]\!]_{\mathcal{E}}\, \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma}\, [\![f]\!]_{\mathcal{E}}$, i.e. $\bigsqcup_n [\![e]\!]^n_{\mathcal{E}}\, \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma}\, [\![f]\!]_{\mathcal{E}}$ by induction on n. That such an induction principle is valid, is a structural property of $\widetilde{\mathcal{M}}$.

Lemma 21. *For any $R : X \nrightarrow Y$, the following induction principle is sound for $\widetilde{\mathcal{M}}$:*

$$\frac{\forall n \geq 0.\ t_n \ \widetilde{\mathcal{M}}R\ s}{\bigsqcup_{n\geq 0} t_n \ \widetilde{\mathcal{M}}R\ s}$$

Moreover, we have $\bot \ \widetilde{\mathcal{M}}R\ s$.

We thus prove $[\![e]\!]^n_{\mathcal{E}} \ \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma} \ [\![f]\!]_{\mathcal{E}}$ by induction on n. The base case holds by Lemma 21. For the inductive step we proceed by case analysis on the definition of $[\![-]\!]^{n+1}_{\mathcal{E}}$ relying on the definition of R^{H}. The case for values follows since we have the general law

$$x\ S\ y \implies \mathit{just}\ x\ \widetilde{\mathcal{M}}S\ \mathit{just}\ y$$

for all relations $S : X \nrightarrow Y$. All cases but sequencing directly follows from Lemma 19. To deal with sequencing we rely on another structural property of $\widetilde{\mathcal{M}}$.

Lemma 22. *For a function $\varphi : X \to Y_\bot$, define $\gg\!\!=\varphi : X_\bot \to Y_\bot$ by:*

$$\gg\!\!=\varphi(t) \triangleq \begin{cases} \varphi(x) & \textit{if } t = \mathit{just}\ x \\ \bot & \textit{otherwise.} \end{cases}$$

As it is customary, we often write $\varphi \gg\!\!= t$ in place of $\gg\!\!=\varphi(t)$ Notice that

$$[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]^{n+1}_{\mathcal{E}} = [\![e]\!]^n_{\mathcal{E}} \gg\!\!= [\![f[-/x]]\!]^n_{\mathcal{E}}.$$

The following law holds, for all functions $\alpha, \beta : X \to A_\bot$ and relations $R : X \nrightarrow X$, $S : A \nrightarrow A$:

$$R \subseteq \alpha; \widetilde{\mathcal{M}}S; \beta^\top \implies \widetilde{\mathcal{M}}R \subseteq \gg\!\!=\alpha; \widetilde{\mathcal{M}}S; (\gg\!\!=\beta)^\top.$$

Pointwise, we have:

$$\frac{\forall x, y \in X.\ x\ R\ y \implies \alpha(x)\ \widetilde{\mathcal{M}}S\ \beta(y) \qquad t\ \widetilde{\mathcal{M}}R\ s}{t \gg\!\!= \alpha\ \widetilde{\mathcal{M}}S\ s \gg\!\!= \beta}$$

Let us now come back to the case of sequencing. By very definition of Howe extension, we see that we have to show $[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]^{n+1}_{\mathcal{E}} \ \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma} \ [\![\mathbf{let}\ x = g\ \mathbf{in}\ h]\!]_{\mathcal{E}}$, i.e.

$$[\![e]\!]^n_{\mathcal{E}} \gg\!\!= [\![f[-/x]]\!]^n_{\mathcal{E}} \ \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma} \ [\![g]\!]^n_{\mathcal{E}} \gg\!\!= [\![h[-/x]]\!]^n_{\mathcal{E}}$$

At this point, we apply Lemma 22, so that it is sufficient to prove $[\![e]\!]^n_{\mathcal{E}} \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma} [\![g]\!]_{\mathcal{E}}$ and $[\![f[v/x]]\!]^n_{\mathcal{E}} \widetilde{\mathcal{M}}(R^{\mathsf{H}})^{\mathcal{V}}_{\sigma}$ $[\![h[w/x]]\!]_{\mathcal{E}}$, for all v, w related by R^{H}. These facts hold by induction hypothesis and Lemma 19. ☐

Theorem 4. *Applicative similarity and applicative bisimilarity are compatible.*

Proof. To prove that applicative similarity is compatible, it is enough to show $\overset{\mathrm{a}}{\preceq}{}^{\mathsf{H}} = \overset{\mathrm{a}}{\preceq}{}^{\circ}$. By Lemma 19 we already know that the open extension of $\overset{\mathrm{a}}{\preceq}$ is included in $\overset{\mathrm{a}}{\preceq}{}^{\mathsf{H}}$, so that it is sufficient to prove: $\overset{\mathrm{a}}{\preceq}{}^{\mathsf{H}} \subseteq \overset{\mathrm{a}}{\preceq}{}^{\circ}$. That follows from $(\preceq^{\mathsf{H}})^{c} \subseteq \preceq$, which is exactly the content of the Key Lemma. From compatibility of applicative similarity it also follows compatibility of applicative bisimilarity (recall that the intersection of compatible term relations is itself compatible). ☐

2.7 Probabilistic Effects

Endowing the calculus we have introduced in Section 2.2 with a primitive for probabilistic choice is relatively simple: it suffices to enrich the syntactic category of computations with a new construct $\mathbf{sample}_q$ parametrized on a rational number $q \in \mathbb{Q}_{(0,1)}$ which returns either **true** or **false**, the former with probability q the latter with probability $1 - q$. The new operator has a very natural static semantics:

$$\frac{}{\Gamma \vdash^{\Lambda} \mathbf{sample}_q : \mathbf{bool}}$$

The resulting sets of computations and values are indicated as $\mathcal{P}\Lambda$ and $\mathcal{PV}$, respectively, and we extend to them the notations we introduced in Section 2.2. We denote the probabilistic calculus by Λ^{PROB}.

The dynamic semantics of the underlying language must be substantially modified, however. More specifically, the semantics of a (closed) computation of type σ is not anymore a (possibly undefined) value or type σ, but rather a subdistribution (or, simply, a distribution) of values of the same type.

Definition 28. *A (discrete) subdistribution over a set X is a map $\mathscr{D}$ from X to $\mathbb{R}_{[0,1]}$ such that:*

1. *$\mathscr{D}(x) \neq 0$ for countably xs only.*
2. *$\sum_x \mathscr{D}(x) \leq 1$.*

We say that $\mathscr{D}$ is a distribution if $\sum_x \mathscr{D}(x) \leq 1$. The set of subdistributions over a set X is denoted by $\mathcal{D}^{\leq 1}(X)$, whereas the set of distributions over X is denoted by $\mathcal{D}(X)$.

In the setting of Λ^{PROB} — where there are only countably many values — we can simply say that a subdistribution is a map $\mathscr{D}$ from $\mathcal{PV}^\bullet$ to $\mathbb{R}_{[0,1]}$ such that

$$\sum_{v \in \mathcal{PV}^\bullet} \mathscr{D}(v) \leq 1$$

The set of all such distributions is indicated as **DV** (so that $\mathbf{DV} = \mathcal{D}(\mathcal{PV})$), and remarkable elements of this set are the everywhere null distributions, indicated as $\emptyset$ and, for every value v, the dirac distribution δ_v returning 1 on v and 0 everywhere else. The convex combination of distributions is itself a distribution, e.g., given a distribution $\mathscr{D} \in \mathbf{DV}$ and a family of distributions $\{\mathscr{E}_v\}_{v \in \mathcal{PV}^\bullet}$ in **DV**, one can form the *convolution*

$$\sum_{v \in \mathcal{PV}^\bullet} \mathscr{D}(v) \cdot \mathscr{E}_v$$

itself as a distribution in **DV**. These constructions, as we will see, are just special instances of a Kleisli triple from the axiomatic definition of a monad. The *support* of a distribution $\mathscr{D} \in \mathbf{DV}$ is the least subset $sup_{\mathscr{D}}$ of $\mathcal{PV}^\bullet$ such that $\mathscr{D}(v) > 1$ implies $v \in sup_{\mathscr{D}}$. The set of all distributions $\mathscr{D} \in \mathbf{DV}$ such that $sup_{\mathscr{D}} \subseteq \mathcal{PV}^\bullet_\sigma$ is indicated as $\mathbf{DV}_\sigma$. Distributions can be endowed with a partial order by considering the pointwise order inherited from $\mathbb{R}_{[0,1]}$. As such, they form a ωCPO. Actually, all these notions apply to arbitrary sets of the form $\mathcal{D}(X)$

We are finally ready to generalize the dynamic semantics. The $\mathbb{N}$-indexed family of functions $[\![-]\!]^n_{\mathcal{E}} : \prod_\sigma \mathcal{P}\Lambda^\bullet_\sigma \to \mathbf{DV}_\sigma$ is inductively defined as follows:

$$\begin{aligned}
[\![e]\!]^0_{\mathcal{E}} &\triangleq \emptyset \\
[\![\mathbf{sample}_q]\!]^{n+1}_{\mathcal{E}} &\triangleq q \cdot \delta_{\mathbf{true}} + (1-q) \cdot \delta_{\mathbf{false}} \\
[\![\mathbf{return}\ v]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_v \\
[\![\mathbf{proj}_l\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_v \\
[\![\mathbf{proj}_r\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_w \\
[\![\mathbf{if\ true\ then}\ e\ \mathbf{else}\ f]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e]\!]^n_{\mathcal{E}} \\
[\![\mathbf{if\ false\ then}\ e\ \mathbf{else}\ f]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![f]\!]^n_{\mathcal{E}} \\
[\![\mathbf{case}\ (\mathbf{inl}\ v)\ \mathbf{of}\ \{\mathbf{inl}\ x \to e \mid \mathbf{inr}\ x \to f\}]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{case}\ (\mathbf{inr}\ v)\ \mathbf{of}\ \{\mathbf{inl}\ x \to e \mid \mathbf{inr}\ x \to f\}]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![f[w/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{unfold}\ (\mathbf{fold}\ v)]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_v \\
[\![(\lambda x.e)v]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]^{n+1}_{\mathcal{E}} &\triangleq \sum_{v \in \mathcal{PV}^\bullet} [\![e]\!]^n_{\mathcal{E}}(v) \cdot [\![f[v/x]]\!]^n_{\mathcal{E}}
\end{aligned}$$

As you can see, most of the clauses in the definition above are identical to those given in the deterministic cases. On the other hand, it should be noted that the codomain of the maps $[\![-]\!]^n_{\mathcal{E}}$ is a distribution of values and not a value.

Lemma 23. *For any type σ, the sequence of maps $(\llbracket - \rrbracket^n_{\mathcal{E}})_{n \geq 0}$ forms an ω-chain in $\mathcal{P}\Lambda^{\bullet}_{\sigma} \to \mathbf{DV}_{\sigma}$.*

By Lemma 23, we define the evaluation map $\llbracket - \rrbracket_{\mathcal{E}} : \prod_{\sigma} \mathcal{P}\Lambda^{\bullet}_{\sigma} \to \mathbf{DV}_{\sigma}$ as $\bigsqcup_{n \geq 0} \llbracket - \rrbracket^n_{\mathcal{E}}$.

Exercise 30.

1. Recall the term Ω from Exercise 22. Show that, when regarded as term of Λ^{PROB}, we have $\llbracket \Omega \rrbracket_{\mathcal{E}} = \emptyset$.
2. Define a Λ^{PROB} term G satisfying $\llbracket G \rrbracket_{\mathcal{E}} = \frac{1}{2} \cdot \delta_{\mathbf{true}} + \frac{1}{2} \cdot \llbracket G \rrbracket_{\mathcal{E}}$. What is $\llbracket G \rrbracket_{\mathcal{E}}(\mathbf{true})$? What is $\llbracket G \rrbracket^n_{\mathcal{E}}(\mathbf{true})$? We say that G *almost sure terminates* as any finite execution diverges with a non-null probability and yet its probability of terminating is 1.

2.7.1 Logical Relations

Let us now extend the logical equivalence of **??** to a probabilistic setting. To avoid difficulties coming from the presence of recursive types, we restrict our analysis to the simply-typed fragment of Λ^{PROB}. Notice that in such a fragment evaluating a program gives a proper distribution (rather than a distribution), meaning that $\sum_v \llbracket e \rrbracket_{\mathcal{E}}(v) = 1$, for any program e.

The relational calculus defined for Λ^{ST} straightforwardly extends to Λ^{PROB}. The main consequence of that is that all notions and results not involving the evaluation semantics of the calculus directly extends to Λ^{PROB}. In particular, the value clauses of notion of a logical relation for Λ^{ST} straightforwardly translates to Λ^{PROB}. To obtain the notion of a probabilistic logical relation, it remains to understand whet the behaviour of a logical relation should be with respect to program evaluation, the latter being probabilistic. To do so, we proceed as we did when defining applicative bisimilarity. There, we faced the problem of extending a relation on values to a relation over $\mathcal{V}_{\perp} \times \mathcal{V}_{\perp}$, as evaluating a program gave an element in $\mathcal{V}_{\perp}$, rather than just a value. Now the evaluation of a program gives a distribution of values, meaning that we have to find a way to extend a relation over values to a relation over distribution of values. Such an extension exists and it plays a central role in optimal transport and optimisation theory (via the notion of a transportation plan as a coupling and the celebrated Wasserstein-Kantorovich duality), differential privacy (via the notion of expected sensitivity and the Strassen Theorem), and program semantics (via the notion of a probabilistic bisimulation due to Lassen and Skou).

Definition 29. *A* coupling *between two distributions $\mathscr{D}$, $\mathscr{E}$ over sets X, Y, respectively, is a a distribution $\mathscr{C}$ over $X \times Y$ such that:*

$$\sum_y \mathscr{C}(x, y) = \mathscr{D}(x)$$
$$\sum_x \mathscr{C}(x, y) = \mathscr{E}(y)$$

We denote the collection of couplings of $\mathscr{D}$ and $\mathscr{E}$ by $\Omega(\mathscr{D}, \mathscr{E})$

Definition 30. *Given a relation $R : X \nrightarrow Y$, define the relation $\widetilde{\mathcal{D}}R : \mathcal{D}(X) \nrightarrow \mathcal{D}(X)$ by:*

$$\mathscr{D} \, \widetilde{\mathcal{D}}R \, \mathscr{E} \overset{\triangle}{\iff} \exists \mathscr{C} \in \Omega(\mathscr{D}, \mathscr{E}). \, \mathscr{C}(x, y) > 0 \implies x \, R \, y.$$

Given a distribution $\mathscr{D}$ over a set X and a distribution $\mathscr{E}$ over a set Y, the notion of a coupling $\mathscr{C} \in \Omega(\mathscr{D}, \mathscr{E})$ formalises the informal notion of a transportation plan. Thinking to a distribution $\mathscr{D}$ as assigning weight to points in X, then a transportation plan from $\mathscr{D}$ to $\mathscr{E}$ is a mapping specifying how to move weights from X to Y in such a way that $\mathscr{D}$ is 'transformed' into $\mathscr{E}$. Formally, a transportation plan from $\mathscr{D}$ to $\mathscr{E}$ is a family of maps $\mathscr{P}_x : Y \to [0, 1]$ specifying for each $y \in Y$ how much weight of the weight $\mathscr{D}(x)$ of x has to be moved to y. Accordingly to such a reading, we have to impose $\mathscr{P}_x$ some constraints. First of all, $\mathscr{P}_x$ does not move more weight than the available one: actually, since we want to move the whole weight mass $\mathscr{D}$, we require $\mathscr{P}$ to transport the whole $\mathscr{D}$. Formally, we require $\sum_y \mathscr{P}_x(y) = \mathscr{D}(x)$, for all $x \in X$.

Applying $\mathscr{P}$ to $\mathscr{D}$ we obtain a new distribution (a new mass, so to speak) on Y, denoted by $T_{\mathscr{P}}(\mathscr{D})$, and defined as:

$$T_{\mathscr{P}}(\mathscr{D})(y) \triangleq \sum_{x \in X} \mathscr{P}_x(y).$$

Obviously, since $\mathscr{P}$ has to transform $\mathscr{D}$ into $\mathscr{E}$, we require $T_{\mathscr{P}}(\mathscr{D}) = \mathscr{E}$. At this point it is straightforward to see that any transportation plan π from $\mathscr{D}$ to $\mathscr{E}$ induces a coupling $\mathscr{C}_{\mathscr{P}} \in \Omega(\mathscr{D}, \mathscr{E})$ defined by $\mathscr{C}_{\mathscr{P}}(x, y) \triangleq \mathscr{P}_x(y)$, and that any coupling $\mathscr{C} \in \Omega(\mathscr{D}, \mathscr{E})$ induces a transportation plan $\mathscr{P}^{\mathscr{C}}$ defined by $\mathscr{P}^{\mathscr{C}}_x(y) \triangleq \mathscr{C}(x, y)$, so that the notions of a transportation plan and of a coupling are equivalent. Definition 30 thus relates two distributions $\mathscr{D}$, $\mathscr{E}$ if there exists a transportation plan $\mathscr{P}$ from $\mathscr{D}$ to $\mathscr{E}$ moving weights between related points only (that is, if $\mathscr{P}_x(y) > 0$, then x and y must be related).

From a logical point of view, Definition 30 is an existential definition: to prove that two distributions are related, we have to come up with a suitable coupling/transportation plan between them. Remarkably, such an existential condition can be equivalently formulated as an *universal* condition. That is the content of the well-known Strassen Theorem.

Theorem 5 (Strassen's Theorem). *Let $R : X \nrightarrow Y$ be a relation between two sets X and Y and $\mathscr{D}$, $\mathscr{E}$ be distributions over X and Y, respectively. Then:*

$$[\exists \mathscr{C} \in \Omega(\mathscr{D}, \mathscr{E}).\, \mathscr{C}(x, y) > 0 \implies x \mathrel{R} y] \iff [\forall \mathcal{X} \subseteq X.\, \mathscr{D}(X) \leq \mathscr{E}(R[\mathcal{X}])],$$

where for a set A, we define $\mathscr{D}(A) \triangleq \sum_{a \in A} \mathscr{D}(a)$. In particular, $\mathscr{D} \widetilde{\mathcal{D}} R \mathscr{E}$ holds if and only if $\forall \mathcal{X} \subseteq X.\, \mathscr{D}(X) \leq \mathscr{E}(R[\mathcal{X}])$.

Last but not least, we notice that the relational operator $\widetilde{\mathcal{D}}$ satisfies the same structural properties satisfied by $\hat{\mathcal{M}}$. Since our (pre)congruence results relied on such properties[13] rather than on the specific definition of $\hat{\mathcal{M}}$, that should already hints the reader that all our results extend in a modular way to the probabilistic setting of Λ^{PROB}.

Proposition 6. 1. *The map $\widetilde{\mathcal{D}}$ satisfies the following laws.*

$$\begin{gathered} \mathsf{I}_{A_\perp} \subseteq \widetilde{\mathcal{D}} \mathsf{I}_A \\ \widetilde{\mathcal{D}} R; \widetilde{\mathcal{D}} S \subseteq \widetilde{\mathcal{D}}(R; S) \\ R \subseteq S \implies \widetilde{\mathcal{D}} R \subseteq \widetilde{\mathcal{D}} S. \end{gathered}$$

2. *Let $\varphi : X \to \mathcal{D}(Y)$ be a function sending elements of a set X to distributions over Y. Define the map $\gg\!\!=\!\varphi : \mathcal{D}(X) \to \mathcal{D}(Y)$ (as usual, we write $\mathscr{D} \gg\!\!= \varphi$ in place of $\gg\!\!=\!\varphi(\mathscr{D})$):*

$$(\mathscr{D} \gg\!\!= \varphi)(y) \triangleq \sum_{x \in X} \mathscr{D}(x) \cdot \varphi(x)(y).$$

Notice that such a map describes the evaluation semantics of sequencing:

$$[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]^{n+1}_{\mathcal{E}} = [\![e]\!]^n_{\mathcal{E}} \gg\!\!= [\![f[-/x]]\!]^n_{\mathcal{E}}.$$

The following law holds, for all functions $\alpha, \beta : X \to \mathcal{D}(A)$ and relations $R : X \nrightarrow X$, $S : A \nrightarrow A$:

$$R \subseteq \alpha; \widetilde{\mathcal{D}} S; \beta^{\top} \implies \widetilde{\mathcal{D}} R \subseteq \gg\!\!=\!\alpha; \widetilde{\mathcal{D}} S; (\gg\!\!=\!\beta)^{\top}.$$

Pointwise, we have:

$$\frac{\forall x, y \in X.\, x \mathrel{R} y \implies \alpha(x) \mathrel{\widetilde{\mathcal{D}} S} \beta(y) \qquad \mathscr{D} \mathrel{\widetilde{\mathcal{D}} R} \mathscr{E}}{\mathscr{D} \gg\!\!= \alpha \mathrel{\widetilde{\mathcal{D}} S} \mathscr{E} \gg\!\!= \beta}$$

Exercise 31. Prove Proposition 6. *Hint.* Use Strassen's Theorem.

Exercise 32. Let I be a finite set and ${p_i}_{i \in I}$ be a family of numbers in $\mathbb{R}_{[0,1]}$ such that $\sum_i p_i \leq 1$. Show that if $\mathscr{D}_i \mathrel{\widetilde{\mathcal{D}} R} \mathscr{E}_i$ holds for any $i \in I$, then $\sum_{i \in I} p_i \cdot \mathscr{D}_i \mathrel{\widetilde{\mathcal{D}} R} \sum_{i \in I} p_i \cdot \mathscr{E}_i$.

[13]That holds true for applicative (bi)similarity, as we defined logical equivalence for Λ^{ST} only. Nonetheless, things remain similar once defining logical equivalence for calculi with full recursion (although in that setting one has to introduce a more involved form of logical relations, called *step-indexed logical relations*, to deal with non-termination inductively): there, we rely on the (structural properties of the) relational construction $\hat{\mathcal{M}}$ in the same way as we do for applicative bisimilarity.

We now have all the ingredients to define probabilistic logical relations.

Definition 31. *A closed term relation R is a logical relation for Λ^{PROB} if:*

$$\begin{aligned}
R^{\mathcal{V}}_{\mathbf{bool}} &\subseteq \mathsf{I}^{\mathcal{V}}_{\mathbf{bool}}\\
R^{\mathcal{V}}_{\sigma\times\tau} &\subseteq \llbracket - \rrbracket_{\mathcal{L}}; (R^{\mathcal{V}}_{\sigma} \times R^{\mathcal{V}}_{\sigma}); \llbracket - \rrbracket^{\top}_{\mathcal{L}}\\
R^{\mathcal{V}}_{\sigma\to\tau} &\subseteq \llbracket - \rrbracket_{\mathcal{L}}; [R^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}}\\
R^{\Lambda}_{\sigma} &\subseteq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{D}} R^{\mathcal{V}}_{\sigma}; \llbracket - \rrbracket^{\top}_{\mathcal{E}}.
\end{aligned}$$

Notice that the only difference between logical relations for Λ^{ST} and Λ^{PROB} is given the computation clause, where we rely on the operator $\widetilde{\mathcal{D}}$ to handle the probabilistic behaviour of Λ^{PROB} computations. As usual, examples of logical relations are the empty relations as well as the (closed restriction of the) identity relation.

Exercise 33. Show that the identity relation is a logical relation (*hint*: use Proposition 6).

As for Λ^{ST}, logical equivalence is defined by induction on types.

Definition 32. *Logical equivalence is defined thus:*

$$\begin{aligned}
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\mathbf{bool}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{bool}}\\
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}}\\
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma\times\tau} &\triangleq \llbracket - \rrbracket_{\mathcal{L}}; (\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma} \times \overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma}); \llbracket - \rrbracket^{\top}_{\mathcal{L}}\\
\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma\to\tau} &\triangleq \llbracket - \rrbracket_{\mathcal{L}}; [\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma} \to \overset{\mathsf{L}}{\simeq}{}^{\Lambda}_{\tau}]; \llbracket - \rrbracket^{\top}_{\mathcal{L}}\\
\overset{\mathsf{L}}{\simeq}{}^{\Lambda}_{\sigma} &\triangleq \llbracket - \rrbracket_{\mathcal{E}}; \widetilde{\mathcal{D}}(\overset{\mathsf{L}}{\simeq}{}^{\mathcal{V}}_{\sigma}); \llbracket - \rrbracket^{\top}_{\mathcal{E}}.
\end{aligned}$$

Lemma 24. *$\overset{\mathsf{L}}{\simeq}$ is a logical relation.*

We extend $\overset{\mathsf{L}}{\simeq}$ to an open term relation by taking its substitutive extension $\overset{\mathsf{L}}{\simeq}{}^{s}$. As $\overset{\mathsf{L}}{\simeq}$ satisfies all the desired logical properties of notion of equivalence by definition, then so does $\overset{\mathsf{L}}{\simeq}{}^{s}$.

Let us not move to the structural properties of $\overset{\mathsf{L}}{\simeq}{}^{s}$. It does not take much to realise that if we assume $\overset{\mathsf{L}}{\simeq}{}^{s}$ to be reflexive, then we show it to be also symmetric, transitive, and compatible essentially in the same way we did for Λ^{ST}.

Exercise 34. Assume $\overset{\mathsf{L}}{\simeq}$ to be reflexive. Show that $\overset{\mathsf{L}}{\simeq}$ is also symmetric, transitive, and compatible. *Hint.* For transitivity you need to use the first part of Proposition 6. For symmetry, prove $\widetilde{\mathcal{D}}(R^{\top}) = (\widetilde{\mathcal{D}}R)^{\top}$.

What we need to to then, is to prove that $\overset{\mathsf{L}}{\simeq}$ is indeed reflexive.

Proposition 7 (Fundamental Lemma)**.** $\mathsf{I} \subseteq \overset{\mathsf{L}}{\simeq}{}^{s}$.

Proof. We simultaneously prove

$$\forall(\Gamma \vdash^{\Lambda} e : \sigma).\Gamma \vdash^{\Lambda} e \overset{\mathsf{L}}{\simeq}{}^{s} e : \sigma \tag{2.3}$$

$$\forall(\Gamma \vdash^{\mathcal{V}} v : \sigma).\Gamma \vdash^{\mathcal{V}} v \overset{\mathsf{L}}{\simeq}{}^{s} v : \sigma \tag{2.4}$$

by induction on the derivation of $\Gamma \vdash^{\Lambda} e : \sigma$ and $\Gamma \vdash^{\mathcal{V}} e : \sigma$. The case of values goes as for Λ^{ST}. The case of computations is mostly straightforward once we notice that $x \mathrel{R} y \implies \delta_x \mathrel{\widetilde{\mathcal{D}}R} \delta_y$ holds for any set X and relation $R : X \nrightarrow X$. The nontrivial case is, as usual, the one of sequencing. Without much of a surprise, such a case is handled relying on the second part of Proposition 6. Let us see how to proceed. For readability, we write $\simeq$ in place of $\overset{\mathsf{L}}{\simeq}$. We have to show $\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f \simeq^{s} \mathbf{let}\ x = e\ \mathbf{in}\ f : \sigma$ given $\Gamma \vdash^{\Lambda} e \simeq^{s} \mathbf{let}\ x = e\ \mathbf{in}\ f : \tau$ and $\Gamma, x : \tau \vdash^{\Lambda} f \simeq^{s} f : \sigma$. Let $\Gamma \xrightarrow{\gamma,\delta} \cdot$ be substitutions such that $(y : \rho) \in \Gamma$ implies $\cdot \vdash^{\mathcal{V}} \gamma(y) \simeq \delta(y) : \rho$. To prove the thesis, we need to show:

$$\llbracket \mathbf{let}\ x = e\gamma\ \mathbf{in}\ f\gamma \rrbracket_{\mathcal{E}}\ \widetilde{\mathcal{D}}(\simeq^{\mathcal{V}}_{\sigma})\ \llbracket \mathbf{let}\ x = e\delta\ \mathbf{in}\ f\delta \rrbracket_{\mathcal{E}}$$

i.e.

$$[\![e\gamma]\!]_{\mathcal{E}} \gg\!= [\![f\gamma[-/x]]\!]_{\mathcal{E}}\ \widetilde{\mathcal{D}}(\simeq^{\mathcal{V}}_{\sigma})\ [\![e\delta]\!]_{\mathcal{E}} \gg\!= [\![f\delta[-/x]]\!]_{\mathcal{E}}.$$

By the second part of Proposition 6, a sufficient condition for the thesis is given by:

$$\begin{aligned}&[\![e\gamma]\!]_{\mathcal{E}}\ \widetilde{\mathcal{D}}(\simeq^{\mathcal{V}}_{\tau})\ [\![e\delta]\!]_{\mathcal{E}}\\ &\forall v, w \in \mathcal{V}^{\bullet}_{\tau}.\ v \simeq^{\mathcal{V}}_{\sigma} w \implies [\![f\gamma[v/x]]\!]_{\mathcal{E}}\ \widetilde{\mathcal{D}}(\simeq^{\mathcal{V}}_{\sigma})\ [\![e\delta]\!]_{\mathcal{E}} \gg\!= [\![f\delta[w/x]]\!]_{\mathcal{E}}\end{aligned}$$

The latter hold by induction hypothesis (viz. $\Gamma \vdash^{\Lambda} e \simeq^{s}$ **let** $x = e$ **in** $f : \tau$ and $\Gamma, x : \tau \vdash^{\Lambda} f \simeq^{s} f : \sigma$) since sequential and simultaneous substitution coincide when mapping variables to *closed* values. □

2.7.2 Applicative Bisimilarity

Having showed $\widetilde{\mathcal{D}}$ has all the relevant structural properties of $\widetilde{\mathcal{M}}$, extending applicative (bi)similarity to Λ^{PROB} is straightforward, except for the fact the presence of recursion forces us to work with subdistributions rather than distribution. As a consequence, we should come up with a suitable relational construction extending relations over values to relations over subdistributions over values and prove that such a construction has all the structural properties of $\widetilde{\mathcal{D}}$. Instead of doing that from scratch, we can take a modular approach by noticing that subdistributions over a set X corresponds to distributions over $X_{\perp}$. Symbolically, $\mathcal{D}^{\leq 1}(X) \cong \mathcal{D}(X_{\perp})$. In fact, given a subdistribution $\mathscr{D}$ we define the distribution $\mathscr{D}_{\perp}$ over $X_{\perp}$ as follows:

$$\mathscr{D}_{\perp}(t) \triangleq \begin{cases}\mathscr{D}(x) & \text{if } t = \mathit{just}\ x\\ 1 - \sum_{x} \mathscr{D}(x) & \text{otherwise.}\end{cases}$$

Exercise 35. Work put the details of the isomorphism $\mathcal{D}^{\leq 1}(X) \cong \mathcal{D}(X_{\perp})$.

This way, we can define the desired relational construction, denoted by $\widetilde{\mathcal{DM}}$, by $\widetilde{\mathcal{DM}}R \triangleq \widetilde{\mathcal{D}}(\widetilde{\mathcal{M}}(R))$. In particular, $\widetilde{\mathcal{DM}}$ inherits all the structural properties of $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{DM}}$ needed to prove reflexivity, transitivity, and compatibility of applicative similarity.

Proposition 8. *The map $\widetilde{\mathcal{DM}}$ satisfies the following laws.*

$$\begin{gathered}\mathsf{I}_{A_{\perp}} \subseteq \widetilde{\mathcal{DM}}\mathsf{I}_{A}\\ \widetilde{\mathcal{DM}}R; \widetilde{\mathcal{DM}}S \subseteq \widetilde{\mathcal{DM}}(R; S)\\ R \subseteq S \implies \widetilde{\mathcal{DM}}R \subseteq \widetilde{\mathcal{DM}}S.\end{gathered}$$

Moreover, the following law holds, for all functions $\alpha, \beta : X \to \mathcal{D}(A)$ and relations $R : X \nrightarrow X$, $S : A \nrightarrow A$, where $\gg\!=\varphi$ is defined as in the case of proper distributions.

$$R \subseteq \alpha; \widetilde{\mathcal{DM}}S; \beta^{\top} \implies \widetilde{\mathcal{DM}}R \subseteq \gg\!=\alpha; \widetilde{\mathcal{DM}}S; (\gg\!=\beta)^{\top}.$$

We can now define the notion of an applicative simulation exactly as we did for Λ^{μ} but replacing $\widetilde{\mathcal{M}}$ with $\widetilde{\mathcal{DM}}$.

Definition 33. *Given a closed term relation R, we define the closed term relation $[R]$ as follows:*

$$\begin{aligned}[R]^{\mathcal{V}}_{\mathbf{bool}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{bool}}\\ [R]^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}}\\ [R]^{\mathcal{V}}_{\mathbf{void}} &\triangleq \emptyset\\ [R]^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; R^{\mathcal{V}}_{\sigma} \times R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}}\\ [R]^{\mathcal{V}}_{\sigma+\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; R^{\mathcal{V}}_{\sigma} + R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}}\\ [R]^{\mathcal{V}}_{\mu t.\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; R^{\mathcal{V}}_{\sigma[t/\mu t.\sigma]}; [\![-]\!]^{\top}_{\mathcal{L}}\\ [R]^{\mathcal{V}}_{\sigma\to\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to R^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}}\\ R^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \widetilde{\mathcal{DM}}R^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}\end{aligned}$$

We say that a term relation R is an applicative simulation *if $R \subseteq [R]$. Moreover, monotonicity of $\widetilde{\mathcal{DM}}$ makes $[-]$ a monotone map on the complete lattice of term relations. We define applicative similarity as the greatest fixed point of $[-]$:*

$$\overset{a}{\preceq} \triangleq \nu R.[R].$$

Exercise 36. Show that $\overset{a}{\preceq}$ is reflexive and transitive.

To prove that $\overset{a}{\preceq}$ is compatible we use Howe's technique. Remarkably, thanks to our relational calculus such a technique is essentially the same one we defined for Λ^{μ}. Additionally, by inspecting the proof of Lemma 20, we notice that the proof of compatibility of $\overset{a}{\preceq}$ relies on three components:

1. Structural properties of the Howe extension operator;
2. Structural properties of $\widetilde{\mathcal{M}}$;
3. A suitable induction principle for $\widetilde{\mathcal{M}}$.

Therefore, as soon as we prove Λ^{PROB} to have such features, we can conclude $\overset{a}{\preceq}$ to be indeed compatible. Since Λ^{μ} and Λ^{PROB} share most of the syntax, point 1 in the above list is obviously satisfied. Point 2 holds by Proposition 8. It thus remains to prove point 3.

Lemma 25. *For any $R : X \nrightarrow Y$, the following induction principle is sound for $\widetilde{\mathcal{DM}}$:*

$$\frac{\forall n \geq 0.\ \mathscr{D}_n \ \widetilde{\mathcal{DM}}R \ \mathscr{E}}{\bigsqcup_{n\geq 0} \mathscr{D}_n \ \widetilde{\mathcal{DM}}R \ \mathscr{E}}$$

Moreover, we have $\emptyset \ \widetilde{\mathcal{DM}}R \ \mathscr{E}$.

Proof. We use the characterisation of $\widetilde{\mathcal{D}}$ given by Strassen's Theorem. Recall that $\bigsqcup_n \mathscr{D}_n = \sup_n \mathscr{D}_n$ and let $\mathcal{X} \subseteq X$. We show that

$$(\sup_n \mathscr{D}_n)(\mathcal{X}) \leq \mathscr{E}(\widetilde{\mathcal{M}}R[\mathcal{X}])$$

given $\mathscr{D}_n(\mathcal{X}) \leq \mathscr{E}(\widetilde{\mathcal{M}}R[\mathcal{X}])$, for any $n \geq 0$. The latter means $\sum_{x\in\mathcal{X}} \mathscr{D}_n(x) \leq \mathscr{E}(\widetilde{\mathcal{M}}R[\mathcal{X}])$, which gives $\sup_n \sum_{x\in\mathcal{X}} \mathscr{D}_n(x) \leq \mathscr{E}(\widetilde{\mathcal{M}}R[\mathcal{X}])$. We conclude the thesis since, by the monotone convergence theorem, we have

$$\sup_n \sum_{x\in\mathcal{X}} \mathscr{D}_n(x) = \sum_{x\in\mathcal{X}} \sup_n \mathscr{D}_n(x) = \sup_n \mathscr{D}_n(\mathcal{X}).$$

□

We have thus achieved the following result.

Theorem 6. *Applicative similarity is compatible: $\widehat{\overset{a}{\preceq}} \subseteq \overset{a}{\preceq}$.*

Additionally, by defining $\overset{a}{\simeq}$ as $\overset{a}{\preceq} \cap \overset{a}{\preceq}^{\top}$, we see that $\overset{a}{\simeq}$ is compatible too, and hence a congruence.

Chapter 3

Program Metrics

3.1 From Relations to Distances

So far, we have focused on *exact* reasoning about programs, whereby we were interested in answering questions such as *have these programs the same semantics?* or *does the semantics of this program refine the semantics of that program?*. Today's software systems, however, are more and more tailored to perform nontrivial numerical calculations oftentimes relying on probabilistic behaviour to improve their performance. For such programs, exact reasoning based on classic program equivalence and denotational semantics turns out to be inadequate to solve several tasks. For instance, if we opitmise a program e by introducing some probabilistic computations in it, say obtaining a new program f, it is hardly the case that will be able to show $e \simeq f$. For that to be the case, in fact, we would need f to behave essentially deterministically, which is rather antithetic with the idea of a probabilistic optimisation. Instead, what we may hope to have is that with high probability f behaves as e or, stated otherwise, that the difference between e and f is small (if not negligeable). Similarly, we may want to optimise a 'numerical' program by replacing some expensive numerical calculations in it with their approximations,[1] this way improving efficiency at the price of introducing an error in the program. Again, what one would like to show then is that, although the programs thus obtained are not equal, they are not even too different.

To achieve the aforementioned goals, a natural way to go is to refine program equivalences into program distances, this way replacing judgments such as *programs e and f are equivalent* with judgments of the form *programs e and f are ε apart* or, equivalently, *the distance between e and f is ε*.

How can we do that? The first observation we make is that any relation $R : X \rightarrow\!\!\!\!\!\!\mapsto Y$ can be seen as a function $R : X \times Y \to \{0, 1\}$ assigning a truth value to any pair of object (x, y) (viz. 1 if x and y are related, and 0 otherwise). A distance can be thus seen as a refinement of the boolean information $\{0, 1\}$ given by a relation to something with a more quantitative flavour. A natural candidate for that is the extended non-negative interval $[0, \infty]$. We can thus define a distance over $X \times Y$ as map $\mu : X \times Y \to [0, \infty]$.

To define notions of program distance, we should then modify the relational calculus developed so far by replacing relations with distances. To do that, we rely on the observation essentially due to Lawvere[2] (Lawvere, 1973) that distances and relations share the same structural properties, and thus the same algebra as summarised by Table 3.1.

For instance, we say that a relation R is reflexive if $R(x, x) = 1$. Applying Table 3.1, we see that a distance μ is reflexive if $\mu(x, x) = 0$, i.e. if identical points are at distance 0. Similarly, we can translate relation composition

$$(R; S)(x, z) = \exists y.\ R\,(x, y) \wedge S(y, z)$$

to distance composition:

$$\mu; \nu(x, z) \triangleq \inf_{z} \mu(x, y) + \nu(y, z).$$

[1] Think, for instance, about computing an integral with one of the many approximated techniques developed in the field of numerical computing.

[2] Lawvere actually shows that ordered and metric spaces (as well as small categories) are all instances of the general notion of an enriched category (Kelly, 2005).

	Relations $R : X \times Y \to \{0,1\}$	Distances $\mu : X \times Y \to [0,\infty]$
Codomain	$\{0,1\}$	$[0,\infty]$
Order	$\leq$	$\geq$
Join	$\exists$	inf
Meet	$\forall$	sup
Tensor	$\wedge$	$+$
Unit	true	0

Table 3.1: Correspondence 2-$[0,\infty]$.

As a consequence, relation transitive $R; R \subseteq R$ now becomes $\mu; \nu \geq \mu$, which gives:

$$\forall y.\ \mu(x,z) \geq \mu(x,y) + \nu(y,z)$$

which is precisely the well-know triangle inequality law. Putting things together, we see that the quantitative counterpart of being a preorder is being a generalised metric, and that the quantitative counterpart of begin an equivalence is being a pseudometric.[3]

Equivalence	Pseudometric
$1 \leq R(x,x)$	$0 \geq \mu(x,x)$
$R(x,y) \leq R(y,x)$	$\mu(x,y) \geq \mu(y,x)$
$R(x,y) \wedge R(y,z) \leq R(x,z)$	$\mu(x,y) + \mu(y,z) \geq \mu(x,z)$

Table 3.2: Correspondences reflexivity-symmetry-transitivity.

3.2 Compositionality, Distance Amplification, and Linear Types

According to previous discussion, the refinement of program equivalences into program distances seems rather smooth. That is indeed the case for what concerns 'general' relational reasoning. But what happens when we consider constructions specific to the calculi considered? Perhaps surprisingly, the notion of *compatibility* turns out to be problematic. Before seeing why such a notion is problematic, let us understand, even informally, what compatibility means in a quantitative setting. We have seen that a relation is compatible when it is closed under language constructs: that is, R is compatible if

$$e\, R\, f \implies \forall C.\, C[e]\, R\, C[f]$$

where C is a context.[4] Therefore, if R is compatible, then language constructs are monotone (with respect to R). In particular, if R is a compatible equivalence, then it is a congruence relation. Applying Table 3.1, we see that the metric-like counterpart of compatibility is given by:

$$\mu(e,f) \geq \sup_{C} \mu(C[e], C[f]).$$

We thus see that if μ is compatible, then language constructs behave as *non-expansive* functions with respect to μ. That means that programs cannot amplify distances: by using programs that are ε apart, we can obtain only programs that are *less* than ε apart.

Finally, we also need to consider a notion of adequacy. For the sake of our argument, we assume our calculus to come with a numerical type, such as the type of integer or real numbers. Say we have a type

[3] Notice that we work with pseudometrics rather than with metrics, so that we do not require the law $\mu(x,y) = 0 \implies x = y$. In fact, programs which are 0-apart should be nothing but equivalent programs, which can obviously be syntactically distinct.

[4] Our notion of compatibility is purely relational, so to avoid the tedious and bureaucratic task of formally defining contexts. For the present purpose, the reader can just think about a context as a program with a hole $[-]$. We write $C[e]$ for the replacement of the hole with e.

Relation	Distances
Monotonicity	Non-expansiveness
$e\, R\, f \implies \forall C.\, C[e]\, R\, C[f]$	$\mu(e, f) \geq \sup_C \mu(C[e], C[f])$.

Table 3.3: Correspondences compatibility.

nat for natural numbers. In such a setting, it is natural to stipulate a relation to be adequate if whenever two programs of type **nat** are related, then they convergence to the same numeral:[5]

$$e\, R_{\mathbf{nat}}\, f \implies [\![e]\!]_{\mathcal{E}} = [\![f]\!]_{\mathcal{E}}.$$

In a metric-like scenario, rather than testing the equality of the output, we should measure their distance. This way we obtain the following notion of adequacy:

$$\mu_{\mathbf{nat}}(e, f) \geq |[\![e]\!]_{\mathcal{E}} - [\![f]\!]_{\mathcal{E}}|.$$

We now have all the ingredients to see what goes wrong with compatibility. Say to have a compatible and adequate distance μ and two programs at a non-null distance. Then we can take a context that copies and duplicates its input several times, this way amplifying the distance between the original program.

Proposition 9. *Given two programs e, f of type* **nat** *such that $|[\![e]\!]_{\mathcal{E}} - [\![f]\!]_{\mathcal{E}}| > 0$. Then, if μ is adequate and compatible, we have $\mu(e, f) = \infty$.*

Proof. Define the following family of contexts C_n:

$$\begin{aligned} C_0 &\triangleq 0 \\ C_n &\triangleq \mathbf{let}\ x = [-]\ \mathbf{in}\ (\mathbf{let}\ y = x + C_{n-1}[x]\ \mathbf{in\ return}\ y) \end{aligned}$$

Notice that $[\![C[e]]\!]_{\mathcal{E}} = n \cdot [\![e]\!]_{\mathcal{E}}$. Then, we have:

$$\begin{aligned} \mu(e, f) &\geq \sup_n \mu(C_n[e], C_n[f]) \\ &= \sup_n |n \cdot [\![e]\!]_{\mathcal{E}} - n \cdot [\![f]\!]_{\mathcal{E}}| \\ &= \sup_n n \cdot |[\![e]\!]_{\mathcal{E}} - [\![f]\!]_{\mathcal{E}}| \\ &= \infty \end{aligned}$$

where the last equality follows from $|[\![e]\!]_{\mathcal{E}} - [\![f]\!]_{\mathcal{E}}| > 0$. □

Each context C_n amplifies the distance between e and f of a factor n, this making e and f maximally distant. During its evaluation, every time the context C_n evaluates its inputs the detected distance between the latter is somehow accumulated to the distances previously observed, thus exploiting the *linear* — as opposed to classical — nature of the act of measuring. Such linearity naturally reflects the monoidal closed structure of categories of metric spaces (see next sections), in opposition with the cartesian closed structure characterising 'classical' (i.e. boolean-valued) observations. Thus, for instance, no matter how we define our program distanceμ, as long as it is adequate and compatible, it will give, e.g., $\mu(2, 3) = \infty$. This phenomenon, which has been first discovered and studied in probabilistic calculi (Crubillé & Dal Lago, 2015), is known as *distance trivialisation* (Crubillé & Dal Lago, 2017), as it essentially makes any distance collapse to an equivalence (programs are either 0 or ∞ apart). Distance trivialisation is thus a consequence of higher-features of calculi. In fact, higher-order programs can freely copy and duplicate their input several times, thus having the testing power to amplify distances between their inputs *ad libitum*.

[5]For simplicity, we assume our calculus to be strongly normalising.

Let us see another example of distance trivialisation in the setting of the probabilistic calculus of Section section 2.7 (Crubillé & Dal Lago, 2017). Recall that in such a setting we say that a relation is adequate if related programs have the same probability of convergence:

$$e\,R\,f \implies \sum_{v} [\![e]\!]_{\mathcal{E}}(v) = \sum_{v} [\![f]\!]_{\mathcal{E}}(v)$$

In a metric-like scenario, rather than testing for equality probability of convergence, we should measure the distance between such probabilities. This way we obtain the following notion of adequacy:

$$\mu(e, f) \geq |\sum_{v} [\![e]\!]_{\mathcal{E}}(v) - \sum_{v} [\![f]\!]_{\mathcal{E}}(v)|.$$

Let us now consider the programs $G \triangleq \mathbf{return}\ (\mathbf{true}) \oplus \Omega$ and $T \triangleq \mathbf{return\ true}$, where $e \oplus f \triangleq \mathbf{let}\ x = \mathbf{sample}_{0.5}\ \mathbf{in}\ (\mathbf{if}\ x\ \mathbf{then}\ e\ \mathbf{else}\ f)$. Since $[\![G]\!]_{\mathcal{E}} = \frac{1}{2} \cdot \delta_{\mathbf{true}}$ and $[\![T]\!]_{\mathcal{E}} = \delta_{\mathbf{true}}$, we would expect any sufficiently well-behaved notion of distance μ to give $\mu(T, G) = \frac{1}{2}$.

Exercise 37. Consider the family of contexts

$$C_n \triangleq (\lambda x.(x\langle\rangle; \cdots ; x\langle\rangle))(\lambda y.[-])$$

where $e; f$ denotes the trivial sequencing of e and f.

1. Show that $[\![C_n T]\!]_{\mathcal{E}} = 1$ and $[\![C_n G]\!]_{\mathcal{E}} = \sum_{i=1}^{n} \frac{1}{2^i}$.
2. Show that if μ is adequate and compatible, then $\mu(T, G) = 1$.
3. Conclude that we have distance trivialisation phenomena in probabilistic calculi.

The moral of distance trivialisation is that not only how a context uses a program matters, but also *how much* it uses the program does. That is the reason why our notion of comaptibility does not work well in a metric scenario, as the latter does not take into account *how much* programs can access their inputs. We overcome this issue by giving a precise meaning to the expression *how much* above and by giving a new notion of compatibility. The key ingredient to do that is the notion of *program sensitivity* (de Amorim, Gaboardi, Hsu, Katsumata, & Cherigui, 2017; Reed & Pierce, 2010). Intuitively, the sensitivity of program is the law describing how much differences in the output are affected by differences in the input. We thus use the notion of sensitivity to formalise the testing power of contexts. More precisely, we define the sensitivity of a context C as a non-negative real number s. The notion of compatibility is refined accordingly: we allow a context C with sensitivity s to increase the distance $\mu(e, f)$ between e and f, but of a factor *at most* s.

$$s \cdot \mu(e, f) \geq \mu(C[e], C[f]).$$

The introduction of the notion of program sensitivity seems to prevent distance trivialisation phenomena. However, it also raises an important question: how can we track program sensitivity? To answer this question, we follow (Reed & Pierce, 2010) and move to linear-like calculi. As already observed, there is no hope to have a nontrivial theory of program distance without a tight control on the copying-capabilities of higher-order calculi. The calculi we consider have powerful type systems inspired by bounded linear logic (Girard, Scedrov, & Scott, 1992) and, compared to other linear type systems, has the novelty of introducing types of the form $!_s\sigma$, where s is a non-negative real number modelling program sensitivity. Moreover, allowing the sensitivity of a program to be ∞, meaning that we are working with non-negative extended real numbers, we can model programs testing their input *ad libitum*. As a consequence, we can recover the full bang type $!\sigma$ as $!_\infty\sigma$, and thus encoding all calculi seen so far in the linear calculus we are going to define.

3.3 Endowing Λ with Linearity Constraints

As just observed, when dealing with program distances a crucial parameter in distance trivialisation is program sensitivity. To deal with such parameter we now introduce F^{μ}, a linear-like refinement of

Λ^μ designed following the calculus Fuzz by Reed and Pierce (Reed & Pierce, 2010). F^μ is characterised by a powerful linear type system inspired by *bounded linear logic* (Girard et al., 1992) giving syntactic information on program sensitivity. Types, (raw) values, and (raw) computations of F^μ are defined in Figure 3.1, where t denotes a type variable and $s \in [0, \infty]$.

$$
\begin{array}{rcl}
\text{Types } \sigma, \tau & ::= & t \\
& \mid & \mathbf{unit} \\
& \mid & \sigma \times \tau \\
& \mid & \sigma \otimes \tau \\
& \mid & \mathbf{void} \\
& \mid & \sigma + \tau \\
& \mid & \sigma \multimap \tau \\
& \mid & \mu t.\sigma \\
& \mid & !_s\sigma
\end{array}
\qquad
\begin{array}{rcl}
\text{Values } v, w & ::= & x \\
& \mid & \langle\rangle \\
& \mid & \langle v, w\rangle \\
& \mid & v \otimes w \\
& \mid & \mathbf{inl}\ v \\
& \mid & \mathbf{inr}\ v \\
& \mid & \lambda x.e \\
& \mid & \mathbf{fold}\ v \\
& \mid & !v
\end{array}
\qquad
\begin{array}{rcl}
\text{Computations } e, f & ::= & \mathbf{return}\ v \\
& \mid & \mathbf{proj}_l\ v \\
& \mid & \mathbf{proj}_r\ v \\
& \mid & \mathbf{let}\ x \otimes y = v\ \mathbf{in}\ e \\
& \mid & \mathbf{abort}\ v \\
& \mid & \mathbf{case}\ v\ \mathbf{of}\ \{\mathbf{inl}\ x \to e \mid \mathbf{inr}\ x \to f\} \\
& \mid & vw \\
& \mid & \mathbf{unfold}\ v \\
& \mid & \mathbf{let}\ !x = v\ \mathbf{in}\ e \\
& \mid & \mathbf{let}\ x = e\ \mathbf{in}\ f
\end{array}
$$

Figure 3.1: Types, values, and computations of F^μ.

Free and bound variables in computations and values are defined as usual, and we adopt the same notational conventions of Λ^μ.

Types of F^μ differ from those of Λ^μ for the presence of tensor types $\sigma \otimes \tau$, linear arrow types $\sigma \multimap \tau$, and bounded exponential types $!_s\sigma$. The first two family of types are standard: for instance, an expression of type $\sigma \multimap \tau$ behaves as a function that takes in input a value of type σ, uses it exactly once[6], and produces a values of type τ. Bounded exponential types are at the heart of F^μ's type system. Intuitively, a program of type $!_s\sigma \multimap \tau$ represent a function with sensitivity s. That is, we use bounded exponential type to statically track information about the sensitivity of programs. We refer to elements $s \in [0, \infty]$ as sensitivities.

3.3.1 Static Semantics

To define the class of well-defined F^μ expression we endow F^μ with a suitable type system. Intuitively, the latter is based on judgments of the form $x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash e : \sigma$, where $s_1, \ldots, s_n$ are sensitivities. The informal meaning of such judgment is that on input x_i ($i \leq n$), the computation e has sensitivity s_i. That is, e amplifies the (behavioural) distance between two input values v_i, w_i of *at most* a factor s_i. Symbolically, we have:

$$s_i \cdot \mu(v_i, w_i') \geq \mu(e[v_i/x_i], e[w_i/x_i]).$$

In order to define typing judgments formally, we need to refine the notion of a *typing environment* introduced in previous chapter.

Definition 34. *A typing environment is a a partial function* Γ *from variables to types and sensitivities such that the domain* $\mathrm{dom}(\Gamma) \triangleq \{x \mid \Gamma(x) \neq \bot\}$ *of* Γ *is finite.*

Given a typing environment Γ, we use the notation $x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n$ to denote it, where $\{x_1, \ldots, x_n\} = \mathrm{dom}(\Gamma)$ and $\Gamma(x_i) = \mathit{just}\ (\sigma_i, s_i)$. Moreover, we write $(x :_s \sigma) \in \Gamma$ to mean that $\Gamma(x) = \mathit{just}\ (\sigma, s)$. Notice that we distinguish between environments Γ such that $\Gamma(x) = \bot$ and those such that $\Gamma(x) = (\sigma, 0)$: that is, there is a distinction between not using a variable and using it with sensitivity zero (i.e. zero times). Nonetheless, we will see that if an expression is typable in an environment Γ

[6] Actually, for our purposes it is enough to consider *affine* functions, i.e. functions using their inputs *at most* once.

undefined on x, then we can weaken the environment by adding $\Gamma(x) = \mathit{just}\ (\sigma, s)$. As usual, we denote by $\cdot$ the totally undefined environment.

As for non-linear calculi, we can join disjoint typing environments together. The typing environment disjointness relation $\perp$ is defined thus:

$$\Gamma \perp \Delta \overset{\triangle}{\iff} \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) = \emptyset$$

The union Γ, Δ of two disjoint typing environments Γ and Δ is then defined in the usual way:

$$(\Gamma, \Delta)(x) \triangleq \begin{cases} \Gamma(x) & \text{if } x \in \mathsf{dom}(\Gamma) \\ \Delta(x) & \text{otherwise.} \end{cases}$$

As usual, whenever we write Γ, Δ, we assume Γ and Δ to be disjoint.

The linear (substructural) nature of F^{μ} makes the disjoint union of typing environments too weak to give a meaningful static semantics to F^{μ} expressions. To see that, consider, for instance, the typing rule for sequencing:

$$\frac{\Gamma \vdash^{\Lambda} e : \sigma \quad \Gamma, x : \sigma \vdash^{\Lambda} f : \tau}{\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f}$$

Say now that e uses a variable y with sensitivity s. According to our typing rule, f needs also to use y with sensitivity s (and that may be fine), but then one would infer that also **let** $x = e$ **in** f uses y with sensitivity s, which is clearly wrong, as **let** $x = e$ **in** f actually uses y with sensitivity s^2. Using a proof-theoretical vocabulary, the additive type systems of previous chapter are unsound in the linear setting of F^{μ}, the latter requiring a multiplicative type systems. Giving such a system ultimately rely on finding ways to combine sensitivities, both in parallel (as in the case of two programs using the same variable independently) and sequentially (as in the case of a term using another term in place of a variable with a given sensitivity). We model these operations using (extended) real number addition and multiplication.

Definition 35. *Extended addition on $[0, \infty]$ is defined by extending the usual real number addition with the clause $x + \infty \triangleq \infty + x \triangleq \infty$. Extended multiplication on $[0, \infty]$ is defined as follows, where $x < \infty$:*

$$\begin{aligned} 0 \cdot \infty &\triangleq 0 \\ \infty \cdot 0 &\triangleq 0 \\ \infty \cdot x &\triangleq \infty \\ x \cdot \infty &\triangleq \infty. \end{aligned}$$

Notice that 0 is stronger than ∞.[7] Let us now come back to typing environments. We begin with the addition of typing environments (cf. Exercise 3).

Definition 36. 1. *Define the consistency relation between typing environments thus:*

$$\Gamma \sim \Delta \overset{\triangle}{\iff} x \in \mathsf{dom}(\Gamma) \cap \mathsf{dom}(\Delta) \implies \pi_1(\Gamma(x)) = \pi_1(\Delta(x)).$$

[7] Our definition of extended multiplication is different from the one given in the usual denotational semantics of F^{μ} (de Amorim et al., 2017), where extended multiplication is defined as follows (with $y \neq 0$):

$$\begin{aligned} x \cdot \infty &\triangleq \infty \\ \infty \cdot 0 &\triangleq 0 \\ \infty \cdot y &\triangleq \infty. \end{aligned}$$

Despite making extended multiplication non-commutative, the above definition does not fit with our intuition about program sensitivity. In fact, according to our informal reading, the equality $0 \cdot \infty = 0$ expresses the principle that if we test observationally distinguishable objects zero times (i.e. we do not test them at all), then we are not able to tell them apart. Dually, the equality $\infty \cdot 0 = 0$ states that observationally indistinguishable objects cannot be told apart, no matter how many times are tested. The reason why some authors need such a unconventional form of multiplication is their treatment of sequencing which impacts on the semantics of sum types creating some problems. As we will see, our typing rule for sequencing prevents such problems, and we can thus use a standard notion of extended multiplication. Nonetheless, we remark that although our definition of extended multiplication agrees with our intuition of program sensitivity, it has the drawback of being non-continuous. For instance, consider the set $X = \{\varepsilon \mid \varepsilon > 0\}$. Continuity requires $\infty \cdot \inf X = \inf(\infty \cdot X)$, which is obviously not the case since $\infty \cdot \inf X = \infty \cdot 0 = 0$ and $\inf\{\infty \cdot \varepsilon \mid \varepsilon > 0\} = \inf \infty = \infty$.

That is $\Gamma \sim \Delta$ *if and only if whenever* $(x :_s \sigma) \in \Gamma$ *and* $x \in \mathrm{dom}(\Delta)$ *, we have* $(x :_r \sigma) \in \Delta$*, for some* $r \in [0, \infty]$.

2. *The sum operation* $+$ *is defined between consistent typing environments as follows:*

$$(\Gamma + \Delta)(x) \triangleq \begin{cases} \mathit{just}\ (\pi_1(\Gamma(x)), \pi_2(\Gamma(x)) + \pi_2(\Delta(x))) & \text{if } x \in \mathrm{dom}(\Gamma) \cap \mathrm{dom}(\Delta) \\ \Gamma(x) & \text{if } x \notin \mathrm{dom}(\Delta) \\ \Delta(x) & \text{if } x \notin \mathrm{dom}(\Delta) \end{cases}$$

Therefore, if we sum two environments Γ *and* Δ *containing the declarations* $x :_s \sigma$ *and* $x :_r \sigma$*, respectively (such declarations necessarily have the same type* σ *by consistency), then* $\Gamma + \Delta$ *will contain* $x :_{s+r} \sigma$*.*

3. *The multiplication (or scaling) operation scale sensitivities in a typing environment by a given sensitivity:*

$$(s \cdot \Gamma)(x) \triangleq \begin{cases} \mathit{just}\ (\sigma, s \cdot r) & \text{if } \Gamma(x) = \mathit{just}\ (\sigma, r) \\ \bot & \text{otherwise.} \end{cases}$$

Lemma 26. *The following laws hold:*

$$\begin{aligned} &\Gamma \sim \Gamma \\ &\Gamma \perp \Delta \implies \Gamma \sim \Delta \\ &\Gamma \perp \Delta \implies \Gamma + \Delta = \Gamma, \Delta \end{aligned}$$

Notice that, contrary to Λ^{ST}, now $\Gamma + \Gamma = \Gamma$ does not hold in general: the failure of this identity reflects the substructural nature of F^{μ}.

Having at our disposal operations on typing environments we can define a type system for F^{μ}. Such a type system is based on two kinds of judgment (exploiting the fine-grained style of the calculus): judgments of the form $\Gamma \vdash^{\mathcal{V}} v : \sigma$ for values and judgments of the form $\Gamma \vdash^{\Lambda} e : \sigma$ for computations. The system is defined in Figure 3.2.

Let us comment some of the typing rules in Figure 3.2. In the variable rule we require $s \geq 1$, meaning our calculus is actually *affine*, rather than linear. That is because affinity is enough to give a proper account of program distance. By replacing s with 1 in the variable rule, we make F^{μ} linear. Typing environment manipulations at this point should be clear. In the introduction rule for the exponential modality, we scale the type environment by a factor s, this way making a value usable in place of variables with sensitivity s. That is the exactly the meaning of the elimination rule for the exponential modality. The most peculiar rule is the rule for sequencing, which uses the binary maximum operation. To see why we need such an operation, let us consider the following instance of the rule, where f is a closed term of type τ (so that we can assume it to have sensitivity 0 on y).

$$\frac{x :_1 \sigma \vdash^{\Lambda} e : \sigma \quad y :_0 \sigma \vdash^{\Lambda} f : \tau}{x :_{\max(0,1)\cdot 1} \sigma \vdash^{\Lambda} \mathbf{let}\ y = e\ \mathbf{in}\ f : \tau}$$

According to our informal intuition, e has sensitivity 1 on input x, meaning that (i) e can possibly detect (behavioural) differences between input values v, w, and (ii) e cannot amplify their behavioural distance of a factor bigger than 1. Formally, point (ii) states that we have the inequality $\mu(v, w) \geq \mu(e[v/x], e[w/x])$, where μ denotes a suitable program distance. On the contrary, f is a closed term having sensitivity 0 on the input y, meaning that it cannot detect any observable difference between input values for which y is a placeholder. In particular, for all values v, w we have $\mu(f[v/y], f[w/y]) = \mu(f, f) = 0$ (provided that μ is reflexive). Replacing $\max(0, 1)$ with 0 in the typing rule for sequencing would allow us to infer the judgment $x :_0 \sigma \vdash^{\Lambda} \mathbf{let}\ y = e\ \mathbf{in}\ f : \tau$, and thus to conclude

$$\mu(\mathbf{let}\ y = e[v/x]\ \mathbf{in}\ f, \mathbf{let}\ y = e[w/x]\ \mathbf{in}\ f) = 0.$$

The latter equality is unsound as evaluating $\mathbf{let}\ y = e[v/x]\ \mathbf{in}\ f$ (resp. $\mathbf{let}\ y = e[w/x]\ \mathbf{in}\ f$) requires to *first* evaluate $e[v/x]$ (resp. $e[w/x]$) thus making observable differences between v and w detectable

$$\frac{}{\Gamma, x :_s \sigma \vdash^{\mathcal{V}} x : \sigma}\ s \geq 1 \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{\Gamma \vdash^{\Lambda} \mathbf{return}\ v : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} e : \tau \quad \Gamma, x :_s \tau \vdash^{\Lambda} f : \sigma}{\max(s,1) \cdot \Gamma + \Delta \vdash^{\Lambda} \mathbf{let}\ x = e\ \mathbf{in}\ f : \sigma}$$

$$\frac{\Gamma, x :_1 \sigma \vdash^{\Lambda} e : \tau}{\Gamma \vdash^{\mathcal{V}} \lambda x.e : \sigma \multimap \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \multimap \tau \quad \Delta \vdash^{\mathcal{V}} w : \sigma}{\Gamma + \Delta \vdash^{\Lambda} vw : \tau}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} \langle\rangle : \mathbf{unit}} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \vdash^{\mathcal{V}} w : \tau}{\Gamma \vdash^{\mathcal{V}} \langle v, w \rangle : \sigma \times \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \mathbf{proj}_l\ v : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \times \tau}{\Gamma \vdash^{\Lambda} \mathbf{proj}_r\ v : \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Delta \vdash^{\mathcal{V}} w : \tau}{\Gamma + \Delta \vdash^{\mathcal{V}} v \otimes w : \sigma \otimes \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \otimes \tau \quad \Delta, x :_s \sigma, y :_s \tau \vdash^{\Lambda} e : \rho}{s \cdot \Gamma + \Delta \vdash^{\Lambda} \mathbf{let}\ x \otimes y = v\ \mathbf{in}\ e : \rho}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \mathbf{void}}{\Gamma \vdash^{\Lambda} \mathbf{abort}\ v : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{\Gamma \vdash^{\mathcal{V}} \mathbf{inl}\ v : \sigma + \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \tau}{\Gamma \vdash^{\mathcal{V}} \mathbf{inr}\ v : \sigma + \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma + \tau \quad \Delta, x :_s \sigma \vdash^{\Lambda} e : \rho \quad \Delta, y :_s \tau \vdash^{\Lambda} f : \rho}{s \cdot \Gamma + \Delta \vdash^{\Lambda} \mathbf{case}\ v\ \mathbf{of}\ \{\mathbf{inl}\ x \to e \mid \mathbf{inr}\ x \to f\} : \rho}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \sigma[\mu t.\sigma/t]}{\Gamma \vdash^{\mathcal{V}} \mathbf{fold}\ v : \mu t.\sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \mu t.\tau}{\Gamma \vdash^{\Lambda} \mathbf{unfold}\ v : \sigma[\mu t.\sigma/t]}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} v : \sigma}{s \cdot \Gamma \vdash^{\mathcal{V}} !v : !_s\sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : !_s\sigma \quad \Delta, x :_{r \cdot s} \sigma \vdash^{\Lambda} e : \tau}{r \cdot \Gamma + \Delta \vdash^{\Lambda} \mathbf{let}\ !x = v\ \mathbf{in}\ e : \tau}$$

Figure 3.2: Static semantics of F^{μ}.

Exercise 38. Show that the weakening rules below are admissible.

$$\frac{\Gamma \vdash^{\Lambda} e : \sigma \quad \Gamma \sim \Delta}{\Gamma + \Delta \vdash^{\Lambda} e : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \sim \Delta}{\Gamma + \Delta \vdash^{\mathcal{V}} v : \sigma}$$

The notion of a substitution in the setting of F^{μ} is more delicate than the ones seen so far. Substitutions are still maps from variables to values: however, when performing the substitution of a variable x with a value $\Delta \vdash^{\mathcal{V}} v : \sigma$ in a term $\Gamma, x :_s \sigma \vdash^{\Lambda} e : \tau$, we do *not* obtain a term $\Gamma + \Delta \vdash^{\Lambda} e[v/x] : \tau$. In fact, the variable x is used according s in e: as a consequence, each variable y in Δ used, say, according to r will be used according to $s \cdot r$ in $e[v/x]$ (intuitively, v is used s times in $e[v/x]$, and every time it is used, y is used r times, thus for a total of $s \cdot r$ times). This gives a first version of the substitution lemma stating admissibility of the following rules:

$$\frac{\Gamma, x :_s \sigma \vdash^{\Lambda} e : \sigma \quad \Delta \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \sim \Delta}{\Gamma + \sum s \cdot \Delta \vdash^{\Lambda} e[v/x] : \sigma} \qquad \frac{\Gamma, x :_s \sigma \vdash^{\mathcal{V}} e : \sigma \quad \Delta \vdash^{\mathcal{V}} v : \sigma \quad \Gamma \sim \Delta}{\Gamma + \sum s \cdot \Delta \vdash^{\mathcal{V}} v[v/x] : \sigma}$$

To deal with arbitrary substitutions $[\boldsymbol{x}/\boldsymbol{v}]$ rather than with single ones, we simply generalises the above rules to vectors. To do so, we introduce the following notation: given vectors $\boldsymbol{a}$, $\boldsymbol{b}$ (of the same length), we write $\sum \boldsymbol{a} \cdot \boldsymbol{b}$ for their dot product $\sum_i a_i \cdot b_i$.

Lemma 27. *The following substitution rules are admissible*

$$\frac{\Gamma, \boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\Lambda} e : \sigma \quad \boldsymbol{\Delta} \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \sim \Delta_i}{\Gamma + \sum \boldsymbol{s} \cdot \boldsymbol{\Delta} \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\Gamma, \boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\mathcal{V}} v : \sigma \quad \boldsymbol{\Delta} \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \sim \Delta_i}{\Gamma + \sum \boldsymbol{s} \cdot \boldsymbol{\Delta} \vdash^{\mathcal{V}} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

Writing without the vectors notation, we have, e.g., the rule:

$$\frac{\Gamma, x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash^{\Lambda} e : \sigma \quad \Delta_i \vdash^{\mathcal{V}} v_i : \sigma_i \quad \Gamma \sim \Delta_i}{\Gamma + \sum_i s_i \cdot \Delta_i \vdash^{\Lambda} e[v_1, \ldots, v_n/x_1, \ldots, x_n] : \sigma}$$

3.3.2 Dynamic Semantics

We extend the evaluation semantics of Λ^{ST} to Λ^{μ} in the natural way.

Definition 37. *The* $\mathbb{N}$*-indexed family of functions* $\llbracket - \rrbracket^n_{\mathcal{E}} : \prod_\sigma \Lambda^{\bullet}_\sigma \rightharpoonup \mathcal{V}^{\bullet}_\sigma$ *is inductively defined as follows:*[8]

$$
\begin{aligned}
\llbracket e \rrbracket^0_{\mathcal{E}} &\triangleq \bot \\
\llbracket \mathbf{return}\ v \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
\llbracket \mathbf{proj}_l\ \langle v, w \rangle \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
\llbracket \mathbf{proj}_r\ \langle v, w \rangle \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ w \\
\llbracket \mathbf{let}\ x \otimes y = v \otimes w\ \mathbf{in}\ e \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \llbracket e[x := v, y := w] \rrbracket^n_{\mathcal{E}} \\
\llbracket \mathbf{case}\ \mathbf{inl}\ v\ \mathbf{of}\ \{\langle i, x \rangle \to e\} f \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \llbracket e[v/x] \rrbracket^n_{\mathcal{E}} \\
\llbracket \mathbf{case}\ \mathbf{inr}\ w\ \mathbf{of}\ \{\langle i, x \rangle \to e\} f \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \llbracket f[w/x] \rrbracket^n_{\mathcal{E}} \\
\llbracket \mathbf{unfold}\ (\mathbf{fold}\ v) \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
\llbracket (\lambda x.e) v \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \llbracket e[v/x] \rrbracket^n_{\mathcal{E}} \\
\llbracket \mathbf{let}\ !x = !v\ \mathbf{in}\ e \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \llbracket e[v/x] \rrbracket^n_{\mathcal{E}} \\
\llbracket \mathbf{let}\ x = e\ \mathbf{in}\ f \rrbracket^{n+1}_{\mathcal{E}} &\triangleq \begin{cases} \llbracket f[v/x] \rrbracket^n_{\mathcal{E}} & \text{if } \llbracket e \rrbracket^n_{\mathcal{E}} = \mathit{just}\ v \\ \bot & \text{otherwise.} \end{cases}
\end{aligned}
$$

Lemma 28. *For any type* σ*, the sequence of maps* $(\llbracket - \rrbracket^n_{\mathcal{E}})_{n \geq 0}$ *forms an* ω*-chain in* $\Lambda^{\bullet}_\sigma \rightharpoonup \mathcal{V}^{\bullet}_\sigma$.

By Lemma 40, we define the evaluation map $\llbracket - \rrbracket_{\mathcal{E}} : \prod_\sigma \Lambda^{\bullet}_\sigma \rightharpoonup \mathcal{V}^{\bullet}_\sigma$ as $\bigsqcup_{n \geq 0} \llbracket - \rrbracket^n_{\mathcal{E}}$.

Lemma 29. *The following identities hold.*

$$
\begin{aligned}
\llbracket \mathbf{return}\ v \rrbracket_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
\llbracket \mathbf{proj}_l\ \langle v, w \rangle \rrbracket_{\mathcal{E}} &= \mathit{just}\ v \\
\llbracket \mathbf{proj}_r\ \langle v, w \rangle \rrbracket_{\mathcal{E}} &= \mathit{just}\ w \\
\llbracket \mathbf{let}\ x \otimes y = v \otimes w\ \mathbf{in}\ e \rrbracket_{\mathcal{E}} &= \llbracket e[x := v, y := w] \rrbracket_{\mathcal{E}} \\
\llbracket \mathbf{case}\ \mathbf{inl}\ v\ \mathbf{of}\ \{\langle i, x \rangle \to e\} f \rrbracket_{\mathcal{E}} &= \llbracket e[v/x] \rrbracket_{\mathcal{E}} \\
\llbracket \mathbf{case}\ \mathbf{inr}\ w\ \mathbf{of}\ \{\langle i, x \rangle \to e\} f \rrbracket_{\mathcal{E}} &= \llbracket f[w/x] \rrbracket_{\mathcal{E}} \\
\llbracket \mathbf{unfold}\ (\mathbf{fold}\ v) \rrbracket_{\mathcal{E}} &= \mathit{just}\ v \\
\llbracket (\lambda x.e) v \rrbracket_{\mathcal{E}} &= \llbracket e[v/x] \rrbracket_{\mathcal{E}} \\
\llbracket \mathbf{let}\ !x = !v\ \mathbf{in}\ e \rrbracket_{\mathcal{E}} &= \llbracket e[v/x] \rrbracket_{\mathcal{E}} \\
\llbracket \mathbf{let}\ x = e\ \mathbf{in}\ f \rrbracket_{\mathcal{E}} &= \begin{cases} \llbracket f[v/x] \rrbracket_{\mathcal{E}} & \text{if } \llbracket e \rrbracket_{\mathcal{E}} = \mathit{just}\ v \\ \bot & \text{otherwise.} \end{cases}
\end{aligned}
$$

3.4 Quantiative Relational Calculus

In this section, we extend the relational calculus developed in Section 2.3 to F^{μ} and program distances. Actually, instead of working with distances directly, we develop a calculus of *ternary relations*. We do so for several reasons: first, finding a light notation for term distances — the latter being the metric counterpart of term relations — is not easy[9]; secondly, we would like our quantitative analysis to be inherently *relational*, so to make the extension of the relational calculus of Section 2.3 as smooth as

[8] As usual, we omit type subscripts.

[9] For instance, we may define a term distance as a map μ associating to each sequent $\Gamma \vdash \sigma$ a distance $\underset{\Gamma \vdash \Lambda_\sigma}{\mu}$ on $\Lambda_{\Gamma \vdash \Lambda_\sigma}$ and a distance $\underset{\Gamma \vdash \mathcal{V}_\sigma}{\mu}$ on $\mathcal{V}_{\Gamma \vdash \mathcal{V}_\sigma}$. Although conceptually clear, the notion employed makes working with term distances rather bureaucratic.

possible (allowing us, for instance, to easily define distances by induction). To this end, we observe that any distance $\mu : X \times Y \to [0,\infty]$ can be characterised as a ternary relation $M \subseteq [0,\infty] \times X \times Y$ defined by:

$$M(a,x,y) \overset{\triangle}{\iff} a \geq \mu(x,y).$$

Vice versa, given such a ternary relation, we define a distance μ as:

$$\mu(x,y) \triangleq \inf\{a \mid M(a,x,y)\}.$$

Therefore, to define a distance μ it is enough to define a suitable ternary relation M. However, not all ternary relations define distances: in fact, ternary relations M induced by distances have two crucial properties: they are monotone and continuous in the first argument, meaning that preserve infimum/intersection. In fact, such features are precisely what is needed to treat distances as ternary relations.

Proposition 10. *There is a one-to-one correspondence between distances $\mu : X \times Y \to [0,\infty]$ and ternary relations $M \subseteq [0,\infty] \times X \times Y$ that are monotone and meet preserving in the first argument. That is:*

$$M(a,x,y) \;\&\; b \geq a \implies M(b,x,y)$$
$$\bigcap_{a \in A} M(a,x,y) = M(\inf A, x, y).$$

In light of Proposition 10, we will work with monotone and meet preserving ternary relations rather than distances, keeping in mind that we can always do back-and-forth between these notions. As a consequence, in the rest of this section, we extend our relational calculus to ternary relations.

Definition 38. 1. *A* closed *term relation is a pair $M = (M^{\Lambda}, M^{\mathcal{V}})$ of maps associating to each type σ monotone and meet preserving ternary relations M^{Λ}_{σ} and $M^{\mathcal{V}}_{\sigma}$ on $[0,\infty] \times \Lambda^{\bullet}_{\sigma} \times \Lambda^{\bullet}_{\sigma}$ and $[0,\infty] \times \mathcal{V}^{\bullet}_{\sigma} \times \mathcal{V}^{\bullet}_{\sigma}$, respectively. We refer to M^{Λ} as the computation component of M, and to $M^{\mathcal{V}}$ as the value component of M.*

2. *An (*open*) term distance M associates to each sequent $\Gamma \vdash \sigma$ a monotone and meet preserving ternary relation envone $\vdash^{\Lambda} M(-,-,-) : \sigma$ on $[0,\infty] \times \Lambda_{\Gamma \vdash^{\Lambda} \sigma} \times \Lambda_{\Gamma \vdash^{\Lambda} \sigma}$ and a monotone and meet preserving ternary relation $\Gamma \vdash^{\mathcal{V}} M(-,-,-) : \sigma$ on $[0,\infty] \times \mathcal{V}_{\Gamma \vdash^{\mathcal{V}} \sigma} \times \mathcal{V}_{\Gamma \vdash^{\mathcal{V}} \sigma}$. We also require term relations to be closed under weakening:*

$$\frac{\Gamma \vdash^{\Lambda} M(a,e,f) : \sigma}{\Gamma + \Delta \vdash^{\Lambda} M(a,e,f) : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} M(a,v,w) : \sigma}{\Gamma + \Delta \vdash^{\mathcal{V}} M(a,v,w) : \sigma}$$

To highlight the intended meaning of a term relation, we use the notation $\Gamma \vdash^{\Lambda} a \geq M(e,f) : \sigma$ in place of $\Gamma \vdash^{\Lambda} M(a,e,f) : \sigma$ (and similarly for values). Indeed, what we are stating is that the M-distance between e and f is (upper) bounded by a, so that the infimum of such as gives the exact distance between e and f.

Example 3. Both the discrete/identity term relation I and the indiscrete relation ∇ defined below are open term relations. The empty relation is a term relation too (notice that the distance associated to such a relation is $\mu(x,y) = \inf \emptyset = \infty$).

$$\frac{}{\Gamma \vdash^{\Lambda} a \geq \mathsf{I}(e,e) : \sigma} \qquad \frac{}{\Gamma \vdash^{\mathcal{V}} a \geq \mathsf{I}(v,v) : \sigma} \qquad \frac{}{\Gamma \vdash^{\Lambda} a \geq \nabla(e,f) : \sigma} \qquad \frac{}{\Gamma \vdash^{\mathcal{V}} a \geq \nabla(v,w) : \sigma}$$

⊠

We overload the notation and write Rel and Rel^{c} for the collections of open and closed term relations, respectively. Formally, we define Rel as $\prod_{\Gamma \vdash \sigma} \mathsf{Rel}(\Lambda_{\Gamma \vdash \sigma}, \Lambda_{\Gamma \vdash \sigma}) \times \mathsf{Rel}(\mathcal{V}_{\Gamma \vdash \sigma}, \mathcal{V}_{\Gamma \vdash \sigma})$. We now endow Rel with a quantale structure. First, let us notice Rel carries a complete lattice structure given by set-theoretic inclusion: $M \subseteq N$ holds if

$$\Gamma \vdash^{\Lambda} a \geq M(e,f) : \sigma \implies \Gamma \vdash^{\Lambda} a \geq N(e,f) : \sigma,$$
$$\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \sigma \implies \Gamma \vdash^{\mathcal{V}} a \geq N(v,w) : \sigma.$$

Next, we give Rel a monoid structure by defining term relation composition $M; N$ as:

$$\frac{\Gamma \vdash^{\wedge} a \geq M(e,g) : \sigma \quad \Gamma \vdash^{\wedge} b \geq N(g,f) : \sigma \quad c \geq a+b}{\Gamma \vdash^{\wedge} c \geq (M;N)(e,f) : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,u) : \sigma \quad \Gamma \vdash^{\mathcal{V}} b \geq N(u,w) : \sigma \quad c \geq a+b}{\Gamma \vdash^{\mathcal{V}} c \geq (M;N)(v,w) : \sigma}$$

Notice that, e.g., in the conclusion of the first rule we have $\Gamma \vdash^{\wedge} c \geq (M;N)(e,f) : \sigma$ rather than $\Gamma \vdash^{\wedge} a + b \geq (M;N)(e,f) : \sigma$. That is a pattern we will always meet when defining relations by mean of rules having more than one premises: in fact, such a patter ensures the resulting relation to be monotone. It is easy to see that $M; N$ is indeed a term relation (it is monotone, closed under weakening, and meet-preserving). Composition is associative and I is its unit, this way making Rel is a monoid. Additionally, easy calculations show that term relation composition is monotone and continuous in both arguments:

$$M \subseteq N \;\&\; L \subseteq O \implies M;L \subseteq N;O$$
$$M;\bigcup_{i\in I} N_i = \bigcup_{i\in I}(M;N_i)$$
$$\bigcup_{i\in I} M_i;N = \bigcup_{i\in I}(M_i;N)$$

This makes Rel a quantale. In particular, we can define term relations both inductively and coinductively. Finally, we observe that Rel also has an involution mapping a term relation M to its converse $M^{\top}$. The latter is defined as follow:

$$\frac{\Gamma \vdash^{\wedge} a \geq M(f,e) : \sigma}{\Gamma \vdash^{\wedge} a \geq M^{\top}(e,f) : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} a \geq M(w,v) : \sigma}{\Gamma \vdash^{\mathcal{V}} a \geq M^{\top}(v,w) : \sigma}$$

As a consequence, we extend the usual notion of a reflexive, transitive, and symmetric relations to term relations.

Definition 39. *A term relation M is reflexive if $\mathsf{I} \subseteq M$; transitive if $M;M \subseteq M$; and symmetric if $M^{\top} \subseteq M$.*

Having analysed the algebra of term relations, we now introduce the ‘syntax-oriented’ constructions upon which our relational calculus is built. Although conceptually similar to the ones introduced in Section 2.3, such constructions are now more involved than their purely (binary) relational counterpart: that is due to the complicated type system of F^{μ}. To simplify our analysis, we introduce right from the beginning some constructions on (quantitative) relations that allow us to extend such relations to structured sets (such product and function spaces).

Definition 40. *Let us write $M : X \twoheadrightarrow Y$ for $M \subseteq [0,\infty] \times X \times Y$. Let $M : X \twoheadrightarrow Y$, $N : U \twoheadrightarrow Z$ be ternary relations.*

1. *Define the Reynolds arrow $[M \to N] : U^X \twoheadrightarrow Z^Y$ by:*

$$[M \to N](a,\alpha,\beta) \overset{\triangle}{\iff} \forall b,c,x,y.\ M(b,x,y) \;\&\; c \geq a+b \implies N(c,\alpha(x),\beta(y))$$

2. *Define the tensor $M \otimes N : X \times U \twoheadrightarrow Y \times Z$ by:*

$$(M \otimes N)(a,(x,u),(y,z)) \overset{\triangle}{\iff} \exists b,c.\ M(b,x,y) \;\&\; N(c,u,z) \;\&\; a \geq b+c.$$

3. *Define the (cartesian) product $M \times N : X \times U \twoheadrightarrow Y \times Z$ by:*

$$(M \times N)(a,(x,u),(y,z)) \overset{\triangle}{\iff} M(a,x,y) \;\&\; N(a,u,z)).$$

4. *Define the sum (or coproduct) $M + N : X + U \twoheadrightarrow Y + Z$ by:*

$$(M+N)(a,t,s) \overset{\triangle}{\iff} [t = in_l(x) \;\&\; s = in_l(y) \;\&\; M(a,x,y)] \text{ or } [t = in_r(u) \;\&\; s = in_r(z) \;\&\; N(a,u,z)]$$

Exercise 39 gives a mathematical explanation of the constructions in Definition 40 in terms of the category of metric spaces. However, such constructions can also be understood operationally. For instance, the relation $[M \to N]$ essentially states that two functions are at at most a-apart if whenever we pass them input at most b-apart, then the outputs are at most $a + b$-apart (as usual, to ensure monotonicity, we actually tale $c \geq a + b$). Notice that $[\mathsf{I} \to M](a, \alpha, \beta)$ holds if and only if $M(a, \alpha(x), \beta(x))$, for any input x, and that all these constructions given obviously extend to term relations.

Exercise 39 (The Category of Generalised Metric Spaces)**.** To motivate the construction given in Definition 40, we move from ternary relations to distances. There, the constructions in Definition 40 give.

1. $[\mu \to \nu] : U^X \to\!\!\!\!\!\to Z^Y$ where:[10]

$$[\mu \to \nu](\alpha, \beta) \triangleq \sup_{x,y} \nu(\alpha(x), \beta(y)) \dot{-} \mu(x, y)$$

2. $\mu \otimes \nu : X \times U \to\!\!\!\!\!\to Y \times Z$ where:

$$(\mu \otimes \nu)((x, u), (y, z)) \triangleq \mu(x, y) + \nu(u, z).$$

3. $\mu \times \nu : X \times U \to\!\!\!\!\!\to Y \times Z$ where:

$$(\mu \times \nu)((x, u), (y, z)) \triangleq \max(\mu(x, y), \nu(u, z)).$$

4. $\mu + \nu : X + U \to\!\!\!\!\!\to Y + Z$ where:

$$(\mu + \nu)(t, s) \triangleq \begin{cases} \mu(x, y) & \text{if } t = in_l(x), s = in_l(y) \\ \nu(u, z) & \text{if } t = in_r(u), s = in_r(z) \end{cases}$$

Notice that $[\mathsf{I} \to \mu](\alpha, \beta) = \sup_x \mu(\alpha(x), \beta(x))$ is the usual sup-metric ones uses to give a metric structure to the space of non-expansive maps. We have a category GenMet whose objects are pairs (X, μ) of sets equipped with reflexive and transitive distances, and whose arrows are non-expansive maps: an arrow $f : (X, \mu) \to (Y, \nu)$ is given by a function $f : X \to Y$ such that $\mu(x, y) \geq \nu(f(x), f(y))$. We denote by $[X, Y]$ the set of non-expansive functions between (X, μ) and (Y, ν).

1. Show that $([X, Y], [\mu \to \nu])$ is an object of GenMet.
2. Show that $(X \times Y, \mu \otimes \nu)$ and $(X \times Y, \mu \times \nu)$ are objects of GenMet.
3. Show that $(X \times Y, \mu \times \nu)$ is the cartesian product of (X, μ) and (Y, ν).
4. Show that $(X + Y, \mu + \nu)$ is the coproduct of (X, μ) and (Y, ν).
5. Is GenMet cartesian closed with exponential object given by $([X, Y], [\mu \to \nu])$?
6. Show that GenMet is a monoidal closed category with tensor product given by $(X \times Y, \mu \otimes \nu)$ and exponential given by $([X, Y], [\mu \to \nu])$. *Hint.* Show that for $\alpha, \beta \in [X, Y]$ one has $[\mu \to \nu](\alpha, \beta) = [\mathsf{I} \to \nu](\alpha, \beta$ (cf. Reynolds-Abramsky Lemma in Exercise 20), using the fact that μ, ν are reflexive and transitive, and that β is non-expansive.

Lemma 30. *The following laws hold.*

$$\begin{aligned}
M \geq L \,\&\, N \geq O &\implies M \otimes N \geq L \otimes O \\
M \geq L \,\&\, N \geq O &\implies M \times N \geq L \times O \\
L \geq M \,\&\, N \geq O &\implies [M \to N] \geq [L \to O] \\
M \geq L \,\&\, N \geq O &\implies M + N \geq L + O
\end{aligned}$$

[10]We write $x \dot{-} y$ for the truncated subtraction of x and y.

Moreover, the following laws hold.

$$\begin{aligned}
\mathsf{I}_{A\times B} &= \mathsf{I}_A \times \mathsf{I}_B = \mathsf{I}_A \otimes \mathsf{I}_B \\
\mathsf{I}_{A\to B} &= [\mathsf{I}_A \to \mathsf{I}_B] \\
\mathsf{I}_{A+B} &= \mathsf{I}_A + \mathsf{I}_B \\
(M \otimes N)^\top &= M^\top \otimes N^\top \\
(M \times N)^\top &= M^\top \times N^\top \\
[M \to N]^\top &= [M^\top \to N^\top] \\
(M + N)^\top &= M^\top + N^\top \\
(M \otimes N); (L \otimes O) &= (M; L) \otimes (N; O) \\
(M \times N); (L \times O) &= (M; L) \times (N; O) \\
[M \to N]; [L \to P] &\geq [M; L \to N; P] \\
(M + N); (L + O) &\geq (M; L) + (N; O)
\end{aligned}$$

The final construction we define is peculiar to F^μ and it acts as a distance scaling operator.

Definition 41. *Given $M : X \nrightarrow Y$ and $s \in [0, \infty]$, we define the relation $[s]M : X \nrightarrow Y$ by:*

$$[s]M(a, x, y) \overset{\triangle}{\iff} \exists b.\ a \geq s \cdot b \ \&\ M(b, x, y).$$

Notice that if M is a term relation, then so is $[s]M$. We use the operator $[s]$ to scale and amplify distances. In fact, if we have $a \geq M(v, w)$ (i.e. $M(a, v, w)$), then by using v, w according to s (viz. s-times), then we expect the distance between v and w to be bounded by $s \cdot a$: and in fact, we have $[s]M(s \cdot a, v, w)$ (i.e. $s \cdot a \geq [s]M(v, w)$).

Lemma 31. *The following laws hold.*

$$\begin{aligned}
\mathsf{I} &\subseteq [s]\mathsf{I} & [1]M &\subseteq M \\
[s]M; [s]N &\subseteq [s](M; N) & [s \cdot r]M &\subseteq [s][r]M \\
M \subseteq N &\implies [s]M \subseteq [s]N & s \leq r &\implies [s]M \supseteq [r]M.
\end{aligned}$$

Exercise 40 (Only if you know category theory)**.** The action of $[s]\mu$ on a distance μ is defined thus: $[s]\mu(x, y) = s \cdot \mu(x, y)$. Show that the map $[s]$ sending a GenMet object (X, μ) to $(X, [s]\mu)$ gives a graded comonad (Gaboardi, Katsumata, Orchard, Breuvart, & Uustalu, 2016; Katsumata, 2018) on GenMet.

For pedagogical reasons, the first construction we introduce is relational substitution. Recall the substitution rule:

$$\frac{\Gamma, \boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\Lambda} e : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \sim \Delta_i}{\Gamma + \sum \boldsymbol{s} \cdot \Delta \vdash^{\Lambda} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma} \qquad \frac{\Gamma, \boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\mathcal{V}} v : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{v} : \boldsymbol{\sigma} \quad \Gamma \sim \Delta_i}{\Gamma + \sum \boldsymbol{s} \cdot \Delta \vdash^{\mathcal{V}} e[\boldsymbol{v}/\boldsymbol{x}] : \sigma}$$

The idea is that given two terms e, f using a variable x according to sensitivity s, if we have two values v, w which are a-apart, then $e[v/x]$ and $f[w/x]$ are essentially $s \cdot a$-apart. The construction $[s]$ models exactly such a behaviour.

Definition 42. *Given term relations M, N, define the* substitution of N into M *as the term relation $M[N]$ defined thus (where we assume $\Gamma \sim \Delta_i$):*[11]

$$\frac{\Gamma, \boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\Lambda} a \geq M(e, f) : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{b} \geq [\boldsymbol{s}]N(\boldsymbol{v}, \boldsymbol{w}) : \boldsymbol{\sigma} \quad c \geq a + \sum \boldsymbol{b}}{\Gamma + \sum \boldsymbol{s} \cdot \Delta \vdash^{\Lambda} c \geq M[N](e[\boldsymbol{v}/\boldsymbol{x}], f[\boldsymbol{v}/\boldsymbol{x}]) : \sigma}$$

$$\frac{\Gamma, \boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\mathcal{V}} a \geq M(e, f) : \sigma \quad \Delta \vdash^{\mathcal{V}} \boldsymbol{b} \geq [\boldsymbol{s}]N(\boldsymbol{v}, \boldsymbol{w}) : \boldsymbol{\sigma} \quad c \geq a + \sum \boldsymbol{b}}{\Gamma + \sum \boldsymbol{s} \cdot \Delta \vdash^{\mathcal{V}} c \geq M[N](v[\boldsymbol{v}/\boldsymbol{x}], u[\boldsymbol{v}/\boldsymbol{x}]) : \sigma}$$

[11] Notice that we write $[\boldsymbol{s}]M(\boldsymbol{v}, \boldsymbol{e}, \boldsymbol{f})$ for $[s_1]M(v_1, e_1, f_1), \ldots, [s_n]M(v_n, e_n, f_n)$.

Lemma 32. *Let M, N, L, P be term relations. Then, the following monotonicity law holds.*

$$M \subseteq L, N \subseteq P \implies M[N] \subseteq L[P]$$

Moreover, we have the following identities.

$$\mathsf{I}[\mathsf{I}] = \mathsf{I}$$
$$M^{\top}[N^{\top}] = (M[N])^{\top}$$
$$(M;L)[N;P] = M[N];L[P]$$

We are now to define the closed restriction and the open/substitutive extension of term relations.

Definition 43. 1. *Given a term relation $M \in \mathsf{Rel}$, define the* closed restriction *$M^{\mathrm{c}} = (M^{\Lambda}, M^{\mathcal{V}})$ of M by*

$$\frac{\cdot \vdash^{\Lambda} a \geq M(e,f) : \sigma}{(M^{\mathrm{c}})^{\Lambda}_{\sigma}(a,e,f)} \qquad \frac{\cdot \vdash^{\Lambda} a \geq M(v,w) : \sigma}{(M^{\mathrm{c}})^{\mathcal{V}}_{\sigma}(a,v,w)}$$

2. *Given a closed term relation M, we define the* open extension *of M as the (open) term relation M^{o} thus defined:*

$$\frac{M^{\Lambda}_{\sigma}(a, e[\boldsymbol{v}/\boldsymbol{x}], f[\boldsymbol{v}/\boldsymbol{x}])}{\boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\Lambda} a \geq M^{\mathrm{o}}(e,f) : \sigma} \qquad \frac{M^{\Lambda}_{\sigma}(a, v[\boldsymbol{v}/\boldsymbol{x}], w[\boldsymbol{v}/\boldsymbol{x}])}{\boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\mathcal{V}} a \geq M^{\mathrm{o}}(v,w) : \sigma}$$

3. *Given a closed term relation M, we define the* substitutive extension *of M as the (open) term relation M^{s} thus defined:*

$$\frac{[\boldsymbol{s}]M^{\mathcal{V}}_{\boldsymbol{\sigma}}(\boldsymbol{b}, \boldsymbol{v}, \boldsymbol{w}) \ \&\ c \geq a + \sum \boldsymbol{b} \implies M^{\Lambda}_{\sigma}(c, e[\boldsymbol{v}/\boldsymbol{x}], f[\boldsymbol{v}/\boldsymbol{x}])}{\boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\Lambda} a \geq M^{\mathrm{s}}(e,f) : \sigma}$$

$$\frac{[\boldsymbol{s}]M^{\mathcal{V}}_{\boldsymbol{\sigma}}(\boldsymbol{b}, \boldsymbol{v}, \boldsymbol{w}) \ \&\ c \geq a + \sum \boldsymbol{b} \implies M^{\mathcal{V}}_{\sigma}(c, v[\boldsymbol{v}/\boldsymbol{x}], w[\boldsymbol{v}/\boldsymbol{x}])}{\boldsymbol{x} :_{\boldsymbol{s}} \boldsymbol{\sigma} \vdash^{\mathcal{V}} a \geq M^{\mathrm{s}}(v,w) : \sigma}$$

Taking advantage of Definition 43, for a *closed* term relation $M = (M^{\Lambda}, M^{\mathcal{V}})$ we will often write $\cdot \vdash^{\Lambda} a \geq M(e,f)f : \sigma$ in place of $M^{\Lambda}_{\sigma}(a,e,f)$ (and similarity for values). In light of that, for an open term relation M we will use the notations $\cdot \vdash^{\Lambda} a \geq M(e,f) : \sigma$ and $(M^{\mathrm{c}})^{\Lambda}_{\sigma}(a,e,f)$ interchangeably (and similarity for values).

Lemma 33. *Let M, N be open term relations, and L, P be closed term relations. The following monotonicity laws hold.*

$$M \subseteq N \implies M^{\mathrm{o}} \subseteq N^{\mathrm{o}}$$
$$M \subseteq N \implies M^{\mathrm{c}} \subseteq N^{\mathrm{c}}$$

Moreover, we have the inclusions:

$$M^{\mathrm{c}}; N^{\mathrm{c}} \subseteq (M;N)^{\mathrm{c}}$$
$$L^{\mathrm{o}}; P^{\mathrm{o}} \subseteq (L;P)^{\mathrm{o}}$$
$$L^{\mathrm{s}}; P^{\mathrm{s}} \subseteq (L;P)^{\mathrm{s}}$$

Exercise 41. Show that the open extension of a reflexive closed term relation is reflexive. Show also that the open extension of a transitive closed term relation is transitive. What goes wrong if you try to prove the same result for substitutive extensions?

Using term relation substitution, we can define the notions of a substitutive and value substitutive term relation.

Definition 44. *A term relation M is* value-substitutive *if $M[\mathsf{I}] \subseteq M$. It is* substitutive *if $M[M] \subseteq M$.*

As usual, a closed term relation is (value) substitutive if its open extension is. Moreover, we notice that the open extension of a closed term relation is trivially value-substitutive.

Exercise 42. Let M be a closed term relation. Show that the open extension of M is value substitutive and that the substitutive extension of M is substitutive:

$$M^{\circ}[\mathsf{I}] \subseteq M^{\circ}$$
$$M^{s}[M^{s}] \subseteq M^{s}$$

Finally, we introduce the notion of a *compatible refinement*. Again, the complicated type systems of F^{μ} makes the definition of a compatible refinement more involved than its purely relational counterparts.

Definition 45. *The* compatible refinement $\widehat{M}$ *of an open term relation M is defined by the rules in* Figure 3.3*. We say that M is compatible if $\widehat{M} \subseteq M$, and that a closed term relation is compatible if its open extension is.*

The notion of *compatibility* captures an abstract form of Lipshitz-continuity with respect to F^{μ} constructors. Notice that in the clause for sequencing the presence of $\max(s, 1)$, instead of s, ensures that for terms like $e \triangleq$ **let** x = (**return true**) **in** (**return true**) and $f \triangleq$ **let** $x = \Omega$ **in** (**return true**) the distance between e and f is determined *before* sequencing (which captures the idea that although **return true** will not 'use' any input, **return true** and Ω will be still evaluated, thus producing observable differences between e and f). In fact, if we replace $\max(s, 1)$ with just s, then by taking $s \triangleq 0$ compatibility would imply that e and f are at distance 0, which is clearly unsound.

Exercise 43. Show that $\widehat{M}$ is a term relation.

Lemma 34. *The following hold:*

$$\widehat{M; N} = \widehat{M}; \widehat{N}$$
$$\widehat{M^{\top}} = (\widehat{M})^{\top}$$
$$M \subseteq N \implies \widehat{M} \subseteq \widehat{N}.$$

Definition 45 induces a map $M \mapsto \widehat{M}$ on the collection of open term relations which is monotone, by Lemma 34. In particular, a term relation is compatible if and only if it is a pre-fixed point of $M \mapsto \widehat{M}$.

Lemma 35. *The discrete term relation I is the least pre-fixed point of $M \mapsto \widehat{M}$. I.e. $\mathsf{I} \subseteq \widehat{\mathsf{I}}$ and $M \subseteq \widehat{M} \implies \mathsf{I} \subseteq M$. As a consequence, any compatible relation is reflexive.*

Exercise 44. Show that for any closed term relation M we have:

$$\widehat{M} \subseteq \mathsf{I}[M]$$

Conclude that $\widehat{M} \subseteq \mathsf{I}[M]$ implies $\widehat{M} \subseteq M$ (i.e. substitutivity of M). What about the converse implication?

Summing up, we have extended the relational calculus of Section 2.3 to ternary, quantitative relations. The resulting calculus is more complicated than its 'qualitative' counterpart, but that is mostly due to the complicated syntax and type system of F^{μ}. From a structural perspective, the qualitative and quantitative relational calculi are surprisingly similar. That allows us to extend all the notions of program equivalence seen so far to a metric-like setting. Moreover, all results whose proofs relied on the relational calculus only extend to F^{μ} essentially for free. We witness that by extending the notions of equivalence seen so far to the new ternary and quantitative setting.

3.5 Metric Logical Relations

Thanks to Definition 40, defining metric logical relations is straightforward. To avoid unnecessary complications hiding the core ideas behind metric logical relations, we restrict our analysis to F^{μ} without recursive types. A standard reducibility argument shows that the calculus thus obtained is strongly normalising, so that we have $[\![e]\!]_{\mathcal{E}} \in \mathcal{V}_{\sigma}^{\bullet}$, for any program of type σ.

As usual, we begin by giving an explicit definition of the logical distance.

$$\frac{}{\Gamma, x :_s \sigma \vdash^{\mathcal{V}} a \geq \widehat{M}(x,x) : \sigma}$$

$$\frac{a \geq \Gamma, x :_1 \sigma \vdash^{\Lambda} M(e,f) : \tau}{\Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(\lambda x.e, \lambda x.f) : \sigma \multimap \tau} \qquad \frac{a \geq \Gamma \vdash^{\mathcal{V}} M(v,u) : \sigma \multimap \tau \quad b \geq \Delta \vdash^{\mathcal{V}} M(w,z) : \sigma \quad c \geq a + b}{\Gamma + \Delta \vdash^{\Lambda} c \geq \widehat{M}(vw, uz) : \tau}$$

$$\frac{}{\Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(\langle\rangle, \langle\rangle) : \mathbf{unit}} \qquad \frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,u) : \sigma \quad \Gamma \vdash^{\mathcal{V}} a \geq M(w,z) : \tau}{\Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(\langle v, w\rangle, \langle u, z\rangle) : \sigma \times \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \sigma \times \tau}{\Gamma \vdash^{\Lambda} a \geq \widehat{M}(\mathbf{proj}_l\ v, \mathbf{proj}_l\ w) : \sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \sigma \times \tau}{\Gamma \vdash^{\Lambda} a \geq \widehat{M}(\mathbf{proj}_r\ v, \mathbf{proj}_r\ w) : \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,u) : \sigma \quad \Delta \vdash^{\mathcal{V}} b \geq M(w,z) : \tau \quad c \geq a + b}{\Gamma + \Delta \vdash^{\mathcal{V}} c \geq \widehat{M}(v \otimes w, u \otimes z) : \sigma \otimes \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq [s]M(v,w) : \sigma \otimes \tau \quad \Delta, x :_s \sigma, y :_s \tau \vdash^{\Lambda} b \geq M(e,f) : \rho \quad c \geq a + b}{s \cdot \Gamma + \Delta \vdash^{\Lambda} c\, \widehat{M}\ (\mathbf{let}\ x \otimes y = v\ \mathbf{in}\ e, \mathbf{let}\ x \otimes y = w\ \mathbf{in}\ f : \rho}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \mathbf{void}}{\Gamma \vdash^{\Lambda} a \geq \widehat{M}(\mathbf{abort}\ v, \mathbf{abort}\ w) : \sigma}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \sigma}{\Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(\mathbf{inl}\ v, \mathbf{inl}\ w) : \sigma + \tau} \qquad \frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \tau}{\Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(\mathbf{inr}\ v, \mathbf{inr}\ w) : \sigma + \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq [s]M(v,w) : \sigma + \tau \quad \Delta, x :_s \sigma \vdash^{\Lambda} b \geq M(e,g) : \rho \quad \Delta, y :_s \tau \vdash^{\Lambda} b \geq M(f,h) : \rho \quad c \geq a + b}{s \cdot \Gamma + \Delta \vdash^{\Lambda} c \geq \widehat{M}(\mathbf{case}\ v\ \mathbf{of}\ \{\langle i, x\rangle \to e\} f, \mathbf{case}\ w\ \mathbf{of}\ \{\langle i, x\rangle \to g\} h) : \rho}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \sigma[\mu t.\sigma/t]}{\Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(\mathbf{fold}\ v, \mathbf{fold}\ w) : \mu t.\sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \mu t.\tau}{\Gamma \vdash^{\Lambda} a \geq \widehat{M}(\mathbf{unfold}\ v, \mathbf{unfold}\ w) : \sigma[\mu t.\sigma/t]}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq [s]M(v,w) : \sigma}{s \cdot \Gamma \vdash^{\mathcal{V}} a \geq \widehat{M}(!v, !w) : !_s\sigma} \qquad \frac{\Gamma \vdash^{\mathcal{V}} a \geq [r]M(v,w) : !_s\sigma \quad \Delta, x :_{r \cdot s} \sigma \vdash^{\Lambda} b \geq M(e,f) : \tau \quad c \geq a + b}{r \cdot \Gamma + \Delta \vdash^{\Lambda} c \geq \widehat{M}(\mathbf{let}\ !x = v\ \mathbf{in}\ e, \mathbf{let}\ !x = w\ \mathbf{in}\ f) : \tau}$$

$$\frac{\Gamma \vdash^{\mathcal{V}} a \geq M(v,w) : \sigma}{\Gamma \vdash^{\Lambda} a \geq \widehat{M}(\mathbf{return}\ v, \mathbf{return}\ w) : \sigma} \qquad \frac{\Gamma \vdash^{\Lambda} a \geq [\max(s,1)]M(e,g) : \tau \quad \Gamma, x :_s \tau \vdash^{\Lambda} b \geq M(f,h) : \sigma \quad c \geq a + b}{\max(s,1) \cdot \Gamma + \Delta \vdash^{\Lambda} c \geq \widehat{M}(\mathbf{let}\ x = e\ \mathbf{in}\ f, \mathbf{let}\ x = g\ \mathbf{in}\ h) : \sigma}$$

Figure 3.3: Compatible refinement F^{μ}.

$$\frac{\forall v, w.\ v \overset{\mathsf{L}}{\sim}_b w \ \&\ c \geq a + b \implies e[v/x] \overset{\mathsf{L}}{\sim}_c f[w/x]}{\lambda x.e \overset{\mathsf{L}}{\sim}_a \lambda x.f : \sigma \multimap \tau}$$

$$\frac{}{\langle\rangle \overset{\mathsf{L}}{\sim}_a \langle\rangle : \mathbf{unit}} \qquad \frac{v \overset{\mathsf{L}}{\sim}_a u : \sigma \quad w \overset{\mathsf{L}}{\sim}_a z : \tau}{\langle v, w\rangle \overset{\mathsf{L}}{\sim}_a \langle u, z\rangle : \sigma \times \tau} \qquad \frac{v \overset{\mathsf{L}}{\sim}_a u : \sigma \quad w \overset{\mathsf{L}}{\sim}_b z : \tau \quad c \geq a + b}{v \otimes w \overset{\mathsf{L}}{\sim}_c u \otimes z : \sigma \otimes \tau}$$

$$\frac{v \overset{\mathsf{L}}{\sim}_a w : \sigma}{\mathbf{inl}\ v \overset{\mathsf{L}}{\sim}_a \mathbf{inl}\ w : \sigma + \tau} \qquad \frac{v \overset{\mathsf{L}}{\sim}_a w : \tau}{\mathbf{inr}\ v \overset{\mathsf{L}}{\sim}_a \mathbf{inr}\ w : \sigma + \tau}$$

$$\frac{v \overset{\mathsf{L}}{\sim}_a w : \sigma \quad b \geq s \cdot a}{!v \overset{\mathsf{L}}{\sim}_b !w) : !_s\sigma} \qquad \frac{[\![e]\!]_{\mathcal{E}} \overset{\mathsf{L}}{\sim}_a [\![f]\!]_{\mathcal{E}}}{e \overset{\mathsf{L}}{\sim}_a f : \sigma}$$

Figure 3.4: Logical distance F^{μ}.

Definition 46. *The* closed *term relation* $\overset{\mathsf{L}}{\sim}$*, called* logical distance *(or metric logical equivalence) is defined recursively on types by the rules in Figure 3.4.*[12] *Logical distance is then extended to open expressions as* $\overset{\mathsf{L}}{\sim}^{s}$.

We now give a more relational definition of logical distance. To do so, we unpack the logical meaning of types exactly as we did for Λ^{ST}.

Definition 47. *For each type* σ*, we define the logical meaning function* $[\![-]\!]_{\mathcal{L}} : \mathcal{V}^{\bullet}_{\sigma} \to \mathcal{L}(\mathcal{V}^{\bullet}_{\sigma})$*, where*

$$\begin{aligned}
\mathcal{L}(\mathcal{V}^{\bullet}_{\mathbf{unit}}) &\triangleq \mathcal{V}_{\mathbf{unit}} \\
\mathcal{L}(\mathcal{V}^{\bullet}_{\mathbf{void}}) &\triangleq \emptyset \\
\mathcal{L}(\mathcal{V}^{\bullet}_{\sigma\times\tau}) &\triangleq \mathcal{V}^{\bullet}_{\sigma} \times \mathcal{V}^{\bullet}_{\tau} \\
\mathcal{L}(\mathcal{V}^{\bullet}_{\sigma\otimes\tau}) &\triangleq \mathcal{V}^{\bullet}_{\sigma} \times \mathcal{V}^{\bullet}_{\tau} \\
\mathcal{L}(\mathcal{V}^{\bullet}_{\sigma+\tau}) &\triangleq \mathcal{V}^{\bullet}_{\sigma} + \mathcal{V}^{\bullet}_{\tau} \\
\mathcal{L}(\mathcal{V}^{\bullet}_{!_s\sigma}) &\triangleq \mathcal{V}^{\bullet}_{\sigma} \\
\mathcal{L}(\mathcal{V}^{\bullet}_{\sigma\multimap\tau}) &\triangleq \mathcal{V}^{\bullet}_{\sigma} \to \Lambda^{\bullet}_{\tau}.
\end{aligned}$$

by:

$$\begin{aligned}
[\![\langle\rangle]\!]_{\mathcal{L}} &\triangleq \langle\rangle \\
[\![\langle v, w\rangle]\!]_{\mathcal{L}} &\triangleq (v, w) \\
[\![v \otimes w]\!]_{\mathcal{L}} &\triangleq (v, w) \\
[\![\mathbf{inl}\ v]\!]_{\mathcal{L}} &\triangleq in_l(v) \\
[\![\mathbf{inr}\ v]\!]_{\mathcal{L}} &\triangleq in_r(v) \\
[\![\lambda x.f]\!]_{\mathcal{L}} &\triangleq v \mapsto f[v/x]
\end{aligned}$$

We can now define logical distance in relational and pointfree style.

Definition 48. *Define logical distance* $\overset{\mathsf{L}}{\sim}$ *as follows:*

$$\begin{aligned}
\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}} \\
\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma} \times \overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma\otimes\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma} \otimes \overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma+\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma} + \overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma\multimap\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma} \to \overset{\mathsf{L}}{\sim}^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{!_s\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; [s]\overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \overset{\mathsf{L}}{\sim}^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}.
\end{aligned}$$

[12] For readability, write $e \overset{\mathsf{L}}{\sim}_a f$ in place of $a \geq e \overset{\mathsf{L}}{\sim} f$.

As usual, we extend $\overset{\mathsf{L}}{\sim}$ to an open term relation by taking its substitutive extension $\overset{\mathsf{L}}{\sim}^{s}$. Let us now move to the structural properties of $\overset{\mathsf{L}}{\sim}$. Since substitutive extensions are always substitutive ($M^{s}[M^{s}] \subseteq M^{s}$ holds for any closed term relation M), $\overset{\mathsf{L}}{\sim}^{s}$ is trivially substitutive. Moreover, exactly the same relational calculation we did for Λ^{ST} show that compatibility and transitivity of $\overset{\mathsf{L}}{\sim}$ follow from reflexivity. As a consequence, the only thing we need is a quantitative, metric version of the fundamental lemma.

Proposition 11 (Fundamental Lemma)**.** $\mathsf{I} \subseteq \overset{\mathsf{L}}{\sim}^{s}$.

Proof. □

Theorem 7. *$\overset{\mathsf{L}}{\sim}$ is a reflexive, symmetric, transitive, compatible, and substitutive term relation.*

Proof. Since $\overset{\mathsf{L}}{\sim}$ is transitive and symmetric (exercise!), then so is $\overset{\mathsf{L}}{\sim}^{s}$. All the rest follows by general relational calculations. □

Exercise 45. Show that $\overset{\mathsf{L}}{\sim}$ is transitive and symmetric.

3.6 Applicative Bisimilarity Distance

At this point, it should be clear that our quantitative relational calculus allows us to extend applicative (bi)similarity to a quantitative setting smoothly, exactly as we did for logical equivalence. Defining a quantitative counterpart of applicative (bi)similarity (known as applicative (bi)similarity distance), however, requires us to introduce an additional construction on term relation, to handle the fact that programs may diverge.

Definition 49. *Given a relation $M : X \twoheadrightarrow Y$, the relation $\widetilde{\mathcal{M}}M : X_{\perp} \twoheadrightarrow Y_{\perp}$ is defined by:*

$$\widetilde{\mathcal{M}}M(a,t,s) \overset{\triangle}{\iff} (t = \mathit{just}\ x \implies s = \mathit{just}\ y \,\&\, M(a,x,y))$$

Lemma 36. *The map $\widetilde{\mathcal{M}}$ satisfies the following laws.*

$$\begin{aligned}
\mathsf{I}_{A_{\perp}} &\subseteq \widetilde{\mathcal{M}}\mathsf{I}_{A}\\
\widetilde{\mathcal{M}}M;\widetilde{\mathcal{M}}N &\subseteq \widetilde{\mathcal{M}}(M;N)\\
M \subseteq N &\implies \widetilde{\mathcal{M}}M \subseteq \widetilde{\mathcal{M}}N.
\end{aligned}$$

We now have all the ingredients to define applicative similarity as the greatest fixed point of a monotone operator.

Definition 50. *Given a closed term relation M, we define the closed term relation $[M]$ as follows:*

$$\begin{aligned}
[M]^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}}\\
[M]^{\mathcal{V}}_{\mathbf{void}} &\triangleq \emptyset\\
[M]^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma} \times M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}}\\
[M]^{\mathcal{V}}_{\sigma\otimes\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma} \otimes M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}}\\
[M]^{\mathcal{V}}_{\sigma+\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma} + M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}}\\
[M]^{\mathcal{V}}_{\mu t.\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma[t/\mu t.\sigma]}; [\![-]\!]^{\top}_{\mathcal{L}}\\
[M]^{\mathcal{V}}_{!_{s}\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; [s]M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}}\\
[M]^{\mathcal{V}}_{\sigma\multimap\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to M^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}}\\
M^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \widetilde{\mathcal{M}}M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}
\end{aligned}$$

A term relation M is an applicative simulation *if $M \subseteq [M]$.*

Lemma 37. *The map $[-]$ is a monotone function on the complete lattice of closed relation.*

Proof. All the relational constructions used in Definition 58 are monotone. □

As a consequence, we can define applicative similarity as the greatest fixed point of $[-]$.

Definition 51. *Define* $\overset{a}{\preceq} \triangleq \nu M.[M]$.

Since applicative similarity is defined as a greatest fixed point, it comes with the usual associated *coinduction proof principle*:

$$M \subseteq [M] \implies M \subseteq \overset{a}{\preceq}.$$

Relying on the latter principle as well as on the relational calculus, we can prove that applicative similarity is reflexive and transitive, pretty much in the same way as we did for Λ^{μ}. Additionally, it is also possible to reproduce Howe's method in the more general setting of quantitative relations to show that applicative similarity is also compatible and substitutive. That, however, is more difficult than its purely relational counterparts, and thus beyond the scope of these notes. We only mention that one first has to show that $\widetilde{\mathcal{M}}$ has to satisfy all the 'good properties' we used to prove applicative (bi)similarity to be compatible. For instance, we need to have the algebraic law

$$M \subseteq \alpha; \widetilde{\mathcal{M}}N; \beta^{\top} \implies \widetilde{\mathcal{M}}M \subseteq {\gg\!=}\alpha; \widetilde{\mathcal{M}}N; ({\gg\!=}\beta)^{\top}.$$

for all functions $\alpha, \beta : X \to A_{\perp}$ and relations $M : X \nrightarrow X$, $N : A \nrightarrow A$. But that is not the end of the story, in fact now one of the crucial point to make Howe's method working, is showing that $[s]$ and $\widetilde{\mathcal{M}}$ properly interacts. That is indeed the case, as proved by the following lax distributive law.

Lemma 38. *For any $s \geq 1$, we have:*

$$[s](\widetilde{\mathcal{M}}M) \subseteq \widetilde{\mathcal{M}}([s]M).$$

3.7 Denotational Semantics Through Metric Spaces

Another way to define distances between programs is by means of denotational semantics. More specifically, we refine the set-theoretic[13] denotational semantics given to Λ^{ST} into a metric space semantics: accordingly, we interpret types σ as metric spaces $[\![\sigma]\!]_{\mathcal{D}} = (X_{\tau}, \mu_{\sigma})$ and expressions as suitable non-expansive functions. Crucial is the interpretation of the bang modality $!_s$: the latter scales the distance μ_{σ} of a factor s. This way, we obtain s-Lipshitz continuous functions from X to Y as non-expansive functions from $!_s X$ to Y.

Definition 52. *The category* Met *has pairs (X, μ) made of a set X and a reflexive, symmetric, and transitive distance (i.e. a pseudometric) on X as objects, and non-expansive functions as arrows: that is, an arrow from (X, μ) to (Y, ν) is a function $f : X \to Y$ such that $\mu(x, y) \geq \nu(f(x), f(y))$. We refer to objects of* Met *as metric spaces and write* $\mathrm{Met}(X, Y)$ *for the collection of non-expansive functions between (X, μ) and (Y, ν).*

Proposition 12. *The category* Met *is a monoidal closed category with cartesian products and coproducts.*

Proof. Repeat the constructions of Exercise 39. For instance, the monoidal product $\mathcal{X} \otimes \mathcal{Y}$ of two metric spaces $\mathcal{X} = (X, \mu)$ to $\mathcal{Y} = (Y, \nu)$ is defined as $(X \times Y, \mu \otimes \nu)$. Similarly, we define $[\mathcal{X}, \mathcal{Y}] \triangleq (\mathrm{Met}(X, Y), [\mathrm{I}_X \to \nu])$ (so that $[\mathrm{I}_X \to \nu](\alpha, \beta) = \sup_x \nu(\alpha(x), \beta(x))$). □

Any $s \in [0, \infty]$ defines a map sending a metric space $\mathcal{X} = (X, \mu)$ to the metric space $!_s\mathcal{X} \triangleq (X, s \cdot \mu)$, where $(s \cdot \mu)(x, y) \triangleq s \cdot \mu(x, y)$. Moreover, the action of $!_s$ extends to non-expansive functions simply by mapping a function f to itself. It is straightforward to see that $!_s$ thus gives a functor on Met. Actually, $!_s$ gives a graded comonad on Met (but for our purposes, such a level of abstraction is not needed).

[13] To avoid domain theoretic arguments, we give denotational semantics to the fragment of F^{μ} without recursive types.

Exercise 46. Show that we have:

$$!_s(\mathcal{X} \otimes \mathcal{Y}) \cong !_s\mathcal{X} \otimes !_s\mathcal{Y}$$
$$!_s!_r\mathcal{X} \cong !_{s\cdot r}\mathcal{X}.$$

Definition 53. *For each type σ, we define a metric space $[\![\sigma]\!]_{\mathcal{D}}$ as follows, where $1 \triangleq (\{*\}, \iota)$ with $\iota(*, *) = 0$.*

$$\begin{aligned}
[\![\mathbf{unit}]\!]_{\mathcal{D}} &\triangleq 1 \\
[\![\sigma \times \tau]\!]_{\mathcal{D}} &\triangleq [\![\sigma]\!]_{\mathcal{D}} \times [\![\tau]\!]_{\mathcal{D}} \\
[\![\sigma \otimes \tau]\!]_{\mathcal{D}} &\triangleq [\![\sigma]\!]_{\mathcal{D}} \otimes [\![\tau]\!]_{\mathcal{D}} \\
[\![\sigma + \tau]\!]_{\mathcal{D}} &\triangleq [\![\sigma]\!]_{\mathcal{D}} + [\![\tau]\!]_{\mathcal{D}} \\
[\![\sigma \multimap \tau]\!]_{\mathcal{D}} &\triangleq [[\![\sigma]\!]_{\mathcal{D}}, [\![\tau]\!]_{\mathcal{D}}] \\
[\![!_s\sigma]\!]_{\mathcal{D}} &\triangleq !_s[\![\sigma]\!]_{\mathcal{D}}.
\end{aligned}$$

Next, we define the denotational semantics of judgments $x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash^{\Lambda} e : \sigma$ and $x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n \vdash^{\mathcal{V}} v : \sigma$ as non-expansive maps[14]

$$[\![e]\!]_{\mathcal{D}} : !_{s_1}[\![\sigma_1]\!]_{\mathcal{D}} \otimes !_{s_n}[\![\sigma_n]\!]_{\mathcal{D}} \to [\![\sigma]\!]_{\mathcal{D}}$$
$$[\![v]\!]_{\mathcal{D}} : !_{s_1}[\![\sigma_1]\!]_{\mathcal{D}} \otimes !_{s_n}[\![\sigma_n]\!]_{\mathcal{D}} \to [\![\sigma]\!]_{\mathcal{D}}.$$

For $\Gamma = x_1 :_{s_1} \sigma_1, \ldots, x_n :_{s_n} \sigma_n$, we define $[\![\Gamma]\!]_{\mathcal{D}} \triangleq !_{s_1}[\![\sigma_1]\!]_{\mathcal{D}} \otimes !_{s_n}[\![\sigma_n]\!]_{\mathcal{D}}$. Notice that $[\![\emptyset]\!]_{\mathcal{D}} = 1$. The following result relates the denotational semantics of typing environments with the operations for adding and scaling such environments.

Lemma 39. *1. The map $\delta : [\![\Gamma + \Delta]\!]_{\mathcal{D}} \to [\![\Gamma]\!]_{\mathcal{D}} \otimes [\![\Delta]\!]_{\mathcal{D}}$ defined by $\delta(x) \triangleq (x, x)$ is non-expansive.*

2. We have $[\![s \cdot \Gamma]\!]_{\mathcal{D}} = !_s[\![\Gamma]\!]_{\mathcal{D}}$.

We now define the denotational semantics of typing judgments by induction. Notice that whenever $s \geq 1$, the map $p : !_s\mathcal{X} \to \mathcal{X}$ sending x to itself is non-expansive. For most terms the definition of $[\![-]\!]_{\mathcal{D}}$ follows the same pattern given for Λ^{ST} (they both rely on exponentials, products, etc). We thus show only the relevant cases.

$$[\![\Gamma, x :_s \sigma]\!]_{\mathcal{D}} : [\![\Gamma]\!]_{\mathcal{D}} \otimes !_s[\![\sigma]\!]_{\mathcal{D}} \to [\![\tau]\!]_{\mathcal{D}} \triangleq p \circ \pi_2$$

- Given $\dfrac{\Gamma \vdash^{\mathcal{V}} v : \sigma \quad \Delta \vdash^{\mathcal{V}} w : \tau}{\Gamma + \Delta \vdash^{\mathcal{V}} v \otimes w : \sigma \otimes \tau}$ we have $[\![v]\!]_{\mathcal{D}} : [\![\Gamma]\!]_{\mathcal{D}} \to [\![\sigma]\!]_{\mathcal{D}}$ and $[\![w]\!]_{\mathcal{D}} : [\![\Delta]\!]_{\mathcal{D}} \to [\![\tau]\!]_{\mathcal{D}}$. Define:

$$[\![v \otimes w]\!]_{\mathcal{D}} \triangleq [\![\Gamma + \Delta]\!]_{\mathcal{D}} \xrightarrow{\delta} [\![\Gamma]\!]_{\mathcal{D}} \otimes [\![\Delta]\!]_{\mathcal{D}} \xrightarrow{[\![v]\!]_{\mathcal{D}} \otimes [\![w]\!]_{\mathcal{D}}} [\![\sigma]\!]_{\mathcal{D}} \otimes [\![\tau]\!]_{\mathcal{D}}$$

- Given $\dfrac{\Gamma \vdash^{\mathcal{V}} v : \sigma \otimes \tau \quad \Delta, x :_s \sigma, x :_s \tau \vdash^{\Lambda} e : \rho}{s \cdot \Gamma + \Delta \vdash^{\Lambda} \mathbf{let}\ x \otimes y = v\ \mathbf{in}\ e : \rho}$ we have $[\![v]\!]_{\mathcal{D}} : [\![\Gamma]\!]_{\mathcal{D}} \to [\![\sigma]\!]_{\mathcal{D}}$ and $[\![e]\!]_{\mathcal{D}} : [\![\Delta]\!]_{\mathcal{D}} \otimes$

[14]For readability, we write $[\![e]\!]_{\mathcal{D}}$ in place of $[\![\Gamma \vdash^{\Lambda} e : \sigma]\!]_{\mathcal{D}}$ (and similarly for values).

$!_s[\![\sigma]\!]_{\mathcal{D}} \otimes !_s[\![\tau]\!]_{\mathcal{D}} \to [\![\rho]\!]_{\mathcal{D}}$. Define $[\![\textbf{let } x \otimes y = v \textbf{ in } e]\!]_{\mathcal{D}}$ as:

$$\begin{tikzcd}
{[\![s \cdot \Gamma + \Delta]\!]_{\mathcal{D}}} \arrow[d, "\delta"] \\
{[\![s \cdot \Gamma]\!]_{\mathcal{D}} \otimes [\![\Delta]\!]_{\mathcal{D}}} \arrow[d, "\cong"] \\
{[\![\Delta]\!]_{\mathcal{D}} \otimes !_s[\![\Gamma]\!]_{\mathcal{D}}} \arrow[d, "1_{[\![\Delta]\!]_{\mathcal{D}}} \otimes !_s[\![v]\!]_{\mathcal{D}}"] \\
{[\![\Delta]\!]_{\mathcal{D}} \otimes !_s(\sigma \otimes \tau)} \arrow[d, "\cong"] \\
{[\![\Delta]\!]_{\mathcal{D}} \otimes !_s\sigma \otimes !_s\tau} \arrow[d, "{[\![e]\!]_{\mathcal{D}}}"] \\
{[\![\rho]\!]_{\mathcal{D}}}
\end{tikzcd}$$

- Given $\dfrac{\Gamma \circ e : \sigma \quad \Delta, x :_s \sigma \vdash^{\Lambda} f}{\max(s,1) \cdot \Gamma + \Delta \vdash^{\Lambda} \textbf{let } x = e \textbf{ in } f : \tau}$ we have $[\![e]\!]_{\mathcal{D}} : [\![\Gamma]\!]_{\mathcal{D}} \to [\![\sigma]\!]_{\mathcal{D}}$ and $[\![f]\!]_{\mathcal{D}} : [\![\Delta]\!]_{\mathcal{D}} \otimes !_s[\![\sigma]\!]_{\mathcal{D}} \to [\![f]\!]_{\mathcal{D}}$. Define $[\![\textbf{let } x = e \textbf{ in } f]\!]_{\mathcal{D}}$ as:

$$\begin{tikzcd}
{[\![\max(s,1) \cdot \Gamma + \Delta]\!]_{\mathcal{D}}} \arrow[d, "\delta"] \\
{[\![\max(s,1) \cdot \Gamma]\!]_{\mathcal{D}} \otimes [\![\Delta]\!]_{\mathcal{D}}} \arrow[d, "\cong"] \\
{[\![\Delta]\!]_{\mathcal{D}} \otimes !_{\max(s,1)}[\![\Gamma]\!]_{\mathcal{D}}} \arrow[d, "1_{[\![\Delta]\!]_{\mathcal{D}}} \otimes !_{\max(s,1)}[\![e]\!]_{\mathcal{D}}"] \\
{[\![\Delta]\!]_{\mathcal{D}} \otimes !_{\max(s,1)}[\![\sigma]\!]_{\mathcal{D}}} \arrow[d] \\
{[\![\Delta]\!]_{\mathcal{D}} \otimes !_s[\![\Gamma]\!]_{\mathcal{D}}} \arrow[d, "{[\![f]\!]_{\mathcal{D}}}"] \\
{[\![\tau]\!]_{\mathcal{D}}}
\end{tikzcd}$$

where we use the fact that the map $(x \mapsto x) : !_{\max(s,1)}\mathcal{X} \to !_s\mathcal{X}$ is non-expansive.

- Given $\dfrac{\Gamma \vdash^{\mathcal{V}} v : !_s\sigma \quad \Delta, x :_{rs} \sigma \vdash^{\Lambda} e : \tau}{r \cdot \Gamma + \Delta \vdash^{\Lambda} \textbf{let } !x = v \textbf{ in } e : \tau}$ we have $[\![v]\!]_{\mathcal{D}} : !_s[\![\Gamma]\!]_{\mathcal{D}} \to [\![\sigma]\!]_{\mathcal{D}}$ and $[\![e]\!]_{\mathcal{D}} : [\![\Delta]\!]_{\mathcal{D}} \otimes !_{rs}[\![\sigma]\!]_{\mathcal{D}} \to$

$\llbracket \tau \rrbracket_{\mathcal{D}}$. Define $\llbracket \textbf{let } !x = v \textbf{ in } e \rrbracket_{\mathcal{D}}$ as:

$$\begin{tikzcd}
\llbracket r \cdot \Gamma + \Delta \rrbracket_{\mathcal{D}} \arrow[d, "\delta"] \\
\llbracket r \cdot \Gamma \rrbracket_{\mathcal{D}} \otimes \llbracket \Delta \rrbracket_{\mathcal{D}} \arrow[d, "\cong"] \\
\llbracket \Delta \rrbracket_{\mathcal{D}} \otimes !_r \llbracket \Gamma \rrbracket_{\mathcal{D}} \arrow[d, "1_{\llbracket \Delta \rrbracket_{\mathcal{D}}} \otimes !_r \llbracket v \rrbracket_{\mathcal{D}}"] \\
\llbracket \Delta \rrbracket_{\mathcal{D}} \otimes !_r !_s \sigma \arrow[d, "\cong"] \\
\llbracket \Delta \rrbracket_{\mathcal{D}} \otimes !_{rs} \sigma \arrow[d, "\llbracket e \rrbracket_{\mathcal{D}}"] \\
\llbracket \tau \rrbracket_{\mathcal{D}}
\end{tikzcd}$$

Exercise 47. Why $\llbracket \Gamma \vdash^{\Lambda} e : \sigma \rrbracket_{\mathcal{D}}$ and $\llbracket \Gamma \vdash^{\mathcal{V}} v : \sigma \rrbracket_{\mathcal{D}}$ are non-expansive? (*Hint.* You do not need any calculation to answer the question).

The denotation of a program e of type σ is thus a non-expansive function $\llbracket e \rrbracket_{\mathcal{D}} : 1 \to \llbracket \sigma \rrbracket_{\mathcal{D}}$, which can be regarded as an element of $\llbracket \sigma \rrbracket_{\mathcal{D}}$. As a consequence, to measure the distance between two programs (of type σ) we simply regard their denotations as elements in the state space of $\llbracket \sigma \rrbracket_{\mathcal{D}}$ and use the denotational pseudometric of $\llbracket \sigma \rrbracket_{\mathcal{D}}$ to measure the distance between $\llbracket e \rrbracket_{\mathcal{D}}$ and $\llbracket f \rrbracket_{\mathcal{D}}$.

3.8 Metrics and Probabilistic Lambda Calculi

We can endow F^{μ} with a family of probabilistic primitives $\mathbf{sample}_q$ exactly as we did for Λ^{ST} and Λ^{μ}. We call the resulting calculus F^{PR}. The static semantics of F^{PR} is obtained from the one of F^{μ} by adding the following rule.

$$\frac{}{\Gamma \vdash^{\Lambda} \mathbf{sample}_q : \mathbf{bool}}$$

The dynamics semantics of F^{PR}, instead, is defined by evaluating programs to subdistributions of values.

Definition 54. *The $\mathbb{N}$-indexed family of functions $[\![-]\!]^n_{\mathcal{E}} : \mathcal{P}\Lambda^\bullet_\sigma \to \mathbf{DV}_\sigma$ is inductively defined as follows:*[15]

$$
\begin{aligned}
[\![e]\!]^0_{\mathcal{E}} &\triangleq \emptyset \\
[\![\mathbf{return}\ v]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_v \\
[\![\mathbf{sample}_q]\!]^{n+1}_{\mathcal{E}} &\triangleq q \cdot \delta_{\mathbf{true}} + (1-q)\cdot \delta_{\mathbf{false}} \\
[\![\mathbf{proj}_l\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_v \\
[\![\mathbf{proj}_r\ \langle v, w\rangle]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_w \\
[\![\mathbf{let}\ x \otimes y = v \otimes w\ \mathbf{in}\ e]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[x := v, y := w]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{case\ inl}\ v\ \mathbf{of}\ \{\langle i, x\rangle \to e\}f]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{case\ inr}\ w\ \mathbf{of}\ \{\langle i, x\rangle \to e\}f]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![f[w/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{unfold}\ (\mathbf{fold}\ v)]\!]^{n+1}_{\mathcal{E}} &\triangleq \delta_v \\
[\![(\lambda x.e)v]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{let}\ !x = !v\ \mathbf{in}\ e]\!]^{n+1}_{\mathcal{E}} &\triangleq [\![e[v/x]]\!]^n_{\mathcal{E}} \\
[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]^{n+1}_{\mathcal{E}} &\triangleq \sum_v [\![e]\!]^n_{\mathcal{E}}(v) \cdot [\![f[v/x]]\!]^n_{\mathcal{E}}
\end{aligned}
$$

Lemma 40. *For any type σ, the sequence of maps $([\![-]\!]^n_{\mathcal{E}})_{n \geq 0}$ forms an ω-chain in $\Lambda^\bullet_\sigma \rightharpoonup \mathcal{V}^\bullet_\sigma$.*

By Lemma 40, we define the evaluation map $[\![-]\!]_{\mathcal{E}} : \mathcal{P}\Lambda^\bullet_\sigma \to \mathbf{DV}_\sigma$ as $\bigsqcup_{n\geq 0}[\![-]\!]^n_{\mathcal{E}}$.

Lemma 41. *The following identities hold.*

$$
\begin{aligned}
[\![\mathbf{return}\ v]\!]_{\mathcal{E}} &\triangleq \mathit{just}\ v \\
[\![\mathbf{proj}_l\ \langle v, w\rangle]\!]_{\mathcal{E}} &= \mathit{just}\ v \\
[\![\mathbf{proj}_r\ \langle v, w\rangle]\!]_{\mathcal{E}} &= \mathit{just}\ w \\
[\![\mathbf{let}\ x \otimes y = v \otimes w\ \mathbf{in}\ e]\!]_{\mathcal{E}} &= [\![e[x := v, y := w]]\!]_{\mathcal{E}} \\
[\![\mathbf{case\ inl}\ v\ \mathbf{of}\ \{\langle i, x\rangle \to e\}f]\!]_{\mathcal{E}} &= [\![e[v/x]]\!]_{\mathcal{E}} \\
[\![\mathbf{case\ inr}\ w\ \mathbf{of}\ \{\langle i, x\rangle \to e\}f]\!]_{\mathcal{E}} &= [\![f[w/x]]\!]_{\mathcal{E}} \\
[\![\mathbf{unfold}\ (\mathbf{fold}\ v)]\!]_{\mathcal{E}} &= \mathit{just}\ v \\
[\![(\lambda x.e)v]\!]_{\mathcal{E}} &= [\![e[v/x]]\!]_{\mathcal{E}} \\
[\![\mathbf{let}\ !x = !v\ \mathbf{in}\ e]\!]_{\mathcal{E}} &= [\![e[v/x]]\!]_{\mathcal{E}} \\
[\![\mathbf{let}\ x = e\ \mathbf{in}\ f]\!]_{\mathcal{E}} &= \begin{cases} [\![f[v/x]]\!]_{\mathcal{E}} & \textit{if } [\![e]\!]_{\mathcal{E}} = \mathit{just}\ v \\ \bot & \textit{otherwise.} \end{cases}
\end{aligned}
$$

Let us now move to program distance. Since in a probabilistic setting we are mostly concerns with probabilistic observations (such as the probability of convergence of a program) whose outcomes belong to $[0, 1]$, it is natural to consider ternary relations with $[0, 1]$ in place of $[0, \infty]$, and thus distances taking values in $[0, 1]$. All we have seen in previous section is left unchanged: we simply replace $[0, \infty]$ with $[0, 1]$ (and addition with truncated addition).

To extend the logical distance and applicative (bi)similarity distance to F^{PR}, we essentially need a construction to extend a ternary relation $M \subseteq [0, 1] \times X \times Y$ to $\widetilde{\mathcal{D}}M \subseteq [0, 1] \times \mathcal{D}(X) \times \mathcal{D}(Y)$. As before, once such an extension is defined, we obtain an extension for $\mathcal{D}^{\leq 1}$ relying on the isomorphism $\mathcal{D}^{\leq 1}(X) \cong \mathcal{D}(X_\bot)$. We define $\widetilde{\mathcal{D}}M$ by generalising couplings to a ternary setting.

Definition 55. *For a relation $M \subseteq [0, 1] \times X \times Y$ defines $\widetilde{\mathcal{D}}M \subseteq [0, 1] \times \mathcal{D}(X) \times \mathcal{D}(Y)$ by:*

$$
\widetilde{\mathcal{D}}M(a, \mathscr{D}, \mathscr{E}) \overset{\triangle}{\iff} \exists \mathscr{C} \in \mathcal{D}([0, 1] \times X \times Y).\ \begin{cases} \sum_{b,y} \mathscr{C}(b, x, y) = \mathscr{D}(x) \\ \sum_{b,x} \mathscr{C}(b, x, y) = \mathscr{E}(y) \\ a \geq \sum_b b \cdot \sum_{x,y} \mathscr{C}(b, x, y) \\ \mathscr{C}(b, x, y) > 0 \implies M(b, x, y) \end{cases}
$$

[15] As usual, we omit type subscripts.

To understand Definition 55 let us consider the extension $\mathcal{D}(\pi_i) : \mathcal{D}(X_1 \times X_2) \to \mathcal{D}(X_i)$ of of the projection map $\pi_i : X_1 \times X_2 \to X_i$ to distributions defined by: $\mathcal{D}(\pi_1)(\mathscr{D})(x) \triangleq \sum_y \mathscr{D}(x,y)$ and $\mathcal{D}(\pi_2)(\mathscr{D})(y) \triangleq \sum_x \mathscr{D}(x,y)$. Notice that a distribution $\mathscr{C}$ is a coupling of $\mathscr{D}$ and $\mathscr{E}$ if and only if $\mathcal{D}(\pi_1)(\mathscr{C}) = \mathscr{D}$ and $\mathcal{D}(\pi_2)(\mathscr{C}) = \mathscr{E}$. Definition 55 stipulates that $\widetilde{\mathcal{D}}M(a,\mathscr{D},\mathscr{E})$ holds if and only if there exists a distribution $\mathscr{C}$ such that:

1. $\mathcal{D}(\pi_2)(\mathscr{C}) \in \mathcal{D}(X \times Y)$ is a coupling of $\mathscr{D}$ and $\mathscr{E}$;
2. a bounds the *expected value*[16] of $\mathcal{D}(\pi_1)(\mathscr{C}) \in \mathcal{D}[0,1]$.
3. Triples (b,x,y) in the support of $\mathscr{C}$ are related by M.

Proposition 13. *1. The map $\widetilde{\mathcal{D}}$ satisfies the following laws.*

$$\begin{aligned} \mathsf{I}_A &\subseteq \widetilde{\mathcal{D}}\mathsf{I}_A \\ \widetilde{\mathcal{D}}M; \widetilde{\mathcal{D}}N &\subseteq \widetilde{\mathcal{D}}(M;N) \\ M \subseteq N &\implies \widetilde{\mathcal{D}}M \subseteq \widetilde{\mathcal{D}}N. \end{aligned}$$

2. The following law holds, for all functions $\alpha, \beta : X \to \mathcal{D}(A)$ and relations $M : X \to\!\!\!\!\!\!+ X$, $N : A \to\!\!\!\!\!\!+ A$:

$$M \subseteq \alpha; \widetilde{\mathcal{D}}N; \beta^{\top} \implies \widetilde{\mathcal{D}}M \subseteq \gg\!\!=\alpha; \widetilde{\mathcal{D}}N; (\gg\!\!=\beta)^{\top}.$$

3. For any $s \geq 1$, we have:

$$[s](\widetilde{\mathcal{D}}M) \subseteq \widetilde{\mathcal{D}}([s]M).$$

At this point of the notes, it should be clear that once we have Proposition 13 we have all we need to define both logical distance and applicative bisimilarity distance and to prove that the latter are compatible pseudometrics. Proving Proposition 13, however, is not that straightforward. We only sketch the key insight behind such a proof. First, we observe that rephrasing our definition of $\widetilde{\mathcal{D}}$ in terms of distances, we obtain the well-known Wasserstein-Kantorovich distance.

Definition 56. *Given a distance $\mu : X \times Y \to [0,1]$, defines $\widetilde{\mathcal{D}}\mu : \mathcal{D}(X) \times \mathcal{D}(Y) \to [0,1]$ by:*

$$\widetilde{\mathcal{D}}\mu(\mathscr{D},\mathscr{E}) \triangleq \inf_{\mathscr{C} \in \Omega(\mathscr{D},\mathscr{E})} \sum_{x,y} \mu(x,y) \cdot \mathscr{C}(x,y).$$

Therefore, to prove Proposition 13 is is sufficient to show the metric-like counterparts of the structural properties of $\widetilde{\mathcal{D}}$, namely:

$$\begin{aligned} \mathsf{I}_A &\geq \widetilde{\mathcal{D}}\mathsf{I}_A \\ \widetilde{\mathcal{D}}\mu; \widetilde{\mathcal{D}}\nu &\geq \widetilde{\mathcal{D}}(\mu;\nu) \\ \mu \geq N &\implies \widetilde{\mathcal{D}}\mu \geq \widetilde{\mathcal{D}}\nu \\ \mu \geq \alpha; \widetilde{\mathcal{D}}\nu; \beta^{\top} &\implies \widetilde{\mathcal{D}}\mu \geq \gg\!\!=\alpha; \widetilde{\mathcal{D}}\nu; (\gg\!\!=\beta)^{\top} \\ [s](\widetilde{\mathcal{D}}\mu) &\geq \widetilde{\mathcal{D}}([s]\mu), \end{aligned}$$

where $s \geq 1$.

To prove such properties, we use the celebrated Kantorovich-Rubinstein Theorem, which here we give in a less general form (Kortanek & Yamasaki, 1995).

[16] Recall that the expected value of a distribution $\mathscr{D} \in \mathcal{D}[0,1]$ is defined as $\sum_x x \cdot \mathscr{D}(x)$.

Theorem 8. *Let $i, j, \ldots$ range over natural numbers. Let m_i, n_j, c_{ij} be non-negative real numbers, for all i, j. Define*

$$M \triangleq \inf\{\sum_{i,j} c_{ij}x_{ij} \mid x_{ij} \geq 0, \sum_j x_{ij} = m_i, \sum_i x_{ij} = n_j, \}$$
$$M^* \triangleq \sup\{\sum_i m_i a_i + \sum_j n_j b_j \mid a_i + b_j \leq c_{ij}, a_i, b_j \textit{ bounded}\}.$$

Then the following hold:

1. *$M = M^*$.*
2. *The linear problem P induced by M has optimal solution.*
3. *The linear problem P^* induced by M^* has optimal solution.*

Corollary 3. *Let $\mathscr{D} \in \mathcal{D}(X), \mathscr{E} \in \mathcal{D}(Y)$ be countable distributions and $\mu : X \times Y \to [0, 1]$. Then:*

$$\widetilde{\mathcal{D}}\mu(\mathscr{D}, \mathscr{E}) = \min\{\sum_{x,y} \mu(x, y) \cdot \mathscr{C}(x, y) \mid \mathscr{C} \in \Omega(\mathscr{D}, \mathscr{E})\}$$
$$= \max\{\sum\nolimits_x a_x \cdot \mathscr{D}(x) + \sum\nolimits_y b_y \cdot \mathscr{E}(y) \mid a_x + b_y \leq \mu(x, y), a_x, b_y \textit{ bounded}\},$$

where a_x, b_y bounded means that there exist $\bar{a}, \bar{b} \in \mathbb{R}$ such that $\forall x.\ a_x \leq \bar{a}$, and $\forall y.\ b_y \leq \bar{b}$.

Exercise 48. Prove Corollary 3

Exercise 49. Let I be a finite set and $p_{i\,i \in I}$ be a family of numbers in $[0, 1]$ such that $\sum_i p_i \leq 1$. Show that $\widetilde{\mathcal{D}}\mu(\mathscr{D}_i, \mathscr{E}_i) = \sum_{i \in I} p_i \cdot \widetilde{\mathcal{D}}\mu(\mathscr{D}_i, \mathscr{E}_i)$.

We now have all the ingredients to define probabilistic logical relations for the fragment of F^{PR} without recursive types.

Definition 57. *Define logical distance $\overset{\mathsf{L}}{\sim}$ as follows:*

$$\begin{aligned}
\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}} \\
\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma \times \tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma} \times \overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma \otimes \tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma} \otimes \overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma + \tau} &\triangleq [\![-]\!]_{\mathcal{L}}; (\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma} + \overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma}); [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma \multimap \tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma} \to \overset{\mathsf{L}}{\sim}{}^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{!_s \sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; [s]\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\
\overset{\mathsf{L}}{\sim}{}^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \widetilde{\mathcal{D}}\overset{\mathsf{L}}{\sim}{}^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}}.
\end{aligned}$$

Relying on Proposition 13, we can mimic the usual argument seen so far to prove good properties of logical relations, this way obtaining the following results.

Proposition 14 (Fundamental Lemma). $\mathsf{I} \subseteq \overset{\mathsf{L}}{\sim}{}^{s}$.

Theorem 9. *$\overset{\mathsf{L}}{\sim}$ is a reflexive, symmetric, transitive, compatible, and substitutive term relation.*

3.8.1 Applicative Bisimilarity

Finally, we can extend our analysis to full F^{PR} by defining a notion of probabilistic applicative bisimilarity distance. Conceptually, we do that exactly in the same way as we defined probabilistic applicative bisimilarity.

First, we define $\widetilde{\mathcal{DM}}$ by $\widetilde{\mathcal{DM}}M \triangleq \widetilde{\mathcal{D}}(\widetilde{\mathcal{M}}(M))$. In particular, $\widetilde{\mathcal{DM}}$ inherits all the structural properties of $\widetilde{\mathcal{M}}$ and $\widetilde{\mathcal{DM}}$ needed to prove reflexivity, transitivity, and compatibility of applicative similarity.

Proposition 15. *1. The map $\widetilde{\mathcal{DM}}$ satisfies the following laws.*

$$\begin{aligned} \mathsf{I}_{A_\perp} &\subseteq \widetilde{\mathcal{DM}}\mathsf{I}_A \\ \widetilde{\mathcal{DM}}M; \widetilde{\mathcal{DM}}N &\subseteq \widetilde{\mathcal{DM}}(M;N) \\ M \subseteq N &\implies \widetilde{\mathcal{DM}}M \subseteq \widetilde{\mathcal{DM}}N. \end{aligned}$$

2. The following law holds, for all functions $\alpha, \beta : X \to \mathcal{D}(A)$ and relations $M : X \twoheadrightarrow X$, $N : A \twoheadrightarrow A$:

$$M \subseteq \alpha; \widetilde{\mathcal{DM}}N; \beta^\top \implies \widetilde{\mathcal{DM}}M \subseteq {\gg\!=}\alpha; \widetilde{\mathcal{DM}}N; ({\gg\!=}\beta)^\top.$$

3. For all $s \geq 1$, we have:

$$[s](\widetilde{\mathcal{DM}}\mu) \geq \widetilde{\mathcal{DM}}([s]\mu).$$

Definition 58. *Given a closed term relation M, we define the closed term relation $[M]$ as follows:*

$$\begin{aligned} [M]^{\mathcal{V}}_{\mathbf{unit}} &\triangleq \mathsf{I}^{\mathcal{V}}_{\mathbf{unit}} \\ [M]^{\mathcal{V}}_{\mathbf{void}} &\triangleq \emptyset \\ [M]^{\mathcal{V}}_{\sigma\times\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma} \times M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\ [M]^{\mathcal{V}}_{\sigma\otimes\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma} \otimes M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\ [M]^{\mathcal{V}}_{\sigma+\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma} + M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\ [M]^{\mathcal{V}}_{\mu t.\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; M^{\mathcal{V}}_{\sigma[t/\mu t.\sigma]}; [\![-]\!]^{\top}_{\mathcal{L}} \\ [M]^{\mathcal{V}}_{!_s\sigma} &\triangleq [\![-]\!]_{\mathcal{L}}; [s]M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{L}} \\ [M]^{\mathcal{V}}_{\sigma\multimap\tau} &\triangleq [\![-]\!]_{\mathcal{L}}; [\mathsf{I}^{\mathcal{V}}_{\sigma} \to M^{\Lambda}_{\tau}]; [\![-]\!]^{\top}_{\mathcal{L}} \\ M^{\Lambda}_{\sigma} &\triangleq [\![-]\!]_{\mathcal{E}}; \widetilde{\mathcal{DM}}M^{\mathcal{V}}_{\sigma}; [\![-]\!]^{\top}_{\mathcal{E}} \end{aligned}$$

A term relation M is an applicative simulation *if $M \subseteq [M]$.*

Since $\widetilde{\mathcal{DM}}$ is monotone, then so is $[-]$ so that we can define applicative similarity as the greatest fixed point of $[-]$:

$$\overset{\mathrm{a}}{\preceq} \triangleq \nu M.[M].$$

As usual, we define $\overset{\mathrm{a}}{\simeq}$ as $\overset{\mathrm{a}}{\preceq} \cap (\overset{\mathrm{a}}{\preceq})^{\top}$.

Finally, by relying on a slight generalisation of Howe's method as highlighted in Section ???, one can prove that applicative bisimilarity distance is a compatible pseudometric.

Theorem 10. *Applicative similarity is reflexive, transitive, and compatible. Applicative bisimilarity is, additionally, symmetric.*

Chapter 4

Contextual Differences

4.1 Making the Quantale To Vary

Notions of program equivalence and program metrics as introduced in the last two chapters are very conservative as for how the environment could act when trying to distinguish between the two compared programs. If you consider, e.g., contextual equivalence, it is clear that *all* contexts, namely, all environments, are taken into account. Other notions of equivalences are even *more discriminating*. Similarly for metrics. As a consequence, if one tries to compare programs which are not equivalent although behaving very similarly *only when* the environment interact with them in a benign way, those programs are put very far away from each other.

4.2 Endowing Λ^{ST} with Real Numbers

For the sake of simplicity, our vehicle calculus will tbe the simply-typed calculus Λ^{ST} where, however, ground types include ot only **bool** and **unit**, but also a type of real number **real**.

4.3 Differential Logical Relations

Types of Λ are built starting from a countable set of type variables (denoted by the letter α in Figure 4.1) and the basic type `real` for real numbers, using finite sum, arrow, and recursive type constructors. In particular, in a type of the form $\sum_{i \in I} \sigma_i$ we assume the letter I to stand for a finite set whose elements are denoted by $\hat{\imath}, \hat{\jmath}, \ldots$. We write **void** and **unit** for the empty sum type and **void** $\to$ **void**, respectively.

Terms of Λ are divided in two classes: values and expressions (also called computations). Intuitively, a value is the result of a computation, whereas an expression is a value producer, i.e. a term that once evaluated may produce a value (the evaluation process might not terminate) as well as side effects (the latter being probabilistic). Accordingly, computations must be explicitly sequenced by means of the sequencing constructor **let** $x = -$ **in** $-$.

We use judgments of the form $\Gamma \vdash^{\Lambda} e : \sigma$ for expressions, and $\Gamma \vdash^{\mathcal{V}} v : \sigma$ for values. In a judgment of the form $\Gamma \vdash^{\Lambda} e : \sigma$ (resp. $\Gamma \vdash^{\mathcal{V}} v : \sigma$), Γ denotes an environment, i.e. a finite sequence $x_1 : \sigma_1, \ldots, x_n : \sigma_n$ of distinct variables with associated *closed* types (we denote by $\cdot$ the empty environment), σ is a *closed* type, and e (resp. v) is an expression (resp. value) with free variables among Γ. Notice that we work with closed types only. We use the letter e (possibly with sup- and subscripts) to denote expressions, and v (possibly with sup- and subscripts) to denote values. We also refer to sequents $\Gamma \vdash^{\Lambda} \sigma$ as computation or expression sequents, and to $\Gamma \vdash^{\mathcal{V}} \sigma$ as value sequents. When the distinction between values and expressions is not relevant, we generically refer to terms.

Each real number r is represented in Λ_{P} by the value `r`. Additionally, Λ_{P} is parametric with respect to a $\mathbb{N}$-indexed family C of *countable* sets C_n, each of which contains (symbols for) measurable functions from $\mathbb{R}^n$ to $\mathbb{R}$. In particular, we assume standard real-valued arithmetic operations to be in C. We use letters $F, G, \dots$ to denote elements in C_n. Notice that a function $F \in C_n$ has values as arguments, rather than expressions. Indeed, we can encode the expression $F(e_1, \dots, e_n)$ as[1] $\mathtt{let}\ x_1 = e_1\ \mathtt{in}\ (\mathtt{let}\ x_2 = e_2\ \mathtt{in}\ \dots(\mathtt{let}\ x_n = e_n\ \mathtt{in}\ F(x_1, \dots, x_n)))$.

Finally, the expression **sample** stands for the uniform distribution over the unit interval, which is nothing but the Lebesgue measure λ on $[0,1]$. As it is shown in e.g. **??**, starting from **sample** it is possible to define several probabilistic measures (e.g. binomial, geometric, and exponential distribution) using functions in C.

Example 4. Given closed terms e_μ, e_σ of type $\mathtt{real}$ (encoding mean μ and standard deviation σ) we represent the normal distribution with mean μ and standard deviation σ as the Λ_{P} expression $\mathtt{normal}\ e_\mu\ e_\sigma$, where the term $\mathtt{normal}_{\mathtt{std}}$ encodes the normal distribution with mean 0 and standard deviation 1. Notice that the expressions e_μ, e_σ may be themselves defined in terms of other distributions:

$$\mathtt{normal}_{\mathtt{std}} \triangleq \mathbf{let}\ x = \mathbf{sample\ in}\ (\mathbf{let}\ y = \mathbf{sample\ in\ return}\ (\sqrt{-2\log(x)}\cos(2\pi y)))$$
$$\mathtt{normal}\ e_\mu\ e_\sigma \triangleq \mathtt{let}\ x_\mu = e_\mu,\ x_\sigma = e_\sigma, y = \mathtt{normal}_{\mathtt{std}}\ \mathtt{in}\ ((x_\sigma * y) + x_\mu).$$

⊠

We adopt the standard syntactical conventions as in Barendregt (1984), notably the so-called variable convention. In particular, we denote by $FV(e)$ (resp. $FV(v)$) the collection of free variables of e (resp. v) (notice that, e.g., in a term of the form **case** v **of** $\{$**fold** $x \to e\}$ the variable x is bound in e). We refer to closed expressions as *programs*. We denote by $e[v/x]$ (resp. $v'[v/x]$) the capture-free substitution of the value v for all free occurrences of x in e (resp. v') and identify terms up to renaming of bound variables. We extend the aforementioned conventions to types. For instance, we denote by $\sigma_1[\sigma_2/\alpha]$ the result of capture-avoiding substitution of the type σ_2 for the type variable α in σ_1. Finally, we use the notation Λ and $\mathcal{V}$ to denote the collections of all typable expressions and values, respectively.

$$\dfrac{}{\Gamma, x:\sigma \vdash^{\mathcal{V}} x:\sigma} \qquad \dfrac{\Gamma \vdash^{\mathcal{V}} v:\sigma}{\Gamma \vdash^{\Lambda} v:\sigma} \qquad \dfrac{r \in \mathbb{R}}{\Gamma \vdash^{\mathcal{V}} \mathtt{r}:\mathtt{real}} \qquad \dfrac{}{\Gamma \vdash^{\Lambda} \mathbf{sample}:\mathtt{real}} \qquad \dfrac{\Gamma \vdash^{\mathcal{V}} v_1:\mathtt{real} \quad \cdots \quad \Gamma \vdash^{\mathcal{V}} v_n:\mathtt{real} \quad F \in C_n}{\Gamma \vdash^{\Lambda} F(v_1,\dots,v_n):\mathtt{real}}$$

$$\dfrac{\Gamma, x:\sigma_1 \vdash^{\Lambda} e:\sigma_2}{\Gamma \vdash^{\mathcal{V}} \lambda x.e:\sigma_1 \to \sigma_2} \qquad \dfrac{\Gamma \vdash^{\mathcal{V}} v_1:\sigma_1 \to \sigma_2 \quad \Gamma \vdash^{\mathcal{V}} v_2:\sigma_2}{\Gamma \vdash^{\Lambda} v_1 v_2:\sigma_2} \qquad \dfrac{\Gamma \vdash^{\Lambda} e_1:\sigma_1 \quad \Gamma, x:\sigma_1 \vdash^{\Lambda} e_2:\sigma_2}{\Gamma \vdash^{\Lambda} \mathbf{let}\ x = e_1\ \mathbf{in}\ e_2:\sigma_2} \qquad \dfrac{\Gamma \vdash^{\mathcal{V}} v:\sigma_{\hat{\imath}}}{\Gamma \vdash^{\mathcal{V}} \langle \hat{\imath}, v\rangle : \sum_{i\in I}\sigma_i}$$

$$\dfrac{\Gamma \vdash^{\mathcal{V}} v:\sum_{i\in I}\sigma_i \quad \Gamma, x:\sigma_i \vdash^{\Lambda} e_i:\sigma\ (\forall i \in I)}{\Gamma \vdash^{\Lambda} \mathbf{case}\ v\ \mathbf{of}\ \{\langle i, x\rangle \to e_i\}:\sigma} \qquad \dfrac{\Gamma \vdash^{\mathcal{V}} v:\sigma[\mu\alpha.\sigma/\alpha]}{\Gamma \vdash^{\mathcal{V}} \mathbf{fold}\ v:\mu\alpha.\sigma} \qquad \dfrac{\Gamma \vdash^{\mathcal{V}} v:\mu\alpha.\sigma_1 \quad \Gamma, x:\sigma_1[\mu\alpha.\sigma_1/\alpha] \vdash^{\Lambda} e:\sigma_2}{\Gamma \vdash^{\Lambda} \mathbf{case}\ v\ \mathbf{of}\ \{\mathbf{fold}\ x \to e\}:\sigma_2}$$

Figure 4.1: Static semantics of Λ_{P}.

4.4 Generalized Metric Domains

[1]This encoding reflects the standard semantics of operations, where to evaluate an expression of the form $F(e_1, \dots, e_n)$ one first sequentially evaluates its arguments, proceeding from left to right.

References

Barendregt, H. (1984). *The lambda calculus: its syntax and semantics.* North-Holland.

Crubillé, R., & Dal Lago, U. (2015). Metric reasoning about lambda-terms: The affine case. In *Proc. of LICS 2015* (pp. 633–644).

Crubillé, R., & Dal Lago, U. (2017). Metric reasoning about lambda-terms: The general case. In *Proc. of ESOP 2017* (pp. 341–367).

de Amorim, A., Gaboardi, M., Hsu, J., Katsumata, S., & Cherigui, I. (2017). A semantic account of metric preservation. In *Proc. of POPL 2017* (pp. 545–556).

Gaboardi, M., Katsumata, S., Orchard, D. A., Breuvart, F., & Uustalu, T. (2016). Combining effects and coeffects via grading. In *Proc. of ICFP 2016* (pp. 476–489).

Girard, J., Scedrov, A., & Scott, P. (1992). Bounded linear logic: A modular approach to polynomial-time computability. *Theor. Comput. Sci.*, *97*, 1–66.

Gordon, A. (1994, September). A tutorial on co-induction and functional programming. In *Workshops in computing* (p. 78-95). Springer London.

Harper, R. (2016). *Practical foundations for programming languages (2nd. ed.).* Cambridge University Press. Retrieved from https://www.cs.cmu.edu/%7Erwh/pfpl/index.html

Hofmann, D., Seal, G., & Tholen, W. (Eds.). (2014). *Monoidal topology. a categorical approach to order, metric, and topology* (No. 153). Cambridge University Press.

Howe, D. (1996). Proving congruence of bisimulation in functional programming languages. *Inf. Comput.*, *124*(2), 103-112.

Katsumata, S. (2018). A double category theoretic analysis of graded linear exponential comonads. In *Proc. of FOSSACS 2018* (pp. 110–127).

Kelly, G. M. (2005). Basic concepts of enriched category theory. *Reprints in Theory and Applications of Categories*(10), 1–136.

Kortanek, K., & Yamasaki, M. (1995). Discrete infinite transportation problems. *Discrete Applied Mathematics*(58), 19–33.

Lassen, S. (1998). *Relational reasoning about functions and nondeterminism* (Unpublished doctoral dissertation). Dept. of Computer Science, University of Aarhus.

Lawvere, F. (1973). Metric spaces, generalized logic, and closed categories. *Rend. Sem. Mat. Fis. Milano*, *43*, 135–166.

Levy, P., Power, J., & Thielecke, H. (2003). Modelling environments in call-by-value programming languages. *Inf. Comput.*, *185*(2), 182–210.

Morris, J. (1969). *Lambda calculus models of programming languages* (Unpublished doctoral dissertation). MIT.

Plotkin, G. (1975). Call-by-name, call-by-value and the lambda-calculus. *Theoretical Computer Science*, *1*(2), 125 - 159.

Reed, J., & Pierce, B. (2010). Distance makes the types grow stronger: a calculus for differential privacy. In *Proc. of ICFP 2010* (pp. 157–168).